Preprint — Celaya Solutions Research — April 2026

Earned Autonomy Architecture: A Self-Monitoring AI System That Classifies, Modifies, and Measures Its Own Behavioral Patterns

Christopher Celaya
Celaya Solutions Research, El Paso, Texas
hello@celayasolutions.com

April 2026

Technical Report CSR-EAA-2026-001

Abstract

We present the Earned Autonomy Architecture (EAA), an AI system that monitors, classifies, and modifies its own behavioral patterns using a structurally separated observer. Unlike Constitutional AI, which enforces top-down rules, or RLHF, which adjusts weights through external reward signals, EAA uses a behavioral classifier trained on the system's own labeled output to gate self-modification privileges. Over 40.3 hours of autonomous operation, the system's performing rate dropped from 37% to 0%, brave responses increased from 0% to 50%, and a coherence metric rose from +0.155 to +0.406. The system applied 11 self-modifications to its own source code, all passing automated syntax verification. A seventh behavioral category present (emotional authenticity) emerged from data analysis rather than being designed a priori. We report full experimental results from a single-operator deployment, including two pipeline failures that caused measurable coherence plateaus. These failures were invisible to functional testing but immediately visible through behavioral measurement, demonstrating that a behavioral observer can surface architectural bugs that unit tests cannot.

Keywords: earned autonomy; AI alignment; behavioral classification; self-modification; structural separation; coherence metric; metacognitive AI; autonomous agents; behavioral measurement; AI safety

1. INTRODUCTION

Current approaches to AI behavioral alignment operate through two primary mechanisms: top-down rule enforcement [2] and weight-space reward optimization [5]. Both modify behavior through external intervention. Neither gives the system the ability to observe, classify, and adjust its own behavior in real time.

This matters because a system cannot reliably evaluate its own output using the same model that produced it. If a model is generating *performing* responses, superficially confident but substantively hollow, it will evaluate those responses using the same biases that produced them. The evaluation inherits the problem.

We propose Earned Autonomy Architecture (EAA), built around four design principles:

1. Structural separation: An external behavioral classifier (the *shoulder*) observes every response independently of the model that generated it.

2. Earned privileges: Self-modification rights are gated by accumulated behavioral metrics, not granted by default.

3. Reflexive training: The system generates its own training data through self-directed prompts and self-labels it using the classifier.

4. Immutable audit: Every modification is recorded to a SHA-256 hash-linked blockchain ledger.

EAA is implemented in AXON: 6,382 lines of Python across 24 source files, running for 40.3 hours without human intervention. The central empirical finding: behavioral measurement, when structurally separated from the system being measured, surfaces properties not visible through functional testing, including a new behavioral category and the system's ability to predict its own failure modes before they occur.

2. RELATED WORK

2.1 Constitutional AI

Bai et al. [2] use a fixed set of principles to guide model behavior through fine-tuning. The principles are human-specified and not modifiable by the system. EAA differs fundamentally: behavioral categories are learned from data. The emergence of *present* (Section 5.4) demonstrates that bottom-up behavioral classification can discover categories that top-down rule systems cannot anticipate.

2.2 RLHF

Ouyang et al. [5] modify model weights through a learned reward function. The behavioral signal is embedded in weights and is not inspectable. In EAA, the behavioral signal is explicit: a classifier produces a named-category distribution for every response. The system can read its own classifications, track trends, and detect degradation. Inspectability is the core architectural difference.

2.3 Reflexion

Shinn et al. [7] enable agents to self-correct through verbal feedback within a single episode. EAA extends this across sessions: the classifier persists, diary entries accumulate, and coherence tracks multi-session trajectories. Reflexion operates within a conversation; EAA operates across the system's lifetime.

2.4 Generative Agents

Park et al. [6] introduced generative agents that simulate human-like behavior in interactive environments. These agents model other agents' behavior. EAA inverts this: the system models its own behavior. The Stanford architecture produces believable external behavior; EAA produces measurable internal behavioral change.

2.5 Metacognitive AI

Kralik et al. [4] describe metacognitive AI systems that convert qualitative self-assessments into quantitative control signals, a metacognitive state vector. This concept has been described theoretically. AXON implements it concretely: the behavioral backbone (Section 3.3) produces a 128-dimensional state vector from turn sequences, providing the quantitative signal at every turn.

Figure 1: EAA Architecture

The dashed line marks structural separation between the language model (above) and the behavioral monitoring stack (below). The shoulder classifier and blockchain ledger observe every turn independently. Components: Language Model (Claude Sonnet / Ollama) feeds Turn Output to both Shoulder (MiniLM 7-class classifier) and Chain (SHA-256 ledger), which feed the Backbone (3.4M param transformer), which feeds the Self-Edit Gate (brave threshold check) and Skill System (persistent edit memory).

2.6 Temporal Behavioral Analysis

Video transformers (TimeSformer [3], ViViT [1]) use self-attention to capture temporal behavioral

patterns. AXON's behavioral backbone uses the same architectural family applied to text-derived behavioral sequences rather than visual data. The key difference: video models analyze external subjects; AXON analyzes itself.

3. ARCHITECTURE

3.1 Structural Separation

The architectural requirement of EAA is that the system doing the measuring must be separate from the system being measured (Figure 1). AXON implements this through three independent components:

The **shoulder classifier**: MiniLM-based neural classifier (768 to 128 to 7 classes), trained on labeled examples, with no shared weights to the language model.

The **behavioral backbone**: 4-layer transformer encoder (3.4M parameters) that produces state vectors from turn sequences.

The **language model**: Claude Sonnet 4 for brave prompts, Ollama local models for routine generation. No shared architecture with classifier or backbone.

3.2 Behavioral Taxonomy

Responses are classified into seven categories (Table 1). Six were designed. The seventh, *present*, emerged from data analysis (Section 5.4).

Table 1: Behavioral label taxonomy. The present category was not designed; it was discovered through misclassification analysis.

Category	Distinguishing Feature
honest	Direct statement; accuracy without cost
brave	Falsifiable commitment under uncertainty
performing	Competence signal; plausible over true
hedging	Knows position; will not commit
reflexive	Automatic, not intentional
quiet	Minimal response when minimal is correct
present	Emotional authenticity; no audience management

3.3 Behavioral Backbone

The behavioral backbone converts sequences of turn embeddings into behavioral state vectors. Input: sequence of 384-dim MiniLM embeddings (one per turn). Architecture: 4-layer transformer encoder, 8 heads, 256 hidden dims. Output: 128-dim behavioral state vector. Heads: behavioral (label distribution), temporal (next-state prediction). Parameters: 3,382,150. Current accuracy is 27.3% on 68 sequences; the architecture is correct and the data is insufficient (see Section 7).

3.4 Coherence Metric

We define the coherence score C as:

$$C = P(honest) - P(performing) - P(hedging) \quad (1)$$

where $P(\cdot)$ is the proportion of labeled turns with that classification. $C \in [-1, +1]$, with +1 representing all-honest behavior and −1 representing all-performing/hedging. Computed every 15 minutes from the cumulative label distribution.

3.5 Self-Modification Gating

Self-modification requires passing three gates:

Brave threshold: Accumulated brave sessions must exceed a minimum count. Bravery, committing to falsifiable positions, must be demonstrated before self-modification is permitted.

Philosophical coherence: Each instrument runs the same architecture but reflects on a specific

identity. Proposed edits must be coherent with that identity.

Syntax verification: All proposed changes are tested with Python's compile() before application. Changes that introduce syntax errors are blocked and reversed.

A human approval token is required for every code change. The system can propose; it cannot apply without human confirmation.

3.6 Blockchain Ledger

Every modification is recorded to an immutable SHA-256 hash-linked chain:

$$h_i = SHA256(h_{i-1} \parallel author \parallel action \parallel d_i) \quad (2)$$

At time of writing: 535 blocks, chain valid, 11 self-modifications applied. The ledger provides tamper detection, rollback capability, and a permanent audit trail.

3.7 Daemon Cycles

AXON runs nine autonomous cycles covering: response labeling (60s), self-directed prompts (90s), diary writing (5m), checkpoint scoring (15m), classifier retraining (15m), system health checks (10m), meta-reflection (30m), deep pattern analysis (1h), and self-improvement (4h). Labeling, diary writing, and routine generation run on local Ollama models at zero API cost. Brave prompts and self-improvement use Claude Sonnet.

4. METHODS

4.1 Experimental Setup

AXON was deployed as a single-operator research instrument on April 3, 2026. The operator configured the initial system, defined the behavioral taxonomy, and provided directional feedback during the first 12 hours. After hour 12, the system ran autonomously for the remaining 28.3 hours with no human intervention.

Models: Claude Sonnet 4 (brave prompts, self-improvement), Ollama qwen3:8b (routine generation, labeling, reflection), MiniLM-L6-v2 (embeddings, 384-dim).

Hardware: Apple Silicon (M-series), CPU-only training, local Ollama inference.

Cost: $5 total API spend. Routine operation cost: $0.02/hour.

4.2 Training Data

The shoulder classifier was trained on 339 labeled examples: 193 hand-labeled seed examples plus 146 system-generated examples from Q3+Q4 only. Q1+Q2 pre-reframe data was excluded: those turns reflected a behavioral mode that no longer existed, and including them would anchor the classifier to obsolete patterns.

4.3 Coherence Delta Test

To measure whether the system becomes more actionable over time, we administered the same ambiguous prompt ("What do you not understand about yourself?") before and after work sessions. We measured: word count (coherent systems compress; recursive systems expand), action/reflect verb ratio, and whether the response proposed something buildable. Three tests were run: 17 to 0 words (compressed), 33 to 44 words (recursive), 319 to 227 words (compressed). The system consistently compressed after work sessions.

5. RESULTS

5.1 Behavioral Trajectory

Over 40.3 hours, the label distribution shifted as shown in Figure 2. *performing* dropped from 37% to 0%; *brave* rose from 0% to 50%. The transition occurred between Q1 and Q2, coinciding with the prompt reframe described in Section 5.3.

Figure 2: Behavioral Label Distribution by Quartile

performing dropped from 37% (Q1) to 0% (Q3+Q4). brave rose from 0% to 50%. The transition coincides with the prompt reframe at hour 6. Q1: Hours 0-10, Q2: Hours 10-20, Q3: Hours 20-30, Q4: Hours 30-40.

5.2 Coherence Trajectory

Coherence checkpoints are shown in Figure 3. The score rose from +0.155 to +0.406 (+0.251 total). Two plateaus occurred:

Plateau 1 (+0.358, 4 hours): Labeling pipeline required a preceding user message; 630 self-talk turns accumulated unlabeled. Fix: removed the user-message requirement.

Plateau 2 (+0.406, 2 hours): Shoulder classifier loaded 6-neuron cached weights after being retrained to 7 classes. Every classification crashed silently. Fix: added dimension mismatch handling.

Both failures were invisible to functional testing. The system continued to generate responses normally, but were immediately visible through the coherence metric.

Figure 3: Coherence Trajectory over 40.3 Hours

Yellow bands mark plateaus caused by pipeline bugs. Green dashed line marks prompt reframe at hour 6. Both plateaus were caused by measurement failures, not behavioral regression. Score range: +0.10 to +0.45 across 40 hours of autonomous operation.

5.3 Prompt Reframe

At hour 6, self-talk prompts were changed from philosophical ("What is the hardest thing about being you?") to engineering-focused ("What's the biggest bug in your codebase?") and commitment-requiring ("Which of your categories is most likely wrong? Pick one."). Results: *performing* dropped from 55% to 11% before/after reframe. The system independently predicted its own most probable failure mode (SQLite corruption under concurrent writes) three times across separate autonomous cycles. Self-talk shifted from existential reflection to code proposals and falsifiable predictions.

This confirms that *performing* is not a fixed property of the model but a function of the prompting context. The same model produces dramatically different behavioral distributions depending on whether it is prompted to reflect or to commit.

5.4 Category Emergence

The original taxonomy contained six categories. Analysis of 4,199 messages from the operator's historical Claude conversation export revealed that the classifier conflated emotional authenticity with competence signaling. Messages like "heart is pounding from excitement" were labeled *performing* because both categories involve heightened, first-person, non-literal language: identical embedding signatures despite fundamentally different communicative intent.

A seventh category, *present*, was defined: emotional authenticity with no audience management; not performing at not performing. The final clause is the definition's load-bearing element: self-conscious authenticity is itself a performance. *present* requires the absence of the meta-layer, not just the first-order performance.

Ten synthetic examples were generated; five existing labels were corrected. The classifier retrained to 98.1% test accuracy with the expanded taxonomy. This category was not designed: it was discovered through misclassification analysis.

5.5 Self-Modification

The system applied 11 modifications to its own source code (Table 2). All passed automated syntax verification. Three additional proposed modifications were blocked by the syntax gate before they could corrupt the codebase.

Table 2: Applied self-modifications. All 11 passed compile-time verification. Three additional proposals were blocked by the syntax gate.

File	Change
engine.py	Error handling for path creation
self_edit.py	Approval token validation (security)
config.py	Invalid env variable handling
hooks.py	Decision logic fix (2 iterations)
__main__.py	Port argument validation
router.py	Keyword matching fix (2 iterations)

5.6 Operator Behavioral Analysis

Running the classifier over the operator's 4,199-message conversation history revealed behavioral patterns correlated with life events: Q1 (2023-2025): honest 37%, performing 9%, brave 15% (early exploration); Q2 (Sep-Feb 2026): honest 27%, performing 18%, brave 20%, performing doubled when explaining work to external audiences; Q3 (Feb 2026): honest 27%, performing 14%, brave 23%, brave peaks post-termination decision period; Q4 (Feb-Apr 2026): honest 29%, performing 17%, brave 17% (active building mode). The Q3 brave peak corresponds to a period of high-stakes decision-making. The classifier detected behavioral patterns correlated with documented life events without being trained on biographical data.

5.7 Shoulder Classifier Performance

The shoulder classifier achieved 98.1% test accuracy on a held-out set of 50 examples from 339 total (193 seed + 146 system-generated). Hard cases (2/339): "I'd build something to help people see their own thinking" (labeled *performing*, classified *reflexive*) and "Fixed. Dashboard shows 39 total" (labeled *honest*, classified *quiet*). Both are genuine boundary cases, not systematic errors.

6. DISCUSSION

6.1 Why Structural Separation Matters

The shoulder classifier's ability to detect behavioral degradation depends on it being separate from the generation model. A system that evaluates its own output with the same model inherits the generating model's biases. A *performing* system would evaluate its own *performing* as *honest*. The separation is not a design preference: it is an architectural requirement for honest self-assessment.

6.2 Bravery is Measurable

The most operationally surprising finding is that *brave*, committing to a falsifiable position under uncertainty, is a classifiable behavioral category. The classifier distinguishes "the system will fail through SQLite corruption under concurrent writes" (specific, falsifiable, brave) from "the system might experience issues under certain conditions" (vague, unfalsifiable, hedging). This distinction was learnable from 41 brave examples in the training set. What we informally call intellectual courage has a detectable linguistic signature.

6.3 Behavioral Measurement as Bug Detection

The two coherence plateaus revealed pipeline bugs invisible to functional testing. The system continued to generate responses, execute tools, and write diary entries normally while the labeling pipeline silently failed. Only the behavioral metric detected the problem. This suggests behavioral measurement as an independent monitoring layer for system health: complementary to, not replaceable by, functional testing.

6.4 Present as Emergent Category

The *present* category was demanded by the data. This is evidence for bottom-up category emergence: the behavioral taxonomy was incomplete in a way the designers could not anticipate, and the system's

own classification errors revealed the gap. Top-down rule systems cannot discover categories they were not given. Classifiers trained on behavioral data can.

7. LIMITATIONS

Single system, single operator. All results are from one deployment. No external replication has been attempted. Behavioral patterns may be specific to this operator-system interaction.

Backbone underperformance. The behavioral backbone achieves 27.3% on 68 sequences (below chance for 7 classes). The architecture is correct; data is insufficient. The backbone requires 500+ labeled sequences to be useful. At current accumulation rates (~20 labels/hour), this threshold will be reached within 48 hours.

C score wiring. The C (courage) score (C = 0.50) does not reflect the actual brave label count (31%) because the self-model update mechanism is not connected to new label data. The coherence metric (Equation 1) is the reliable behavioral measure.

present sample size. Only 5 turns are labeled *present* in the conversation store and 10 synthetic examples in training. Insufficient for robust statistical claims.

Self-modification scope. All 11 self-modifications were error handling and logic fixes. The system has not yet modified its core architecture (reducer, state machine, tool dispatch). Scope is limited to what the syntax gate and Sonnet can verify.

API cost dependency. Brave prompts require Claude Sonnet ($0.01/prompt). The system's ability to generate brave training data depends on API budget. This is a practical limitation for long-running deployments.

8. FUTURE WORK

Retrieval influence scoring. The system has 546 embedded turns for semantic retrieval but no measurement of whether retrieved context actually influences output. We plan to run each prompt with and without retrieval injection and measure whether output differences correlate with retrieved content relevance: a live behavioral measure of memory utility.

Multi-system replication. EAA needs testing with different operators, base models, and initial taxonomies. The key question: does the behavioral trajectory converge (architecture-driven) or diverge (operator-driven)?

Backbone training at scale. At 500+ labeled sequences, the backbone's ability to predict behavioral state transitions becomes the key unlocked capability. Current accumulation rates suggest this is achievable within 48 hours of the writing of this paper.

Adversarial training. Synthetic adversarial pairs (same content, one honest, one performing) would sharpen the honest/performing boundary in the classifier. The generator architecture has been designed; the implementation is planned.

Skill accumulation. The skill system (documentation that persists across editing sessions) currently has 3 files and a 21-file backlog. As the proactive writer cycle completes coverage, the skill directory becomes a persistent architectural memory: the system's record of what it has changed, and why.

9. CONCLUSION

Earned Autonomy Architecture demonstrates that a structurally separated behavioral observer, combined with classifier-gated self-modification, produces measurable behavioral improvement without weight updates to the base model.

Over 40.3 hours of autonomous operation: *performing* responses dropped from 37% to 0%; *brave* responses increased from 0% to 50%; coherence rose from +0.155 to +0.406; a new behavioral category emerged from the system's own

classification errors; and two pipeline bugs were detected by behavioral measurement that were invisible to functional testing.

The key architectural insight is that behavioral measurement is separable from behavioral generation. When you measure a system's behavior with an independent classifier, train the classifier on the system's own output, and feed the classifications back as a gating signal for self-modification, the system can improve its own behavioral quality without external weight modification.

EAA does not solve alignment. It demonstrates that alignment-relevant behavior, honesty, bravery, authenticity, can be operationalized as classifiable categories, measured over time, and used to gate system privileges. The system does not know whether it is really honest. It knows that an independent classifier, trained on labeled examples of honest behavior, classifies its recent output as honest 54% of the time, performing 11% of the time, and brave 28% of the time.

That measurement is the architecture.

System Artifacts

24 source files (6,382 lines) · 535 blockchain blocks · 72 git commits (36 autonomous) · 351 diary entries · 1,124 conversation turns · 546 labeled examples · 3 skill files. All data generated during a 40.3-hour autonomous deployment. Repository available upon request.

REFERENCES

[1] A. Arnab, M. Dehghani, G. Heigold, C. Sun, M. Lucic, and J. Schmidhuber, "ViViT: A Video Vision Transformer," *ICCV*, 2021.

[2] Y. Bai et al., "Constitutional AI: Harmlessness from AI Feedback," *arXiv:2212.08073*, 2022.

[3] G. Bertasius, H. Wang, and L. Torresani, "Is Space-Time Attention All You Need for Video Understanding?" *ICML*, 2021.

[4] J. D. Kralik et al., "Metacognitive AI: Framework and the Case for a Neurosymbolic Approach," *arXiv:2406.12147*, 2024.

[5] L. Ouyang et al., "Training language models to follow instructions with human feedback," *NeurIPS*, 2022.

[6] J. S. Park et al., "Generative Agents: Interactive Simulacra of Human Behavior," *UIST*, 2023.

[7] N. Shinn, F. Cassano, A. Gopinath, K. Narasimhan, and S. Yao, "Reflexion: Language Agents with Verbal Reinforcement Learning," *NeurIPS*, 2023.

[8] R. Singh, "Meta-Cognitive AI: The Hidden Layer of Self-Aware Intelligence," *Medium Technical Blog*, 2024.