Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

From Human Approval to Current Authority: Adaptive Runtime Governance at the Execution Boundary of AI-Enabled Organizational Workflows

Maria Kollia

karnw79@gmail.com

Research Article

Keywords: AI governance, human authorization, meaningful human oversight, accountability, runtime governance, execution control

Posted Date: August 13th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-10668771/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: Competing interest reported. The author is the creator and developer of the ARG research architecture and the prototype evaluated in this study. No licensing income or other financial benefit has been received.

Abstract

Human oversight in AI-enabled workflows can become ethically nominal when an approval remains recorded but the authority supporting it has changed by the time an action is executed. This study develops Adaptive Runtime Governance (ARG) as a broader governance architecture and evaluates one implemented component: a deterministic execution gate that revalidates current human authorization and organizational governance evidence immediately before execution. The evaluated prototype, implemented in Python and FastAPI with SQLAlchemy persistence, checks decision state, persisted approvals, approver status and role continuity, approval freshness, policy-version continuity, and delegation conditions. Evidence consists of an anonymized retrospective snapshot containing 101 decisions and a controlled validation in an isolated SQLite database comprising 223 cases. All controlled cases matched their predefined outcomes: 30 positive execution cases returned HTTP 200, 192 negative or robustness cases returned the expected refusal status, and a final verification reported a valid linked chain across 492 audit events. The broader ARG architecture is presented as a proposed research framework rather than as an empirically validated system. The findings support a bounded conclusion: historical human approval and current execution authority are not equivalent, and selected conditions of attributable human authority can be made technically enforceable at the execution boundary. The study does not establish model safety, legal compliance, production security, organizational effectiveness, or adaptive behavior.

1 Introduction

Artificial intelligence is increasingly embedded in organizational workflows in which model outputs or automated recommendations can become financial, administrative, technical, or public-sector actions. The governance problem is therefore not confined to model development, training, or inference. It also arises at the transition between a proposed action and the moment at which that proposal acquires an external effect. Once execution is permitted, the relevant question is no longer only what an AI-enabled system can generate, predict, classify, recommend, or rank. It is also whether the organization still has valid grounds to permit the proposed action to occur.

This distinction is important because authorization is temporally contingent. An approval may be legitimate when it is issued and no longer legitimate when execution occurs. An approver may have been disabled, may no longer hold the required organizational role, or may have lost the relevant authority. A delegation may have expired. The applicable policy version may have changed. Approval evidence may become stale or internally inconsistent. The decision itself may have entered a state in which execution is no longer permissible. In each case, historical approval exists, but current authority cannot be inferred from that historical fact alone.

Organizational AI governance is commonly described through governance structures, risk-management practices, accountability mechanisms, assurance processes, and technical controls [1–6, 15]. Meaningful human control scholarship similarly links responsibility to traceability, responsiveness to

human reasons, and actionable design properties [7–9]. Recent work in AI and Ethics has reinforced the need to move beyond symbolic human involvement toward oversight architectures that preserve genuine human evaluative agency and accountability [17–20]. These approaches explain why oversight matters, but they do not define a universal implementation for deciding whether a particular organizational action remains admissible immediately before execution.

Existing technical mechanisms provide important components. Role-based access control (RBAC) assigns permissions through organizational roles [10]. Attribute-based access control (ABAC) evaluates subject, object, action, and environmental attributes [11]. Zero trust architecture rejects implicit trust and requires explicit verification before access [12]. Ongoing authorization recognizes that earlier authorization decisions may need to be revisited as conditions change [13]. Runtime assurance systems evaluate specified properties during operation and intervene when those properties are threatened [14]. The contribution of the present work is not to replace these mechanisms but to compose selected governance checks around one decision-specific transition: execution.

The broader research program is described here as Adaptive Runtime Governance (ARG). ARG is an architectural research framework for maintaining governance conditions across the lifecycle of an AI-enabled organizational action. Only one part of that broader framework is empirically evaluated in this study: a deterministic, rule-based execution gate. Adaptive risk classification, dynamic validation-profile selection, emergency governance, automatic policy discovery, and continuous monitoring are not treated as demonstrated capabilities. They are presented only as proposed architectural extensions requiring separate implementation and evaluation.

The central research question is therefore deliberately bounded: can selected conditions of human authorization and organizational governance be represented as decision-specific evidence, revalidated at execution time, and enforced through backend refusal when those conditions no longer hold? The paper contributes (1) a conceptual distinction between historical approval and current execution authority; (2) a broader ARG architecture that locates execution control within an AI-governance lifecycle; (3) an implemented execution-gate pattern that makes selected governance conditions technically consequential; and (4) controlled evidence covering both successful and refusal paths.

2 Ethical and governance foundations

The ethical problem addressed by ARG is not whether a human appears somewhere in an AI-enabled workflow. Human presence alone is an inadequate proxy for meaningful oversight. A person may click an approval button without adequate authority, may approve under a policy that later changes, may cease to occupy the role under which the approval was valid, or may remain nominally associated with an action while the organizational conditions supporting responsibility have changed. The relevant governance question is whether responsibility remains attributable and operationally connected to the action when execution occurs.

This paper therefore advances a bounded normative claim: continuity of valid, attributable authorization is a necessary but not sufficient condition for accountable AI-enabled organizational action. It is necessary because execution without current authority creates a gap between the organizational record and the actual basis for permission. It is not sufficient because an authorized person can still approve an inaccurate, unfair, unsafe, unlawful, or ethically unjustified action. Those dimensions require separate model assurance, application controls, domain expertise, legal analysis, and substantive ethical judgment.

Meaningful human control emphasizes that responsibility should remain connected to identifiable humans and to reasons that can guide or constrain system behavior [8, 9]. Recent work on meaningful human oversight similarly distinguishes genuine evaluative agency from human participation that functions as a rubber stamp [17]. The execution-gate approach developed here addresses a narrower part of that problem. It does not attempt to decide whether the human judgment is normatively correct. Instead, it asks whether the organization can still demonstrate a valid line of authority for allowing the action to occur.

This distinction also matters for operational responsibility. Recent AI-governance scholarship emphasizes that accountability attaches across socio-technical arrangements rather than residing entirely inside an AI model [18–20]. An organization that deploys an AI-enabled workflow remains responsible for the institutional conditions under which generated proposals acquire real-world consequences. From this perspective, execution is a governance event, not merely a technical state change.

A refusal mechanism has a practical governance role. A requirement has limited value if a violation is recorded only after the action has already occurred. When an encoded condition is mandatory, failure of that condition must be able to change the outcome. In the evaluated gate, execution is therefore refused when current authority, decision state, policy continuity, approval evidence, or delegation validity cannot be demonstrated under the encoded rules.

3 Adaptive Runtime Governance as a broader architecture

ARG is designed as a governance layer between AI-enabled proposal generation and externally consequential execution. The aim is to preserve an explicit governance boundary whether the upstream proposal comes from an AI model, an automated service, a human decision-support workflow, or a combination of these. The governance layer does not attempt to reproduce the reasoning of the upstream system. It asks a narrower question: what evidence and authority must still be valid before the proposal is allowed to cross the execution boundary?

The architecture is built around a persistent decision object. Rather than treating approval as a transient interface event, the proposed action is represented as an identifiable governance object with state, provenance, risk input, approval requirements, authority records, policy version, delegation context, and

audit history. This representation allows governance conditions to be evaluated more than once and at different moments in the lifecycle.

For clarity, this paper separates three different levels of evidence. The first is demonstrated capability: behavior exercised in the controlled validation reported here. The second is implemented but unevaluated, or only partly evidenced, capability that may exist in the wider codebase but is not supported by the controlled evidence used in this study. The third is proposed architecture: future modules and design directions that still require implementation and empirical evaluation. This distinction is necessary because architectural plans should not be presented as experimental results.

The principal layers of the broader ARG architecture and the evidentiary status assigned to each in this study are summarized in Table 1.

Table 1
ARG layers and evidentiary status in the present study

Layer	Governance function	Status in this paper
Proposal interface	Receives an AI-enabled, automated, or human-supported proposed action and contextual fields.	Boundary specified; model quality not evaluated.
Decision persistence	Creates a stable decision object with state, actor, risk input, policy and approval metadata.	Implemented and evaluated as part of the gate.
Authorization	Records named human approvals, roles, timestamps, rationale, thresholds and diversity constraints.	Implemented and evaluated.
Execution revalidation	Rechecks state, approval records, authority continuity, role continuity, freshness, policy continuity and delegation immediately before execution.	Implemented and evaluated.
Refusal / safe outcome	Prevents transition to executed status when an encoded condition fails.	Implemented and evaluated.
Audit evidence	Records decisions, approvals, blocks, executions and linked event hashes.	Implemented and evaluated within one application database.
Adaptive criticality	Would dynamically classify criticality and select validation profiles.	Proposed; not demonstrated.
Emergency governance	Would define exceptional authorization and recovery behavior under declared emergencies.	Proposed; not demonstrated.
Continuous governance monitoring	Would detect material changes after approval and trigger revalidation or suspension.	Proposed; not demonstrated.

3.1 Governance lifecycle and execution boundary

A simplified ARG lifecycle is: proposal -> decision creation -> risk input -> approval acquisition -> pre-execution revalidation -> execute or refuse -> audit. The sequence does not imply that every system must use the same approval threshold or role names. Those parameters are organizational policy choices. The architectural invariant is that execution is treated as a distinct transition at which current governance evidence can be evaluated again.

The governed object may be a recommendation to release a payment, change a user entitlement, approve an administrative request, apply a classification, modify an infrastructure setting, or trigger another external process. The evaluated prototype begins after a proposed action has been produced and ends before the downstream effect is invoked. It does not inspect model weights, training data, prompt quality, predictive accuracy, fairness metrics, hallucination risk, or semantic correctness. Those questions belong to model and application assurance. The gate asks a different question: does the organization currently possess sufficient encoded authority and evidence to let this particular action proceed?

This separation is particularly important for agentic AI. An AI system may retain operative capacity to generate or plan actions while the organization retains a non-delegable governance boundary around execution. Recent work on agentic AI stresses that meaningful human oversight should preserve human evaluative agency rather than reduce it to a symbolic checkpoint [17, 20]. ARG operationalizes one narrow form of that evaluative agency by making current authorization a precondition of organizational effect.

3.2 Governance invariants

The evaluated gate expresses a set of governance invariants rather than a general-purpose ethics engine. First, approval and execution are separate events. Second, self-approval is prohibited for the evaluated high-risk path. Third, duplicate actor approval is rejected, and duplicate-role approval is rejected when role diversity is required. Fourth, an approval does not remain indefinitely valid: every approval used at execution is subject to a freshness rule. Fifth, an approver's current role and active status must remain compatible with the approval. Sixth, the current policy version must match the version under which the decision was approved. Seventh, delegated execution requires an active and valid delegation from an eligible grantor. Eighth, invalid state transitions and repeat execution are blocked.

These invariants are deliberately limited. They do not decide whether a policy is ethically adequate, whether a role allocation is legitimate, or whether the supplied risk category is correct. They represent institutional conditions that an organization may choose to make mandatory. ARG makes those conditions executable and inspectable; it does not claim that they are a complete account of ethical governance.

The governance invariants exercised by the evaluated gate and their accountability rationale are summarized in Table 2.

Table 2
Governance invariants exercised by the evaluated gate

Invariant	Why it matters for accountability	Evaluated response
No self-approval	Separates proposal/creation from authorization for the tested high-risk path.	Self-approval request refused.
Actor uniqueness	Prevents a single actor from satisfying a multi-approval threshold more than once.	Duplicate actor refused.
Role diversity	Prevents nominal multiplicity when configured governance requires distinct role perspectives.	Duplicate approval role refused where required.
Approval freshness	Treats authorization as time-bounded rather than permanent.	Approvals older than 24 hours refused.
Authority continuity	Links historical approval to the approver's current active status and role.	Inactive, deleted or role-changed approvers cause refusal.
Policy continuity	Prevents execution after the governing policy version has changed.	Policy-version mismatch causes refusal.
Delegation validity	Requires delegated execution to remain within a valid authority relationship.	Missing, expired or invalid delegation causes refusal.
State admissibility	Prevents execution from pending, rejected, cancelled or already executed states.	Invalid state transition refused.

4 Relationship to existing governance and control mechanisms

ARG is best understood as a composition of established control ideas at a specific decision boundary rather than as a claim to invent authorization, workflow, audit, or runtime monitoring. RBAC provides role-based permission foundations [10]. ABAC provides a conceptual basis for evaluating decision, actor, action, and environmental attributes [11]. Zero trust and ongoing authorization motivate explicit re-verification when relevant conditions change [12, 13]. Runtime assurance contributes the pattern of checking properties during operation and moving to a safe outcome when those properties fail [14]. Human-in-the-loop governance contributes the requirement that consequential actions remain connected to identifiable human judgment and authority [8, 9, 17].

The contribution is operational rather than foundational. The evaluated pattern binds a persistent organizational decision to approval evidence, authority, policy, delegation, and state, and checks those conditions again when execution is requested. It does not propose a new general theory of access control. It proposes and tests a governance-by-design pattern for deciding whether a specific AI-enabled organizational action remains admissible at execution time.

This placement also distinguishes ARG from ordinary workflow management. Both systems can represent states and transitions. A generic workflow engine, however, may advance a task because a prior state transition was completed. The governance-specific requirement is that the transition to

execution remains conditional on current evidence of authority. Similarly, audit logging alone provides retrospective traceability but does not prevent a disallowed action. ARG treats refusal as part of the governance mechanism rather than as an optional post-hoc response.

The relationship between the evaluated gate and established authorization, workflow, audit, and runtime mechanisms is summarized in Table 3.

Table 3
Relationship between the evaluated gate and established mechanisms

Mechanism	Primary function	Use in evaluated gate	Boundary
RBAC	Permissions associated with organizational roles	Restricts roles that may approve or grant delegation	Role membership alone does not establish sufficient current evidence for a specific decision
ABAC	Authorization using subject, object, action and context attributes	Conceptual basis for evaluating actor, decision, risk, state and context	Fixed application rules; no general policy language or external attribute authority
Human-in-the-loop approval	Human review or confirmation	Records named approval, rationale, role and timestamp	Human presence alone does not establish freshness, diversity or authority continuity
Workflow management	State transitions, routing and task sequencing	Represents pending, approved, executed, rejected and cancelled states	Generic workflows need not revalidate governance evidence
Audit logging	Retrospective trace of events	Records decision, approval, refusal and execution events	Logging does not itself prevent invalid execution
Zero trust / ongoing authorization	Explicit and repeated verification	Motivates rechecking authority, policy, delegation and state	Prototype is application-level, not enterprise identity or network zero trust
Runtime assurance	Runtime checking and safe intervention	Shares check-and-refuse pattern	Evaluated properties concern organizational authorization rather than continuous physical-system safety

5 Prototype implementation

The evaluated prototype is implemented as a modular FastAPI application with SQLAlchemy persistence. The controlled evaluation used SQLite to isolate the validation run. The application contains decision, approval, execution, delegation, user, policy, audit, and integration routes. Enforcement occurs in backend endpoints rather than only in the user interface.

Each decision stores a stable identifier, action, justification, normalized risk level, status, creation actor, configured approval threshold, current approval count, named authorizer, authorization timestamp, authorization note, policy version, execution timestamp, and execution result. Separate approval records preserve approving actors, roles, timestamps, and notes. Execution snapshots preserve selected revalidation outcomes. Audit events preserve lifecycle events together with a linked hash field.

The state model includes pending, approved, executed, rejected, and cancelled. High-risk proposals are created as pending and require at least one approval. Low-risk proposals follow a deterministic zero-approval demonstration path and may be created as approved. This is not autonomous risk assessment: risk is supplied to the governance layer and normalized to a permitted value.

Approval is limited to configured organizational roles. The implementation blocks self-approval, duplicate approval by the same actor, and duplicate-role approval when role diversity is required. High-risk approval requires a non-empty justification. A decision becomes approved only when the configured threshold is supported by persisted approval state.

Execution triggers a second validation stage. The endpoint evaluates decision state, named authority for high-risk actions, persisted approval-record count, freshness of every approval record, existence and current status of approving users, continuity of approval role, policy-version continuity, and delegation conditions when an operator executes on behalf of another actor. A failed condition produces a refusal response and an execution-blocked record. A successful path creates an execution result, timestamp, audit event, and successful revalidation snapshot.

5.1 Corrections made before controlled evaluation

Retrospective inspection and preliminary tests identified defects that would have made broader claims inappropriate. The controlled evaluation was therefore run on a revised copy of the sanitized source while the original package was retained unchanged. The study does not treat corrected code as evidence that the historical implementation had always behaved correctly.

The revised copy normalizes risk input and rejects unsupported categories; rejects negative approval thresholds; enforces a minimum approval requirement for high-risk decisions; implements a coherent zero-approval low-risk path; verifies that persisted approval records meet the configured threshold; checks the timestamp of every approval; restricts delegation creation to administrator or manager roles; blocks self-delegation; normalizes execution roles for delegation checks; and calculates audit-event hashes from the canonical fields actually persisted and later verified.

An earlier description referred to sequential approval stages. The supplied source snapshot did not contain an approval-stage table or a validated stage sequence. That claim is therefore excluded. The evaluated multi-approval property is approval counting with actor uniqueness and role diversity, not an empirically demonstrated ordered-stage workflow.

6 Threat model and security assumptions

The evaluation assumes that requests reach the governance endpoint and that the application database is available. It tests policy violations encoded in the application, not every insider, infrastructure, identity, concurrency, or distributed-systems threat. This boundary is important because an execution gate can only govern effects that are actually routed through it.

The strongest architectural dependency is therefore exclusive routing. If a downstream service can invoke the external effect without passing through the governance endpoint, the gate can be bypassed. Production deployment would need to make the governance boundary structurally mandatory, for example through service architecture, credentials, network policy, or downstream verification. That deployment property is not tested here.

The linked audit chain is also limited. It can detect changes to the canonical event sequence when verification is performed against the stored hashes, but the database is not an external append-only ledger, trusted timestamp service, digital-signature scheme, or hardware-backed evidence store. A privileged database actor remains a residual risk.

The principal threats exercised by the evaluation, the implemented responses, and the residual risks are summarized in Table 4.

Table 4
Threat model, implemented responses, and residual risks

Threat or failure	Scenario	Implemented response	Residual risk
Unauthorized approval	Disallowed role, self-approval, repeated actor or duplicate role	Role restriction, no-self-approval, actor uniqueness and role-diversity checks	Compromised accounts and collusion not detected
Stale authority	Approver disabled, deleted, role-changed, loses authority or approval > 24h	Execution-time user, role, authority and freshness checks	No external identity-provider integration
Policy drift	Approved policy version differs from current version	Execution refused and change recorded	Semantic equivalence between policy versions not evaluated
Invalid delegation	Operator acts without valid delegation or valid grantor	Delegation existence, validity interval and grantor-authority checks	Delegation scope coarse and not cryptographically signed
Invalid state / replay	Pending, rejected, cancelled or already executed decision sent to execution	State-based refusal and repeat-execution block	Distributed replay across services not tested
Approval inconsistency	Decision counter suggests threshold met but approval records do not	Persisted record-count check at execution	Concurrent races not stress-tested
Audit alteration	Events changed or reordered	Linked hashes detect canonical sequence changes	Privileged modification remains possible
Gate bypass	Downstream service invokes effect directly	Outside tested application boundary	Deployment must enforce exclusive routing
Harmful AI proposal	Proposal inaccurate, biased, manipulated or unsafe	Outside evaluated objective	Requires separate model, data, application and domain assurance
Credential compromise	Valid token stolen or incorrectly issued	Only partly represented by overridden identity dependency	Production identity, revocation and key management require independent testing

7 Evaluation design

Two evidence sets were used for different purposes. The retrospective set was an anonymized database supplied with the technical review package. It contained 101 decisions, 101 approval records, 272 audit events, 27 execution snapshots, nine users, five delegations, and one integration-system record. The

snapshot was inspected for status distributions, event counts, snapshot outcomes, missing fields, threshold inconsistencies, and audit-chain reproducibility.

The controlled evidence set was created from the revised source in a new isolated SQLite database. Tests used Python 3.13.5, FastAPI TestClient, and fixed synthetic users representing administrator, manager, compliance officer, operator, user, and auditor roles. The authentication dependency was overridden to supply the designated synthetic actor for each request. This isolates application authorization and execution logic after identity resolution; it does not test passwords, token issuance, browser behavior, network security, or an external identity provider.

Thirty-seven scenario types were repeated six times, producing 222 API cases. A final audit-chain verification produced the 223rd case. Each test defined the expected HTTP status and, where relevant, a required response detail. A case passed only when the observed status and required detail matched the predefined expectation. Request latency was recorded around the in-process TestClient call using a monotonic high-resolution timer.

The scenario set covered positive execution, negative approval, negative execution, malformed input, risk normalization, delegation controls, robustness, and audit integrity. Repetition reduces the chance that one accidental record or ordering effect explains an outcome, but the exercise is not statistical sampling from a production population and should not be interpreted as such.

7.1 Translation assistance

OpenAI ChatGPT was used only for translation into English.

8 Results

After normalization of risk-label capitalization, the 101 retrospective decision records comprised 27 approved, 45 executed, 28 pending, and one rejected decision. The audit table contained 15 execution-blocked events. Among 27 execution snapshots, 14 recorded successful revalidation and 13 recorded blocked revalidation. Recorded block reasons included non-approved state, repeat execution, inactive approver, changed approver role, policy-version change, missing approval records, and missing or invalid delegation.

The snapshot also exposed legacy inconsistencies. Two executed records lacked named authority, authorization time, authorization note, and approval records. One approved record was below its configured threshold, one record exceeded its configured approval count, and event hashes could not be reproduced from the fields retained in the audit table. These are reported as defects rather than governance successes. They were used to define corrections and controlled scenarios.

Because execution snapshots were introduced after many historical executions, the absence of a successful snapshot for an earlier record cannot establish whether the current revalidation sequence

was performed during that earlier call. The retrospective evidence therefore documents implementation history, observed controls, and defects. It is not the primary evidence for current execution behavior.

All 223 controlled cases matched the predefined result. The 30 positive execution cases included single-approval high-risk execution, two-approval role-diverse execution, operator execution under a valid delegation, and deterministic low-risk execution without a human approval record. Every positive case returned HTTP 200.

The 42 negative approval cases blocked unauthorized approval, self-approval, missing high-risk justification, duplicate actors, duplicate approval roles, approval after approved state, and approval after rejection. The 108 negative execution cases blocked pending, rejected, cancelled, and already executed decisions; missing named authority; approvals older than 24 hours; inactive, deleted, or role-changed approvers; policy-version change; absent or invalid delegation; a stale earlier approval in a multi-approval decision; and persisted approval counts below the configured threshold.

Input and robustness cases rejected malformed identifiers, unknown decisions, unsupported risk categories, and negative thresholds, and confirmed normalization of uppercase HIGH to high. Delegation creation by an unauthorized role and self-delegation were blocked. The final verification endpoint reported a valid linked chain across 492 hashed audit events with no disconnected event.

Across the 223 cases, mean latency was 7.331 ms, median latency was 8.071 ms, the 95th percentile was 12.393 ms, the minimum was 1.732 ms, and the maximum was 23.312 ms. These are descriptive timings from local in-process requests. They are not network benchmarks, load tests, throughput measurements, or evidence of production scalability.

Controlled validation outcomes by category are summarized in Table 5.

Table 5
Controlled validation outcomes by category

Category	Cases	Passed	Observed HTTP statuses	Mean ms	Median ms	P95 ms
Positive	30	30	200: 30	10.388	9.998	12.646
Negative Approval	42	42	400: 12, 403: 30	2.967	2.946	3.799
Negative Execution	108	108	403: 108	9.446	8.764	13.068
Input Validation	24	24	400: 18, 404: 6	2.682	2.615	3.680
Robustness	6	6	403: 6	11.438	10.024	16.479
Negative Delegation	12	12	403: 12	2.751	2.724	3.840
Audit Integrity	1	1	200: 1	12.393	12.393	12.393

9 Discussion

The controlled evaluation supports a narrow but concrete claim: in the tested deterministic implementation, selected governance conditions were enforced through backend API responses before a decision entered executed status. This is stronger evidence than design description or source inspection alone because refusal paths were invoked and compared with predefined outcomes. It remains prototype evidence and should not be generalized to production deployment.

The ethical relevance of the results lies in the distinction between human presence and current human authority. A workflow can contain a human approval and still fail to provide meaningful oversight if that approval is stale, unsupported by the actor's present role, inconsistent with current policy, or disconnected from a valid delegation. The gate makes these conditions executable rather than leaving them solely as retrospective audit questions. This complements recent work arguing that meaningful human oversight requires design mechanisms through which human evaluative agency can actually steer or stop system outcomes [17].

The results also help locate operational responsibility. AI governance is not exhausted by the quality of the upstream model output. Organizations still decide which outputs are allowed to acquire institutional effect, under whose authority, and on the basis of what evidence. Work on AI in operations similarly emphasizes accountability around deployment and operational decision-making [18], while research on operational responsibility locates responsibility in the human and institutional actors able to control context-specific deployment [19]. The execution boundary is one concrete point at which those responsibilities can be represented technically.

The architecture should not be interpreted as transferring moral agency to software. The gate does not determine whether an action is ethically right. It verifies selected institutional facts encoded by policy. Its normative role is secondary but important: it makes the absence of those facts operationally consequential. A refusal is meaningful precisely because the software cannot silently reinterpret, extend, or inherit human authority when the required conditions are no longer present.

This addresses a practical weakness of checklist-oriented governance. A policy may require human review, named accountability, valid delegation, or current approval, but those requirements remain documentary if they cannot affect whether the system acts. The prototype shows one way to connect selected governance requirements to the action boundary. It does not show that these requirements are sufficient for every domain; it shows that at least some governance rules can be made operational before execution.

The relationship to existing mechanisms is also important for interpreting novelty. RBAC and ABAC provide authorization foundations. Human-in-the-loop processes provide review points. Audit systems preserve evidence. Zero trust and ongoing authorization motivate repeated verification. Runtime assurance contributes the pattern of evaluating conditions during operation and shifting to a safe outcome when they fail. ARG composes selected versions of these mechanisms around a persistent

decision and evaluates them again at execution. The contribution is an implementation pattern for organizational action admissibility, not a new general theory of access control.

The role of AI is deliberately bounded. The model or automation component may act as a proposer, but the gate does not decide whether the proposal is factually correct, unbiased, safe, aligned, explainable, or lawful. A production system would need to connect model-level assurance, application risk controls, domain validation, contestability, and execution governance. Recent scholarship on agentic AI similarly shows that governance must address the socio-technical allocation of agency and responsibility rather than treating autonomy as a purely model-internal property [20].

The retrospective defects are not incidental. They illustrate why intended design and code branches are insufficient evidence for a governance system. Historical records exposed missing authority evidence, inconsistent counters, and audit-verification logic that did not reproduce from retained fields. Reporting these defects narrows the claims, but it also demonstrates why governance systems themselves require verification. A system that exists to create accountability cannot simply assume the integrity of its own evidence.

Independent validation remains absent. The system designer prepared the evaluation package and the controlled test harness was executed within the same study. Detailed scenario definitions and machine-readable outcomes improve inspectability, but they do not eliminate confirmation bias. Replication by an external software-security or governance research group would materially strengthen the evidence.

10 What is and is not demonstrated about ARG

The broader ARG architecture is larger than the execution gate evaluated here. For that reason, the paper does not treat architectural specification as implementation evidence. The distinction below is important for interpreting the claims.

Demonstrated in the controlled evaluation are decision-state enforcement; high-risk approval requirements; named human authorization; self-approval blocking; actor uniqueness; role diversity where configured; approval freshness; approver existence, activity, and role continuity; policy-version continuity; approval-record consistency; selected delegation validity; repeat-execution blocking; refusal recording; and linked audit-chain verification within the tested application.

Not demonstrated by this study are adaptive learning, automatic risk or criticality classification, dynamic policy selection, emergency-mode governance, continuous external monitoring, production identity integration, distributed transaction safety, concurrency behavior, infrastructure failover, external append-only audit storage, penetration resistance, real-world organizational effectiveness, or legal compliance.

Proposed ARG extensions include dynamic criticality assessment, policy-specific validation-profile selection, emergency governance, offline or communications-degraded policy behavior, mandatory evidence binding, and continuous revalidation triggers. These capabilities are part of the research

direction because they address conditions under which authority and risk can change over time. They require independent implementation and evaluation before they can support empirical claims.

The evidentiary status of selected ARG capabilities is summarized in Table 6.

Table 6
Evidentiary status of selected ARG capabilities

Capability	Category	Evidence basis
Execution-time state and authority revalidation	Demonstrated	Controlled API cases
No-self-approval, actor uniqueness, role diversity	Demonstrated	Controlled approval cases
Approval freshness and role continuity	Demonstrated	Controlled negative execution cases
Policy-version continuity	Demonstrated	Controlled negative execution cases
Delegation validity checks	Demonstrated	Controlled delegation and execution cases
Linked audit-chain verification	Demonstrated within application boundary	Final audit verification across 492 events
Production identity / SSO / token revocation	Not demonstrated	Authentication dependency overridden
Concurrency, load, failover, distributed replay	Not demonstrated	Outside local TestClient evaluation
Adaptive criticality and dynamic validation selection	Proposed ARG extension	No empirical evidence in this study
Emergency governance and continuous monitoring	Proposed ARG extension	No empirical evidence in this study
Legal compliance and organizational effectiveness	Not established	Requires domain and institutional evaluation

11 Limitations

The test environment used FastAPI TestClient and a local SQLite database. It did not evaluate network latency, PostgreSQL behavior, distributed transactions, concurrent requests, high load, failover, container orchestration, or deployment security. Authentication was dependency-overridden with synthetic actors after identity resolution, so password storage, token issuance and revocation, compromised credentials, single sign-on, identity-provider synchronization, and session security were outside the test boundary.

The test oracle was derived from the intended policy and implemented response contracts. There was no independent test author, blinded evaluation, external penetration test, or institutional user study. The

results demonstrate encoded rules, not legal validity or governance effectiveness within a real organization. Role names, thresholds, expiry periods, policy versions, and delegation rules require deployment-specific authority and legal analysis.

The prototype does not evaluate AI model outputs, training data, explainability, accuracy, fairness, robustness, prompt injection, adversarial examples, or model autonomy. The linked audit chain is maintained over canonical persisted event fields in one application database; it is not an immutable ledger, trusted timestamp service, digital-signature scheme, hardware-backed log, or external evidence repository.

The low-risk path is a deterministic demonstration rule based on supplied risk input. Incorrect risk classification can therefore create inappropriate execution behavior. Production use would require a trusted risk source and policy-specific validation. The evaluated source does not empirically demonstrate ordered approval stages; multi-approval evidence is limited to thresholds, actor uniqueness, and role diversity. The latency measurements are descriptive local timings and should not be generalized to production performance or scalability.

Finally, the present paper tests only a subset of the wider ARG architecture. The term adaptive refers to the research direction of maintaining governance as context changes, not to machine learning or autonomous policy adaptation demonstrated here. To avoid overstating the evidence, all untested adaptive modules are treated as research propositions rather than system results.

12 Research agenda toward a full ARG system

The next stage should broaden the evidence before broadening the claims. The first priority is independent replication of the execution gate with a separately authored test suite. The second is production identity integration, so that role changes, account suspension, delegation, and revocation come from an external authority rather than synthetic application state. The third is concurrency and distributed-systems testing, because governance guarantees may fail when multiple services attempt to approve or execute the same decision at the same time.

Evidence binding should also be strengthened. The current prototype binds a decision to approval and policy metadata, but a production governance system should ensure that the action being executed is exactly the action that was approved. This suggests cryptographic or content-addressed binding among proposal version, relevant evidence, policy version, approval set, and downstream execution request. External append-only audit storage or independently verifiable signatures would further reduce reliance on one privileged application database.

Adaptive criticality is another research direction, but it should not be implemented as unconstrained model self-classification. A defensible design would require a trusted risk source, explicit uncertainty handling, and conservative escalation when criticality cannot be established. Dynamic validation profiles

could then select stronger approval, evidence, or delay requirements for higher-risk actions. Such a module would need separate benchmarks and failure-oriented testing.

Emergency governance creates a different problem. Organizations sometimes need exceptional authority under time pressure, but an emergency path can become a permanent bypass if it is not tightly scoped. Future ARG work should therefore treat emergency mode as a separate governed state with explicit activation authority, time limits, enhanced logging, retrospective review, and automatic expiry. Similar principles apply to communications-degraded or offline operation, where the system must distinguish previously delegated authority from unauthorized authority inheritance.

Finally, institutional evaluation is essential. Technical refusal behavior does not demonstrate that the governance rules are appropriate, usable, proportionate, or legally sufficient in a real organization. Future studies should evaluate how different roles understand the gate, whether escalation delays create unsafe workarounds, how frequently legitimate actions are blocked, whether operators can contest incorrect governance data, and how accountability is perceived before and after deployment. These questions would connect the technical architecture more directly to the socio-technical concerns emphasized in AI ethics and governance research.

13 Conclusion

This study presents Adaptive Runtime Governance as a broader research architecture for preserving governance conditions around AI-enabled organizational action and evaluates one implemented component: a deterministic execution gate. The gate represents a proposed action as a persistent decision and revalidates selected state, approval, authority, policy, delegation, and evidence conditions immediately before execution.

Across 223 controlled API-level cases, every predefined outcome was observed, including successful execution and explicit refusal of unauthorized, stale, inconsistent, or invalid conditions. The evidence supports a limited technical and ethical conclusion: historical human approval and current execution authority are not equivalent, and selected conditions needed to preserve attributable human authority can be made technically enforceable at the execution boundary.

The architecture does not establish that an authorized action is fair, safe, accurate, lawful, or ethically justified, and it does not demonstrate production security, organizational effectiveness, adaptive behavior, or scalability. Its contribution is narrower but operationally significant: it provides a tested pattern in which loss of current authorization becomes a reason to stop an AI-enabled organizational action rather than merely an event to document after execution. The broader ARG architecture remains a research agenda whose additional modules require independent implementation and empirical validation.

Declarations

Competing Interests

The author is the creator and developer of the ARG research architecture and the prototype evaluated in this study. No licensing income or other financial benefit has been received.

Author Contribution

M.K. conceived the study and the ARG governance architecture, developed the prototype, designed the evaluation, analysed the results, and reviewed and approved the final manuscript.

Data Availability

The anonymized source snapshot, revised prototype copy, controlled validation script, detailed CSV and JSON outcomes, retrospective analysis, and validation supplement are available from the corresponding author upon reasonable request.

References

1. Tabassi, E.: Artificial Intelligence Risk Management Framework (AI RMF 1.0). NIST AI 100-1. National Institute of Standards and Technology (2023). https://doi.org/10.6028/NIST.AI.100-1
2. Organisation for Economic Co-operation and Development: Advancing Accountability in AI: Governing and Managing Risks Throughout the Lifecycle for Trustworthy AI. OECD Digital Economy Papers No. 349. OECD Publishing: (2023). https://doi.org/10.1787/2448f04b-en
3. European Union: Regulation (EU) 2024/1689 laying down harmonised rules on artificial intelligence. Official J. Eur. Union (2024)
4. Mäntymäki, M., Minkkinen, M., Birkstedt, T., Viljanen, M.: Defining organizational AI governance. AI Ethics. **2**, 603–609 (2022). https://doi.org/10.1007/s43681-022-00143-x
5. Papagiannidis, E., Enholm, I.M., Dremel, C., Mikalef, P., Krogstie, J.: Toward AI governance: identifying best practices and potential barriers and outcomes. Inform. Syst. Front. **25**, 123–141 (2023). https://doi.org/10.1007/s10796-022-10251-y
6. Mökander, J., Floridi, L.: Operationalising AI governance through ethics-based auditing: an industry case study. AI Ethics. **3**, 451–468 (2023). https://doi.org/10.1007/s43681-022-00171-7
7. Novelli, C., Taddeo, M., Floridi, L.: Accountability in artificial intelligence: what it is and how it works. AI Soc. **39**, 1871–1882 (2024). https://doi.org/10.1007/s00146-023-01635-y
8. de Santoni, F., van den Hoven, J.: Meaningful human control over autonomous systems: a philosophical account. Front. Rob. AI. **5**, 15 (2018). https://doi.org/10.3389/frobt.2018.00015
9. Cavalcante Siebert, L., Lupetti, M.L., Aizenberg, E., et al.: Meaningful human control: actionable properties for AI system development. AI Ethics. **3**, 241–255 (2023). https://doi.org/10.1007/s43681-022-00167-3

10. Sandhu, R., Ferraiolo, D., Kuhn, D.R.: The NIST model for role-based access control: towards a unified standard. Proceedings of the Fifth ACM Workshop on Role-Based Access Control, 47–63 (2000). https://doi.org/10.1145/344287.344301
11. Hu, V.C., Ferraiolo, D., Kuhn, D.R., Schnitzer, A., Sandlin, K., Miller, R., Scarfone, K.: Guide to Attribute Based Access Control (ABAC) Definition and Considerations. NIST Special Publication 800 – 162. National Institute of Standards and Technology (2019). https://doi.org/10.6028/NIST.SP.800-162
12. Rose, S., Borchert, O., Mitchell, S., Connelly, S.: Zero Trust Architecture. NIST Special Publication 800 – 207. National Institute of Standards and Technology (2020). https://doi.org/10.6028/NIST.SP.800-207
13. Dempsey, K.L., Ross, R., Stine, K.: Supplemental Guidance on Ongoing Authorization: Transitioning to Near Real-Time Risk Management. National Institute of Standards and Technology (2014). https://doi.org/10.6028/NIST.CSWP.06032014
14. Slagel, J.T., White, L.M., Dutle, A., Muñoz, C.A., Crespo, N.: A Verification Framework for Runtime Assurance of Autonomous UAS. 2024 AIAA DATC/IEEE 43rd Digital Avionics Systems Conference (2024). https://doi.org/10.1109/DASC62030.2024.10748654
15. Batool, A., Zowghi, D., Bano, M.: AI governance: a systematic literature review. AI Ethics. **5**, 3265–3279 (2025). https://doi.org/10.1007/s43681-024-00653-w
16. Laux, J.: Institutionalised distrust and human oversight of artificial intelligence: towards a democratic design of AI governance under the European Union AI Act. AI Soc. **39**, 2853–2866 (2024). https://doi.org/10.1007/s00146-023-01777-z
17. Zhu, L., Lu, Q., Ding, M., Lee, S.U., et al.: Designing meaningful human oversight in AI. AI Ethics. **6**, 286 (2026). https://doi.org/10.1007/s43681-026-01147-7
18. Pang, G., Szliter, M.: Governing AI in operations: ethical risks, accountability mechanisms, and responsible digital operations. AI Ethics. **6**, 338 (2026). https://doi.org/10.1007/s43681-026-01191-3
19. Chen, Z.: Operational responsibility in AI governance: a user-centric liability framework. AI Ethics. **6**, 306 (2026). https://doi.org/10.1007/s43681-026-01163-7
20. Hahn, M., Tretter, M., Dabrock, P.: Ethical perspectives on AI Agents and Agentic AI. AI Ethics. **6**, 218 (2026). https://doi.org/10.1007/s43681-026-01027-0
21. Szabó, G.: Embedding ethical governance in personal AI systems: design and evaluation of a user-centred oversight agent. AI Ethics. **6**, 336 (2026). https://doi.org/10.1007/s43681-026-01193-1