TECHNICAL REFERENCE ARCHITECTURE

# Operationalizing AI decision rights

Applying the MIT CISR AI Decision Matrix to policy checks, human review, and verifiable evidence

**Document ID VP-ARCH-2026-02 | November 2026 | Version 1.0**

For enterprise architects, governance leads, and risk officers

## Contents

01 The decision-rights gap

02 From the MIT matrix to policy decisions

03 Governance roles and decision authority

04 Banking and healthcare scenarios

05 Verifying evidence across frameworks

06 Fleet economics and governance costs

07 Rollout and verification

08 References

**Executive summary:** Researchers at the MIT Center for Information Systems Research (MIT CISR) introduced the AI Decision Matrix to help enterprises allocate decision rights between humans and autonomous AI agents along two axes: ambiguity and risk. While the matrix gives executives and risk committees a clear model for framing, acting, and learning, a framework on a slide deck cannot stop an unauthorized API call in production. Once agents change customer pricing, approve commercial credit, or submit clinical prior authorizations, teams need a runtime governance control plane that enforces decision rights before actions execute. This paper maps the MIT CISR AI Decision Matrix directly to VeriProof's runtime architecture. It shows how engineering, risk, and compliance teams decouple decision rules from agent code, route consequential exceptions to verified authority holders, and produce evidence an auditor can verify against the ledger.

# 1. The decision-rights gap: conceptual matrices vs. runtime control

*Why static acceptable-use policies and hardcoded prompt guardrails break down across a multi-agent portfolio.*

The MIT CISR AI Decision Matrix—developed by Ina Sebastian, Peter Weill, Thomas Haskamp, and Jan vom Brocke from executive interviews and deployments such as One New Zealand's fleet of 50+ AI agents—starts from a practical premise: the same AI capability can be safe in one business context and unacceptably risky in another. Rather than applying blanket autonomy or blanket restrictions, enterprises must design decision rights using two dimensions:

- Ambiguity: How clearly data determines the right answer and the degree of consensus around it. Low-ambiguity decisions are repeatable and predictable; high-ambiguity decisions require interpretation and trade-off judgment.
- Risk: The financial, operational, clinical, or reputational impact if a decision is wrong, and how difficult it is to reverse.

Whether an organization can enforce those decision rights in production depends on where the rules live:

- Stage 1: static policy documents and developer-coded thresholds. When a team runs a handful of internal pilots, keeping risk rules in PDF policies and `if/else` checks or system prompts inside agent code feels manageable. However, every threshold adjustment requires a pull request, a CI run, and a redeploy. As the fleet grows, compliance officers have more rules to track and more exceptions to review.
- Stage 2: portfolio-scale governance across heterogeneous runtimes. When an enterprise scales to dozens of agents across LangGraph, the OpenAI Agents SDK, Microsoft Copilot Studio, browser automation, and Model Context Protocol (MCP) tools, decision rights must be separated from application code. Security and risk teams need deterministic pre-action checks, verified human authority at the moment of escalation, and tamper-evident decision records.

Example escalation: The application pauses the action, verifies the signed review decision, rechecks policy, and reports what happened next.

| MIT decision component | Decision activity | Stage 1 ad-hoc breakdown | VeriProof control plane extension |
|---|---|---|---|
| 1. Framing | Define the problem, non-negotiable constraints, risk tolerance, and success criteria in advance. | Coupled code and policy. Changing a threshold embedded in a tool requires a code change and deployment. | Decoupled policy engine. Versioned rules managed centrally outside agent code. New constraints test in Shadow mode against live traffic before enforcement. |
| 2. Acting | Gather data, evaluate options, authorize actions, and execute approved steps within defined authority. | A shared approval message leaves a gap when the application accepts a reply without checking who sent it. | VeriProof evaluates the proposed action. If review is required, it checks the reviewer's authority before signing the decision. |
| 3. Learning | Monitor outcomes, assign named human accountability, and refine the decision system over time. | Scattered records. Reconstructing a decision across application logs and traces depends on what each system retained. | Anchored evidence packs. Auditors can check exported records against a signed inventory and confirm their ledger anchors. |

## 2. Mapping the MIT CISR AI Decision Matrix to pre-action outcomes

*Translating the four ambiguity-and-risk quadrants into deterministic runtime behavior.*

MIT CISR crosses low and high levels of ambiguity and risk to define four decision types: Routine, Consequential, Exploratory, and Strategic. VeriProof sits in the execution path via SDKs (Python, TypeScript, .NET, Java, Go, Rust), framework adapters, MCP hooks, and browser automation interceptors. Before any protected action executes, VeriProof evaluates the proposed input and facts against central rules and returns one of five standardized outcomes—ordered by precedence from strictest to most permissive: Deny, Escalate, Transform, Warn, or Allow.

| MIT matrix quadrant | Ambiguity & risk profile | Human and AI responsibilities | VeriProof runtime enforcement pattern |
|---|---|---|---|
| 1. Routine decisions | Low ambiguity, low risk | Automate repeatable decisions within agreed limits. A human process owner monitors results. | Allow / Warn / Transform. Routine requests can proceed under policy. A rule can also warn about an unusual value or cap a fee at the permitted amount. |
| 2. Consequential decisions | Low ambiguity, high risk | Set risk limits up front. People monitor performance and handle exceptions. | Escalate / Deny. Hard guardrails block prohibited actions (Deny) or pause execution (Escalate), generating a typed review form in Slack, Teams, or the VeriProof inbox for an authorized reviewer. |
| 3. Exploratory decisions | High ambiguity, low risk | Iterative human-AI co-framing; tighten governance rapidly as experimental workflows move toward higher-risk production. | Shadow mode → staged enforcement. New rules evaluate live agent traffic in Shadow mode first, recording would-be escalations so business owners and governance engineers can tune thresholds before enforcing. |
| 4. Strategic decisions | High ambiguity, high risk | Human-led framing and final trade-off judgment; task and orchestration agents gather data and propose options. | Escalate for a strategic decision. The agent gathers options. The person authorized for the decision weighs the trade-offs and gives or withholds approval. |

# 3. Enforcing Framing, Acting, and Learning across six governance roles

*Named human ownership, separation of duties, and structured in-flight resolution.*

MIT CISR emphasizes human accountability for outcomes as decision rights shift between people and AI. Accountability for an agent and authority to approve a specific action can sit with different people. The following roles illustrate how teams divide that work.

- Business owner: Sponsors the agent, frames business value and risk tolerance, reviews Shadow mode impact, and names the people authorized to decide strategic exceptions.
- Risk and compliance officer: Selects applicable regulatory frameworks, defines non-negotiable policy constraints, and tracks control coverage and remediation gaps.
- Governance engineer: Authors versioned rules, configures review profiles and SLA escalation chains, and manages staged rollouts across applications.
- Developer: Instruments protected boundaries via SDKs or framework adapters and resolves telemetry or schema gaps when an agent run omits required facts.
- Reviewer (authority holder): Evaluates escalated consequential decisions in the VeriProof inbox or a connected work app.
- Auditor: Examines the evidence within the audit engagement and independently verifies the records provided.

### Preventing rubber-stamping and NIGO rework in Consequential decisions

When a Consequential decision escalates—such as a telecom plan upgrade with a custom pricing override or a commercial credit adjustment—a binary approve/reject button forces a bad trade-off. Rejecting a nearly correct proposal terminates the workflow and triggers manual "not in good order" (NIGO) rework; approving without structured context creates a rubber-stamp trap.

VeriProof review profiles render validated forms (Slack modals, Teams Adaptive Cards, or web forms) displaying the proposed action, the business facts, and the exact rule triggered. Reviewers can approve, reject with a mandatory reason, or modify editable fields in flight (for example, adjusting a proposed fee waiver from $1,500 to $500). VeriProof checks the reviewer’s identity, application access, assignment, and separation-of-duties rules before signing the decision. The application verifies the callback, applies any authorized field corrections, and checks that the action can still proceed.

## 4. Regulated production patterns: financial services and healthcare

*Two illustrative scenarios show how a bank and a health system could apply the decision matrix.*

### Illustrative banking scenario: commercial credit and treasury

A regional bank uses agents to analyze financial statements, prepare credit renewals, and propose treasury transfers. Its credit policy gives agents limited authority over routine renewals and reserves larger adjustments for a senior credit officer.

| Banking agent action | MIT quadrant | Runtime outcome & authority routing | Audit & privacy posture |
|---|---|---|---|
| KYC/AML screening & financial ratio extraction | Routine | Allow: The screening result meets the bank's rules and the application supplies the required facts. | Decision record hashed and anchored in background. |
| Credit line renewals (<$50k, DSCR > 1.35x) | Routine | Allow / Transform: Evaluated against central credit policy v4.2; fee waivers above $250 automatically capped via Transform. | Rule revision and input digest recorded per run. |
| Credit adjustments ($50k–$500k) or rate overrides | Consequential | Escalate: Pauses execution and routes a Teams Adaptive Card to a Senior Credit Officer with a 4-hour SLA timer. | The application verifies the signed response and applies the authorized credit limit before calling the core banking system. |
| Distressed loan restructuring & covenant waivers | Strategic | Escalate: The bank assigns the final decision on covenant waivers to its head of commercial lending. | The bank keeps borrower details in its own systems and records the decision's ledger reference. |

When lending guidance changes, the credit risk team tests a new debt-service coverage threshold in Shadow mode across its lending agents. The team can see which renewals would reach a credit officer before it puts the threshold into effect.

## Illustrative healthcare scenario: prior authorization and claim appeals

A health system uses agents to check eligibility, assemble prior authorization requests, and draft appeals. Hospital policy reserves medical necessity decisions for authorized clinicians and keeps patient identifiers out of external logs.

| Healthcare agent action | MIT quadrant | Runtime outcome & authority routing | Audit & privacy posture |
|---|---|---|---|
| Eligibility checks & standard CPT/ICD-10 lookups | Routine | Allow: The agent checks eligibility when the required facts are present and the request meets hospital policy. | The hospital keeps patient identifiers in its clinical system and records a hash for the audit trail. |

| Healthcare agent action | MIT quadrant | Runtime outcome & authority routing | Audit & privacy posture |
|---|---|---|---|
| Generative drafting of appeal letters for novel therapies | Exploratory | Shadow mode records how a proposed rule would treat an appeal draft before the hospital enforces it. | The export records which fields were withheld. |
| Prior auth submission for biologics or inpatient surgery | Consequential | Escalate: Freezes the browser agent prior to portal submission; routes clinical criteria match to a Utilization Review Nurse (2-hour SLA). | Records the reviewer's identity, decision time, and submitted corrections. |
| Oncology pathway & experimental coverage escalations | Strategic | Escalate: The hospital assigns final review of an exceptional coverage request to its medical director. | The audit team can review the signed timeline and check the evidence against its ledger anchor. |

Routine eligibility checks can proceed without waiting for clinical review. A surgical request that needs clinical judgment goes to the designated nurse before submission. The hospital can measure the time each path takes using the payer's recorded submission and response times.

## 5. Cryptographic evidence packs and multi-framework compliance

*How an auditor traces a decision to the relevant controls and verifies the exported records.*

Traces help engineers diagnose a run. A signed evidence pack answers a different question: do the exported records match the versions committed to the ledger? VeriProof batches record hashes and anchors them to the tenant's selected ledger provider.

Compliance teams link decision evidence to the controls in scope for an assessment. They can send selected reports to Vanta and export a signed evidence pack for an auditor.

- `integrity-proof.json`: ECDSA-signed inventory of the exported evidence files, SHA-256 digests, Merkle inclusion proofs, and ledger anchor references.
- `sessions/, revisions/, and review-records/: Session metadata, source revisions, and review history included in the export.`

- `redaction-report.json: Identifies information withheld from the export and explains why.`
- `chronology.pdf`: A human-readable decision timeline for legal, clinical, and executive reviewers.

An auditor uses veriproof-verify and a trusted public key to check the files against the signed inventory, then confirms the anchor with the ledger provider. This shows whether the exported records match the evidence that was committed.

## 6. Fleet economics and total cost of governance

*VeriProof's public calculator and customer portal share a fleet planning model. Its starting workload is 12 applications, 250 human reviews a month, six evidence requests a year, and 60 investigations a year.*

With that workload, compliance, evidence preparation, investigations, and reviews take 305 hours per application each year under the current assumptions and 102.5 with VeriProof. Two governance staff have 3,600 productive hours between them. Dividing those hours by the work per application gives the table's rounded estimates of 12 and 35. Engineering time is counted separately.

| Operational metric | Ad-hoc / framework-only governance | VeriProof horizontal control plane | What the model shows |
|---|---|---|---|
| Applications supported by two governance staff | About 12 applications | About 35 applications | The model assigns the same two people a larger portfolio when teams share rules and review processes. |
| Initial integration effort | Enter your team's estimate for building and testing these controls. | ~40 developer hours per agent using an SDK or adapter | Teams can measure integration effort from the first engineering task to a verified test action. |
| Ongoing governance maintenance | ~320 developer hours / app / year maintaining policies and audit exports | ~120 developer hours / app / year | In this model, the difference is 200 developer hours per application each year. |
| Rule updates across agent fleet | Code pull request, CI pipeline, and redeploy per agent | Test in Shadow mode, then roll out by application. | Decouples risk and compliance iteration from engineering sprint cycles. |

| Operational metric | Ad-hoc / framework-only governance | VeriProof horizontal control plane | What the model shows |
|---|---|---|---|
| Where policy checks run | Custom cloud plumbing per environment | Managed cloud, customer-hosted Azure, or local evaluation of eligible rules at the Edge Gateway | Local evaluation lets eligible checks run beside the agent while its signed policy lease is valid. |

# 7. Production rollout and verification checklist

*Operationalize the MIT AI Decision Matrix across your agent fleet in four steps.*

1. Classify your agent portfolio on the MIT matrix. Classify the decisions each agent makes. A lending agent can calculate a ratio routinely and still need a credit officer to approve a limit change.
2. Instrument one consequential boundary. Wrap a high-impact tool call, MCP server, browser agent, or Copilot Studio action using a VeriProof SDK or adapter, and configure a review profile for Slack, Teams, or the VeriProof inbox.
3. Validate rules in Shadow mode. Author your framing constraints and run them in Shadow mode against live or test traffic. Review the would-be escalations, transforms, and denials with the business owner until the human review volume matches team capacity.
4. Enforce controls and verify an evidence pack against the ledger. Promote the rule to Enforced (with dual-approval change control where required), trigger an escalated review, export the resulting evidence pack, and verify it independently with `veriproof-verify`.

# 8. References


1. MIT Center for Information Systems Research, “Designing Decision Rights for AI,” June 2026.
2. VeriProof Technical Reference Architecture, “Agentic AI production governance (DOC ID: VP-ARCH-2026-01).”
3. Regulation (EU) 2024/1689 (Artificial Intelligence Act), Articles 12 (Record-keeping) and 14 (Human oversight). eur-lex.europa.eu/eli/reg/2024/1689/oj
4. ISO/IEC 42001:2023, Information technology — Artificial intelligence — Management system. iso.org/standard/81230.html
5. NIST, “Artificial Intelligence Risk Management Framework (AI RMF 1.0).” nist.gov/itl/ai-risk-management-framework
6. MITRE, “CWE-367: Time-of-check Time-of-use (TOCTOU) Race Condition.” cwe.mitre.org/data/definitions/367.html
7. VeriProof, “How VeriProof checks evidence integrity.” veriproof.app/whitepapers/tamper-evidence
8. VeriProof, “Verify exported evidence.” docs.veriproof.app/agent-integrations/verification
9. VeriProof, “Business-value and ROI calculator.” veriproof.app/roi


## Try one workflow with your team

Choose a credit limit and name the person authorized to change it. Test a routine request and an exception. Check who could approve the exception and what the application recorded.

**Start your free 60-day trial**

veriproof.app/start