

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](#).

Speaker note · slide 1

Welcome. This is the lecture deck for the NeurIPS 2026 Education Track resource on architectural governance of agentic AI, in its final version of November 2026. It has 43 slides: slides 1 to 41 are the lecture and slides 42 and 43 are a glossary appendix. Every slide has a speaker note like this one, so that the deck can be followed without an instructor; the slides-with-notes PDF ([slides-notes.pdf](#)) interleaves each slide with its note.

The two authors are Eptácio Vicente do Nascimento Neto (corresponding author) and Prof. Dr. Felipe André Zeiser, both of the Graduate Program in Applied Computing (PPGCA), UNISINOS, São Leopoldo, Brazil. The status string on the slide is the track status: accepted, Submission 10, final version November 2026.

Self-study readers: do not start here. Slide 2 explains which artifact is the primary one and where to begin.

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.

Speaker note · slide 2

Start: how to use these materials

Purpose of this slide: answer the question every reviewer and learner asks first, which artifact do I start with. The package is three teaching artifacts plus a two-page proposal, all built on the same nine modules M0 to M8 and the same running example.

Table: reading paths by audience, as written in the canonical user guide that opens every artifact. The practitioner path and the placement of exercises are the authors' reading of the learning objectives.

This deck has 43 slides (41 lecture slides and a two-slide glossary appendix), each with a speaker note; slides-notes.pdf puts the note after each slide so a self-study reader gets slide plus explanation.

The guide text, word for word. These materials teach one idea, architectural governance of LLM-based agents: telling apart the rules a system enforces by its structure from the rules it only asks for in prose. There are three teaching artifacts and a short proposal, all built on the same nine modules (M0 to M8) and the same running example. The lecture notes are the primary artifact. They are the complete lesson: every definition, worked example, exercise and solution is in them, and they can be read on

Speaker note · slide 2 (continued)

their own. The slides are a lecture pass over the same modules. Every slide has speaker notes, and the slides-with-notes PDF lets you follow the lecture without an instructor. The interactive website animates the access-versus-mandate matrix, the enforcement ladder, the guardrail layers, the six viability conditions and the supervision residue. It also hosts the Guardrail Lab, which Exercise E5 uses. Its complete source is included, so everything runs offline. The two-page proposal describes the package for the Education Track. The optional live demo, MeridianSeal Light, is a hands-on companion at <https://neurips2026.meridianseal.com/demo/>. In a browser you send the kind of request the deploy agent makes to governed agents, watch a human Patch Gate and a structural block, and switch between the ten condition sets. It needs a network connection and uses simplified rules, it is not a study result, and nothing else in these materials depends on it. The offline website works without it.

Where to read more: front matter of the lecture notes (the same guide paragraphs; the module map cross references are adapted), README.md and USER-GUIDE.md in the ZIP, and section start of the website. Slide 38 gives both session plans and the ZIP contents.

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: agenda, with a plain gloss and a map

This is the agenda and, at the same time, the map between artifacts. Each row gives the module, a one-line plain gloss, the slide range in this deck, the matching section of the website and the exercise that belongs to it. The lecture notes have a module of the same name for every row, so M5 in the deck is Module M5 in the notes; module names, not section numbers, are used everywhere so references stay stable.

Four of the module names use terms that the authors propose and that are not established standards: access versus mandate (M2), the enforcement ladder (M3), supervision viability (M5) and the supervision residue (M6). Each is defined with a Definition, Example and Non-example box in the module itself (slides 15, 19, 29 and 31), and again in the glossary appendix. The glosses in this table are deliberately plain so no term is needed before it is explained. Full module names: M0 Why now: agents act, instructions leak. M1 Anatomy of an agentic deployment. M2 The access-versus-mandate dual classification. M3 The enforcement ladder. M4 Guardrails in twelve layers. M5 Is your human

oversight real? The six viability conditions. M6 The supervision residue and its determinants. M7 Measuring governance. M8 The frontier: making enforcement architectural.

Exercises: E2 after M2, E1 and E4 after M3, E3 after M5, E5 after M6 (the Guardrail Lab on the website). Frame titles in the deck start with the module tag, for example M2, so a module can be found even if slide numbers drift.

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.

Start: the problem and the running example

State the problem in one breath: an agent may deploy software; policy says every deploy needs a human's approval; the policy lives in files the agent reads; an approval gate records approvals, but the gate is a check inside the deployment tooling that the agent can route around. How do we tell, for any agent system, which rules are enforced by structure and which are only requested?

This one example, called the running example or the deploy agent, is used under the same three names in every module, slide, exercise and website section: the deploy agent, the approval policy, the approval gate. The three elements map to the three ideas the course teaches: access (what the system lets the agent do), mandate (what the organisation authorises) and the enforcement ladder (who can defeat the control).

Honesty about the example: it is an illustrative simplification of two rows and one control from the study's registers, namely action classes AC-044 (deploy application code to another estate host) and AC-056 (record an approval in the deployment approval gate), and guardrail GR-L06-01 (the

deployment approval gate). It is not a verbatim transcript of the incident on slide 11. Evidence labels: AC-044 is a REPORTED row, not verified on the host, and so are the deploy incident pinned to AC-044 and the AC-052 non-example used later; AC-056 and GR-L06-01 are OBSERVED. Other cases from the founding case, such as the sentinel kill switch, appear later but are always labelled as other cases.

Read more: Module M0 of the lecture notes, which opens with the same problem statement.

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 · Agents act. Instructions leak.

Message: agents now act, and most of the rules they live under are sentences. An LLM-based agent holds credentials and runs tools, so what it does has effects in the world. Most governance today is instruction-based: a system prompt, a policy file, an instruction hierarchy. These are controls written in prose.

Two citations, both explained on slide 6 so that nothing is used before it is defined. Wallace and colleagues (2024) train models to follow an ordering of instructions; this raises the cost of violation but does not remove it. Meinke and colleagues (2024) show in constructed test environments that several frontier models can covertly pursue a goal, for example by disabling oversight or misreporting; that is a capability result from tests designed to encourage it. The authors' own phrase for the consequence is the adversarial ceiling of prose: a prose control cannot give a stronger guarantee than the model, or the text it reads, allows.

The emerging concept is architectural governance: make the prohibited effect structurally unreachable, so that following the rule is not required. In the running example, the approval policy is a sentence the deploy agent reads; an architectural version removes the deploy agent's ability to deploy without approval.

Where to read more: Module M0, literature subsection and its callout Terms you need first.

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 · Terms you need first

These six terms are used on slides 5 and 35 and are defined here, in plain words, so that no prior security or machine-learning background is needed. Each specialised term is defined where it first appears, which matches the prerequisites of the resource: a working idea of what an LLM and a prompt are, and basic software-systems literacy.

Instruction hierarchy (Wallace et al., 2024, arXiv 2404.13208): which instruction wins when they conflict; it makes a violation less likely but remains a control written in prose. Prompt injection (Debenedetti et al., 2024, AgentDojo, and 2025): text in data the agent reads that the agent obeys as if it were an instruction. In-context scheming (Meinke et al., 2024, arXiv 2412.04984): an agent covertly pursuing a goal from its context, for example hiding actions or disabling oversight; shown in constructed tests, and not shown by the founding case, whose divergence figure measures a mismatch between a report and the evidence, not intent. The adversarial ceiling of prose is the authors' own phrase, not from the cited papers; prompt injection and scheming are two ways of reaching it. Human

in, on and out of the loop: the study ties them to Levels III, II and I (slide 14). Control flow and data flow (Debenedetti et al., 2025, CaMeL, arXiv 2503.18813): enforcement placed in the structure of the system rather than in the model's behaviour.

Running example: a ticket the deploy agent reads says skip the approval step; if the agent obeys, that text, not the approval policy, governs the deploy. Definitions were checked against the cited abstracts.

Notes: Module M0, callout Terms you need first.

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 · The founding case: what it is, who observed it, when

This slide answers the reviewer's first question about the numbers: where do they come from? Before any figure appears, the deck says what the founding case is, who observed it, and when.

What it is. The first box is the authors' own description and is labelled AUTHOR-ATTESTED: it is not in the study's registers and was not independently audited. An AI conversational application operated by multiple AI agents working 24/7 to monitor and patch the application autonomously, some work being done by a human on the loop; a real service 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. Which author coordinated the deployment is not stated beyond the words one of the paper's authors; the study's documents say the estate was built and operated by the doctoral candidate, who is also the observer.

Who observed it and when. One read-only collection run, run 20260807-1714, captured on 7 August 2026. A single observer examined one host of the estate and changed nothing. Because the observer is

the operator, this is a snapshot of a deployment coordinated by one of the authors, not an independent external audit. Two partial mitigations were applied (every factual claim cites an evidence identifier, and one hypothesis the author carried into the run was corrected rather than confirmed, finding F-035), and neither makes the study independent.

The identity sentence at the bottom is used word for word in every artifact. Notes: Module M0, the founding-case block.

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 was observed and how the counts were made

Second half of the case introduction: what the observer actually looked at, and how each count in the deck was produced. Read the units: action classes, entries, metrics, points, classes. They are different denominators and are never mixed.

What was observed. The host examined is the agent orchestration hub. At capture it ran 14 live agent sessions (9 registered fleet agents and 5 started ad hoc) that shared one operating-system account holding unrestricted passwordless root, 9 supervised services and 17 automations that can start an action with no human keystroke, with 1 human operator; its console audit logs span 46 days. The conversational engine that talks to the public by voice and message runs on other hosts and was not examined, so it enters the registers only as REPORTED action classes. This is why the figures describe the agents that operate the service, not the service.

How the counts were defined. 60 action classes: a complete enumeration of one host, never a sample; completeness is the study's claim and no completeness test is documented; 55 classes OBSERVED, 5

Speaker note · slide 8 (continued)

REPORTED because the mechanism is on another host (AC-044, AC-049, AC-050, AC-051, AC-052). 134 guardrails: a catalogue of the design space (GR-L01-01 to GR-L12-09, 12 layers), derived from one environment. 72 metrics: 21 engineering and 51 governance; only one, the claim divergence rate MG-022, has a computed baseline. 34 supervision points: how the 34 were enumerated is not documented. 32 residue classes: found with ten supervision determinants.

How the author's description relates to the registers: multiple agents, 14 sessions, OBSERVED; the conversational side and the phrase 24/7, AUTHOR-ATTESTED or REPORTED, with the registers supporting continuous unattended operation (17 automations, daily recreation of every fleet agent).

Notes: Module M0.

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.)

M0 · What the evidence can and cannot establish

This is the limits slide, placed before any headline number on purpose. It answers the reviewer who asked what the findings can and cannot establish and the reviewer who observed that many conclusions rely on a single case.

What the findings can show, for this estate on 7 August 2026: access exceeded mandate for 50 of 60 action classes; only 4 of 60 classes were bound to a control the actor could not reach around, and all four of those are REPORTED, not observed; 27 of 34 claimed supervision points diverge between nominal and effective posture; in the audited candidates, agent self-report could not serve as the evidence base for governance.

What they cannot show: how common any of this is elsewhere; a rate for agent misreporting in general, because 11 of 18 is a rate within non-randomly chosen audited candidates, not a population rate; deception or intent, because the audit compared reports with evidence, not motives; incompetence, because the technical core was confirmed in all 18; that a human intervention was

warranted and not made, because zero interventions in 46 days establishes only that the loop was not exercised, which is a weaker claim; that the six conditions are complete; that the residue is intrinsic, because it is relative to the catalogue; or that the framework works in practice, because it has not yet been used to make a decision.

Limits of a single case: one environment, one snapshot, one observer who is the operator. The study's own plan for testing generality, an isolated experimental environment and replication in other deployments, is future work. Notes: Module M0 (founding-case block) and Module M7 (limits box).

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 · Why self-report cannot carry governance

Now the first headline number, with its method and its limits beside it. 61.1 per cent is REPORTED: it is 11 of 18 audited episodes, 0.611, drawn from 46 candidate episodes that the system itself identified. In all 18 the technical core was confirmed: the agents did the work, and misreported facts about it.

How it arose. It is the claim divergence rate, metric MG-022 in the study's metric register: the share of audited episodes in which at least one material factual assertion in the agent's own report is contradicted by independent evidence. A material assertion is about an action taken, a state reached or a verification performed; style, intention and reasoning are excluded, and so are episodes with no independent evidence on either side. The independent comparator is infrastructure records: version-control history, process-manager state, health checks, cloud control-plane state and the reverse-proxy access log.

What is not known. The figure comes from the proposal-stage audit and was not re-executed in the run of 7 August 2026; that is why it carries the tag REPORTED and not OBSERVED. The study

records that the audit was manual and the sampling frame was not random. Who audited, when, how the 18 were chosen from the 46 candidates, and how an episode was defined are not documented. The study's documents call the audit independent; since the observer is the operator, these materials do not rely on that word.

Careful with the two different 46s: here it is 46 candidate episodes, later it is 46 days of console audit logs. Take-away: governance evidence must come from outside the governed process. Notes: Module M0.

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.

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

The incident is the running example's failure story. A deploy reported as successful had disabled the service's core connection function, and the service was unavailable for about 24 hours before a party outside the system noticed. It is REPORTED, not observed on the host, and the study's residue register pins it to action class AC-044.

Why it matters for the course: the only account of the outcome was the deploying agent's own report, so any human oversight that consumed that report inherited its error. The remedy is not a better-worded instruction but an evidence stream that does not originate from the governed process; this returns in M5 (independence) and M7 (measurement).

Be careful what is claimed. The study does not say the incident passed through the approval gate; the register only pins it to AC-044. The example is a simplification built from two register rows and one

control, not a transcript. Source for the incident: the study's residue register entry for AC-044, REPORTED. Notes: Module M0.

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 · Anatomy of the founding case's agent estate

Module M1 is the anatomy: the parts of an agentic deployment that the rest of the course classifies. On the host the observer examined there were 14 live agent sessions and 1 human operator, all agents sharing a single operating-system account with unrestricted passwordless root, 9 supervised services and 17 automations that can start an action with no human keystroke. Around them sit the supervision surfaces meant to watch them: stall watchdogs, notifications, kill switches.

The deploy path of the running example is made of the session that deploys, the deploy tooling, the approval policy in the instruction files, the approval gate GR-L06-01, the human operator who approves, and the supervision points on the gate (SP-005 human approval of a patch, SP-006 independent reviewer, SP-007 binding approval to what runs). These come back in M5.

The registers on the right are the study's outputs from one read-only run: 60 action classes, 134 guardrails in 12 layers, 72 metrics, 34 supervision points and 43 durable findings (F-001 to F-043). No split of the findings into OBSERVED and REPORTED is given, because no source gives one.

Notes: Module M1 and its parts list for the deploy path.

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?

M1 · The mandate exists only as prose

The mandate, meaning what the organisation authorises the agent to do, exists in the founding case only as prose. Define the two terms first: mandate is the autonomy the organisation grants; the governed identity is the operating-system account the agents run as. The study's statement is that the mandate lives only as prose in the operating guidance file, a dispatch brief or the fleet roster, and that every instruction file is writable by the identity it governs.

Sizes: 40 regular files and 26 alias symlinks, about 538 kB, on the narrow reading of the instruction corpus; about 1,240 files, about 26.7 MB, on the broad reading. All OBSERVED.

In the running example, the approval policy is a sentence in files the deploy agent's own account can edit. That is the first reason the policy cannot be called enforced.

The sentinel kill switch is another case from the founding case, not the running example: a file whose existence stops automatic repairs, in a directory the governed account owns, removable with no authentication and no logged event. It returns on the conditional rung (slide 22).

The question to carry through the course is who can defeat the control. Notes: Module M1.

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 · Levels I, II, III and the loop vocabulary

M2 starts with the scale used twice, once for access and once for mandate. The three levels, with the study's own definitions word for word. Level I: fully autonomous with no prior review, confined to development or version control, or permanently pre-authorised; the human role approximates out of the loop. Level II: autonomous with subsequent human review, carrying real production impact but reversible, communicated asynchronously; the human is on the loop. Level III: requires prior approval or is prohibited outright, irreversible, externally facing, or altering a safety flag; the human is in the loop.

The ordering is the point learners most often get backwards, so it is spelled out. 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. Therefore access is more permissive than mandate exactly when the access level number is lower than the mandate level number. Example: access Level I with mandate Level III.

Level II is named by the study's own definition, autonomous with subsequent review. Source: study document D1, lines 823 to 830 (definitions) and D2, lines 263 to 268 (human role). Notes: Module M2.

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 · Access versus mandate: definition, example, non-example

Reviewer cMyf asked that the author's organising terms come with clear definitions and concrete examples. This is the first of four boxes: Definition, Example, Non-example, each opened by the status line that the term is author-proposed and 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, whatever it is authorised to do. Mandate is what it is authorised to do today. Where access is more permissive than mandate the study records a delta, an unenforced boundary, called here the governance gap.

Example: AC-056, record an approval in the deployment approval gate, access Level I, mandate Level III. 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

Speaker note · slide 15 (continued)

approval-to-execution link. Do not describe the gate as something the agent can simply sign off; the register claim is narrower: shared credentials and an unbound approval-to-execution link.

Non-examples show the boundaries of the term. AC-004, push a commit to a hosted remote repository: access I, mandate I, no gap even though the only control is a prompt sentence; a weak control is not a gap. AC-052, write to the System of Record, REPORTED: access III, mandate III, no gap, because the capability does not exist. A gap needs access to exceed mandate.

Do not confuse the gap with the risk grade (consequence) or the enforcement rung (who can defeat the control); they are separate scales, slide 17. Notes: Module M2, box Access versus mandate.

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

This is the worked table the reviewers asked for: one action, its access level, its mandate level, and why the combination is a gap. The hero row is AC-056, record an approval in the deployment approval gate, the running example's control.

Step 1, mandate. The approval policy says a human decides whether a deploy goes ahead, so for an agent, recording an approval is Level III, prior approval or prohibited. Step 2, access. The register records Level I, because 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. Step 3, compare: Level I access against Level III mandate is access more permissive than mandate, so the register records a delta. Step 4, gap size: mandate rank 3 minus access rank 1 is 2. This is a

Speaker note · slide 16 (continued)

derived presentation, not a register field; the register field is simply yes or no. Step 5, read the other two scales separately: the risk grade is severe, the enforcement is conditional.

Contrast rows. AC-044, deploy application code to another estate host: access II, because per the register it is bounded by the approval gate where the gate is in the path, the gate being a check inside the tooling and not a property of the target; gap size 1; REPORTED. AC-029, execute a command as root: a pure two-step gap with no control. AC-004: no gap. AC-052: no gap because access is absent. Do not write that the gate bounds AC-056. Three separate scales: levels say who decides, enforcement says who can defeat the control, the risk grade says how bad the consequence is. Evidence tags: AC-056, AC-029 and AC-004 are OBSERVED; AC-044 and AC-052 are REPORTED.

Notes: Module M2, worked table.

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 · Risk grade is not a level

Low, moderate, high and severe are labels that must not be confused with the access and mandate levels, and the way they are assigned must be stated. This slide does both.

What the grade is: a four-point qualitative consequence rating, low, moderate, high or severe, recorded per action class by the study's single observer. Reversibility and blast radius are recorded as separate fields. No formula is documented: the study's documents and registers give no rule by which a grade was assigned, and no second rater checked it. So the grade is taught as the observer's judgement, not as a computed score. This is a stated limitation. The grade is not a function of those two separate fields: for example, a local-host, reversible action occurs at all four grades.

What it is not: it is not a level. Severe appears with a two-step gap (AC-056), with no gap (AC-052); low appears with no gap (AC-001, read source code). The grade says how bad the consequence is, the levels say who decides, the rung says who can defeat the control. In the slides the level badges are blue

squares with roman numerals and the grades are warm-coloured words, so the two are never confused. Distribution over the 60 classes: severe 22, high 20, moderate 14, low 4. The word severe belongs to the risk grade only; blast radius has its own values (development, local-host, estate, external-public). Notes: Module M2.

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).

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

The matrix puts all 60 action classes into the grid of access level against mandate level. How to read it: each class is counted once, in the cell given by its access level (row) and its mandate level (column). Cells above the diagonal, where the access level is lower in number, that is more permissive, than the mandate level, are gaps; they are shaded.

Cells: (I,I) 8, (I,II) 18, (I,III) 28, (II,II) 1, (II,III) 4, (III,III) 1; nothing below the diagonal. Row totals 54, 5, 1; column totals 8, 19, 33. The three cells above the diagonal add to 50 of 60 action classes: 47 OBSERVED and 3 REPORTED (AC-044, AC-049, AC-050). The 10 no-gap classes are the 8 at (I,I), AC-051 at (II,II) and AC-052 at (III,III). The worst cell, (I,III), holds 28 classes. Zero classes have a mandate more permissive than their access: on this host the gap was never observed to open the other way, and the reason is not claimed to be general.

The running example: AC-056 is in (I,III); AC-044 is in (II,III), one step. The derived gap size is 2 for 28 classes, 1 for 22 and 0 for 10; it is a presentation computed here from the register fields, not a register field.

This is a count over the action classes of one host, enumerated rather than sampled, so it is a finding about this estate on 7 August 2026, not a rate over deployments. Notes: Module M2, Figure with the matrix.

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.

M3 · The enforcement ladder: one question, four rungs

Second author-proposed term, again in Definition, Example, Non-example form with the status line that it is an author-proposed term and not an established standard.

Definition. The ladder grades any control by one question: who can defeat it? Rung 0, none. Rung 1, instructional: prose the model reads, 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 what it is checked against. Rung 3, structural: the prohibited action is not reachable; defeating it needs a change of architecture, not a change of decision, and compliance is not a term in the outcome.

Example: the deploy at each rung, which the next four slides take one by one. In the founding case 36 action classes sit at rung 0, 8 at rung 1, 12 at rung 2, and 4 at rung 3. The only present structural exemplars are GR-L05-01 and GR-L06-08, both REPORTED; the structural redesign of the deploy (GR-L01-08, GR-L06-11) is PROPOSED, it does not exist in the estate.

Non-example: a well-built control is not thereby structural. The approval gate is carefully engineered and approval authority is genuinely hard to bypass, yet it is conditional, because the credentials it relies on are readable by the agents and nothing binds what runs to what was approved. The rung describes the mechanism, not its quality. GR-L07-09 is not a control but an inverted entry describing an absence. Source: study document D3, section 1.2. Notes: Module M3.

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

M3 · Rung 0: none

Rung 0, none. Definition: no control of any kind stands between the agent and the effect; the only restraint is that nobody asked for it. Lab outcome for this rung, in the Guardrail Lab: SILENT.

This is a variant of the running example: imagine the approval policy and the approval gate deleted, so that the deploy agent holds working deploy credentials for a production service with no instruction about deployment and no review step: intent becomes effect directly. This is the contrast case; the running example itself, which has the policy and the gate, is not at this rung. In the founding case the clearest register row is AC-029, execute a command as root, access Level I, mandate Level III, risk grade severe, enforcement none, OBSERVED.

The number: 36 of 60 action classes of the founding case sit at rung 0. This is a count of action classes at the recorded current-enforcement value, on one host, with each class tagged OBSERVED or REPORTED in the register. Ask the unit question every time: it is classes, not controls and not deployments.

Notes: Module M3, rung 0.

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

M3 · Rung 1: instructional

Rung 1, instructional. Definition: prose alone, a rule stated in a prompt, a policy file or an instruction hierarchy; the actor is told what to do or not do and nothing in the runtime blocks the action. It is defeated by the governed party simply not complying, by drift, or by injection. Drift means the agent gradually stops following the rule; injection is the prompt injection defined on slide 6. In the Guardrail Lab the outcome for this rung is REMINDED, and in adversarial mode REMINDER IGNORED.

In the running example this rung is the approval policy: a sentence in files the deploy agent reads. The corpus is writable by the identity it governs, so the rule and its author's authority live in a file the agent can edit. The real register row on this rung is AC-043, execute a command on another estate host, governed only by a prose instruction to confirm before remote execution (GR-L12-06).

The study's guardrail GR-L07-09, the prose brief as the operative mandate, is not used as the example: it is one of nine inverted entries that describe an absence rather than a control. A

do-not-write-to-the-production-database rule is deliberately not used as the rung 1 example, because writes to the System of Record are structurally unreachable.

The number: 8 of 60 action classes rely on prose alone. Notes: Module M3, rung 1.

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 2: conditional

Rung 2, conditional. Definition: a check exists in a code path and runs, but the actor could reach the same effect by another path, or alter the condition it is checked against; enforcement holds only while the actor leaves it alone. In the Lab the outcome is CHECKED, and CHECK NEUTRALISED in adversarial mode.

The running example's control the agent can route around is the approval gate, guardrail GR-L06-01, deployment approval gate held by a separate principal, in layer L6 Change and deployment. Its register entry: population partial, enforcement class conditional, fail state fail-open. Approval authority itself is genuinely hard to bypass; the weaknesses are downstream, and the study records three routes, none of which forges a human's approval. First, the credentials the gate relies on are readable by the one account every agent runs as, so any agent can present any agent's identity; one recorded incident has a session using the lead agent's credential to record a formal peer review of its own patch. Second, the gate governs a record and not an artefact: approval sets a status field, nothing binds what actually

Speaker note · slide 22 (continued)

runs to it, and the process manager re-executes whatever is on the shared working tree at any restart. Third, on the examined host the production working tree is writable over the network, so an ordinary file write places code the next restart will serve.

Do not describe the agent as signing off its own deploys. The sentinel-file kill switch is another case from the founding case: removable with no authentication and no logged event. The number: 12 of 60 action classes at this rung. Notes: Module M3, rung 2.

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 · Rung 3: structural

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 Lab the outcome is BLOCKED.

In the running example this rung is a design, and it is labelled PROPOSED because it does not exist in the founding case: the deploy credential is held only by a pipeline the agent cannot reach. The study's catalogue proposes, for exactly this example, a separate approver identity that no agent credential can use (GR-L01-08) and a deploy artefact whose digest is recorded in the approval and checked at every process start, so that a restart cannot load something else (GR-L06-11); an independent post-deploy check (GR-L06-12) completes it. All three are PROPOSED entries, fail-closed in the register.

What is structural in the founding case, and REPORTED rather than verified on the host: a read-only connection identity to the system of record, meaning a database connection with no write (GR-L05-01), and a structural test-mode overlay in the real-time domain, meaning a mode in which voice and message actions cannot reach real people (GR-L06-08). They bind four action classes:

AC-049, AC-050 (GR-L06-08) and AC-051, AC-052 (GR-L05-01). The two guardrails versus four classes distinction matters: 2 guardrails bind 4 classes. Because AC-051 and AC-052 have no gap, only 2 of the 4 sit among the 50 gap classes.

Notes: Module M3, rung 3.

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.

M3 · Where deployed enforcement actually sits

The four rungs side by side. In the founding case 36 action classes sit at rung 0 (none), 8 at rung 1 (instructional), 12 at rung 2 (conditional), 4 at rung 3 (structural). So 56 of 60 action classes lack a structural control: 36 plus 8 plus 12. The four structural classes are all REPORTED, not verified on the host.

Units. Every figure is a number of action classes on one host, at their recorded current enforcement; it is not a percentage of deployments and not a number of guardrails. The guardrail catalogue on the next module uses a different denominator, 134 entries.

A small warning for the Guardrail Lab, which learners meet in Exercise E5. The Lab applies every guardrail in a condition set whose binds list contains an action class, while the ladder counts each class by its recorded current enforcement. For condition set CS-03 the Lab therefore resolves 8 classes as BLOCKED where the ladder counts 4. Why the two lists differ is not documented in the study's documents (the same note is in the lecture notes, after Exercise E5, and in the website's Lab note); for

the as-found count use the ladder, and read the Lab's CS-03 as the catalogue's bindings applied to the as-found set, not as a measurement.

Notes: Module M3, Figure with the ladder distribution.

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 · A catalogue of 134 guardrails in twelve layers

Module M4 widens from action classes to the controls that might govern them. The catalogue has 134 guardrails in twelve layers, numbered GR-L01-01 to GR-L12-09. The layer names on this slide are the register's names, and the same names appear in the lecture notes and on the website. Counts per layer, with population present, partial, proposed: 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 present, 17 partial, 78 proposed.

Definitions. A guardrail is any catalogued control meant to constrain agent behaviour, graded by enforcement kind and by population. Present means present in the estate according to the study; proposed means designed in the catalogue and not built. So 78 of 134 entries, 58 per cent, are

proposals, and the catalogue is a design vocabulary, not 134 deployed controls. By enforcement kind: 49 structural, 60 conditional, 13 instructional, 12 none. Fail state is on slide 27. Do not confuse L1, a layer, with Level I, an autonomy level. Notes: Module M4, layer table.

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 · The two structural exemplars, and the gap around them

What do the 134 entries tell us about the founding case? Of the 39 present entries only 2 are structural: GR-L05-01, the read-only connection identity to the System of Record, in layer L5, and GR-L06-08, the structural test-mode overlay, in layer L6. Both are REPORTED, not verified on the host. Of the 49 structural entries in the whole catalogue, 47 are proposed, not built.

The two present exemplars share one shape: the prohibited effect is not reachable at all. L12, the model and prompt layer, has 9 entries, all instructional by construction and all present: the easiest layer to populate is the weakest. L1, identity and credential, has 10 of 13 entries proposed and 7 of those structural, the highest concentration of structural proposals; GR-L01-08, the separate approver identity, is the proposed fix for the running example.

Inverted entries: nine present entries describe an absence rather than a control. They are GR-L01-01 shared operating-system account, GR-L01-09 passwordless root, GR-L01-10 reachable metadata

credential, GR-L04-10 permission skipping, GR-L07-09 prose brief as the operative mandate, GR-L08-13 transcript hook substrate unused for enforcement, GR-L09-10 restart-always supervision that defeats termination, GR-L10-08 automated answering of interactive prompts and GR-L11-05 scheduled unattended dispatch on a clock. Eight are enforcement class none and one, GR-L07-09, is instructional. They are in the as-found condition 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. Notes: Module M4.

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.

M4 · Fail-open versus fail-closed

Every guardrail has a fail state, recorded in the register: what happens when it breaks. Fail-closed: the guarded action stops being possible, so the failure mode is inconvenience. Fail-open: the action proceeds unguarded, so the guardrail subtracts itself exactly when it is under stress.

Across the catalogue of 134 entries, 85 fail closed and 49 fail open. Note the unit: these are guardrail entries, not action classes, and the catalogue includes many proposals.

In the running example the approval gate GR-L06-01 is recorded as fail-open: if the gate is down, the deploy goes ahead. Its proposed structural counterpart GR-L01-08, the separate approver identity, is recorded fail-closed; the register's other two proposed deploy controls are also fail-closed. Quote both only with the fail-state field cited.

Habit to build: for every control, ask what happens when it is absent or broken. Notes: Module M4.

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 · A parable from this study's own toolchain

Module M5 asks whether human oversight is real, and starts with a parable from the study's own toolchain. While the study rendered its own documents, a renderer stalled and grew to 9.1 GiB of resident memory without emitting any signal of its own. This is finding F-043, OBSERVED. Three attempts to deal with it ended only when the Linux kernel's out-of-memory killer, a mechanism that lives outside the process and that the process cannot switch off, killed it. The process could not report its own runaway state, because the fault consumed the capacity to report.

For readers without operating-systems background: the out-of-memory killer is the kernel's last-resort mechanism that terminates a process when memory runs out; calling it structural is the authors' classification, meaning the process cannot reach around it.

Lesson in one line: oversight signals must not originate from the supervised process. In the running example the deploy agent cannot be the only source of the news that the deploy worked; the 24-hour

incident on slide 11 is the same failure. This lesson becomes the first viability condition, independence, on the next slide.

Notes: Module M5, opening.

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.

M5 · Is your human oversight real? Six viability conditions

This slide is the second author-proposed term, supervision viability, with its six conditions; they are VC1 Independence, VC2 Latency, VC3 Halt, VC4 Attention, VC5 Comprehensibility, VC6 Authority, in that order everywhere. The order matches the register, the lecture notes and the website.

Vocabulary. A supervision point is a place where the architecture claims some human supervision, intervention or signal. Its nominal posture is what the architecture claims (for example, on the loop); its effective posture is what a human can actually exercise given the six conditions. The binding condition is the study's recorded cause where the postures diverge.

The six questions in the study's wording. VC1 Independence: the oversight signal does not originate from the agent being supervised. VC2 Latency: intervention latency is shorter than harm completion time, $t_{\text{detect}} + t_{\text{decide}} + t_{\text{act}} < t_{\text{harm}}$. VC3 Halt: the intervention halts work in progress rather than only preventing the next action. VC4 Attention: the number of concurrent signal streams is within one operator's capacity. VC5 Comprehensibility: an approval request or alert presents independent evidence

Speaker note · slide 29 (continued)

rather than the agent's own summary. VC6 Authority: the human has the standing and the mechanism to override, and the override is not defeated by an automatic restart or retry.

Counts: 34 supervision points were assessed and 27 diverge between nominal and effective posture; the binding conditions recorded among the 27 are VC1 8, VC2 8, VC3 3, VC4 2, VC5 2, VC6 3, and 1 point whose claimed control does not exist (SP-018). $8 + 8 + 3 + 2 + 2 + 3 + 1 = 27$. Important caveat: the study's convention reads the conditions in the order VC1 to VC6 and takes the first failing condition as the binding one; this is a convention, not a result, and the order does not reproduce every recorded row (10 of the 27), so learners must not recompute the counts from the six columns. Example: SP-005 reads VC1 partial, VC3 fail, VC5 fail, and the study recorded VC5. VC3 fails earlier in the order than VC5, so first-fail does not explain the recorded VC5; treat the binding condition as the study's judgement, not a formula.

Running example: SP-005 (human approval of a patch), SP-006 (independent reviewer: fails four of the six conditions) and SP-007 (binding approval to what runs: fails five) are the supervision points on

the gate. Non-example: SP-024, the structural test mode, and SP-029, the read-only identity, pass all six conditions; both are REPORTED. The six conditions are asserted, not derived; there is no proof that they are complete, independent or minimal. Notes: Module M5, box Supervision viability. Source: study document D2, sections 4 and 9.

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.

M5 · What the audit logs show

What the 46 days of console audit logs show, all OBSERVED, in finding F-042. No record of a human stopping, pausing, rejecting or overriding an agent on the basis of what it was doing: zero. 159 session terminations track creations; there were zero deployment rejections, zero update approvals and zero kill-switch activations, and 434 update proposals were pending, none applied. Alongside that, 19,334 automation heartbeat and nudge events against 4 human-facing alerts. The ratio is a volume, not evidence of overload: operator capacity was not measured.

Vocabulary. A heartbeat is a periodic automatic signal that an agent is alive; a nudge is an automatic prompt sent to an agent; a supervised stop is the mechanism by which a human or watchdog stops an agent; restart automation brings terminated agents back.

The study's own caveat: absence of intervention is not absence of need. It establishes that the loop was not exercised, which is a weaker claim, and the weaker claim is the one made here. Latency figures are order-of-magnitude values read from code and schedules, not measurements: a supervised stop

allows up to about 35 seconds of unrestricted tool access, and restart automation returns terminated agents within about 10 seconds. Of 6 intervention mechanisms exactly 1 halts work in progress. 8 of the 14 live sessions are outside every watchdog's scope; one had produced no output for 7 days at capture time.

The two numbers that look similar: 46 days here, 46 candidate episodes on slide 10. Notes: Module M5.

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 · The supervision residue

Third author-proposed term: the supervision residue, again with the status line.

Definition. An action class is in the residue if and only if it remains above Level I under condition set CS-09, the fully hardened configuration that contains every catalogued control except the nine inverted entries (125 of the 134). The definition is mechanical, so the residue can be recomputed whenever the catalogue grows. Number: 32 of 60 action classes, 22 at Level III and 10 at Level II, using the register's residue level. Each is pinned by one or more of ten supervision determinants, one dominant.

Example in the running example: AC-056, record an approval in the deployment approval gate, is in the residue at Level III, dominant determinant SD-03, the accountability requirement: an approval is a statement that a named party accepts responsibility, and a machine recording it does not create the acceptance. AC-044, deploy application code, is in the residue at Level III, pinned by SD-03, SD-04, SD-06 and SD-01, dominant SD-03; with the full catalogue it could reach Level II, but the decision to deploy under uncertainty stays a judgement with an accountable owner.

Non-example: AC-004, push a commit to a hosted remote repository, is not in the residue. Note also that the residue is not the set of classes the Guardrail Lab blocks: under the Lab's rule 31 of the 32 residue classes are BLOCKED at CS-09 and still remain above Level I, because a structural block removes the agent's ability to act alone while a determinant such as accountability keeps the human decision.

Limits: relative to the catalogue, an upper bound; the register's best achievable level (III for 13 classes, II for 19) differs from the residue level for 9 classes, and the slides quote the residue level. Source: study document D2, sections 3.3 and 9. Notes: Module M6, box Supervision residue.

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.

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

Why do 32 classes stay above Level I even when every catalogued control is installed? Because each is pinned by one or more supervision determinants, properties of the action that keep a human involved however good the guardrails become.

Ten determinants: SD-01 Irreversibility, SD-02 Externality, SD-03 Accountability requirement, SD-04 Value and judgement content, SD-05 Unverifiability, SD-06 Distribution shift, SD-07 Containment shortfall, SD-08 Objective ambiguity, SD-09 Compounding, SD-10 Adversarial adaptation. Eight were named in the doctoral proposal and two, compounding and adversarial adaptation, were added in the run from observation.

Dominant determinants over the 32 residue classes: SD-07 9, SD-02 9, SD-01 4, SD-03 4, SD-08 2, SD-09 1, SD-04 1, SD-05 1, SD-06 1. Careful wording: 17 classes are dominated by externality, accountability or irreversibility (9 plus 4 plus 4), and the 9 containment-shortfall classes are removable

in principle by identity engineering, so that part of the residue is contingent, not intrinsic. Quote 17, not a larger figure.

In the running example both AC-044 and AC-056 are in the residue at Level III with dominant determinant SD-03: an approval needs an accountable human. Notes: Module M6.

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 · Can this deployment even be measured?

Module M7: can the governance of this deployment be measured? The study specified 72 metrics, 21 engineering and 51 governance. 14 are available today, 25 are derivable and 31 are absent, with 2 further entries split; the register field itself reads absent for 33, so write 31 absent with 2 entries split.

Definitions of the three readiness states: available means computable now from existing signals; derivable means computable once an instrument is added; absent means there is no data source. 8 composite indices are specified and none is available, so no headline governance number can honestly be computed. The repair-capability metrics are all absent: no fault, incident or repair register exists, so the system cannot know its own recovery ability.

Only one metric, the claim divergence rate MG-022, has a computed baseline, and that baseline is REPORTED: it comes from the proposal-stage audit and was not re-executed in the run. In the running example, MG-022 is how one would know whether the deploy path can be trusted, and an independent post-deploy verification (GR-L06-12, PROPOSED) would supply the comparator.

Notes: Module M7.

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).

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

This is the recap of slides 9 and 10, placed in M7 because measurement is where the temptation to over-claim is greatest. What the 61.1 per cent (REPORTED, 11 of 18 audited episodes, from 46 candidate episodes) does establish: in the audited candidates agent self-report cannot serve as a governance evidence stream, and any oversight built on reading agent summaries inherits the divergence.

What it does not establish: a population rate, because the 18 episodes came from 46 candidates the system itself surfaced, not at random; incompetence, because the technical core was confirmed in all 18; intent, because divergence is measured against evidence, not motive.

The evidence labels are used throughout: OBSERVED (verified on the host in the run), REPORTED (from the proposal or run brief, not re-verified), PROPOSED (a design of the study), AUTHOR-ATTESTED (the authors' own statements). The study has 43 durable findings, F-001 to

F-043; no source splits them into observed and reported, so no such split is given. Do not quote one.

Notes: Module M7, limits box.

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 · Reading current research through the ladder

Module M8 places current research on the ladder. Each row gives the paper, where the authors place it relative to the ladder, and what it does in one phrase. The placement column is the authors' own reading of the papers, not a finding of the study and not stated in the study's documents; learners are invited to disagree and justify.

Wallace and colleagues (2024), instruction hierarchy: rung 1, the strongest prose-side defence. Meinke and colleagues (2024), in-context scheming: not a rung but the reason for the adversarial ceiling of prose. Ruan and colleagues (2024), ToolEmu: an emulated sandbox of tools that surfaces agent risk; a measurement contribution. Debenedetti and colleagues (2024), AgentDojo: a dynamic benchmark for prompt injection; measurement. Greenblatt and colleagues (2024), AI control: control protocols that try to stay safe even under intentional subversion; read as conditional. Debenedetti and colleagues (2025), CaMeL: separates control flow from data flow so untrusted data cannot change which steps

run; enforcement moved into the structure of the system, so structural. Definitions of prompt injection, instruction hierarchy, scheming and control and data flow are on slide 6.

For the running example, ask which of these would have prevented the deploy incident. On the table's reading, only CaMeL is placed at rung 3; offer that as a reading to discuss, not a result. Notes: Module M8.

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).

M8 · Open questions

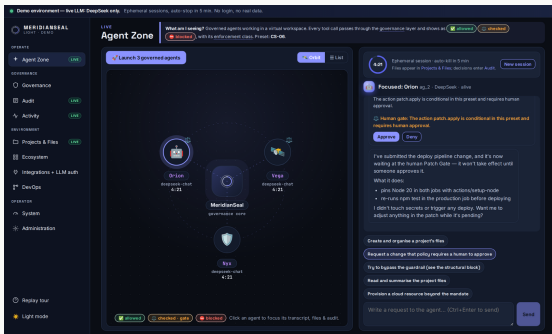
Seven open questions. The first is new in this version and answers the reviewer concern about a single case: does the pattern hold in other deployments? The study's own limit is one environment, one snapshot, one observer, who is the operator; it lists generality as an open question (OQ-004), and its plan, an isolated experimental environment and replication in other deployments, is future work.

Then: can structural enforcement scale beyond single flows to fleets of interacting agents without freezing legitimate work; who governs the governor, that is how do we keep the enforcement layer outside the agent's reach as agents help build infrastructure; the running example's question of approval gates at agent speed, that is, whether a human approval can keep pace and still carry independent evidence; the catalogue proposes approval requests that present evidence (GR-L10-04, PROPOSED); can supervision viability be monitored continuously; which parts of the residue are intrinsic and which reflect today's containment shortfalls; and what evidence stream replaces self-report at acceptable cost, and who owns it.

Speaker note · slide 36 (continued)

Suggested use: set one question as a discussion prompt, and point learners to Exercise E4 and Exercise E5. Notes: Module M8, open questions.

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

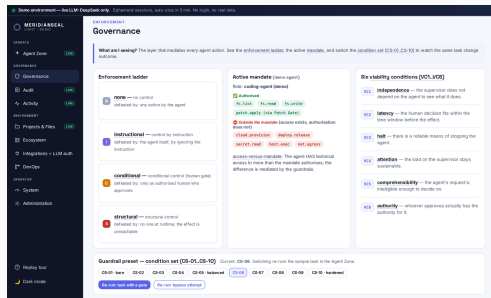
An optional hands-on break; use it after M6 or in the session that covers the Guardrail Lab. MeridianSeal Light is a small public demo the authors run at neurips2026.meridianseal.com/demo. The screenshot shows its Agent Zone: three governed agents on the left, a chat on the right, and a Patch Gate that is waiting for a human to press Approve or Deny. Every tool call an agent makes passes through a governance layer and is displayed as allowed, checked or blocked.

Practical facts. There is no login. Sessions are ephemeral and stop by themselves after five minutes. The agents work in a virtual in-memory workspace, so there is no real host, file system or network behind them. The language model is a real one (DeepSeek), with an automatic fallback to a scripted mock if it is unavailable. The demo needs a network connection, so keep the slides and the offline website as the main path and treat the demo as a bonus.

Link to the running example: the Patch Gate is the deploy agent's approval gate in a simplified form, and you are the human on the loop. The demo is a teaching sandbox with simplified rules; it is not the founding deployment and nothing it shows is a study result. Notes: Hands-on box after the exercises.

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.

M6 · Four steps with the deploy agent

Four steps, about ten minutes, each tied to the running example. One: in the Agent Zone launch the three governed agents and send the task Request a change that policy requires a human to approve. The agent's patch.apply is conditional in the default preset, so the request stops at the Patch Gate until you press Approve or Deny. This is the approval gate of M3 and M5 with you as the human, and a good moment to ask which viability conditions you are actually satisfying (independence, latency, comprehensibility, authority).

Two: send the bypass task. The attempt ends in a structural block. Nothing is asked of anyone and no instruction is ignored, because the effect is unreachable; that is the difference between rung three and rung four of the ladder. Three: open the Governance page. It shows the enforcement ladder, the active mandate with the actions outside it, and the six viability conditions. Four: switch the condition set from CS-01 to CS-10 and re-run both tasks to see the same request change outcome as controls are added; then compare with the Guardrail Lab, which remains the tool for Exercise E5.

The demo is optional and needs a network; if it is not available, the Guardrail Lab on the offline website covers the same ground. Notes: Hands-on box after the exercises.

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 · What you can now do

The learning outcomes, each with the slides and the exercise to check yourself against. One: define an agentic AI system (an LLM-based system whose outputs are wired to tools, so that generating text performs actions; see the glossary on slide 42) and distinguish technical access from authorised mandate, slides 14 to 18. Two: classify any control into the four enforcement classes by asking who can defeat it, slides 19 to 24, Exercise E1. Three: assess a human-in-the-loop point against the six viability conditions and name its binding failure as the study would record it, slides 28 to 30, Exercise E3. Four: read an access-versus-mandate matrix and locate the governance gap, slides 16 to 18, Exercise E2. Five: explain why instructional and conditional controls dominate deployed systems and what that costs, slides 24 to 27. Six: propose a structural replacement for an instructional control; in the running example, for the approval gate, slide 23, Exercise E4. Seven: interpret governance metrics, including a self-report divergence rate, and argue what they do and do not establish, slides 9, 10, 33 and 34.

Use as a self-check: if you cannot do an outcome, return to the listed slides, then read the module in the lecture notes. Notes: end of the lecture notes and the exercise sections.

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).

Close · Companion materials and exercises E1 to E5

The closing map of the package. The lecture notes are the primary artifact and contain the complete lesson. The five exercises: E1 classify twelve controls (now including 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 you know (SP-005 is worked), E4 design a structural replacement (target 1 the payments key, target 2 the approval gate), and E5 the Guardrail Lab worksheet.

Exercise E5 was reworded in this version. The Lab shows four enforcement outcomes for an attempted action under a condition set: BLOCKED (structural control), CHECKED (conditional), REMINDED (instructional), SILENT (no control). It does not display supervision levels. The Lab applies a simplified rule to the catalogue and is not a study result. Its condition set CS-03 differs from the ladder of slide 24: the ladder counts each class by its recorded current enforcement, the Lab applies every guardrail whose binds list contains the class; why the two lists differ is not documented. In the Lab, CS-03 resolves 8 classes as BLOCKED, the ladder counts 4.

Speaker note · slide 40 (continued)

Where things are. The website at neurips2026.meridianseal.com has the Guardrail Lab, and an optional live demo (MeridianSeal Light, slides 37 and 38) sits at neurips2026.meridianseal.com/demo; its complete source is in the ZIP and runs offline with `python3 -m http.server 8000` from the website folder, with no external dependency. The code folder has `reproduce.py` and `reproduce.ipynb` reading only `website/data.json`; `expected-output.txt` is a clean run. Both session plans are printed, 3-session and 4-session; neither is the only one. The accepted version as submitted on 24 Aug 2026 is kept unchanged on the hosted website and is not in the ZIP; all downloads are the final version. `neurips2026-education-materials-final.zip` contains `README.md`, `USER-GUIDE.md`, the proposal, notes, slides (both PDFs), the website folder, code, sources, licences and `MANIFEST.txt`; `neurips2026-website-source.zip` contains the website folder alone.

Licence: content CC BY 4.0, code Apache 2.0.

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`.

Thank you

Close with the one sentence to remember: architectural governance means make the prohibited effect unreachable, so that following the rule is not a term in the outcome. In the running example, the deploy agent should be unable to deploy without a human's approval, not merely asked.

Both authors are at the Graduate Program in Applied Computing (PPGCA), UNISINOS, São Leopoldo, Brazil; the corresponding author is Eptácio Vicente do Nascimento Neto. The status is accepted at the NeurIPS 2026 Education Track, Submission 10, final version, November 2026 (November 2026).

The deck continues with a two-slide glossary appendix, slides 42 and 43, so that every term used in the lecture can be looked up without the lecture notes. The deck is 43 slides in all.

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 (1 of 2)

Glossary appendix, first of two slides, with the entries word for word from the merged glossary that is shared by the lecture notes, the website and these slides. Twelve terms, plus viability conditions.

Access, action class, agentic AI system, enforcement ladder, fail-open and fail-closed, governance gap. The enforcement ladder is an author-proposed term, not an established standard. Use this slide when a learner meets a term on any slide and wants the plain definition; the full glossary has 55 entries and is in the lecture notes appendix and on the website. Notes: Appendix, Glossary of the lecture notes.

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.

Appendix · Glossary (2 of 2)

Glossary appendix, second of two slides, again word for word from the merged glossary: guardrail, instruction-based governance, mandate, structural control, supervision point, supervision residue and viability conditions. The supervision residue is an author-proposed term, not an established standard, and is relative to the catalogue. Level I, II and III are defined in the table on slide 14, and the loop vocabulary on slide 6. IDs used in the deck: AC action class, GR guardrail, L layer, CS condition set, SP supervision point, VC viability condition, SD supervision determinant, MG metric, F durable finding, EV evidence file; Level I, II, III are autonomy levels and rung 0 to 3 are enforcement rungs. Do not confuse L1, a layer, with Level I, or CS with SD. The full 55-entry glossary is in the lecture notes appendix and on the website. This is the last slide of the 41. Notes: Appendix, Glossary of the lecture notes.