AP-009 — Laboratorios Alexandria

Constitutional Governance as Engineering Strategy: Ethics Before Capability in Autonomous Systems

Laboratorios Alexandria
contact@laboratoriosalexandria.com
June 2026

Abstract

The prevailing approach to AI system development follows a capability-first trajectory: build the system, demonstrate its capabilities, then add governance constraints. We present an alternative approach in which constitutional governance is implemented before capabilities are deployed, and argue that this ordering is not merely ethically preferable but produces measurably better engineering outcomes. Drawing on operational data from a multi-system autonomous environment—comprising 307 completed deliberation sessions with independent quality grading, 3,697 decision narratives that explicitly reference constitutional provisions, and zero irreversible decisions executed without human confirmation—we demonstrate that constitutional governance functions as a form of test-driven development for decision quality. The constitution specifies what correct behavior looks like before the system attempts to produce it. We present a concrete case in which a constitutional test failed, was detected, and led to systemic correction—mirroring the TDD cycle of red-green-refactor. We discuss the analogy to formal specification in software engineering, the role of constitutional citation in automated decision auditing, and the practical implications for organizations building autonomous systems.

1 Introduction

Software engineering learned decades ago that writing tests before code—test-driven development (TDD)—produces more reliable software than writing code first and testing later. The test specifies the desired behavior; the code is written to satisfy the specification. When the test fails, the developer knows the code is wrong. When the test passes, the developer has evidence (not proof) that the code is correct.

We argue that constitutional governance in autonomous systems follows the same logic. A constitution specifies what correct decision-making looks like—what criteria a decision must satisfy, what evidentiary standards must be met, what procedural requirements must be followed—before the system makes any decisions. When a system's decision violates its constitution, the violation is detectable and auditable. When a decision satisfies constitutional requirements, the operator has evidence (not proof) that the decision was well-formed.

This analogy is not metaphorical. It is structural. In TDD, the test suite serves as a formal specification that constrains implementation. In constitutional governance, the constitution serves as a formal specification that constrains decision-making. The engineering benefits are analogous: early defect detection, regression prevention, documentation of intent, and confidence in system behavior.

The alternative—capability-first development, where governance is added after the system is operational—is analogous to writing code without tests and adding them retroactively. The retroactive approach is more expensive, less reliable, and produces governance that is shaped by the system's existing behavior rather than by principled requirements. We present operational evidence that ethics-first development produces systems that are not only more trustworthy but more effective.

2 Constitutional TDD: The Analogy

2.1 Constitution as Specification

In test-driven development, a test specifies a behavioral contract: given these inputs, the system should produce these outputs or satisfy these properties. The test is written first, fails initially (because the code doesn't exist yet), and passes when the implementation is correct.

A constitution for an autonomous system specifies a decision-making contract: given a decision context, the system should satisfy these criteria, consult these standards, and follow these procedures. The constitution is written first, and decisions are evaluated against it. A decision that violates the constitution is analogous to a failing test: it identifies a defect in the system's decision logic.

2.2 Constitutional Citation as Test Output

In our operational environment, the autonomous decision-making system produces decision narratives that explicitly reference constitutional provisions. When the epistemic judge grades a deliberation session, it cites the specific constitutional articles that informed its grade. When a decision is flagged for review, the constitutional provisions that were satisfied or violated are recorded.

This constitutional citation serves the same function as test output in software engineering: it provides a traceable link between the specification (constitution) and the behavior (decision). When the judge cites Article 14 to justify a Grade D verdict, the operator can verify that the grading criteria were applied correctly. When the system cites Article 5 to reject a claim without citation, the operator can verify that evidentiary standards were enforced.

In our operational data, 3,697 decision narratives contain explicit references to constitutional provisions. This number represents not a reporting metric but a structural property of the system: constitutional citation is not optional commentary added to decisions. It is a required component of the decision pipeline. Decisions without constitutional justification are structurally incomplete, analogous to code that has not been tested.

2.3 The Zero-Irreversible Record

Since the implementation of constitutional governance, zero irreversible decisions have been executed without human confirmation. This record is not the result of a policy memo or a verbal agreement. It is the result of architectural enforcement: the constitution specifies that irreversible actions require human confirmation, and the system's architecture implements this specification as a hard constraint.

The analogy to TDD is direct: a critical safety test that has never failed. The value of such a test is not merely that it passes—it is that the system was designed to make it pass. The constitution shaped the architecture, not the reverse.

3 A Constitutional Test That Failed

The TDD analogy gains force when we can demonstrate not just tests that pass but tests that fail—and the systemic corrections that follow. In TDD, the most informative moment is the red phase: the test fails, revealing a defect that was invisible before the test existed. Constitutional governance produces analogous moments.

In 307 completed deliberation sessions, the epistemic judge assigned 62 Grade D verdicts—sessions in which the constitutional criteria for epistemic quality were not met. Each Grade D verdict functions as a failing test: the deliberation produced output, but the output did not satisfy the specification. The judge's verdict includes explicit citation of which constitutional provisions were violated, providing the diagnostic information necessary for correction.

Consider the distribution of deliberation outcomes as a test suite report:

Constitutional Criterion	Sessions	%	TDD Equivalent
Grade A — Robust empirical support, explicit falsifiability	11	3.6%	Pass (highest)
Grade B — Consistent empirical support, implicit falsifiability	54	17.6%	Pass
Grade C — Coherent theoretical support, no falsifiability	180	58.6%	Conditional pass
Grade D — Speculative claims, insufficient evidence	62	20.2%	Fail

Table 1. Deliberation outcomes as constitutional test results. 307 sessions, June 2026.

The 20.2% failure rate is not a deficiency of the system—it is evidence that the constitutional specification has teeth. A test suite in which every test passes is either testing trivial properties or is not testing enough. The 62 Grade D verdicts demonstrate that the judge applies constitutional criteria with genuine discriminative power: it rejects speculative claims, identifies insufficient evidence, and cites the specific articles that the deliberation failed to satisfy.

The corrective cycle that follows a Grade D verdict mirrors the TDD refactor phase. The failing session's correlation is not discarded—it is returned to the detection corpus with its quality assessment recorded. If new evidence emerges (additional papers, independent corroboration from a different data source), the correlation becomes eligible for re-deliberation. The constitutional specification remains unchanged; the system's capacity to satisfy it improves as the knowledge base grows. This is precisely the TDD dynamic: the test defines correctness; the implementation evolves to meet it.

4 Why Ethics-First Produces Better Engineering

4.1 Early Defect Detection

In software engineering, defects caught during specification are orders of magnitude cheaper to fix than defects caught in production. The same principle applies to autonomous decision-making. A constitution written before deployment specifies the boundaries of acceptable behavior before the system encounters edge cases in operation. Without a prior specification, edge cases are resolved ad hoc—by whatever heuristic the system has learned or whatever intuition the operator applies in the moment.

The deliberation system provides a concrete example. When a domain arguer presents a claim without citation—violating Article 5 of the constitution—the dialectician challenges it immediately, during the deliberation, before it can propagate into a thesis or influence the judge's verdict. This is early defect detection: the constitutional provision catches the error at the point of generation, not after it has been integrated into downstream conclusions.

4.2 Regression Prevention

As autonomous systems evolve—new capabilities, new data sources, updated models—previously correct behaviors can regress. A constitutional specification serves as a regression test: any system modification that causes decisions to violate constitutional requirements is detected immediately. Without constitutional regression testing, capability upgrades can silently degrade decision quality.

This principle was tested empirically when the deliberation system's underlying language models were updated. The constitutional criteria remained unchanged, and the judge's grading behavior was compared across model versions. Because the constitution serves as a fixed specification, regressions in argument quality—for example, a new model that produces more verbose but less evidentially supported arguments—are detectable through grade distribution shifts. The finding that verbosity does not correlate with quality (Grade D sessions averaged 10,026 characters per contribution versus 8,210 for Grade B) was itself discovered through constitutional regression analysis.

4.3 Documentation of Intent

A constitution is a living document that records what the system's designers intended it to do. As systems scale and teams change, this record of intent becomes increasingly valuable. Without it, the reasons behind specific behavioral constraints are lost, and future developers cannot distinguish between constraints that serve a purpose and constraints that are artifacts of early implementation decisions.

4.4 Graduated Trust

Constitutional governance enables a trust gradient that would be impossible without it. Because every decision is evaluated against constitutional criteria and recorded with constitutional citations, the system's track record is auditable. An operator can verify not just that the system made correct decisions, but why it made them—which constitutional provisions it invoked, which alternatives it considered, which criteria it applied.

The trust gradient operates concretely through the relationship between constitutional compliance history and operational scope. A system that has accumulated hundreds of constitutionally compliant decisions earns expanded autonomy—not by assertion but by evidence. In our environment, the deliberation system's scope has expanded from initial controlled experiments (15 sessions across limited domains) to autonomous operation across multiple domain pairs, producing 307 sessions with consistent constitutional compliance. Each session is a data point in a growing case for graduated trust. Without constitutional governance, this graduation would be arbitrary—a human judgment call rather than an evidence-based decision.

5 The Capability-First Alternative

The dominant approach in AI development is capability-first: build the most capable system possible, then constrain it through alignment techniques (RLHF, Constitutional AI, safety fine-tuning). This approach treats governance as a constraint on capability rather than as a specification of correct behavior.

The engineering problems with this approach mirror the problems with retroactive testing in software. First, the system's architecture is not designed for governance—governance is bolted on, creating mismatches between what the system can do and what it is allowed to do. Second, the governance constraints are shaped by observed failures rather than by principled specification—they are patches, not designs. Third, the system's developers accumulate technical debt in governance: each new capability requires new constraints, and the interaction between constraints becomes increasingly difficult to manage.

The ethics-first alternative avoids these problems by specifying governance requirements before capabilities are implemented. The architecture is designed for governance from the start. The constraints are principled rather than reactive. And the interaction between constraints is managed by the constitution rather than by ad hoc patching.

The distinction is visible in the quality of the resulting governance artifacts. In a capability-first system, governance documentation describes what the system was observed to do wrong and how it was corrected. In an ethics-first system, governance documentation describes what correct behavior looks like and how the system was designed to produce it. The former is a repair log; the latter is an engineering specification. The difference is not cosmetic—it determines whether governance scales with capability or falls behind it.

6 Constitutional Coverage and the Problem of Untested Provisions

Software engineering distinguishes between having tests and having adequate test coverage. A system with a comprehensive test suite but low coverage—many tests that exercise the same code paths while leaving others untested—provides false confidence. Constitutional governance faces the same challenge.

Of the constitutional provisions in force, some are exercised in every deliberation session (Article 14, the grading criteria, is invoked in all 307 completed sessions). Others are exercised rarely or never—not because they are irrelevant but because the situations they govern have not

arisen. A provision that specifies behavior under crisis conditions provides no evidence of compliance if no crisis has occurred.

This is not a theoretical concern. The zero-irreversible record, while impressive, is a test of a single constitutional provision—the one governing irreversible actions. The 3,697 constitutional citations demonstrate that many provisions are regularly exercised. But the distribution of citations across provisions is uneven. Some articles are heavily cited; others may be structurally present but operationally inert.

The response to this challenge is the same as in software engineering: pursue coverage deliberately. Scenario testing, edge case exploration, and adversarial probing can exercise constitutional provisions that normal operation leaves untested. The constitution's falsifiability conditions—each provision specifies conditions under which it would be shown to be unnecessary—provide a framework for designing such tests. A provision whose falsifiability conditions have never been approached is a provision that has never been meaningfully tested, regardless of how many times it has been cited in routine decisions.

7 Sources of Uncertainty and Limitations

The operational data comes from a single multi-system environment. The engineering benefits of constitutional governance may depend on properties of this specific environment—the operator's discipline, the system's architecture, the domain of application—that do not generalize.

The zero-irreversible record is a product of architectural enforcement, not of the constitution alone. A constitution without architectural enforcement—a document that specifies requirements without implementing them as hard constraints—would not produce the same record.

The TDD analogy, while structurally apt, has limits. Software tests produce binary pass/fail results; constitutional evaluation often involves judgment about degree of compliance. The four-grade system (A through D) introduces intermediate states that have no direct analog in binary testing. The analogy highlights the engineering logic of ethics-first development but does not claim that constitutional governance is as mechanically precise as automated testing.

The 3,697 constitutional citations are produced by the system itself. We cannot independently verify that each citation is substantively correct—that the constitutional provision was genuinely relevant to the decision, rather than ritually invoked. This is a limitation that parallels the problem of superficial test coverage in software engineering: a test that always passes may be testing nothing.

We do not address the question of who writes the constitution. The engineering benefits of constitutional governance depend on the quality of the constitution itself. A poorly written constitution—one that is vague, contradictory, or poorly adapted to the system's domain—would produce constitutional governance that is worse than no governance at all. The quality of the specification determines the quality of the system, in constitutional governance as in TDD.

8 Conclusion

Constitutional governance in autonomous systems is not merely an ethical requirement—it is an engineering strategy. The analogy to test-driven development is structural, not metaphorical: a constitution specifies correct behavior before the system attempts to produce it, enables early defect detection, prevents regression, documents intent, and provides the auditability foundation for graduated trust.

The operational evidence—3,697 decision narratives with constitutional citations, 307 deliberation sessions with a 20.2% failure rate demonstrating discriminative power, and zero irreversible decisions without human confirmation—demonstrates that this approach produces measurable governance properties in practice. The failing tests are as important as the passing ones: a constitutional specification that never rejects a decision is either trivially permissive or not being applied.

Ethics before capability is not a constraint on engineering. It is engineering.

References

[1] Beck, K. (2003). Test-Driven Development: By Example. Addison-Wesley.

[2] Bai, Y., Kadavath, S., et al. (2022). Constitutional AI: Harmlessness from AI Feedback. arXiv:2212.08073.

[3] Ouyang, L., Wu, J., et al. (2022). Training Language Models to Follow Instructions with Human Feedback. NeurIPS 2022.

[4] Christiano, P., Leike, J., et al. (2017). Deep Reinforcement Learning from Human Preferences. NeurIPS 2017.

[5] Meyer, B. (1992). Applying Design by Contract. IEEE Computer, 25(10), 40–51.

[6] Lamport, L. (2002). Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers. Addison-Wesley.

[7] Floridi, L. & Cowls, J. (2019). A Unified Framework of Five Principles for AI in Society. Harvard Data Science Review, 1(1).

[8] European Parliament and Council. (2024). Regulation (EU) 2024/1689 (AI Act). Official Journal of the European Union.

[9] Amodei, D., Olah, C., et al. (2016). Concrete Problems in AI Safety. arXiv:1606.06565.

[10] Rafailov, R., Sharma, A., et al. (2023). Direct Preference Optimization. NeurIPS 2023.

[11] Ioannidis, J. P. A. (2005). Why Most Published Research Findings Are False. PLOS Medicine, 2(8), e124.