Policy-Enforced Delivery (PED)

PED community

2026-06-14

Policy-Enforced Delivery (PED)

Abstract

Modern software organizations face a growing imbalance between the rate at which changes can be produced and the rate at which those changes can be evaluated.

Advances in automation, cloud infrastructure, continuous integration, and Large Language Models (LLMs) have dramatically reduced the effort required to create code, infrastructure definitions, configuration changes, documentation, and operational workflows. The ability to generate change has accelerated. The ability to evaluate change has not.

In many organizations, human approval remains the dominant control mechanism governing delivery. As change volume increases, approval queues grow, review latency increases, and engineering throughput becomes constrained by a finite number of decision makers.

Policy-Enforced Delivery (PED) is a governance framework designed to address this imbalance. PED does not seek to eliminate governance. It makes governance observable, measurable, and increasingly deterministic.

The framework uses evidence-driven controls to evaluate change, reserves cognitive decision-making for ambiguity, and continuously converts repeatable decisions into automated policy. PED is intentionally technology-agnostic. It can be applied to infrastructure, software delivery, platform engineering, security governance, operational change management, documentation systems, and any environment where changes must be evaluated before execution.

1. The Problem

1.1 Change Production Has Become Cheap

Historically, producing software changes was expensive. Engineering teams spent substantial time writing code, creating infrastructure, producing documentation,

and implementing operational changes. The cost of producing a change naturally limited change volume.

Modern tooling has altered this equation. Automation platforms reduce operational effort. Infrastructure can be defined declaratively. Continuous integration systems perform validation automatically. LLMs have further reduced the cost of generating implementation artifacts.

The result is simple: organizations can now generate proposed changes far faster than they can evaluate them.

This trend is expected to continue. A delivery system that depends primarily on manual review will increasingly be limited by review capacity, not implementation capacity.

1.2 Human Approval Does Not Scale Linearly

Most organizations respond to increased change volume by introducing additional reviews, approvals, committees, signoffs, or governance checkpoints. These controls often improve confidence, but they create a new bottleneck: decision-making capacity.

Little's Law states:

```
L = A * W
```

Where:

- `L` is work in the system
- `A` is arrival rate
- `W` is time spent in the system

As the arrival rate of changes increases, work-in-progress grows unless evaluation capacity grows proportionally. Organizations typically respond in one of three ways:

- hire additional reviewers
- accept growing queues
- reduce change volume

None of these approaches fundamentally solves the problem. Decision-making capacity becomes the throughput constraint.

1.3 Cognitive Capacity Is a Scarce Resource

Every delivery system contains two categories of decision making:

- **Deterministic decisions**, where outcomes can be derived directly from available evidence.
- **Cognitive decisions**, where ambiguity, uncertainty, incomplete information, competing objectives, or risk acceptance require interpretation.

PED assumes that cognitive capacity is constrained regardless of whether it is provided by humans, LLMs, autonomous agents, review boards, or other decision-support systems. The framework intentionally does not distinguish between these implementations.

Instead, PED focuses on ensuring that cognitive resources are engaged only when deterministic evaluation has been exhausted. The objective is not to eliminate cognition. The objective is to make every escalation to a cognitive system necessary, explainable, and useful.

1.4 Most Reviews Contain Deterministic Questions

A significant percentage of change reviews do not involve judgment. Reviewers often answer questions such as:

- Does the actor own the affected system?
- Does the change conform to policy?
- Is rollback possible?
- Does required evidence exist?
- Is the blast radius acceptable?
- Are previous exceptions still valid?

These questions often have deterministic answers. Yet organizations frequently require cognitive systems to evaluate them repeatedly.

The result is governance overhead without corresponding governance value. PED seeks to identify and eliminate these forms of non-contributory review by converting repeatable questions into observable controls.

2. Foundational Requirement: Observable Decisions

A decision that cannot be explained cannot be automated.

All PED decisions must be:

- attributable
- reproducible
- timestamped
- traceable to evidence
- reconstructable after the fact

PED does not trust outcomes alone. PED trusts evidence supporting outcomes. Every control, policy, evaluation, and decision within the framework must satisfy this requirement.

A control that cannot produce an observable audit trail is not eligible for automation. It may still be useful, but it should be treated as a cognitive intervention until it can explain itself through evidence.

Observable decisions require more than logs. A useful decision record captures:

- the proposed change
- immutable inputs
- the actor requesting or performing the change
- control evaluations
- evidence references
- policy versions
- timestamps
- final outcome
- cognitive reviewer and rationale when escalation occurs

This record becomes the common language between delivery systems, security teams, platform teams, auditors, and application owners.

3. Core Principles

Principle 1: Deterministic Decisions Belong to Machines

If a decision can be evaluated consistently from available evidence, it should not require cognitive intervention.

Humans, LLMs, and other cognitive systems are valuable because they resolve ambiguity. They should not be consumed by repeatable evaluation tasks.

Principle 2: Cognitive Resources Are Reserved for Ambiguity

Cognitive review should occur only when uncertainty exists. Examples include:

- conflicting objectives
- incomplete evidence
- novel situations
- exception handling
- risk acceptance decisions
- unclear ownership boundaries

PED treats cognitive capacity as a scarce resource that should be preserved for situations where deterministic evaluation cannot produce a defensible outcome.

Principle 3: Governance Must Scale

The cost of governance should grow more slowly than the rate of change being governed. Organizations that require governance effort to scale proportionally with change volume will eventually become constrained by governance itself.

Principle 4: Controls Must Demonstrate Value

Manual controls should continuously justify their existence. A review process that never changes outcomes should be considered a candidate for automation.

Controls should be retained because they provide measurable value, not because they have historically existed.

Principle 5: Automation Follows Evidence

Automation should not be introduced because it appears safe. Automation should be introduced because evidence demonstrates that a deterministic evaluation consistently produces the same outcome as cognitive review.

Principle 6: Exceptions Are First-Class Evidence

Exceptions are not failures of governance. They are evidence about where policy is incomplete, ownership is unclear, or the organization intentionally accepts risk. A PED implementation records exceptions with the same care as approvals and rejections.

4. Control Domains

PED evaluates changes across independent control domains. Organizations may implement them in any order. The order of implementation is not important. The decision outcome is based on the combined evaluation of applicable domains.

The five core domains are ownership, conformance, reversibility, blast radius, and evidence.

4.1 Ownership

Ownership determines whether the actor has authority to perform the proposed action.

Questions include:

- Who owns the affected resource?
- Is authority delegated?
- Is the requester authorized?
- Is the owning team still active and accountable?
- Does the change cross ownership boundaries?

Ownership establishes accountability. Without ownership, governance cannot function effectively.

Common evidence includes repository metadata, CODEOWNERS rules, service catalog records, infrastructure tags, incident ownership, approval delegation records, and team membership snapshots.

4.2 Conformance

Conformance evaluates whether the proposed change aligns with organizational policy.

Examples include:

- security requirements
- architectural standards

- compliance obligations
- operational standards
- data handling rules
- documentation quality rules

PED does not prescribe specific policies. It requires policy evaluation to be observable and evidence-based.

Common evidence includes policy-as-code results, static analysis output, configuration checks, dependency scans, architecture decision records, and documentation metadata.

4.3 Reversibility

Reversibility evaluates the ability to recover from an incorrect decision. Not all changes carry equal recovery risk.

Examples include:

- configuration updates
- service deployments
- infrastructure modifications
- database migrations
- data destruction activities
- documentation publication

Organizations should understand the cost and complexity of recovery before approving change.

Common evidence includes rollback plans, migration down scripts, feature flag plans, backup verification, canary strategy, restoration testing, and immutable artifact references.

4.4 Blast Radius

Blast radius evaluates potential impact.

Questions include:

- What systems are affected?
- What dependencies exist?
- How many users may be impacted?
- What business functions depend on this component?
- Is the change contained to one environment or shared infrastructure?
- Does the change affect production data?

Blast radius is not necessarily a rejection criterion. It influences the burden of proof required for approval. Higher-impact changes may require stronger evidence than lower-impact changes.

Common evidence includes dependency maps, service catalog relationships, traffic metrics, asset inventory, deployment target metadata, and user impact estimates.

4.5 Evidence

Evidence evaluates whether sufficient proof exists that the change behaves as intended.

Examples include:

- testing outcomes
- validation activities
- operational health indicators
- runtime observations
- deployment verification
- screenshots or rendered documentation checks
- manual test attestations when automation is not yet available

Evidence transforms approval from opinion into observation. A PED system should make missing evidence visible rather than hiding it behind an approval button.

5. Decision Outcomes

PED does not require controls to execute in a particular sequence. Control domains are independent sources of evidence.

A final decision is rendered from the combined outcome of all applicable evaluations. Organizations may distribute evaluation across multiple systems, workflows, pipelines, or governance processes. Different controls may execute at different stages of the delivery lifecycle.

The important requirement is that each evaluation produces observable evidence and contributes to the final decision record.

PED uses three primary decision outcomes:

- `approved`: deterministic controls supplied enough evidence to proceed.
- `rejected`: deterministic controls identified a blocking failure.
- `needs_cognitive_review`: deterministic controls found ambiguity, missing evidence, exception paths, or risk acceptance requirements.

Control evaluations commonly use four statuses:

- `pass`: the control requirement is satisfied.
- `fail`: the control requirement is not satisfied.
- `needs_review`: the control cannot produce a deterministic answer.
- `not_applicable`: the control does not apply to this change.

The final decision should be attributable to immutable inputs, recorded evidence, control evaluations, policy versions, and timestamps. It should not depend on a specific workflow implementation.

6. The PED Decision Record

The decision record is the portable artifact produced by a PED evaluation. It is intentionally simple enough to emit from CI, infrastructure tools, internal developer platforms, documentation pipelines, and manual workflows.

At minimum, a decision record should include:

- record identifier
- schema version
- requested change
- actor
- affected resources
- control results
- evidence references
- final outcome
- timestamps
- policy version
- reviewer and rationale when cognitive review is required

The decision record is not a compliance trophy. It is operational data. Teams use it to debug slow approvals, identify missing evidence, compare control effectiveness, and decide what to automate next.

7. Incremental Adoption

PED does not require complete implementation. Organizations frequently delay governance improvements because they believe a comprehensive solution is necessary. PED rejects this assumption.

A useful PED implementation may begin with:

- an audit trail
- documented decisions
- ownership tracking
- recorded exceptions
- evidence links in pull requests
- a single policy-as-code control

The framework can evolve incrementally. Each improvement increases observability and creates new opportunities for deterministic evaluation.

Progress is measured by improved decision quality and reduced reliance on cognitive review, not by framework completeness.

8. Implementation Patterns

PED can be implemented in many kinds of codebases. The framework is concerned with decision quality, not tool preference.

Infrastructure as Code

Infrastructure systems often have clear deterministic signals: resources, providers, tags, environments, plans, and ownership metadata. PED controls can evaluate whether a plan touches production, changes persistent storage, lacks rollback evidence, or crosses ownership boundaries.

Application Services

Application services can emit decision records from CI. A Go, Java, Python, or Node service can combine test results, deployment metadata, ownership, and runtime health checks into a delivery gate.

UI Work

UI changes often need rendered evidence. A PED control may require screenshots, accessibility checks, visual diff results, or explicit rationale when a user experience change cannot be proven by automated checks.

Documentation Repositories

Documentation changes can be governed by ownership, audience, published surface, links, build output, and reviewer rationale. A docs-only change may have lower reversibility risk than a database migration, but it still deserves observable evaluation when it affects public or regulated content.

Operational Workflows

Operational changes can emit decision records from runbooks, change tickets, incident tooling, and automation platforms. PED does not require everything to start in Git.

9. The Feedback Loop

The feedback loop is the mechanism through which PED improves over time. Without the feedback loop, PED becomes a static governance model. With the feedback loop, PED becomes self-improving.

Observe

Capture decisions, evidence, outcomes, and cognitive interventions. Every decision becomes a source of learning.

Explain

Require justification whenever cognitive review is necessary. The goal is not approval. The goal is understanding why deterministic evaluation was insufficient.

Analyze

Identify recurring intervention patterns. Examples include:

- ownership ambiguity
- missing dependency information
- policy exceptions
- insufficient evidence
- undefined standards
- repeated low-risk approvals

These patterns reveal governance bottlenecks.

Encode

Convert recurring decision logic into observable policy. If a decision can be consistently explained, it can often be represented as a deterministic evaluation.

Automate

Replace deterministic cognitive decisions with automated controls. Automation is not the objective. Improved governance scalability is the objective. Automation is one mechanism used to achieve it.

Measure

Continuously evaluate outcomes. Organizations should measure:

- intervention rates
- approval rates
- review latency
- policy effectiveness
- exception frequency
- outcome modification rates
- deterministic approval percentage
- time to encode recurring decisions

The purpose of measurement is to determine whether controls continue to provide value.

10. Evaluating Governance Effectiveness

PED encourages organizations to ask a simple question:

> What value did this intervention provide?

Possible outcomes include:

- policy violation identified
- risk identified
- additional evidence requested
- decision changed
- exception granted
- owner corrected
- rollback plan improved

When interventions repeatedly produce no change in outcome, organizations gain evidence that deterministic evaluation may be sufficient. This allows governance evolution to be driven by observation rather than assumption.

11. Relationship to Existing Practices

PED complements existing delivery and governance practices.

Continuous Delivery

Continuous delivery improves the flow of change. PED improves the evaluation of change. A mature delivery system should be able to move quickly without turning every approval into an act of trust.

Policy as Code

Policy as code is one implementation technique. PED provides the governance model around it: which decisions should be encoded, what evidence is required, when cognition is necessary, and how controls prove value over time.

Change Advisory Boards

PED does not require organizations to delete existing approval bodies. It gives them better inputs and a path to spend less time on deterministic questions. Review boards become more valuable when they focus on ambiguous risk rather than repeatable checklist work.

AI-Assisted Engineering

AI-assisted engineering increases change production. PED provides a way to scale change evaluation without relying on the same scarce reviewers for every generated artifact. The framework applies equally whether the change was written by a human, generated by an LLM, or produced by automation.

12. Continuous Governance Evolution

PED assumes that governance systems are never complete. Every intervention that cannot be automated today becomes a candidate for future analysis. Each

explanation provided by a cognitive reviewer creates additional information about why deterministic evaluation was insufficient.

Over time, organizations accumulate a body of evidence describing:

- where ambiguity exists
- where standards are incomplete
- where ownership is unclear
- where evidence is insufficient
- where policy cannot yet be expressed deterministically

This information becomes the roadmap for future improvement. As organizations mature, more deterministic decisions are automated and cognitive capacity is increasingly concentrated on genuinely ambiguous problems.

Governance evolution is driven by observed decisions rather than theoretical requirements.

13. Conclusion

Policy-Enforced Delivery is not an automation framework. It is a governance framework.

Its goal is to ensure that decisions are observable, evidence-based, and scalable. By treating cognitive capacity as a constrained resource and deterministic evaluation as the preferred mechanism for routine decisions, organizations can continuously improve governance while preserving accountability.

The objective is not to remove humans, LLMs, or other cognitive systems from decision making. The objective is to ensure they are engaged only when their involvement is necessary.

PED creates a path toward governance systems that become more effective, more scalable, and more observable as they operate.