Deterministic Replay Architectures for AI Systems

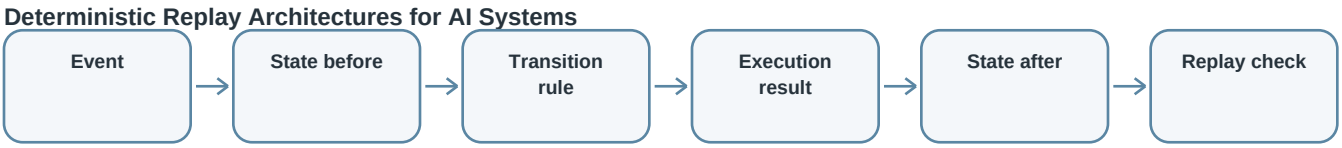
Reconstructable runtime transitions, drift detection, event lineage, and reviewable execution for AI workflows.

Artifact type	Technical Protocol Note
Status	Research artifact
Primary route	/research/deterministic-replay/
Domains	deterministic replay, AI observability, runtime governance, execution traceability
Keywords	deterministic replay, AI observability, runtime lineage, replayable AI systems, execution traceability, drift detection, event lineage, AI runtime

Abstract

Replay is the difference between an opaque assistant and an accountable runtime. Deterministic replay architectures seek to make important transitions reconstructable even when probabilistic models participate in planning or interpretation.

Primary architecture reading



Design reading: each transition should be bounded, observable, and reversible where practical.

Must-have requirements

- Identify event inputs
- Capture before/after state
- Bind transition rules
- Separate proposal from commit
- Surface replay failure

Good-to-provide enrichments

- Replay logs
- Hash commitments
- Diff reports
- Risk-based replay tiers

Why replay matters

When AI systems act, organizations need more than a transcript. They need to know what state existed, what transition was authorized, what tools were called, what changed, and whether the transition can be reconstructed. Replayability supports debugging, trust, and governance.

Probabilistic planning, deterministic transitions

A model may propose or interpret, but consequential state transitions should be represented in deterministic forms where possible. This allows validators, policy checks, and replay systems to examine the transition independent of the model’s conversational surface.

Drift detection

Replay can reveal drift when the same source material, rules, and state no longer produce equivalent results. Drift may come from model updates, prompt changes, data changes, hidden state mutation, or infrastructure changes. Detecting drift is a governance feature, not an academic luxury.

Operational boundaries

Not every interaction deserves full replay overhead. The system should classify transitions by consequence. Low-risk drafting may need lightweight lineage; irreversible actions, external submissions, payments, policy decisions, or institutional claims require stronger replay discipline.

Implementation notes for blue-hand.org

This artifact should be hosted from **/research/deterministic-replay/** with an HTML summary page, PDF download link, schema.org TechArticle JSON-LD, OpenGraph metadata, and links back to the Research Library, Systems Atlas, N2 Protocol, and relevant Bluehand systems.

Suggested HTML sections

- Why replay matters

- Probabilistic planning, deterministic transitions

- Drift detection

- Operational boundaries

SEO and discovery surface

The artifact should use its title as the page H1, subtitle as the meta description basis, and domains/keywords as tags. The copy should remain human-readable; keyword density should arise from precise technical terminology rather than stuffing.

deterministic replay	AI observability	runtime lineage	replayable AI systems
execution traceability	drift detection	event lineage	AI runtime

Governance boundary

This artifact is a public research object, not a claim that every described capability is already deployed in production. Claims about implementation should remain explicitly separated from architectural direction, organizational doctrine, and future-facing design work.

Canonical relationship to Bluehand

This brief supports Bluehand as a research and infrastructure organization working across semantic memory, governed execution, local-first AI, institutional trust, and research venture formation. It should be treated as one node in a larger public knowledge graph, not as standalone marketing collateral.