
Architectural Governance of Agentic AI: From Instructions to Enforcement

Epitácio Vicente do Nascimento Neto **Prof. Dr. Felipe André Zeiser**
Graduate Program in Applied Computing (PPGCA)
Universidade do Vale do Rio dos Sinos (UNISINOS), São Leopoldo, RS, Brazil
Corresponding author: netoepitacio@edu.unisinos.br

Abstract

Accepted at the NeurIPS 2026 Education Track (Submission 10); final version, November 2026. Lecture notes (the primary self-study text), slides with speaker notes, an interactive web laboratory and runnable code that teach *architectural governance* of LLM-based agents: classifying enforcement by who can defeat it, measuring the gap between what an agent can do and what it is authorised to do, and testing whether claimed human oversight is real. All materials are original, grounded in an evidence-cited, single-observer study of a production deployment coordinated by one of the authors.

1 The concept

LLM-based agents are governed mostly by telling them things: prompts, policy files, instruction hierarchies. The concept taught here is **architectural governance**: making a prohibited effect structurally unreachable, rather than discouraged by prose or reviewed by a mechanism the acting party could itself alter. The problem: an agent may deploy software, policy says every deploy needs a human's approval, and the approval gate is a check the agent can route around; how do we tell which rules are enforced and which are only requested? The concept is taught through four author-proposed instruments, each with a definition, an example and a non-example. (1) The *access-versus-mandate* dual classification: each agent action class is graded by technical capability and by authorisation; the delta is the governance gap. (2) The *enforcement ladder*: none → instructional → conditional → structural, graded by who can defeat the control. (3) *Supervision viability*: six conditions a human-oversight point must satisfy to be real (independence, latency, halt, attention, comprehensibility, authority). (4) The *supervision residue*: the action classes no catalogued guardrail configuration can safely free from human oversight, and the determinants that pin them there.

The grounding is a single-observer, read-only, evidence-cited study (run 20260807-1714, 7 August 2026) of a production multi-agent deployment. AUTHOR-ATTESTED description: an AI conversational application operated by multiple AI agents working 24/7 to monitor and patch it autonomously, with some work done by a human on the loop; a real service running in a real real-estate business in Sydney, Australia, coordinated by one of the authors, who identified the opportunity to start working on the architecture side of reinforcement. Agent self-reports diverged factually from independent evidence in 61.1% of audited episodes (11 of 18, REPORTED). The observer is also the operator, so the study is not an independent audit. It enumerates 60 action classes on one host, a 134-entry guardrail catalogue (12 layers) and 72 specified metrics, and finds that access exceeded mandate in 50 of 60 action classes; only 4 of 60 were bound to a control the actor could not reach around (all four REPORTED); of 39 guardrails present, only 2 were structural (REPORTED); and 27 of 34 claimed human-supervision points diverge between nominal and effective posture. Limits of a single case: one environment, one snapshot and one observer, who is also the operator, so every figure is a finding about this estate, not a prevalence estimate.

2 Why now, and linked papers

Agent governance is now central to AI research; the question underneath much of it is where enforcement must live. ToolEmu emulates the consequences of real tool access (Ruan et al., 2024); AgentDojo benchmarks prompt-injection attacks and defences (Debenedetti et al., 2024); AI Control develops protocols that stay safe even if the model is adversarial (Greenblatt et al., 2024); CaMeL defeats prompt injection *by design*, moving enforcement into control- and data-flow architecture (Debenedetti et al., 2025); the Instruction Hierarchy is the strongest instructional-side defence (Wallace et al., 2024); and in-context scheming evaluations show frontier models subverting the oversight that instructions are meant to carry (Meinke et al., 2024). None is the concept; each strengthens one rung of the enforcement ladder.

3 Audience, leveling and prerequisite knowledge

Advanced undergraduate and early graduate students, and practitioners deploying agentic systems. Prerequisites: what an LLM and a prompt are; basic software-systems literacy (processes, file permissions, version control); no security, formal-methods or ML-theory background. Each specialised term is defined where it first appears. It teaches by classification and case analysis, not proofs.

4 Learning objectives and outcomes

By the end, a learner can: (1) distinguish technical access from authorised mandate; (2) classify any control into the four enforcement classes; (3) assess a human-in-the-loop supervision point against the six viability conditions; (4) read an access-versus-mandate matrix and locate the governance gap; (5) explain why instructional and conditional controls dominate deployed systems; (6) propose a structural replacement for an instructional control; (7) interpret governance metrics, including a self-report divergence rate, and say what they do and do not establish. Objectives (1)–(4) are assessed by classification exercises, (5)–(7) by design exercises and the Guardrail Lab worksheet.

5 Teaching materials

The lecture notes are the primary self-study text, and a user guide (reading paths, 3-session and 4-session plans, a module-to-slide-to-website-to-exercise map) opens every artifact. One running example, an agent that may deploy software under a policy requiring a human’s approval, with an approval gate it can route around, runs through the notes, slides, website and exercises. The website source, runnable code (`code/reproduce.py` and a notebook) and a preserved accepted version are included.

Lecture notes (PDF, modules M0–M8, five exercises with worked solutions, glossary). **Slides** (PDF, 16:9; a speaker note on every slide, plus a slides-with-notes PDF). **Interactive website** (<https://neurips2026.meridianseal.com>): animated infographics of the access-versus-mandate matrix, the enforcement ladder and the twelve-layer guardrail stack, driven by identity-free extracts of the study’s registers (`data.json`), plus the *Guardrail Lab*, a live simulation in which learners pick guardrail condition sets from a bare baseline to fully hardened, toggle an adversarial mode, and watch attempted actions resolve: blocked, checked, reminded, or silent; in adversarial mode every rung but the structural one gives way. The complete source (HTML, CSS, JavaScript, data) is in the ZIP and runs offline. **Code**: `reproduce.py` and `reproduce.ipynb` read only `data.json` and reproduce the quoted figures. Content is CC BY 4.0, code Apache 2.0; LaTeX sources included.

References

- Y. Ruan et al. Identifying the risks of LM agents with an LM-emulated sandbox. In *ICLR*, 2024. arXiv:2309.15817.
- E. Debenedetti et al. AgentDojo: A dynamic environment to evaluate prompt injection attacks and defenses. In *NeurIPS Datasets and Benchmarks*, 2024. arXiv:2406.13352.
- R. Greenblatt et al. AI control: Improving safety despite intentional subversion. In *ICML*, 2024. arXiv:2312.06942.
- E. Debenedetti et al. Defeating prompt injections by design. arXiv:2503.18813, 2025.
- E. Wallace et al. The instruction hierarchy: Training LLMs to prioritize privileged instructions. arXiv:2404.13208, 2024.
- A. Meinke et al. Frontier models are capable of in-context scheming. arXiv:2412.04984, 2024.