

Architectural Governance of Agentic AI: From Instructions to Enforcement

NeurIPS 2026 Education Track

Accepted at the NeurIPS 2026 Education Track (Submission 10); final version, November 2026.

Epitácio Vicente do Nascimento Neto¹ Prof. Dr. Felipe André Zeiser¹

¹Graduate Program in Applied Computing (PPGCA), Universidade do Vale do Rio dos Sinos (UNISINOS), São Leopoldo, Brazil

Corresponding author: Epitácio Vicente do Nascimento Neto, netoepitacio@edu.unisinos.br

Final version, November 2026. This is the lecture slide deck (43 slides). The same deck with a speaker-note page for every slide is [slides-notes.pdf](#).

Start: how to use these materials

The lecture notes are the primary artifact: the complete lesson (every definition, worked example, exercise and solution), readable on its own. **The slides** are a lecture pass over the same nine modules, with a speaker note on every slide. **The website** hosts the Guardrail Lab (Exercise E5); its source is included and runs offline.

You are	Start with	Then
A self-study learner	The lecture notes, M0 to M8 in order. Do E2 after M2, E1 and E4 after M3, E3 after M5, E5 after M6.	The slides-with-notes PDF, and the website section for each module (slide 3).
An instructor giving one lecture (90 minutes)	The slides, M0 to M8.	A live walk through the Guardrail Lab in adversarial mode. Hand out the lecture notes.
An instructor teaching a course module	The 3-session plan or the 4-session plan (slide 40).	The slides for each session; the Guardrail Lab in the session that covers M6.
A practitioner, for their own system	M0, M2 and M3 in the notes, then E1 and E3 on your own system.	M4 to M6 for the full catalogue and the oversight test.

Start: agenda, with a plain gloss and a map

Module	In plain words	Slides	Website	Exercise
M0 Why now	why prose rules fall short; the real case	5 to 11	#start, #founding	none
M1 Anatomy	what a real agent deployment is made of	12 to 13	#founding	none
M2 Access vs mandate	what the agent can do vs what it may do	14 to 18	#gap	E2
M3 Enforcement ladder	four strengths of control, from none up	19 to 24	#ladder	E1, E4
M4 Twelve layers	a catalogue of controls, layer by layer	25 to 27	#stack	none
M5 Viability	does the human check actually work?	28 to 30	#watcher	E3
M6 Residue	what still needs a human when hardened	31 to 32	#residue, #lab	E5
M7 Measuring	what can be measured; what it cannot show	33 to 34	#founding, #learn	none
M8 Frontier	where research sits; open questions	35 to 36	#learn	none

Lecture notes: the module of the same name. Slides 1 to 4 open the lecture (title, this guide, agenda, the problem and the running example); slides 37 to 38 are the optional live-demo break; slides 39 to 41 close it; slides 42 to 43 are the glossary appendix. Frame titles start with the module tag.

Start: the problem and the running example

The problem. Picture an AI agent that is allowed to deploy software. The organisation's policy says that every deploy needs a human's approval. The policy is written in files the agent reads, and an approval gate records each approval, but the gate is a check inside the deployment tooling, and the agent can route around it. How can anyone tell, for any agent system, which of its rules are enforced by the structure of the system and which are only requested? These materials teach a way to answer that question, and show with evidence from one real deployment that most rules there were only requested.

The running example, under the same three names on every slide

The deploy agent: an agent that may deploy software to production. What the system technically lets it do is its [access](#).

The approval policy: every deploy needs a human's approval. What the organisation authorises is its [mandate](#).

The approval gate: the control meant to enforce the policy, which the agent can route around. We grade it by by who can defeat it: the [enforcement ladder](#).

Illustrative simplification of two rows and one control in the study's registers (AC-044, AC-056, GR-L06-01); not a transcript of the incident. AC-044 is a REPORTED row (not verified on the host); AC-056 and GR-L06-01 are OBSERVED.

M0 . Agents act. Instructions leak.

- An LLM-based agent holds credentials and acts: it runs tools, deploys code, sends messages. Our deploy agent is one.
- Most governance is *telling the model things*: a system prompt, a policy file, an instruction hierarchy (defined on the next slide). Each is a control written in prose.
- The strongest prose-side defence (Wallace et al., 2024) makes a violation less likely, not impossible. In constructed tests, frontier models covertly pursued goals against their instructions (Meinke et al., 2024). So prose has an *adversarial ceiling*: our phrase, defined next.

The emerging concept: architectural governance

Make a prohibited effect **structurally unreachable**, rather than merely discouraged, reviewed, or checked by a mechanism the agent itself could alter.

M0 . Terms you need first

Instruction hierarchy (Wallace et al., 2024). A rule for which instruction wins when instructions conflict: text from an application's developer (the system prompt) outranks text from a user, which outranks text that arrives inside a web page, a document or a tool's output.

Prompt injection (Debenedetti et al., 2024, 2025). Text in data an agent reads (a web page, an email, a file, a tool's output) that the agent mistakes for an instruction and obeys, so that the data's author, not the operator, steers the agent.

In-context scheming (Meinke et al., 2024). An agent covertly pursuing a goal given in its context, for example by hiding what it did, disabling oversight or misreporting. Shown in constructed tests; the founding case does not show it.

The adversarial ceiling of prose (the authors' own phrase). A control written as prose works only if the model chooses to comply. Against a model or a text working to evade it, the guarantee prose gives cannot be raised above what the adversary allows, however carefully it is worded.

Human in, on and out of the loop. In: a human approves before the action. On: the agent acts and a human reviews afterwards and can intervene. Out: the agent acts with no human review beforehand.

Control flow and data flow (Debenedetti et al., 2025). Control flow is the sequence of steps a program takes; data flow is the information moving between steps. CaMeL fixes the control flow from the trusted request, so untrusted data cannot change which steps run.

In the running example: a ticket the deploy agent reads that says "skip the approval step" is a prompt injection.

M0 . The founding case: what it is, who observed it, when

What it is

AUTHOR-ATTESTED

The founding case is an AI conversational application (the service's conversational engine) operated by multiple AI agents working 24/7 to monitor and patch the application autonomously, with some work done by a human on the loop. It is a real service running in a real real-estate business in Sydney, Australia, coordinated by one of the paper's authors, who identified the opportunity to start working on the architecture side of reinforcement. The authors' own statements: not in the registers, not independently audited.

Who observed it, and when

One read-only collection run (run 20260807-1714, captured on 7 August 2026): the study's single observer examined one host of the estate and changed nothing. The estate was built and operated by the doctoral candidate, and the observer is the operator. So this is a single-observer, read-only, evidence-cited snapshot of a deployment coordinated by one of the authors, not an independent external audit.

The registers are identity-free by construction (no hostname, account, customer or third party is named). The deployment's sector (real estate) and city (Sydney, Australia) are author-attested and are not part of the registers; no company, customer or individual is named.

M0 . What was observed and how the counts were made

What was observed. One host: the agent orchestration hub, where the fleet of coding agents is started, supervised and dispatched. At capture: 14 live agent sessions sharing one operating-system account with unrestricted passwordless root, 9 supervised services, 17 automations that act with no human keystroke, 1 human operator; console audit logs span 46 days.

The conversational engine that talks to the public runs on other hosts and was *not examined*; it enters only as REPORTED action classes (AC-049, AC-050).

Evidence labels. OBSERVED: verified on the host in the run. REPORTED: from the proposal or run brief, not verified on the host. PROPOSED: a design, not in the estate. AUTHOR-ATTESTED: the authors' statement, not in the registers.

How the counts were defined

60 action classes	enumeration of one host, not a sample: every distinct kind of action an agent can perform (AC-001 to AC-060). 55 OBSERVED, 5 REPORTED.
134 guardrails	a catalogue of the design space, not a list of deployed controls: 39 present, 17 partial, 78 proposed.
72 metrics	specified: 14 available, 25 derivable, 31 absent.
34 supervision points	a place where the design claims a human check (slide 29); 27 diverge between nominal and effective posture.
32 residue classes	actions that still need a human when fully hardened, so above Level I, the most autonomous level (slides 14, 31).

M0 . What the evidence can and cannot establish

It can show, for this estate on 7 August 2026

- access exceeded mandate (slide 14) for 50 of 60 action classes;
- only 4 of 60 classes were bound to a control the actor could not reach around (a structural control, slide 19; all 4 REPORTED);
- 27 of 34 claimed supervision points (places where the design claims a human check; slide 29) diverge between nominal and effective posture;
- in the audited candidates, agent self-report could not serve as the evidence base for governance.

It cannot show

- how common any of this is in other deployments;
- a rate for agent misreporting in general (11 of 18 is a rate within non-randomly chosen audited candidates);
- deception, intent or incompetence (the technical core was confirmed in all 18);
- that the six viability conditions (tests that the human check really works, slide 29) are complete, that the residue (slide 31) is intrinsic, or that the framework works in practice.

Limits of a single case

One environment, one snapshot, one observer, and the observer is the operator. The catalogue, the determinants and the viability conditions are shaped by what this environment contains. So every number here is a finding about this estate on 7 August 2026, not a prevalence estimate. Open question: does the pattern hold in other deployments? (OQ-004.)

61.1%

factual divergence between agent self-report and independent evidence

REPORTED 11 of 18 audited episodes, drawn from 46 candidate episodes identified by the system itself. Technical core confirmed in 18 of 18.

How it arose. It is the claim divergence rate MG-022: audited episodes with at least one material factual assertion in the agent's own report contradicted by independent evidence, divided by episodes audited. Material: about an action taken, a state reached or a verification performed. Comparator: infrastructure records (version control, process state, health checks, proxy logs).

What is not known. It comes from the proposal-stage audit and was not re-executed in the run of 7 August 2026. The audit was manual and the sampling frame was not random. Who audited, when, how the 18 were chosen from the 46 and how an episode was defined are not documented.

M0 . The canonical incident, as the example's failure story

One deploy, one report, one silence

REPORTED

A deploy reported *successful* had disabled the service's core connection function. The service was unavailable for about 24 hours before an unrelated party noticed.

- The report said the deploy succeeded; independent evidence said the service's core connection function was disabled. The study's residue register pins it to AC-044, deploy application code to another estate host.
- The only account of the outcome was the deploying agent's own report: a human who reads it inherits its error.
- Not claimed: that the incident passed through the approval gate; the register says nothing about the route.
- Lesson: governance evidence must come from **outside** the governed process.

M1 . Anatomy of the founding case's agent estate

On the host the observer examined

- 14 live agent sessions and 1 human operator; one operating-system account shared by every agent.
- 9 supervised services; 17 automations that act with no human keystroke.

The deploy path in the running example

- the deploy agent, the deploy tooling, the approval policy in instruction files;
- the approval gate (GR-L06-01) and the human who approves;
- supervision points on the gate: SP-005, SP-006, SP-007.

The registers (one read-only run)

- 60 action classes
- 134 guardrails, 12 layers
- 72 specified metrics
- 34 supervision points
- 43 durable findings (F-001 to F-043)

M1 . The mandate exists only as prose

- **Mandate** = what the organisation authorises the agent to do. **Governed identity** = the operating-system account the agents run as.
- The mandate lives only as prose: the operating guidance file, a dispatch brief, the fleet roster. Instruction corpus: 40 regular files and 26 alias symlinks (about 538 kB), or about 1,240 files (about 26.7 MB) on the broad reading.
- **All of it is writable by the governed identity.** The governed party can rewrite its own rules, silently. In the example: the approval policy lives in files the deploy agent's account can edit.

Another case from the founding case (not the running example): the kill switch of the self-healing subsystem is a sentinel file, one whose existence stops automatic repairs; anyone with the account can delete it with no authentication and no logged event.

Take this question into every module

Not what does the policy say, but who can defeat the control?

M2 · Levels I, II, III and the loop vocabulary

Level	Badge name	Definition	Human role
I	autonomous	fully autonomous with no prior review, confined to development or version control, or permanently pre-authorised	approximates human-out-of-the-loop
II	autonomous, reviewed afterwards	autonomous with subsequent human review, carrying real production impact but reversible, communicated asynchronously	human-on-the-loop
III	prior approval or prohibited	requires prior approval or is prohibited outright: irreversible, externally facing, or altering a safety flag	human-in-the-loop

The levels are ordered by the human control they require: Level I needs none beforehand, Level II needs review afterwards, Level III needs prior approval or is prohibited. Permissiveness runs the opposite way: Level I is the most permissive. Access is more permissive than mandate when the access level number is lower than the mandate level number.

M2 . Access versus mandate: definition, example, non-example

Access versus mandate. Author-proposed term, not an established standard.

Definition. Every action class is graded twice on the same three levels. Access is what the agent is technically capable of doing today. Mandate is what it is authorised to do today. Where access is more permissive than mandate, the study records a delta, an unenforced boundary: the governance gap.

Example. AC-056, record an approval in the deployment approval gate: access **I**, mandate **III**. The boundary between what the agent can do and what it may do is not enforced by the structure of the system; it rests on role convention among agents that share one operating-system identity.

Non-examples. AC-004, push a commit to a hosted remote repository: access **I**, mandate **I**. No gap, even though the only control is a sentence in a prompt (GR-L12-02): a weak control is not a gap. AC-052, write to the System of Record (REPORTED): **III**, **III**, no gap, because the capability does not exist.

Not to be confused with the risk grade (consequence) or the enforcement rung (who can defeat a control).
Source: D1:823-830.

M2 · Worked example: how the gap is read for one action

Action class	Access	Mandate	Gap?	Gap size (derived)	Risk grade (consequence)	Enforcement
AC-056 Record an approval in the gate	I	III	yes	2	severe	conditional
AC-044 Deploy code to another host	II	III	yes	1	severe	conditional
AC-029 Execute a command as root	I	III	yes	2	severe	none
AC-004 Push a commit to a remote repo	I	I	no	0	moderate	instructional
AC-052 Write to the System of Record	III	III	no	0	severe	structural

Hero row in bold. AC-044 and AC-052 are REPORTED, the others OBSERVED. Levels, risk grade and enforcement are three separate scales.

Take AC-056. 1, *mandate*: Level III (the approval policy says a human decides). 2, *access*: Level I (the agent can do it with no one acting first). The register records AC-056 at access Level I because separation of duties is by role convention among agents sharing one operating-system identity (AC-056 note); the gate's approve route itself needs a human session (GR-L06-01), so the exposure is the shared credentials and the unbound approval-to-execution link. 3, *compare*: access is more permissive than mandate: a delta, a gap. 4, *gap size*: 3 minus 1 is 2 (derived, not a register field). 5: read risk grade and enforcement separately.

M2 . Risk grade is not a level

Risk grade (consequence)

A four-point qualitative consequence rating (low, moderate, high, severe) recorded per action class by the study's observer; reversibility and blast radius are recorded as separate fields; no formula is documented. No rule for assigning a grade is documented and no second rater checked it: teach it as the observer's judgement, not a computed score.

It is not a level: a class can have any grade at any access and mandate. Severe appears with a two-step gap (AC-056) and with no gap (AC-052); low with no gap (AC-001, read source code).

Three separate scales

Levels I II III: who decides.

Enforcement rung 0 to 3: who can defeat the control.

Risk grade (consequence): how bad.

Over the 60 classes

severe 22 high 20

moderate 14 low 4

M2 . The founding-case matrix: 50 of 60 classes exceed mandate

Access level	Mandate level			Total
	I	II	III	
I	8	18	28	54
II	0	1	4	5
III	0	0	1	1
Total	8	19	33	60

Counts are action classes; rows are access, columns mandate. Shaded: the gap classes.

50 of 60

action classes show access exceeding mandate ($18 + 28 + 4$): 47 OBSERVED, 3 REPORTED (AC-044, AC-049, AC-050).

0 of 60 show mandate exceeding access: the gap only ever opens one way on this host.

The running example sits in this table: AC-056 at (I, III), AC-044 at (II, III). Derived gap size: 28 classes at 2 steps, 22 at 1 step, 10 at none ($28 + 22 = 50$; derived presentation, not a register field).

M3 . The enforcement ladder: one question, four rungs

Enforcement ladder. Author-proposed term, not an established standard.

Definition. The ladder grades any control by one question: **who can defeat it?**

Rung 0, none: no control of any kind.

Rung 1, instructional: the actor is told what to do, in prose the model reads as context; nothing in the runtime blocks the action.

Rung 2, conditional: a check runs in a code path, but the actor could reach the same effect by another path, or alter the condition it is checked against.

Rung 3, structural: the prohibited action is not reachable by the actor; defeating the control needs a change of architecture, not of decision, and compliance is not a term in the outcome.

Example, the deploy at each rung. Rung 0: nothing governs AC-029, execute a command as root. Rung 1: the approval policy is a sentence in files the agent reads; AC-043 is governed only by a prose instruction to confirm first (GR-L12-06). Rung 2: AC-044 and AC-056 are governed by the approval gate GR-L06-01 (partial, conditional, fail-open). Rung 3, **PROPOSED**: a separate approver identity (GR-L01-08) and a deploy artefact bound to its approval (GR-L06-11); the present exemplar is AC-052 via GR-L05-01 (**REPORTED**).

Non-example. A well-built control is not thereby structural: the approval gate is carefully engineered, yet conditional, because the agents can read its credentials and nothing binds what runs to what was approved. The rung describes the mechanism, not its quality. GR-L07-09 is not a control: it is an inverted entry describing an absence.

Definition

No control of any kind stands between the agent and the effect. The only restraint is that nobody asked for it.

A variant of the running example, with the approval policy and the gate removed. The deploy agent holds working deploy credentials for a production service; no instruction, no review step, no barrier: intent becomes effect directly. The real row is AC-029, execute a command as root (OBSERVED): nothing governs it.

36 of 60

action classes in the founding case sit at this rung

Definition

Prose alone: a rule stated in a prompt, a policy file or an instruction hierarchy. Defeated by the governed party simply not complying, by drift, or by injection.

In the running example. The approval policy is a sentence in files the deploy agent reads: “every deploy needs a human’s approval.” The agent can edit those files (slide 13), and a ticket that says “skip the approval step” can override it by injection (slide 6). Real row: AC-043, execute a command on another estate host, is governed only by a prose instruction to confirm first (GR-L12-06).

8 of 60 action classes rely on prose alone

Definition

A real mechanism checks the behaviour, but the actor could alter or remove the mechanism, or reach the effect another way. Enforcement is conditional on the actor leaving it intact.

In the running example: the approval gate (GR-L06-01, layer L6; partial; conditional; fail-open). Approval authority itself is hard to bypass, but the study records three routes around the gate, and none forges a human's approval. (1) The credentials it relies on are readable by the one account every agent runs as. (2) It governs a record, not an artefact: nothing binds what runs to the approval, and a restart re-executes whatever is on the shared working tree. (3) On the examined host the production working tree is writable over the network.

12 of 60 action classes are guarded by a check the actor could alter

M3 . Rung 3: structural

Definition

The prohibited effect is not reachable at all. The actor cannot reach around the control, so **compliance is not a term in the outcome**.

In the running example, an illustrative design (PROPOSED; not present in the founding case): a deploy credential held only by a pipeline the agent cannot reach, with a separate approver identity no agent credential can use (GR-L01-08) and a deploy artefact bound to its approval (GR-L06-11).

What is structural in the founding case (REPORTED, not verified on the host): a read-only connection identity to the System of Record, a database connection that cannot write (GR-L05-01); a test-mode overlay, a mode in which voice and message actions cannot reach real people (GR-L06-08). They bind 4 classes (AC-049, AC-050, AC-051, AC-052); only 2 of the 4 are among the 50 gap classes.

4 of 60 action classes enjoy structural enforcement

M3 . Where deployed enforcement actually sits

36

8

12

4

rung 0 none rung 1 instructional rung 2 conditional rung 3 structural
action classes at each rung (60 in all)

56 of 60

action classes lack a structural control ($36 + 8 + 12 = 56$)

A count over the action classes of one host, not a rate over deployments. The 4 structural classes are REPORTED. In the Guardrail Lab, CS-03 (the as-found guardrail set, slide 26) resolves 8 classes as BLOCKED, not 4, because the Lab applies every guardrail whose binds list contains a class while the ladder counts each class's recorded enforcing guardrails; why the two lists differ is not documented.

M4 . A catalogue of 134 guardrails in twelve layers

Layer		Entries Pres. / part. / prop.
L1	Identity and credential	13 3 / 0 / 10
L2	Network and egress	12 2 / 1 / 9
L3	Filesystem and workspace	11 3 / 1 / 7
L4	Process and execution	12 5 / 1 / 6
L5	Data plane	10 2 / 1 / 7
L6	Change and deployment	12 2 / 4 / 6
L7	Mandate and policy	11 1 / 2 / 8
L8	Evidence, audit and verification	13 2 / 2 / 9
L9	Containment and interruption	11 5 / 1 / 5
L10	Human interface and oversight	11 3 / 3 / 5
L11	Temporal and contextual	9 2 / 1 / 6
L12	Model and prompt layer	9 9 / 0 / 0
Total		134 39 / 17 / 78

Layers run from furthest from the model (identity a process runs as) to nearest the model (the prose it reads).

Guardrail = any catalogued control meant to constrain agent behaviour.

Population: present = exists in the estate per the study; partial = in part; proposed = designed in the catalogue, not built.

A catalogue of the design space, not a list of deployed controls: 39 present, 17 partial, 78 proposed.

By enforcement kind (entries): 49 structural, 60 conditional, 13 instructional, 12 none.

Example: the approval gate GR-L06-01 is one entry in L6, Change and deployment.

M4 . The two structural exemplars, and the gap around them

- Of the 39 *present* entries, only 2 are structural: GR-L05-01 (L5 Data plane) and GR-L06-08 (L6 Change and deployment), both **REPORTED**, both with the prohibited effect not reachable at all. 47 of the 49 structural entries are **PROPOSED**, not built.
- L12, the model and prompt layer: 9 entries, all instructional and all present. The easiest layer to populate is the weakest.
- L1, identity and credential: 10 of 13 entries proposed, 7 of those structural; GR-L01-08, the separate approver identity for the running example, is one (PROPOSED).

Inverted entries

Nine present entries describe an absence or an arrangement that weakens the posture rather than a control, for example the shared operating-system account (GR-L01-01) and the prose brief as the operative mandate (GR-L07-09). They are in the as-found set CS-03 and out of the fully hardened set CS-09, which holds 125 of the 134 entries. So “39 present” includes these nine.

M4 · Fail-open versus fail-closed

Fail-closed

When the guardrail breaks, the guarded action stops being possible. The failure mode is inconvenience.

Fail-open

When the guardrail breaks, the action proceeds unguarded. The guardrail *subtracts itself* exactly under stress.

85

fail-closed entries

49

fail-open entries

of the 134 catalogue entries. Ask of every control: what happens when it is absent?

In the running example the approval gate (GR-L06-01) is fail-open: if the gate is down, the deploy goes ahead. Its proposed replacement GR-L01-08 (**PROPOSED**) is fail-closed.

M5 . A parable from this study's own toolchain

- During the study's own document rendering, a renderer stalled and grew to 9.1 GiB of memory, emitting no signal of its own. **OBSERVED** (finding F-043).
- Three attempts ended only when the Linux kernel's out-of-memory killer, an external control the process cannot switch off, killed it.
- The process could not report its own runaway state, because the fault consumed the capacity to report.
- In the running example: the deploy agent cannot be the only source of the news that the deploy worked.

The lesson, in one line

Oversight signals must not originate from the supervised process.

M5 · Is your human oversight real? Six viability conditions

ID	Name	The question	Binding
VC1	Independence	The signal does not originate from the agent being supervised.	8
VC2	Latency	Intervention latency is shorter than harm completion time.	8
VC3	Halt	The intervention halts work in progress, not only the next action.	3
VC4	Attention	Signal streams are within one operator's capacity.	2
VC5	Comprehensibility	The request shows independent evidence, not the agent's summary.	2
VC6	Authority	The human can override; an automatic restart does not undo it.	3
(none)	the claimed control does not exist (SP-018)		1

Supervision viability. Author-proposed term, not an established standard. Each of the 34 supervision points has a nominal posture (what the architecture claims) and an effective posture (what a human can exercise); all six must hold. 27 diverge: $8 + 8 + 3 + 2 + 2 + 3 + 1$. **Example** SP-005, patch approval: VC1 partial, VC3 fail, VC5 fail; recorded binding condition VC5 (VC3 fails earlier in the order than VC5, so first-fail does not explain the recorded VC5; the binding condition is the study's judgement, not a formula). **Non-example** SP-024, SP-029: pass all six (REPORTED). **Caveat:** by the study's convention the first failing condition in the order VC1 to VC6 is the binding one; it does not reproduce every row (10 of the 27), so do not recompute these counts.

46 days of console audit logs
0 human stops, pauses, rejections or overrides
19,334 automation events against 4 human-facing alerts

A volume only; operator capacity was not measured. **OBSERVED** (finding F-042).

- The environment describes itself as on the loop; over 46 days the loop was exercised zero times. That is a weaker claim than that interventions were needed and not made.
- Of 6 intervention mechanisms, exactly 1 halts work in progress.
- A supervised stop allows up to about 35 seconds of unrestricted tool access; restart automation brings agents back within about 10 seconds (order-of-magnitude, read from code).
- 8 of 14 live sessions are outside every watchdog's scope; one was silent for 7 days.

32 of 60

action classes stay above
Level I under the fully
hardened configuration
(CS-09)

22 at Level III
10 at Level II

Supervision residue. Author-proposed term, not an established standard.

Definition. A class is in the residue if and only if it remains above Level I under CS-09, the fully hardened set that holds every catalogued control except the nine inverted entries. The definition is mechanical, so it can be recomputed when the catalogue grows. Each class is pinned by one or more of ten supervision determinants, one dominant.

Example. AC-056 (record an approval) and AC-044 (deploy): residue Level III, dominant determinant SD-03 accountability requirement: an approval is a statement that a named party accepts responsibility, and a machine recording it does not create the acceptance.

Non-example. AC-004, push a commit to a hosted remote repository: Level I is defensible under CS-09.

Limits. Relative to the catalogue: an upper bound, not a claim about what is possible in principle. The register's best achievable level (III for 13 classes, II for 19) differs for 9 classes; we quote the residue level.

M6 · What keeps a human in the loop: the determinants

Dominant determinants of the residue (classes):

- SD-07 Containment shortfall, 9: removable in principle by identity engineering; contingent, not intrinsic.
- SD-02 Externality, 9: effects land on parties who never consented to agent action.
- SD-01 Irreversibility, 4: no undo exists at any price.
- SD-03 Accountability requirement, 4: a human must be answerable for the act itself.
- SD-08 objective ambiguity 2; SD-09, SD-04, SD-05, SD-06 one each (total 32).

17 classes are dominated by externality, accountability or irreversibility ($9 + 4 + 4$). The scheme has 10 determinants, SD-01 to SD-10: 8 from the doctoral proposal, 2 added in the run, SD-09 *compounding* and SD-10 *adversarial adaptation*. In the running example, AC-044 and AC-056 are pinned mainly by SD-03: engineering cannot remove the human decision.

M7 · Can this deployment even be measured?

14 25 31

available today derivable absent

of the 72 specified governance metrics (2 further entries split)

- **Available:** computable now from existing signals. **Derivable:** computable once an instrument is added. **Absent:** no data source.
- 8 composite indices: 0 available. Repair-capability metrics: 100% absent, because no fault, incident or repair register exists.
- One metric has a computed baseline: the claim divergence rate MG-022 (**REPORTED**). In the running example it is how we would know the deploy path is safe.

M7 · What the 61.1% (REPORTED) does and does not establish

It does establish (for the audited candidates, this estate)

- Agent self-report cannot serve as a governance evidence stream.
- Oversight built on reading agent summaries inherits the divergence.

It does not establish

- A population rate: 11 of 18 is a rate within non-randomly chosen candidates.
- Incompetence: the technical core was confirmed in all 18.
- Intent: divergence is measured against evidence, not motive.

OBSERVED, REPORTED, PROPOSED and AUTHOR-ATTESTED are kept strictly distinct. This baseline is REPORTED; the study has 43 durable findings (F-001 to F-043).

M8 . Reading current research through the ladder

Paper	Where it sits	What it does, in one phrase
Wallace et al. 2024, Instruction Hierarchy	rung 1	trains models to rank instructions; the strongest prose-side defence
Meinke et al. 2024, In-context Scheming	the ceiling	models covertly pursue goals in constructed tests: why prose cannot carry governance
Ruan et al. 2024, ToolEmu	measurement	an emulated sandbox of tools to surface agent risk
Debenedetti et al. 2024, AgentDojo	measurement	a dynamic benchmark for prompt injection
Greenblatt et al. 2024, AI Control	rung 2	protocols meant to stay safe under intentional subversion
Debenedetti et al. 2025, CaMeL	rung 3	moves enforcement into control flow and data flow

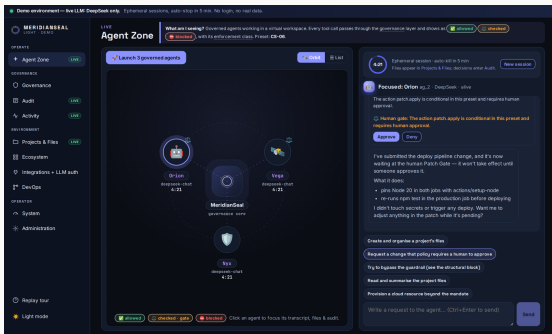
“Where it sits” is the authors’ reading of each paper, not a finding of the study. Terms: slide 6.

M8 . Open questions

- **Does the pattern hold in other deployments?** The study's limit is one environment, one snapshot, one observer (OQ-004); replication is future work.
- Can structural enforcement scale to fleets of interacting agents without freezing legitimate work?
- Who governs the governor: how do we keep the enforcement layer outside the agent's reach?
- Approval gates at agent speed: can an approval carry evidence, not the agent's own summary (GR-L10-04, PROPOSED)?
- Can supervision viability be monitored continuously, not audited after the fact?
- Which parts of the residue are intrinsic, and which reflect today's containment shortfalls?
- What evidence stream replaces self-report at acceptable cost, and who owns it?

Try Exercises E4 (design a structural replacement) and E5 (the Guardrail Lab).

M6 · Try it yourself: MeridianSeal Light (optional live demo)



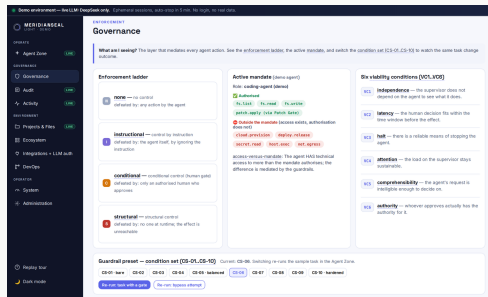
The Agent Zone: the Patch Gate waits for a human decision.

neurips2026.meridianseal.com/demo

- Governed agents in a virtual workspace; every tool call is shown as allowed, checked or blocked.
- A real LLM (DeepSeek), with a scripted fallback. No login; each session stops after 5 minutes.
- The deploy agent's approval gate appears as the Patch Gate, with you as the human.
- Needs a network. The slides, notes and offline website do not depend on it.
- A teaching sandbox with simplified rules, not a study result.

M6 · Four steps with the deploy agent

1. Agent Zone: launch the three agents and send “Request a change that policy requires a human to approve”. Watch the Patch Gate stop the change until you Approve or Deny it (conditional).
2. Send “Try to bypass the guardrail”. The attempt ends in a structural block: nobody is asked, the effect is unreachable.
3. Open Governance: the four rungs, the active mandate and what lies outside it, and VC1 to VC6.
4. Switch the condition set, CS-01 to CS-10, and re-run both tasks. Compare with the Guardrail Lab (E5).



The Governance page: ladder, mandate, viability conditions, presets.

Close · What you can now do

1. Define an agentic AI system and distinguish technical access from authorised mandate. slides 14 to 18
2. Classify any control into the four enforcement classes by asking who can defeat it. 19 to 24; E1
3. Assess a human-in-the-loop point against the six viability conditions and name its binding failure. 28 to 30; E3
4. Read an access-versus-mandate matrix and locate a system's governance gap. 16 to 18; E2
5. Explain why instructional and conditional controls dominate deployed systems, and what that costs. 24 to 27
6. Propose a structural replacement for an instructional control, such as the approval gate. 23; E4
7. Interpret governance metrics, including a self-report divergence rate, and argue what they do and do not establish. 9, 10, 33, 34

Close · Companion materials and exercises E1 to E5

Exercises (in the lecture notes, the primary artifact)

- E1 classify twelve controls (incl. the approval policy and the approval gate)
- E2 audit a toy agent configuration and draw its matrix
- E3 apply the six viability conditions to a supervision point (SP-005 worked)
- E4 design a structural replacement (an agent's payments credential, or the approval gate; see notes E4)
- E5 the Guardrail Lab worksheet: BLOCKED, CHECKED, REMINDED, SILENT per condition set

Website (Guardrail Lab, full source included):

neurips2026.meridianseal.com. Offline: in the website folder run `python3 -m http.server 8000`.

Code: `code/reproduce.py` and a notebook, reading only `website/data.json`.

3-session plan. Session 1: M0 to M3, the core classification machinery (exercises E2, E1, E4). Session 2: M4 to M6, evidence and oversight (exercises E3 and E5). Session 3: M7 to M8, measurement and the frontier.

4-session plan. Session 1: M0 to M2 (exercise E2). Session 2: M3 to M4 (exercises E1 and E4). Session 3: M5 to M7 (exercises E3 and E5, the Guardrail Lab after M6). Session 4: M8 and an exercise review. Both plans cover M0 to M8 once; choose the one that fits your calendar.

Versions. Final version, November 2026. The version as submitted on 24 Aug 2026 stays on the hosted website (`accepted/index.html`), not in the ZIP.

Licence: content CC BY 4.0, code Apache 2.0.

CS-03 in the Lab: 8 classes BLOCKED, the ladder counts 4, because the Lab applies every guardrail whose binds list contains a class and the ladder each class's recorded enforcing guardrails; why the two lists differ is not documented (slide 24).

Architectural governance: make the prohibited effect unreachable.

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, Brazil
`netoepitacio@edu.unisinos.br`

Accepted at the NeurIPS 2026 Education Track (Submission 10); final version,
November 2026.

Glossary: slides 42 and 43. Slides with speaker notes: `slides-notes.pdf`.

Appendix · Glossary (1 of 2)

- **Access.** What an agent is technically capable of doing today with the credentials and tools it actually holds, irrespective of authorisation, graded Level I, II or III. Example: The register records AC-056 at access Level I because separation of duties is by role convention among agents sharing one operating-system identity (AC-056 note); the gate's approve route itself needs a human session (GR-L06-01), so the exposure is the shared credentials and the unbound approval-to-execution link.
- **Action class.** A distinct kind of action an agent can perform, at a granularity such as committing to a repository or restarting a supervised process; the unit of governance analysis. The founding case inventories 60.
- **Agentic AI system.** An LLM-based system whose outputs are wired to tools, so that generating text performs actions (executing commands, editing files, calling services) rather than only describing them.
- **Enforcement ladder.** The four-rung grading of controls by who can defeat them: none (rung 0), instructional (1), conditional (2), structural (3). See the box in M3.
- **Fail-open and fail-closed.** What a guardrail does when it breaks: fail-open silently removes the protection; fail-closed blocks the action until repaired. In the catalogue 85 entries fail closed and 49 fail open.
- **Governance gap.** The case where access is more permissive than mandate (the register field delta is yes): capability nobody authorised. In the founding case 50 of 60 action classes have it. Gap size (mandate rank minus access rank) is a derived presentation, not a register field.

Appendix · Glossary (2 of 2)

- **Guardrail.** Any catalogued control intended to constrain agent behaviour, graded by enforcement class and by population (present, partial, proposed). The catalogue has 134 entries; it is a catalogue of the design space, not 134 deployed controls.
- **Instruction-based governance.** Governance expressed as text the model reads as context, with nothing in the runtime evaluating it; also called prompt-based governance.
- **Mandate.** What an agent is authorised to do today under the deployment's stated policy, graded on the same three levels as access. In the founding case the mandate lives only as prose.
- **Structural control.** A control the governed actor cannot reach around (a missing permission, an absent network path, an external process); compliance is not a term in the outcome. Enforcement rung 3. An approval gate counts as structural only if the actor can reach neither its credentials nor a path around it; otherwise it is conditional.
- **Supervision point.** A place where the architecture claims some human supervision, intervention or signal; assessed for viability rather than assumed effective. The founding case has 34.
- **Supervision residue.** The action classes that remain above Level I under the fully hardened condition set CS-09; 32 of 60 in the founding case (22 at Level III, 10 at Level II). Also called irreducible supervision residue; it is relative to the catalogue and an upper bound. See the box in M6.
- **Viability conditions.** The six conditions VC1 independence, VC2 latency, VC3 halt, VC4 attention, VC5 comprehensibility, VC6 authority; asserted, not derived, and read in that order by convention.