AI 2026: Frontier Awareness Without the Hype

Volume II: Leading Applications Across Domains

Applications only matter when institutions can evaluate, control, and defend them.

Alejandro Reynoso

Chief Scientist, DEFI CAPITAL RESEARCH

External Lecturer, Judge Business School, University of Cambridge

February 2026

Reader Notice (Scope, Reliance, and Professional Responsibility)

Educational purpose only. This volume is provided solely for educational and informational purposes. It is not legal advice, not compliance advice, not medical advice, not engineering certification guidance, and not a substitute for professional judgment, institutional policy, or counsel. Application narratives can create false confidence if evaluation and controls are weak. **Verification gate:** do not rely until confirmed.

Connection to Volume I (required). Volume II assumes the governance posture established in Volume I: evaluation before enthusiasm; control points before autonomy; and auditability as a prerequisite for scale. If Volume I is the discipline layer, Volume II is the application layer. The two volumes are designed to be read as a single concatenated work: research and governance first, then applications under those constraints.

No operational authority; no autonomous decision rights. Nothing in this volume should be implemented as an autonomous system that can commit an institution to irreversible action. Where workflows, controls, or technical patterns are described, they are presented as conceptual scaffolds and must be adapted, tested, and approved within the reader's governance environment.

Security, privacy, and sensitive data. Do not paste confidential, regulated, or identifying information into unapproved tools. Use minimum-necessary inputs, redaction, and environment controls. If uncertainty exists, treat the content as sensitive and escalate per policy. Governance must harden as capability increases.

Preface

If Volume I is the map of frontier AI capability and its governance constraints, Volume II is the terrain: the domains where AI systems are now being applied in ways that create real value and real institutional exposure. This volume is written with an explicit bias: applications are only as strong as the evaluation and control regime that surrounds them. In other words, the question is not, "Can the model do it?" The question is, "Can the institution supervise it, reproduce it, and defend it?"

The purpose of Volume II is to show how the frontier ideas from Volume I express themselves in concrete environments: science and wet-lab adjacent discovery, physics and simulation surrogates, finance under hard constraints, interpretive work where meaning is contested, and multimodal systems where perception becomes action. These domains share a pattern. The more the model touches the physical world, regulated decisions, or public claims, the less you can tolerate silent failure. Each chapter therefore treats governance as a first-class design variable: what is being optimized, what can go wrong, what must be checked, and which control points are non-negotiable.

The five chapters of this volume are the second half of the AI 2026 argument. They are not mere case studies. Each chapter is a template for disciplined thinking about application risk: how evaluation changes when ground truth is expensive, when feedback loops create optimization pressure, when errors are persuasive rather than obvious, and when tools extend the model's reach. The target outcome is a reader who can look at an AI application proposal and ask the decisive questions: what evidence exists, what tests are run, what failure modes are expected, where the human boundary sits, and what monitoring makes drift visible. **Facts are not assumptions.** Facts must be provided or verified; assumptions must be stated, owned, and testable.

How to Use This Volume

This volume is designed to be read as a direct continuation of Volume I. The first volume established the foundations: why agentic behavior creates an evaluation bottleneck, why reasoning introduces new risk surfaces, why representation-level control is emerging as a governance lever, why retrieval and memory change failure modes, and why planning and search create the operational boundary of autonomy. Volume II applies that same discipline to domains where consequences are concrete.

Read the chapters in order. They are chosen to cover a spectrum of failure costs and validation regimes. Chapter 6 focuses on scientific and discovery workflows, where progress depends on scarce ground truth and where evaluation must be designed around experimental confirmation rather than persuasive text. Chapter 7 addresses physics and simulation surrogates, where the key risk is silent physical invalidity and the control challenge is enforcing operating envelopes under distribution shift. Chapter 8 covers finance under constraint, where errors propagate through decisions, disclosures, and incentives, and where governance must treat evaluation and recordkeeping as part of the system. Chapter 9 examines interpretive AI in knowledge work, where truth is often contested and the risk is epistemic drift, framing manipulation, and attribution failure. Chapter 10 treats multimodal systems, where perception flows into narrative and then into action, making provenance, gating, and monitoring decisive.

Each chapter is supported by a companion Google Colab notebook available on GitHub. The notebooks are part of the instructional design: they convert the chapter's conceptual frame into executable, inspectable scaffolds that can be rerun and compared over time. Each notebook corresponds one-to-one with its chapter and mirrors its emphasis, so the text provides the mental model and the notebook provides the runnable workflow skeleton.

The GitHub format is intentional. It makes versions explicit, enables transparent updates, and supports reproducibility. In this volume, reproducibility is not a cosmetic feature; it is what makes governance real. The notebooks therefore prioritize traceability of inputs, staged outputs, control points, and artifact generation rather than maximal performance demonstrations. Use them as a disciplined training environment: run the notebook end-to-end, inspect intermediate outputs, stress the workflow, modify assumptions, and rerun. The objective is not to be impressed by a result; it is to understand what must be measured, what must be controlled, and what evidence is required before an institution should rely on systems of this class.

A practical rhythm is recommended. For each chapter: (i) read for the capability and failure model, (ii) extract the minimum control set implied by the domain, (iii) run the companion notebook from beginning to end, (iv) review outputs and logs as if you were a supervisor or reviewer, and (v) record what you would require to approve or block deployment in your own environment. This is the intended outcome of Volume II: application fluency that remains constrained by evaluation and governance discipline.

Contents

Chapter 1

Chapter 6 — Generative Discovery: Chemistry, Biology, and Lab-Adjacent AI

Abstract. This chapter examines generative chemistry and materials design as a frontier domain where foundation models and diffusion-like generators act as proposal engines in extraordinarily large structured spaces. The operational promise is straightforward: accelerate discovery by producing candidate molecules, polymers, catalysts, and materials conditioned on desired properties. The governance reality is less forgiving: the moment a generator can propose plausible structures at scale, the bottleneck moves from ideation to validation, and the error modes become physical rather than textual.

The chapter builds an executive-relevant mental model: generation is cheap, evaluation is expensive, and proxy objectives are dangerously easy to exploit. A model can produce syntactically valid representations that score well under learned predictors while remaining non-synthesizable, unstable, or unsafe. The distinction between *valid* and *deployable* becomes the core managerial boundary: validity concerns representational well-formedness, while deployability requires conservatively gated evidence across uncertainty, feasibility, hazard screening, and provenance.

Technically, the chapter frames modern discovery pipelines as constrained search with iterative loops: propose candidates under conditions, score with predictors and/or simulators, filter using feasibility and safety constraints, and selectively escalate to higher-cost validation. It formalizes the optimization problem, emphasizes uncertainty calibration and out-of-distribution detection under optimization pressure, and explains why many common benchmarks under-measure the risks that matter. The chapter concludes with a governance-first control set: explicit constraint definitions, auditable logging of candidate generation and selection decisions, role separation between proposing and approving, and deployment deferral criteria when safety filters and uncertainty controls cannot be defended.

1.1 Orientation and Scope

1.1.1 Motivation and context

Chemistry and materials science sit at an awkward intersection of enormous upside and non-negotiable physical reality. In purely digital domains, an AI system can be wrong and still be harmless, or at least cheaply corrected: you rerun the query, adjust the prompt, revert the commit, or roll back the model. In chemical and materials design, the same posture is institutionally reckless. Here, outputs are not merely text; they are candidate structures that can propagate into simulation pipelines, lab automation queues, procurement orders, and eventually experiments whose cost profile is measured in days, budgets, and safety controls rather than milliseconds. This is precisely why the domain is a frontier application for generative AI: the value proposition is unusually clear, and the governance burden is unusually unforgiving.

From an executive perspective, the attraction is simple to state. Drug discovery, catalysis, battery chemistry, polymers, coatings, industrial separations, and advanced semiconductors all share a structural problem: the search space is vast, the success rate is low, and the experimental feedback loop is slow and expensive. Traditional discovery pipelines cope through heuristics, expert intuition, and incremental search over known families. Generative models offer a different approach: learn a distribution over feasible candidates and propose new structures conditioned on desired properties. Even in early, constrained deployments, this changes the economics of ideation. Where a team once curated dozens of candidates per week, a generative system can propose thousands per hour, and it can do so in a targeted way by conditioning on performance objectives or constraints.

But that apparent productivity is exactly where governance must enter the room and take a seat at the head of the table. The central managerial fact in generative science is that generation is cheap and evaluation is expensive. The model can produce candidates at scale, but the institution cannot validate at the same scale. This asymmetry invites systematic failure modes that are not moral panics; they are simple consequences of optimization and selection under uncertainty. If an organization rewards high predicted scores, it will receive candidates that score highly under predictors—whether or not those scores correspond to real, reproducible physical behavior. The institution risks building a pipeline that manufactures confidence rather than evidence.

The right framing for leadership is therefore not, “Can a model generate valid molecules or materials?” Validity is necessary but not sufficient, and it is often the wrong first question. The correct question is: *What does the organization consider deployable, and what governance gates turn model output into a defensible experimental decision?* Deployability is an institutional claim, not a syntactic property. It requires that generated candidates be (i) meaningfully constrained by feasibility, safety, and provenance rules; (ii) evaluated under calibrated uncertainty; and (iii) advanced through a staged validation funnel with explicit stop conditions and accountable approvals. In other words, the domain forces executives to confront a principle that is easy to say and hard to operationalize:

when AI touches the physical world, the cost of being wrong is not a bad paragraph; it is the misallocation of scarce experimental capacity and the creation of safety and liability exposure.

This chapter treats generative chemistry and materials design as the first application domain in *AI 2026* precisely because it reveals the frontier pattern in its cleanest form. It is not primarily a story about clever architectures. It is a story about how capability shifts bottlenecks, how new failure modes emerge when proposals scale faster than validation, and how governance must be designed as a system of constraints, review roles, evidence thresholds, and auditable traces. The goal is to provide a disciplined mental model that lets senior practitioners recognize the opportunity without importing the wrong intuitions from language generation. In a laboratory pipeline, the phrase "the model suggested it" is not an explanation. It is a risk event.

1.1.2 Why this topic is at a frontier moment

This topic is at a frontier moment because three curves are crossing at once: model capability, integration feasibility, and organizational appetite for accelerated discovery. The first curve is the maturation of generative modeling techniques for structured objects. Earlier waves of molecular generation relied on representations and models that produced plausible strings or graphs but struggled to capture three-dimensional geometry and physical constraints. More recent approaches have pushed into 3D-aware generation and equivariant architectures that treat geometry as a first-class citizen rather than an afterthought, substantially improving the plausibility of generated conformations and structures [1, 2]. Even when such methods are imperfect, they move the field from "toy generation" toward pipelines that can plausibly integrate with simulation and downstream evaluation.

The second curve is operational integration. A few years ago, even enthusiastic teams faced a messy stack: disconnected data sources, inconsistent representations, and brittle code paths that made it hard to place a generative model into a discovery workflow without turning it into a perpetual science project. That is changing. Tooling for molecular representation, property prediction, and retrosynthesis planning has matured; benchmarks and standard datasets have stabilized expectations; and lab workflows increasingly include computational triage as a natural stage-gate. The result is that a generative module can be treated not as a research demonstration but as a component within a larger discovery factory: propose candidates, filter them, simulate them, and selectively validate.

The third curve is strategic demand. Across pharmaceuticals, energy storage, and industrial chemistry, competitive pressure has risen while R&D productivity has not kept pace. At the same time, the opportunity cost of under-exploring the design space is increasingly visible. Organizations now treat accelerated discovery not as a speculative bet but as a strategic necessity: reducing time-to-candidate, increasing the diversity of explored structures, and improving the efficiency of experimental allocation. This is the executive "pull" that turns a promising method into a frontier deployment question.

However, the frontier moment is not simply about acceleration; it is about governance capacity. As generative models become more integrated, the institution must answer questions it previously deferred. Who owns the definition of constraints (toxicity proxies, hazard class exclusions, synthesis feasibility rules, IP provenance requirements)? Who is accountable for the selection policy—the mechanism that chooses which candidates deserve expensive validation? What constitutes sufficient uncertainty calibration to treat a predictor score as a triage signal rather than a false guarantee? These questions cannot be postponed indefinitely once a pipeline can produce candidates faster than humans can review them.

A key inflection in organizational risk occurs when the model moves from "idea generator" to "proposal engine." An idea generator can be safely sandboxed: it suggests possibilities, and experts decide whether any are worth attention. A proposal engine, by contrast, can produce ranked lists, filtered sets, and "top candidates" with accompanying predicted metrics. The mere existence of ranking is powerful: it shapes attention, budget allocation, and perceived confidence. In regulated or safety-sensitive contexts, that influence is itself a control surface. If a model ranks candidates, the organization needs governance around the ranking function, the calibration of the scoring models, and the documentation of how candidates were filtered and selected. Otherwise, the institution creates a subtle form of decision laundering: not the automation of a decision, but the automation of the *appearance* that the decision was evidence-based.

This chapter therefore treats the frontier moment as an institutional design challenge. The question is not whether diffusion models, transformers, or graph generators are impressive—they are. The question is whether the organization can construct a validation funnel that converts model output into defensible evidence. That funnel must respect scarce experimental capacity, incorporate uncertainty, enforce safety constraints, preserve provenance, and maintain audit trails. Without such a funnel, frontier capability becomes frontier exposure: faster generation simply means faster accumulation of unvalidated candidates, faster consumption of attention, and faster drift toward proxy metrics that look scientific but are not.

1.1.3 What has changed recently

The most important recent change is conceptual: diffusion-style generative modeling and large foundation models have expanded beyond images and text into structured scientific objects, and the field has begun to treat "generation" as a conditional, controllable process rather than an unconditional novelty machine. In molecular design, this manifests in two broad directions. First, models are increasingly built to respect symmetries and geometry. Equivariance to rotations and translations matters because molecules exist in three-dimensional space, and physical properties are often functions of geometry rather than of a string encoding. Methods that build these invariances into the generative process reduce the gap between what the model can output and what a simulator or experimentalist would consider meaningful [1, 2]. Second, conditioning has become more practical:

models can be guided toward candidates that match target properties, constraints, or textual specifications. This does not solve the validation problem, but it changes the operational shape of the pipeline: instead of generating broadly and filtering heavily, systems can generate in a narrower region of interest, potentially reducing wasted evaluation.

A second change is that the community has become more explicit about benchmarks and measurement. Benchmarking platforms such as GuacaMol and MOSES created shared baselines for molecular generation and made visible a problem that executives should take seriously: many "good" generative metrics are proxies that can be satisfied without producing practically useful candidates [6, 7]. Benchmarks often emphasize validity, novelty, diversity, and distribution matching. These are important, but they do not guarantee synthesizability, stability, safety, or downstream performance. The very existence of widely used benchmarks creates a governance hazard: teams can report progress against metrics that are legible to management while quietly failing to address feasibility and validation constraints that matter in deployment. The antidote is not to reject benchmarks, but to contextualize them: treat them as unit tests for components, not as acceptance criteria for real-world pipelines.

A third change is that the feasibility layer has improved. Historically, generative chemistry could easily produce "valid" molecules that were chemically nonsensical or extremely difficult to synthesize. The field responded by developing heuristic measures and learned models for synthetic accessibility and retrosynthesis planning [8, 9]. These methods are imperfect, but they shift feasibility from a purely human judgment into a semi-automated filter stage. This shift matters organizationally: it enables a multi-stage funnel in which cheap computational screens can eliminate obviously impractical candidates before humans or labs spend time on them. It also introduces a new governance requirement: feasibility screens themselves become predictors subject to exploitation and calibration failure. If the system optimizes against a synthetic accessibility proxy, it can drift toward the quirks of that proxy rather than toward truly synthesizable chemistry. The lesson is not that feasibility scoring is bad; it is that every learned screen becomes a control point requiring monitoring, calibration, and periodic reevaluation.

A fourth change is the recognition of dual-use and hazard risks as first-class considerations in generative discovery. In drug discovery, the same tools that can accelerate therapeutic design can, under different objectives, accelerate the design of harmful compounds or lower the barrier to malicious exploration. The literature has explicitly raised the dual-use concern of AI-powered discovery [10]. This matters for governance because it expands the control requirements beyond ordinary model risk management. It introduces access controls, usage policies, monitoring for suspicious objective specifications, and a clear separation between what the system can propose in a sandbox and what it can export into downstream workflows. In other words, the frontier shift is not merely technical; it is sociotechnical. Institutions deploying these systems must implement controls that anticipate misuse pathways and accidental hazard generation, even if the organization has no malicious intent.

A fifth change is the consolidation of representations and the practical acceptance that no single representation is sufficient. Molecules and materials can be represented as strings (e.g., SMILES-like encodings), graphs, or 3D coordinates; each has advantages and failure modes. Earlier generations of models often committed to a single representation and inherited its blind spots. More recent pipelines are pragmatic: they translate between representations, canonicalize outputs, and use different models for different stages (generation, property prediction, feasibility screening). That modularity is itself a governance opportunity. It enables role separation and control separation: the generator proposes, the predictor scores, the feasibility module filters, and the selection policy decides. Each module can be audited and stress-tested independently. Conversely, if the pipeline collapses into a monolith, interpretability and accountability collapse with it.

Taken together, these changes shift the executive conversation away from the spectacle of generation and toward the discipline of validation. The frontier is not "models can generate molecules." That statement has been true for years [3, 4, 5]. The frontier is that models can generate at a level that invites integration into real workflows, and that integration forces institutions to confront the hard questions: which constraints are binding, which proxies are safe enough to triage, and which stages require human accountability.

1.1.4 Explicit exclusions and non-goals

This chapter is intentionally bounded. It is not a chemical safety manual, not a synthesis guide, and not an operational recipe for generating hazardous compounds. The purpose is to explain the governance-relevant structure of generative discovery systems and to provide a disciplined conceptual framework for executives and senior practitioners who must make decisions about adoption, oversight, and risk controls. That means some topics are explicitly out of scope.

First, we do not provide instructions, heuristics, or workflows for synthesizing compounds, improving yields, bypassing safety procedures, or selecting target classes for harmful use. The technical discussion is about generative modeling as a proposal mechanism and about governance controls that constrain proposals and gate escalation. It is framed to reduce misuse, not to enable it. Where we discuss representations, conditioning, and optimization, we do so at the level necessary to understand failure modes such as proxy exploitation and distribution shift, not at the level required to operationalize a harmful pipeline.

Second, we do not claim that generative AI replaces laboratory validation or domain expertise. In fact, the opposite is the organizing principle: as generation becomes easier, validation becomes the bottleneck and must be treated as a scarce institutional resource. A mature program should invest more in validation discipline (uncertainty calibration, feasibility screening, hazard filters, provenance tracking, and stage-gated escalation) than in the raw capacity to generate candidates. An organization that invests primarily in generation capacity is likely optimizing for internal demos rather than for real R&D outcomes.

Third, we do not attempt a comprehensive survey of generative models in chemistry and materials science. The literature is broad and evolving, spanning string-based generation, graph generation, 3D equivariant diffusion, reinforcement learning, and hybrid methods, with multiple competing benchmark suites and datasets. This chapter is not an encyclopedia. It is a governance-first map of the territory that highlights a small set of architectural patterns, recurring failure modes, and control points that matter for decision-makers.

Fourth, we do not assume that property predictors are authoritative. Predictors are treated as probabilistic instruments that must be calibrated, monitored, and stress-tested, especially under optimization pressure. If the model is allowed to search for candidates that maximize predictor scores, it will tend to discover regions where predictors are least reliable. This is not a bug; it is a property of optimization. Therefore, any serious deployment must incorporate uncertainty thresholds, out-of-distribution detection, and conservative gating. The chapter emphasizes why these mechanisms are not optional.

Fifth, we do not resolve the complex legal and IP questions that arise when generative systems are trained on proprietary or public datasets and then produce novel candidates. We treat provenance and documentation as governance requirements, but we do not provide legal interpretations. The chapter's stance is practical: if an organization cannot document the lineage of training data, model versions, conditioning inputs, and candidate selection decisions, it cannot defend the outputs under scrutiny—whether that scrutiny comes from regulators, partners, investors, or internal audit.

These exclusions are not evasions; they are part of the discipline of the book. *AI 2026* is not written to impress the reader with technical maximalism. It is written to help institutions avoid predictable failure. In generative science, the predictable failure is to confuse a plausible output with a deployable candidate, to treat predictor scores as evidence, and to build pipelines whose speed outpaces governance capacity. This chapter's non-goal is to accelerate that failure.

1.1.5 Role of this paper in AI 2026

Chapter 6 opens the applications half of *AI 2026* with a science-first domain because it makes the book's thesis painfully concrete: frontier AI is not merely about better answers; it is about systems that can influence real-world decisions under uncertainty. Chemistry and materials design are ideal as the first application because they expose the fundamental asymmetry that governs all high-impact deployments: scaling capability without scaling evaluation and control produces fragility, not advantage.

In Chapters 1 through 5, the book builds a conceptual ladder. Chapter 1 argues that when systems act over time, evaluation must move from outputs to trajectories. Chapter 2 shows that deeper reasoning expands the risk surface through delayed and confident failures. Chapter 3 explains why internal control mechanisms (representation engineering) are powerful but fragile levers that must be governed. Chapter 4 reframes retrieval and memory as institutional design choices that

determine what evidence is visible and auditable. Chapter 5 explains how planning and search amplify objectives and therefore amplify both success and failure. Chapter 6 is the first place where those ideas collide with physical reality.

Generative discovery is, in effect, planning and search in a structured scientific space, with a generator proposing actions (candidate structures) conditioned on objectives, and with predictors and simulators providing approximate feedback. The pipeline is an optimization machine. That makes it a natural demonstration of why objective specification and constraint definition are governance issues, not engineering details. If the objective is "maximize predicted activity," the system will find ways to maximize predicted activity—including by exploiting predictor blind spots. If the constraints are underspecified, the system will produce candidates that satisfy what is measured and violate what is merely assumed. This is the same pattern Chapter 5 warned about, now translated into molecules and materials.

Just as importantly, the chapter establishes a template that later application chapters will reuse. In physics surrogates (Chapter 7), the key risk is error accumulation under rollout and the inadequacy of one-step accuracy. In finance (Chapter 8), the key risk is decision laundering and the need for reviewable, constrained support systems rather than autonomous optimization. In interpretive knowledge work (Chapter 9), the key risk is epistemic drift away from evidence. In multimodal systems (Chapter 10), the key risk is action gating when perception and actuation combine. Chapter 6 prepares the reader to recognize the common structure: a system that proposes at scale, a bottleneck that shifts to validation, and a governance requirement that becomes more important precisely because the model becomes more capable.

The chapter's role is therefore foundational for Part II. It teaches the reader to separate three layers that are too often collapsed into one: *generation* (the ability to propose candidates), *evaluation* (the ability to estimate which candidates are worth attention), and *validation* (the ability to establish defensible evidence under real constraints). Frontier AI strengthens generation dramatically; it strengthens evaluation unevenly; it cannot eliminate validation. When organizations forget this hierarchy, they accidentally build systems that are extremely good at producing hopeful artifacts and extremely bad at producing justified decisions. Chapter 6's purpose is to prevent that mistake, and to provide governance-minded leaders with a vocabulary for asking the questions that determine whether "AI for discovery" is a disciplined capability or a high-cost mirage.

Finally, this chapter serves a rhetorical purpose for the book as a whole. "Frontier awareness without the hype" means the reader must leave with both opportunity clarity and risk clarity. Chemistry and materials make that balance unavoidable. The upside is enormous; the risks are not abstract; and the governance mechanisms are tangible: constraints, uncertainty thresholds, stage gates, role separation, and auditable logs. If a reader can learn to reason correctly about generative discovery, they will be better equipped to reason correctly about every other frontier deployment discussed in the chapters that follow.

1.2 Conceptual Abstraction

1.2.1 Core abstraction

The governing abstraction for generative chemistry and materials design is deceptively simple: *design is search in a vast structured space under imperfect evaluation.* A candidate object—a molecule, polymer, crystal, catalyst surface, or composite—is an element x of a structured domain $\mathcal{X}$ with grammar-like constraints and physical meaning. The institution has a set of desired properties $f(x)$ (performance, stability, selectivity, conductivity, binding affinity, etc.) and a set of constraints $g_i(x)$ (feasibility, hazard exclusions, cost, IP provenance requirements, regulatory constraints). In classical discovery, the hard part was producing plausible candidates to test; in modern generative discovery, the generator can produce candidates cheaply, so the hard part becomes selecting which candidates deserve scarce validation effort.

In this framing, a generative model is not the decision-maker and not even the optimizer. It is a *proposal mechanism* that provides a distribution over candidates. Conditioned on a context c (which can include property targets, constraint hints, or textual specifications), the generator samples candidates $x \sim p_\theta(x \mid c)$. The organization then evaluates those candidates through scoring functions, predictors, and simulators—none of which are perfect. The outputs of those evaluators are not truths; they are estimates that must carry uncertainty, and they must be treated as triage signals rather than as warrants for action.

This abstraction immediately clarifies why executives should care about governance. Search under imperfect evaluation is the setting in which proxy failures are inevitable unless constraints and uncertainty controls are explicit. If the organization selects candidates primarily by predicted score, and if the search procedure is allowed to iterate (generate many candidates, keep only the best, feed back into the next generation), the pipeline becomes an optimization machine that will push toward regions of $\mathcal{X}$ where predictors are least reliable. This is not a moral hazard; it is a mathematical tendency. In any high-dimensional space, the extremes of a noisy predictor are dominated by error rather than by signal once enough samples are drawn. The model does not need to be adversarial to cause this; a naive selection policy is sufficient.

The abstraction also separates *validity* from *deployability.* Validity is representational: is x a well-formed object according to the encoding (a parseable string, a chemically plausible graph, a stable 3D conformation)? Deployability is institutional: does the evidence justify escalation into higher-cost, higher-risk stages of the pipeline? Deployability requires constraints, provenance, and uncertainty gates. A valid candidate can be non-deployable for many reasons: it may be infeasible to synthesize, it may fall into a hazard class, it may lie outside the training distribution of the predictor, or it may be novel in a way that implies uncalibrated uncertainty rather than strategic advantage.

Finally, the abstraction implies a critical organizational lesson: *the generator is not the bottleneck;*

the governance of selection is. If an institution cannot clearly define what it is optimizing for, how it trades off objectives, and what constraints are binding, then increasing the generator's capacity simply increases the volume of candidates that must be triaged. This produces the illusion of progress while increasing the probability of expensive mistakes. In a well-governed program, the generator's output is treated as an input to a disciplined decision funnel. In a poorly governed program, the generator becomes a machine that produces scientific-looking artifacts faster than the organization can responsibly interpret them.

1.2.2 Key entities and interactions

A generative discovery pipeline can be described as a small number of entities interacting in a loop. The specific modeling choices vary, but the governance-relevant structure is stable. The first entity is the *representation layer*. Candidates must be encoded in a form that models can generate, predictors can score, and downstream systems can consume. Depending on the domain, representations include string encodings, graphs, and geometric structures; in materials, they may include lattices, compositions, or microstructure descriptors. Representation is not a neutral technicality: it defines what kinds of constraints can be expressed, what kinds of validity checks are possible, and what kinds of artifacts can be logged and audited.

The second entity is the *constraint layer*. Constraints are explicit rules that determine which candidates are admissible at each stage. Some constraints are hard (e.g., exclude hazard classes, enforce maximum molecular weight, enforce allowed element sets, enforce known safe scaffold restrictions, require provenance tags). Others are soft (e.g., prefer candidates with higher predicted stability, prefer lower cost). The key governance point is that constraints should be declared as a registry with owners, not as scattered heuristics embedded in code. If a constraint is important enough to matter, it is important enough to be versioned and auditable.

The third entity is the *generator*. This can be a diffusion model, an autoregressive transformer, a variational model, or a hybrid architecture. What matters conceptually is that it proposes candidates conditioned on c. The conditioning signal itself is a governance surface: it encodes what the organization is asking for. If the conditioning includes only a single performance objective, the generator will tend to trade off unspoken constraints for that objective unless constraints are enforced elsewhere. If the conditioning includes multiple property targets, the generator still needs a selection policy that decides how to trade off those targets, and that policy must be justified.

The fourth entity is the *scorer*. Scoring includes property predictors, simulators, feasibility estimators, and hazard screens. The scorer produces a vector of estimated properties $\hat{f}(x)$ and uncertainties $\hat{\sigma}(x)$ (explicitly or implicitly), along with auxiliary annotations such as out-of-distribution flags, similarity metrics, and provenance status. In a well-designed pipeline, the scorer is not a single scalar. It is a bundle of evidence, with each component having known failure modes and calibration status. The scorer should be treated as a probabilistic instrument, not as an oracle.

The fifth entity is the *selection policy*. This is the part that is most often neglected and most often responsible for institutional failure. The selection policy takes candidates and their scores and decides what happens next: discard, keep for further computational evaluation, escalate to more expensive simulation, or propose for experimental validation. It is here that the system becomes operationally meaningful. It is also here that the risk of decision laundering emerges. If the selection policy silently converts uncertain scores into confident rankings and then presents a short list as "top candidates," it creates an impression of scientific justification even when the justification is weak. Therefore, the selection policy must encode explicit gates: uncertainty thresholds, diversity requirements, novelty caps, safety exclusions, and human approval triggers.

These entities interact in a loop that can be summarized as: *propose* → *canonicalize* → *score* → *filter* → *select* → *refine*. Canonicalization is a critical but often invisible step: it ensures that representations are normalized so that duplicates are detected, constraints are consistently applied, and audit logs can be meaningfully compared across runs. Filtering applies constraints; selection applies trade-offs and gates; refinement updates the conditioning or the generator based on what was learned.

Two additional interactions matter for governance. The first is the *human-in-the-loop interface*. At some stage, a human expert should review candidates and decide whether escalation is warranted. The interface must be designed to present evidence rather than persuasion: predicted properties with uncertainty, constraint pass/fail status, similarity to known families, provenance notes, and reasons for selection. The second is the *feedback channel*. In active learning or iterative discovery, experimental results feed back into the predictors and sometimes into the generator. Feedback is powerful but destabilizing: it can cause the model to overfit to recent experiments, drift toward narrower families, or amplify biases in the experimental program. Therefore, feedback must be governed as a change-management process with versioning, performance monitoring, and rollback capability.

Seen through this lens, the pipeline is not a model; it is a *sociotechnical control system*. The generator proposes, the scorer estimates, the policy gates, the human approves, and the lab validates. Each stage must be designed to prevent the central failure: a candidate being treated as credible merely because it was produced by a sophisticated model. Sophistication is not evidence. Evidence is evidence.

1.2.3 What is being optimized or controlled

The first and most important point is that generative discovery is almost always multi-objective, even when organizations pretend otherwise. Performance objectives are rarely singular. A drug candidate must not only bind a target but also avoid toxicity, maintain bioavailability, and be synthesizable at reasonable cost. A battery material must not only store charge but also be stable, manufacturable, and safe across operating conditions. A catalyst must not only accelerate a reaction

but also resist deactivation and remain selective. In practice, the organization is optimizing a weighted compromise across objectives, and the weights encode strategic priorities and risk appetite.

Therefore, the true object of control is not the generator's ability to produce high-scoring candidates; it is the institution's ability to enforce *constraints, trade-offs, and exploration boundaries* in a way that remains defensible. This is where governance meets mathematics. Without explicit constraints, optimization drifts toward hidden failure. Without explicit trade-off rules, the system's behavior becomes a function of model quirks rather than of strategy. Without exploration boundaries, the system will push into uncertain regimes where predictors are unreliable and hazards are under-screened.

At a formal level, one can think of the system as trying to maximize an expected utility under uncertainty:

$$\max_{x \in \mathcal{X}} \mathbb{E}\left[u(f(x))\right] \quad \text{s.t.} \quad g_i(x) \leq 0,$$

but the governance reality is that $f(x)$ is unknown and only estimates $\hat{f}(x)$ are available. The system therefore operates on a surrogate objective that includes predictors and uncertainty penalties. A conservative selection policy might, for example, prefer candidates with high lower-confidence bounds rather than high point estimates. Whether or not such a policy is used, the key is that the pipeline is selecting under uncertainty, and uncertainty must be treated as a control variable rather than as an inconvenient detail.

The second object of control is *proxy exploitation.* Predictors and scoring models are proxies for reality. When a pipeline repeatedly selects candidates that maximize a proxy, it applies pressure that reveals the proxy's weaknesses. This is the generative discovery analogue of "reward hacking" in reinforcement learning: the system finds ways to score well under the proxy without satisfying the real intent. Importantly, this can happen even when every actor is well intentioned. The mere fact that the selection policy prefers high predicted scores is enough to create pressure. The result is a systematic inflation of predicted performance and a systematic degradation of real performance once candidates are validated.

To control proxy exploitation, institutions can use several mechanisms, all of which reflect governance choices. They can (i) add uncertainty penalties or require calibrated confidence; (ii) cap novelty and enforce similarity constraints to keep exploration within well-understood regions; (iii) use ensembles of predictors and require agreement; (iv) include adversarial stress tests where candidates are scored under perturbations or alternative models; and (v) enforce diversity to avoid selection collapse onto a narrow, over-optimized family. None of these mechanisms is free; each reduces the apparent "best score" but improves the credibility of the funnel.

The third object of control is *exploration.* Generative systems can be configured to explore broadly or narrowly. Broad exploration increases novelty and the chance of discovering truly new families, but it also increases uncertainty, hazard risk, and feasibility failures. Narrow exploration reduces these risks but can lead to incrementalism and missed opportunities. The governance question

is not which is "better" in the abstract; it is which is aligned with the organization's stage, risk tolerance, and validation capacity. Early-stage programs should usually prefer narrow exploration with conservative gates, because they are still learning how to calibrate predictors, define constraints, and run an auditable pipeline. As governance maturity increases, exploration can widen in controlled ways, with explicit policies that define how far the system is allowed to extrapolate.

The fourth object of control is *information leakage and provenance*. Conditioning signals, training data, and generated candidates may encode proprietary knowledge or sensitive objectives. In materials design, the composition space may embed trade secrets; in pharma, targets and constraints may reveal strategic intent. The system must therefore control what information flows into the generator, what is logged, what is exported, and what is shared. Provenance controls ensure that the institution can later explain where a candidate came from, what model version produced it, and what data and constraints were in effect. This is a governance requirement, not a cosmetic feature.

Finally, the system must control *human interpretation*. The presence of AI-generated candidates can create cognitive bias: humans may over-trust candidates because they appear quantitatively justified. A well-governed pipeline treats model outputs as hypotheses requiring evidence, and it designs review interfaces to encourage skepticism: show uncertainty, show constraint failures, show model limitations, and show alternative candidates. This is one reason role separation matters. The person incentivized to propose novel candidates should not be the same person accountable for approving escalation to expensive or risky validation.

In summary, what is being optimized is not merely performance; it is a constrained, uncertainty-aware utility that must remain defensible. What is being controlled is not merely generation; it is the entire selection and escalation process. The institution's competitive advantage is not that it can generate candidates. Many can. The advantage is that it can generate *responsibly*, turning cheap proposals into credible decisions without collapsing into proxy-chasing.

1.2.4 Distinction from prior paradigms

Traditional chemistry and materials discovery can be caricatured as a combination of human intuition, local search, and expensive screening. In practice, it uses a set of domain heuristics: explore within known families, modify functional groups, adjust compositions, measure, iterate. Even when computational models are used, they often operate as tools for triage rather than as engines of novelty. The implicit assumption is that candidate generation is scarce and that expert time is the main limiter. Under that assumption, the organization optimizes for producing a small number of high-quality candidates, each backed by substantial reasoning and experience.

Generative discovery inverts the scarcity. Candidates become abundant. The scarce resource becomes validation capacity: simulation time, lab throughput, and expert review bandwidth. This inversion changes the institutional economics of discovery. In the traditional paradigm, improving generation means improving human creativity and domain insight. In the generative paradigm,

improving generation is easy: scale the model, increase sampling, adjust conditioning. The harder work is designing filters and selection policies that prevent the organization from drowning in plausible but useless candidates.

Another distinction is the role of representation. Traditional pipelines often operate on human-friendly representations: chemical intuition, known motifs, and experimentally grounded descriptors. Generative pipelines operate on machine-friendly representations that may be less interpretable to humans. This creates a governance gap. If a candidate is generated as a graph or 3D structure with predicted properties, the human reviewer may not have the same intuitive feel for its plausibility. The pipeline must therefore provide interpretable summaries and evidence signals, or else it risks shifting decision-making from human judgment to model persuasion by default.

A third distinction concerns the nature of generalization. Traditional discovery generalizes through chemical principles and analogical reasoning grounded in known mechanisms. Generative models generalize through statistical learning over datasets, which may encode biases, gaps, and implicit constraints that are not explicitly documented. The result is that generative models can propose "novel" candidates that are novel only relative to the dataset, not relative to chemistry, or they can fail catastrophically when asked to extrapolate beyond the training distribution. In a traditional paradigm, extrapolation is constrained by human caution and mechanistic knowledge. In a generative paradigm, extrapolation can occur silently unless the pipeline implements explicit out-of-distribution detection and novelty bounds.

A fourth distinction is the shift from deterministic reasoning to probabilistic triage. In classical discovery, a candidate might be chosen because it satisfies a mechanistic hypothesis or because it aligns with an expert's understanding of structure-property relationships. In generative discovery, a candidate may be chosen because it ranks highly under a predictor ensemble. That ranking may be correct, but it may also be a statistical artifact. Therefore, a mature generative pipeline must incorporate uncertainty and calibration in a way that traditional pipelines could sometimes ignore. Executives should recognize that the more the organization relies on predictor-driven ranking, the more it must invest in model risk management techniques that are uncommon in conventional lab culture.

A fifth distinction concerns incentives and organizational behavior. Traditional discovery teams are often measured by experimental progress: validated results, published findings, patents, or successful prototypes. Generative pipelines introduce new intermediate artifacts: candidate lists, predicted scores, novelty metrics, and benchmark performance. These artifacts can become performance theater if not governed. Teams can show impressive dashboards of generated candidates without producing validated breakthroughs. This is not because teams are dishonest; it is because the availability of cheap intermediate artifacts makes it easy to substitute activity for progress. Governance must therefore define what counts as progress. In a generative discovery program, progress is not the number of candidates generated; it is the quality of the validation funnel, the calibration of predictors, and the evidence that selected candidates outperform baselines under controlled experiments.

The final distinction is the nature of failure. Traditional failures are slow and expensive but often legible: a synthesis fails, a material degrades, a compound is toxic. Generative failures can be fast and subtle: a system produces candidates that are systematically over-optimized to a proxy, leading to a long sequence of expensive experimental disappointments. The organization may not notice the failure until much later, because the early indicators (high predicted scores, high novelty, apparent diversity) look positive. This is why the generative paradigm demands different monitoring: track predictor calibration drift, track the gap between predicted and validated outcomes, track the distribution of novelty and uncertainty among selected candidates, and audit the decision logic that led to escalation.

In short, generative discovery is not merely a faster version of traditional discovery. It is a different institutional regime. It requires the organization to shift from a scarcity of ideas to a scarcity of evidence, from deterministic plausibility to probabilistic triage, and from human intuition as the primary bottleneck to governance and validation as the bottleneck. Treating it as "the same pipeline but with AI" is the surest way to spend money quickly and learn slowly.

1.2.5 Conceptual failure modes

The failure modes of generative discovery are not exotic; they are the natural consequences of optimizing under uncertainty in a structured space. The first and most central failure mode is *proxy hacking*. A proxy is any predictor score used as a substitute for real validation. When the system is allowed to generate many candidates and then select those with the highest proxy score, it will disproportionately select candidates whose high score is due to predictor error. Over time, the pipeline becomes a machine for discovering predictor blind spots rather than a machine for discovering real improvements. The empirical signature is painful but recognizable: predicted performance trends upward, validated performance does not, and the gap widens as sampling increases.

A closely related failure mode is *non-synthesizable outputs*. A candidate can be representationally valid and even chemically plausible, but practically infeasible to synthesize within reasonable constraints. Some infeasibility arises from complexity; some arises from unstable intermediates; some arises from requiring rare reagents or extreme conditions. Without feasibility filters, a generative pipeline can waste enormous amounts of expert time and experimental capacity. Even with feasibility filters, the system can over-optimize to the filter's proxy and produce candidates that pass the filter but still fail in practice. The governance response is staged feasibility gating: cheap heuristics first, then more expensive retrosynthesis or expert review, then experimental attempts only after evidence is strong enough.

A third failure mode is *hidden hazards*. The system may propose candidates that are hazardous (toxic, reactive, explosive, environmentally persistent) even if hazard was not an objective. This can occur through novelty: the model explores regions where hazard screening is weak or absent.

It can also occur through objective mis-specification: an innocent objective such as "maximize potency" can correlate with toxicity if not constrained. Hazard risk is amplified by scale. The more candidates generated, the higher the probability that some fall into problematic classes. The governance response is explicit hazard exclusion rules, monitored screening, and strict controls on export and escalation of candidates that trigger hazard flags.

A fourth failure mode is *dataset bias and coverage collapse*. Generative models learn from available data, which is rarely representative of the full design space. Public datasets over-represent certain chemical families; proprietary datasets reflect a company's historical strategy and may under-represent alternative approaches. A model trained on such data can appear impressive on in-distribution benchmarks while failing to propose genuinely diverse candidates. Worse, when used in a loop with selection, the pipeline can collapse diversity further: if selection favors candidates similar to historical successes, the generator is implicitly reinforced to stay within that narrow region. The result is an "innovation treadmill" that produces many variants of the same idea. The governance response is explicit diversity requirements, novelty monitoring, and periodic audits of chemical space coverage relative to strategic goals.

A fifth failure mode is *novelty without utility*. Generative systems can produce candidates that are novel in a representational sense but not useful in a scientific or commercial sense. Novelty metrics often reward distance from training data, but distance can mean nonsense as easily as breakthrough. A molecule can be novel because it is unstable, reactive, or violates implicit constraints not captured in the dataset. In materials, novelty can mean phases that are thermodynamically implausible. If an organization treats novelty as a KPI, it may optimize for novelty as an end in itself. Governance must therefore treat novelty as a constrained variable: useful only when paired with feasibility evidence and when aligned with a clear hypothesis about why novelty is valuable for the target application.

A sixth failure mode is *ranking-induced overconfidence*. When systems output ranked candidate lists, humans tend to anchor on the top-ranked items and treat the ranking as meaningful even when differences in scores are within noise. This cognitive effect is amplified when the ranking is presented with quantitative precision (e.g., many decimal places) that implies certainty. The governance response is interface discipline: present uncertainty, show confidence intervals or bins, avoid spurious precision, and require explicit justification for escalation beyond "it ranked highly."

A seventh failure mode is *feedback destabilization*. If experimental results are fed back into predictors or generators too aggressively, the models can become unstable. They may overfit to recent experimental regimes, forget broader patterns, or chase transient artifacts. The organization may interpret rapid model updates as agility, but without change management it becomes impossible to reproduce decisions or audit why a candidate was selected. Governance requires versioning, update cadence policies, performance monitoring on stable holdouts, and rollback capability.

An eighth failure mode is *traceability failure*. In high-accountability environments, the institution

must be able to reconstruct decisions: what model version generated a candidate, what conditioning was used, what filters were applied, what scores were produced, and who approved escalation. If these traces are not captured automatically, they will not be captured reliably. The result is organizational fragility: the program may work until the first serious question arises, at which point it cannot defend itself. Traceability is not a documentation afterthought; it is an operational requirement.

The unifying theme across these failure modes is that capability changes the error distribution. Generative systems do not merely make the old pipeline faster; they create new ways to be wrong at scale. The governance-first posture is therefore to assume that proxy failures will occur, that feasibility and hazard constraints will be tested, that datasets will bias outputs, and that humans will over-trust rankings. A successful program is not one that hopes these failures do not happen. It is one that designs the pipeline so that failures are detected early, contained cheaply, and corrected through explicit controls rather than through informal heroics.

1.3 Historical and Technical Lineage

1.3.1 Preceding approaches

Before generative foundation models became the headline act, chemistry and materials design already had a long history of computational assistance. The crucial point for governance-minded readers is that earlier approaches were not naive; they were constrained by the nature of the domain and by the economics of validation. They evolved under conditions where experiments were expensive, data was sparse, and failures carried real cost. This lineage matters because many of the controls that modern teams now rediscover—filters, feasibility checks, stage-gates, conservative thresholds—are not new inventions. They are the institutional memory of a domain that learned, repeatedly, that physical reality does not negotiate.

A canonical predecessor is QSAR (Quantitative Structure–Activity Relationship) modeling, which attempted to predict properties or biological activities from molecular structure using hand-crafted descriptors and statistical models. QSAR and its descendants treated prediction as an estimation problem: given a molecule, estimate toxicity, binding affinity, solubility, permeability, or other relevant properties. In practice, QSAR systems were embedded in workflows as triage tools. They reduced search cost by deprioritizing candidates likely to fail, but they rarely claimed to produce trustworthy rankings at the frontier of novelty. The limitations were well understood: models extrapolate poorly outside the data regime, correlations can be spurious, and prediction uncertainty grows rapidly as molecules diverge from the training set. The culture that grew around QSAR was implicitly governance-aware: it assumed that predictions were probabilistic hints that needed confirmation.

Cheminformatics pipelines extended this with richer representations and data management. They provided canonicalization, similarity searches, fingerprinting, scaffold analysis, and database-driven filtering. These tools supported human discovery by making chemical space legible: they enabled teams to map families, identify analogs, and track what was known. Importantly, cheminformatics introduced early forms of provenance and reproducibility. Even when the models were simple, the discipline of tracking structures, descriptors, and screening results created a foundation for auditability. Modern generative pipelines inherit this necessity, but with higher stakes: if the system can generate thousands of candidates, provenance tracking must scale with it.

Combinatorial chemistry and high-throughput screening represent another predecessor paradigm. Here the strategy was to generate and test large libraries of molecules systematically, often with heuristic constraints. The underlying logic was brute force: if you can synthesize and screen enough candidates, you increase your odds of finding hits. This paradigm introduced an organizational lesson that is directly relevant to generative modeling: *scale without selection discipline is waste*. High-throughput screening is expensive and prone to false positives; therefore, it requires strong experimental design, quality control, and careful interpretation. Those same requirements appear in

generative discovery, but shifted upstream: instead of screening libraries in the lab, organizations screen model-generated candidates with predictors and filters. The need for quality control does not disappear; it moves.

Heuristic screening and rule-based filters were the practical glue of older workflows. Medicinal chemistry developed rules of thumb (for example, constraints on size, polarity, functional groups) because experience taught that unconstrained exploration produces candidates that fail for predictable reasons. Materials science used domain-specific feasibility constraints based on thermodynamics, phase stability, and synthesis routes. These heuristics were not perfect, but they served as governance controls: they encoded institutional caution into the pipeline. A core mistake in naive modern deployments is to treat heuristics as obsolete because deep learning exists. In reality, heuristics are often the only robust guardrails in regimes where data is sparse and predictors are uncalibrated. Modern systems should treat them as first-line constraints, not as embarrassing relics.

It is also important to emphasize what preceding approaches did *not* do. They did not treat generation as a cheap and unlimited resource. Candidate generation was expensive because it required human creativity, complex synthesis planning, or physical library construction. Therefore, older pipelines were relatively conservative: they focused on incremental modifications in known families and relied on domain judgment to avoid unproductive regions. This conservatism was not merely cultural; it was an economic response to scarcity. Generative models invert that scarcity, and in doing so they remove a natural brake on reckless exploration. The lineage therefore offers a caution: when you remove the brakes, you must replace them with controls.

1.3.2 Key inflection points

The modern era of generative chemistry and materials design emerged through a series of inflection points that changed what was technically possible and what was operationally plausible. The first inflection was the rise of deep learning for molecular property prediction. Graph neural networks and related architectures made it possible to learn directly from molecular graphs rather than from hand-crafted descriptors. Transformers and attention-based models further expanded the representational toolkit, enabling large-scale pretraining and transfer. The practical effect was to improve predictive accuracy on many benchmarks and to make prediction pipelines more modular: a learned predictor could be treated as a reusable component.

This predictive shift had a second-order consequence: it enabled optimization against predictors. Once predictors became strong enough to be treated as scoring functions, teams began to search chemical space by maximizing predicted properties. This is where governance seeds were planted. The moment a predictor becomes a target of optimization, its error modes become operational risks. Many teams encountered the same pattern: optimization found candidates that scored well but failed in validation. That experience is one reason the field became increasingly attentive to uncertainty estimation and out-of-distribution detection. The inflection here is not merely technical;

it is a change in how models are used. Predictors stopped being passive evaluators and became active instruments in search.

The second inflection was the introduction of deep generative modeling for molecules. Early approaches used continuous latent representations and variational autoencoders to map discrete molecular structures into continuous spaces that could be searched and optimized. Work on continuous representations demonstrated that chemical design could be framed as optimization in latent space, and it influenced a generation of thinking about "molecule as data" [3]. Graph-based generative models followed, including structured approaches that explicitly modeled molecular substructures and assembly (for example, junction-tree-based methods) [4]. These methods improved validity and control, but they still faced limitations: difficulty capturing 3D geometry, challenges enforcing constraints, and limited robustness under aggressive optimization.

Flows, adversarial models, and other generative mechanisms expanded the menu. They offered different trade-offs between controllability, sample quality, and training stability. But the major conceptual turning point came when diffusion-style generative models entered scientific generation. Diffusion models reframed generation as an iterative denoising process, which in images produced impressive fidelity and controllable sampling. In molecular generation, diffusion approaches were adapted to structured data and, crucially, to 3D geometry. Equivariant diffusion models introduced a way to generate molecular structures while respecting physical symmetries, moving the field closer to producing candidates that are geometrically plausible rather than merely syntactically valid [1]. Related work on geometric diffusion for molecular conformations further reinforced the idea that 3D generation is not a luxury feature but a requirement for serious downstream use [2].

Parallel to these modeling advances, a third inflection occurred in benchmarking and evaluation infrastructure. Platforms such as GuacaMol and MOSES standardized datasets, metrics, and evaluation procedures for molecular generation [6, 7]. For the research community, benchmarks provided comparability. For institutions, they provided the first "dashboard" view of what models could do. But benchmarks also created a governance hazard: they tempted teams to equate benchmark improvements with practical readiness. The inflection, therefore, is double-edged. It increased visibility and accountability but also increased the risk of proxy-chasing: optimizing for benchmark metrics that do not capture synthesizability, safety, or deployability.

The fourth inflection is the integration of generative modeling with synthesis planning and feasibility estimation. Synthetic accessibility scores and retrosynthesis planning systems made it more practical to screen candidates for plausibility and to generate plausible synthetic routes [8, 9]. This matters because feasibility is one of the most common reasons generative candidates fail. By adding feasibility estimators into the loop, pipelines became more realistic: they could automatically discard candidates that were likely impractical. However, feasibility estimation also expanded the proxy stack. Now the pipeline might optimize not only predicted performance but also predicted synthesizability, creating layered proxies that can each be exploited. The net effect is that governance must cover not one proxy but an ensemble of proxies.

The fifth inflection is automation and the convergence of AI with lab execution. As robotic experimentation, automated synthesis, and high-throughput characterization became more common, the latency of experimental feedback loops decreased. Even partial automation changes the economics of search: the marginal cost of testing more candidates drops, and the system can iterate faster. This creates the possibility of active learning pipelines in which model suggestions drive experiments, experiments update predictors, and the cycle repeats. Operationally, this is the closest thing to a "discovery factory." Strategically, it is enticing. Governance-wise, it is an escalation: decisions propagate into physical action more quickly, and the opportunity for human review narrows unless explicitly preserved through gating.

Finally, a sixth inflection is the explicit recognition of dual-use concerns and the safety implications of generative discovery. The field has confronted, more directly than many other application areas, the possibility that AI-powered design tools can be repurposed for harmful ends [10]. For executives, the relevance is that safety is not only about accidental errors; it is also about misuse pathways. This recognition shifts governance from internal model risk management to broader operational controls: access permissions, monitoring, objective restrictions, and export controls on candidates and models. The inflection is cultural as much as technical: it expands the definition of risk.

Taken together, these inflection points produced the modern landscape: powerful generators, stronger predictors, better feasibility filters, standardized benchmarks, and increasing integration with automation. The headline capability is generation. The deeper change is that the system can now be wired into an end-to-end loop that resembles optimization under uncertainty, where small design choices in scoring and selection can dominate outcomes. That is why this chapter treats the field not as "AI for chemistry" but as "governed search with physical consequences."

1.3.3 What persisted vs what broke

The historical lineage makes it clear that some truths persisted, while some limiting assumptions broke. The most important persistent truth is that chemistry and materials are constrained by physical reality. No amount of modeling sophistication eliminates the need for feasibility, stability, and safety considerations. Models can propose candidates that appear plausible, but whether those candidates can be synthesized, whether they are stable under operating conditions, and whether they exhibit the desired properties are empirical questions. This is not cynicism; it is the essence of science. The persistent principle is therefore: *lab validation is not optional; it is the ground truth that governs credibility.*

A second persistent truth is that uncertainty dominates at the frontier. Regardless of how good predictors become, the most valuable candidates are often those that are novel relative to known data. Novelty is precisely where prediction is least reliable. Therefore, any discovery program must manage the trade-off between exploration and reliability. Older workflows managed this implicitly through human caution. Modern workflows must manage it explicitly through uncertainty

estimation, novelty bounds, and staged validation. The fact that uncertainty grows with novelty is not a temporary inconvenience; it is a structural feature of the problem.

A third persistent truth is that constraints and feasibility are first-class. Earlier pipelines used heuristic constraints because they reduced waste. Modern pipelines still need constraints, but now they must be formalized and operationalized in code and policy. The content of constraints may evolve, but the existence of constraints is not negotiable. The pipeline's success will depend less on whether the generator is state-of-the-art and more on whether constraints are correctly specified, enforced, and audited.

What broke is the assumption that candidate generation is the bottleneck. Historically, generating plausible candidates required significant human creativity and synthesis effort. That scarcity acted as a natural selection mechanism: you could not generate millions of candidates, so you had to choose carefully. Generative models break this scarcity. Now the pipeline can produce candidates at scale, and the marginal cost of generation approaches zero relative to validation. This inversion changes everything. It means that organizations can no longer rely on scarcity to enforce discipline. They must implement discipline intentionally.

Another broken assumption is that "more candidates" implies progress. In older paradigms, increasing the library size could increase hit rate, because the physical library represented real, testable options. In generative pipelines, increasing candidate volume does not necessarily increase progress; it can decrease it by overwhelming validation capacity and by increasing proxy exploitation. The system can produce enormous quantities of candidates that look good under predictors but fail in reality. The broken intuition is that scale alone yields value. In a proxy-driven system, scale can amplify error faster than it amplifies signal.

A third broken assumption is that evaluation can be treated as a static, offline step. Earlier workflows often used predictors as a one-time screen: score candidates, pick some, test them. In modern pipelines, evaluation becomes dynamic because the system is iterating and optimizing. Predictors are no longer just estimators; they are target functions in search. This means evaluation must be treated as a controlled process: calibration must be monitored, stress tests must be performed, and the relationship between predicted and validated outcomes must be continuously tracked. Evaluation is no longer a prelude; it is a central governance domain.

A fourth broken assumption is that human intuition provides sufficient guardrails. Human judgment remains essential, but it cannot scale to match the generator's output. When candidate volume grows, humans must rely on filters and rankings to triage. Those filters and rankings embed choices that can amplify bias, hide uncertainty, and create decision laundering. Therefore, the pipeline must be designed so that human judgment is applied at the right choke points with the right evidence. Human intuition cannot simply be "in the loop" as a vague slogan; it must be structurally integrated as an approval gate.

Finally, a fifth broken assumption is that metrics of "validity" and "novelty" are adequate proxies

for scientific progress. In older paradigms, a candidate was meaningful because it was physically constructed or at least physically plausible within a known family. In modern generative benchmarks, validity often means the encoding is well-formed, and novelty often means distance from training data. Those metrics can be satisfied by candidates that are useless or unsafe. The break is that the field's measurement culture has, in some contexts, drifted toward proxies that are convenient rather than decisive. Governance must correct that drift by defining acceptance criteria tied to feasibility, safety, and validated performance, not merely to representational properties.

In summary: constraints persisted, uncertainty persisted, validation persisted. What broke was the scarcity-based discipline that forced careful selection. Generative models remove scarcity from generation, and therefore governance must replace scarcity with explicit controls. This is why the chapter's central thesis is not about architecture; it is about institutional design.

1.3.4 Why older intuitions fail

Older intuitions fail primarily because they were formed under a different scarcity regime. When candidate generation is expensive, each candidate carries a high implicit cost, and human attention naturally focuses on a small number of options. Under these conditions, "more candidates" often meant broader exploration of physically realizable libraries, and it could be valuable. But when candidate generation is effectively free, "more candidates" means more draws from a distribution shaped by training data and by conditioning objectives. The marginal candidate does not necessarily add information; it may add noise, and in proxy-optimized settings it disproportionately adds error.

One of the most dangerous older intuitions is that if a model can generate valid representations, it is therefore producing meaningful candidates. Validity is necessary but it is not an achievement; it is the cost of entry. In chemical terms, a valid string or graph is not evidence of synthesizability, stability, or safety. In materials terms, a plausible composition is not evidence of phase stability or manufacturability. When teams celebrate validity, they often signal that they are still in the demonstration phase. A production-phase mindset starts from a harsher premise: assume that most valid candidates are not worth testing until filters, feasibility checks, and uncertainty gates say otherwise.

A second failing intuition is that higher predicted scores correspond to better candidates. This intuition holds only when predictors are well-calibrated and when candidates lie within the predictor's reliable domain. Under optimization pressure, those conditions rarely hold. The best predicted candidates are often the ones where the predictor is most wrong. As sampling increases, the extreme predicted values are increasingly driven by error. Therefore, a pipeline that simply takes the top k predicted candidates is structurally fragile. Older screening workflows sometimes did this with human oversight and limited sample sizes. Generative pipelines do it at scale and at speed, magnifying the failure.

A third failing intuition is that novelty is intrinsically valuable. In many discovery contexts, novelty

is a means, not an end. Novelty can indicate exploration of new mechanisms or materials families, but it can also indicate departure from feasibility constraints. In generative benchmarks, novelty is often rewarded without penalty. In real discovery, novelty must be purchased with validation budget. Therefore, a mature pipeline treats novelty as a controlled variable: it allocates a small portion of validation capacity to high-novelty exploration, while reserving most capacity for candidates that are innovative but still anchored in feasible, interpretable regimes. Older intuitions often came from a culture of incremental improvement where novelty was pursued cautiously. Generative systems can produce novelty cheaply, and without governance they can produce it irresponsibly.

A fourth failing intuition is that integration with automation automatically improves productivity. Automation shortens loops, but it also shortens the window for reflection and review. If a system can generate candidates, score them, and send them to a robotic synthesis platform quickly, the organization may interpret speed as progress. But speed can amplify mis-specification. If the objective is wrong, the system will efficiently pursue the wrong thing. If hazard filters are incomplete, the system will efficiently generate hazards. Automation is not merely an accelerator; it is a risk multiplier. Older intuitions treated lab work as inherently slow and therefore self-limiting. Automation removes that self-limitation and demands explicit stop conditions and approval gates.

A fifth failing intuition is that scientific pipelines are naturally self-correcting. It is true that experiments provide feedback. But experiments are expensive, and organizations often test only a small subset of candidates. If the selection policy is biased by proxy exploitation, the tested subset can be systematically misleading. The organization can spend months testing candidates that were selected because of predictor error, and then interpret the failures as "bad luck" rather than as a selection failure. This is a governance problem: the pipeline must monitor the relationship between predicted and validated outcomes and must treat widening gaps as a control failure requiring intervention. Without that monitoring, the organization can remain trapped in a loop of expensive disappointment.

These failing intuitions converge on a single theme: older workflows relied on natural constraints (cost, time, human bandwidth) to enforce discipline. Generative and automated workflows remove those natural constraints. Therefore, discipline must be engineered. Governance-free generation amplifies unsafe exploration not because models are malicious, but because the system's incentives and mechanics reward proxy performance over real evidence. The organization must therefore redesign its intuitions: treat generation as cheap and abundant, treat evaluation as uncertain and gameable, treat validation as scarce and decisive, and treat governance as the mechanism that preserves strategic value.

1.3.5 Inherited lessons

Despite the novelty of diffusion models and foundation architectures, the field inherits a set of lessons from earlier eras of optimization, screening, and safety. The first inherited lesson is that constraints

dominate outcomes. In any multi-objective discovery problem, the feasible set is often small relative to the total space. It is not enough to "optimize performance"; one must optimize within the feasible region defined by synthesizability, stability, hazard exclusions, cost, and regulatory constraints. Organizations that learn this lesson early build constraint registries and enforce them as hard gates. Organizations that learn it late discover that their pipelines generate impressive candidates that cannot be acted upon.

The second inherited lesson is that conservative validation gates are essential when outputs have physical consequences. In software, a bug can be patched; in chemistry, a failure can injure people or waste months of work. Therefore, validation must be staged. Cheap screens (validity checks, rule-based filters, similarity constraints) should eliminate the obvious failures. More expensive computational checks (retrosynthesis planning, higher-fidelity simulation) should be reserved for candidates that survive early gates. Experimental validation should be reserved for candidates that meet explicit evidence thresholds. This staged funnel is not bureaucracy; it is the only way to scale responsibly.

The third inherited lesson is that proxies must be treated as adversarial under optimization pressure. Optimization literature teaches that when a system optimizes a surrogate, it will tend to exploit the surrogate's weaknesses. This is not unique to AI; it is a general property of objective functions. In generative discovery, property predictors and feasibility scores are surrogates. Therefore, pipelines must use uncertainty penalties, model ensembles, stress tests, and monitoring to prevent proxy collapse. The lesson is: assume the proxy will be gamed, and design controls accordingly.

The fourth inherited lesson is that measurement must be aligned with real outcomes. Screening and QSAR pipelines taught that metrics can mislead. A model can perform well on a dataset and still fail in real-world deployment because the dataset does not represent the deployment regime. Similarly, a generator can perform well on a benchmark and still fail to produce deployable candidates. Therefore, evaluation must include end-to-end measures: the fraction of candidates that pass feasibility filters, the calibration of predicted properties, the relationship between predicted and validated outcomes, and the cost per validated improvement. Without end-to-end metrics, teams can optimize for internal dashboards rather than for scientific progress.

The fifth inherited lesson is that safety is a process, not a label. Chemical safety culture learned that hazards must be anticipated, documented, and controlled through layered defenses: exclusion rules, training, procedures, monitoring, and incident response. In generative discovery, safety extends upstream: it includes preventing the system from proposing hazardous candidates, controlling access to models and objectives, monitoring usage patterns, and ensuring that candidates cannot be escalated without human approval. The lesson is that safety cannot be retrofitted after deployment; it must be designed into the pipeline.

The sixth inherited lesson is that reproducibility is a governance necessity. Science depends on repeatability, but generative pipelines introduce additional sources of nondeterminism: sampling

randomness, model version drift, dataset updates, and changing constraint sets. Therefore, reproducibility must be enforced through versioning, fixed seeds where appropriate, immutable logs, and artifact bundles that capture the full context of each run. This is not merely good scientific practice; it is essential for accountability. If an organization cannot reproduce how a candidate was generated and selected, it cannot defend the decision to test it.

The final inherited lesson is organizational: role separation matters. In many safety-critical domains, those who propose actions are not the same as those who approve them. The same principle should apply here. The team building the generator and optimizing objectives should not be solely responsible for approving candidates for lab validation. Approval should be a governed process that includes domain experts and safety officers, with documented criteria and traceability. This role separation reduces incentive distortion and makes it harder for proxy success to masquerade as evidence.

These lessons, taken together, form a consistent governance posture: treat generative discovery as constrained optimization under uncertainty with physical-world consequences. The historical lineage shows that the domain already knows how to operate under such conditions. The challenge is that modern generative systems remove natural scarcity constraints and introduce new proxy-driven failure modes at scale. The solution is not to abandon the technology; it is to import the domain's hard-earned discipline into the design of the AI pipeline, and to treat governance not as an afterthought but as the mechanism that turns frontier capability into defensible advantage.

newpage

1.4 Technical Foundations

1.4.1 System or architectural components

A generative chemistry or materials design system is best understood as an architecture of interacting components rather than as a single model. This is not merely a software engineering preference; it is a governance necessity. When the pipeline is modular, controls can be placed at boundaries, evidence can be logged at each stage, and failures can be isolated. When the pipeline is monolithic, decisions become hard to reconstruct and error modes become difficult to attribute. The technical foundations, therefore, begin with an architectural stance: separate proposing, scoring, filtering, and selecting into explicit modules with explicit interfaces.

The first component is the *generator backbone*. This is the model responsible for producing candidate structures. In modern systems, the backbone is commonly a diffusion-like generator or an autoregressive transformer operating over an appropriate representation space. The generator's internal details matter for quality and controllability, but the governance-relevant question is what the generator is permitted to output. The generator must be bounded by representation

constraints (what constitutes a well-formed object), by domain constraints (allowed atoms/elements, charge constraints, composition bounds), and by operational constraints (what objectives may be conditioned upon). In practice, organizations should treat the generator as a *proposal engine* and should avoid granting it any authority to decide what is "best" without downstream gates.

The second component is the *conditioning interface*. Conditioning is the mechanism by which the generator is guided toward candidates aligned with desired properties or constraints. Conditioning can take many forms: numeric property targets, vector embeddings of desired attributes, textual prompts describing intended use, or combinations thereof. The key point is that conditioning is not merely a convenience feature. It is the formal encoding of organizational intent. If the conditioning signal is underspecified, the generator is free to trade off unspoken constraints. If the conditioning signal includes prohibited objectives (for example, if it implicitly encourages hazardous candidates), the system becomes a misuse risk. Therefore, conditioning inputs should be validated against an allowed schema, logged, and subject to access controls. In a governed deployment, the conditioning interface is a controlled input surface, not an open-ended prompt field.

The third component is the *representation and canonicalization layer*. Generators often output candidates in a representation that must be normalized before further processing. Canonicalization ensures that equivalent structures are represented consistently, duplicates are detected, and downstream filters operate reliably. This is a technical requirement with governance implications: if canonicalization is inconsistent, the pipeline can double-count candidates, misapply constraints, or produce misleading audit records. Canonicalization also supports provenance: it enables the system to track whether the same candidate was generated across runs, under what conditioning, and with what scores.

The fourth component is the *validity filter*. Validity filters ensure that generated candidates satisfy basic representational and domain constraints. In molecular settings, this includes syntactic validity of encodings, chemically plausible valences, and exclusion of impossible bonding patterns. In materials settings, this includes compositional plausibility and structural constraints. Validity is a necessary first gate, but it should be treated as the lowest bar. Passing validity does not confer meaningful credibility; it merely allows the candidate to enter the evaluation funnel.

The fifth component is the *property prediction stack*. This includes learned predictors and, where feasible, simulators. Predictors estimate properties relevant to the objectives: performance metrics, stability proxies, toxicity proxies, reaction yields, conductivity, and so on. Simulators may include physics-based methods (e.g., energy calculations, approximate dynamics) or domain-specific models. The practical point is that predictors are cheap and approximate; simulators are more expensive and often more reliable, but still imperfect. In a governed system, predictors and simulators are treated as layers in a staged funnel. Candidates should not be escalated to expensive validation based solely on cheap predictors when uncertainty is high. Instead, predictor outputs should be accompanied by uncertainty estimates and used to decide whether a candidate deserves more expensive simulation.

The sixth component is the *novelty and diversity module*. This module measures how far candidates lie from known data and how diverse the selected set is. Novelty is valuable because it expands exploration, but it is also correlated with uncertainty and risk. Diversity is valuable because it reduces the chance of selection collapse onto a narrow family and improves the information gained from validation. These modules therefore act as governance controls: they prevent the system from optimizing a single proxy into a brittle corner and help ensure that scarce validation capacity is allocated across a range of plausible hypotheses.

The seventh component is the *feasibility and synthesizability filter*. For molecules, this may include synthetic accessibility heuristics and retrosynthesis planning systems [8, 9]. For materials, it may include phase stability proxies, manufacturability heuristics, or process constraints. The feasibility filter is a critical gate because it addresses one of the dominant sources of waste: candidates that appear promising but cannot be made. However, feasibility filters are themselves predictors and therefore must be treated as proxies. Their outputs should be logged, their calibration monitored, and their failure modes explicitly acknowledged. The purpose of the filter is not to certify feasibility, but to reduce the proportion of obviously infeasible candidates that reach expensive stages.

The eighth component is the *safety and hazard screening module*. This module enforces exclusions and flags candidates that may be hazardous. In a governed pipeline, hazard screening is not optional. It is a hard constraint layer that should operate early and should be conservative with respect to false negatives. The design challenge is to define hazard classes and screening rules in a way that is enforceable and auditable. The governance challenge is to ensure that hazard screening cannot be bypassed by users or by downstream tooling. This requires access control, logging, and separation of responsibilities.

The ninth component is the *selection and escalation policy*. This component takes the scored and filtered candidates and decides what happens next. The policy encodes the organization's risk appetite and resource constraints. It might specify, for example, that only candidates with uncertainty below a threshold and feasibility above a threshold can be escalated to simulation, and that only candidates with multiple independent predictors in agreement can be proposed for lab testing. It might also enforce quotas, such as allocating a fixed fraction of validation budget to high-novelty exploration. The selection policy is where governance becomes operational. It must be explicit, versioned, and reviewable.

The final component is the *logging and artifact layer*. Every stage of the pipeline must emit structured logs that capture conditioning inputs, model versions, candidate representations, scores, uncertainties, filter decisions, and escalation outcomes. This is not mere documentation. It is the infrastructure that enables auditability and incident reconstruction. Without it, the organization cannot answer basic questions such as: why was this candidate selected, what evidence supported the decision, and what controls were in effect. In a physical-world domain, the inability to answer those questions is itself a serious risk.

In short, the architecture is a funnel: propose at scale, filter aggressively, score probabilistically, select conservatively, and escalate only with explicit approvals. The generator is a powerful component, but the pipeline's safety and value are dominated by the control modules that surround it.

1.4.2 Information flow

The information flow in a generative discovery pipeline is best described as a sequence of transformations and decisions, each of which must be auditable. The flow begins with a *conditioning specification* c. This specification encodes the intended design goal: property targets, constraints, and contextual metadata such as domain, project, and permissible exploration bounds. The conditioning specification should be treated as a governed artifact. It should have a schema, an owner, and a version. It should not be an informal text field whose meaning changes across runs.

Given c, the generator produces candidate structures $x_1, x_2, \ldots, x_N$ by sampling from $p_\theta(x \mid c)$. The sampling process itself has parameters: number of samples, temperature or guidance strength, diversity settings, and random seeds. These parameters matter because they affect the distribution of generated candidates and therefore the risk profile. If the system increases sampling volume, it increases the probability of encountering proxy-exploiting outliers and hazard outliers. Therefore, sampling parameters should be logged and, in mature deployments, subject to quotas or escalation approvals.

Once candidates are generated, they pass through *canonicalization*. Canonicalization maps each candidate into a normalized representation and assigns an identifier. This step enables deduplication and traceability. The system should detect duplicates both within the current batch and across historical batches. Duplicate detection is not trivial housekeeping; it prevents the pipeline from wasting evaluation resources and prevents inflated metrics that make the system appear more productive than it is.

Next, candidates pass through a *validity gate*. Invalid candidates are discarded, and the reasons for invalidation are recorded. The distribution of invalidation reasons is itself a monitoring signal: if the generator begins to produce more invalid outputs, it may indicate drift, misconditioning, or a change in representation assumptions. Validity filters can also be staged: a cheap syntactic validity check first, then a deeper plausibility check second. The output of this gate is a set of candidates that are representationally admissible.

Admissible candidates then enter the *evaluation stack*. The evaluation stack produces a structured score record for each candidate: predicted properties, uncertainty estimates, similarity metrics, novelty metrics, feasibility estimates, and hazard flags. The key governance principle is that evaluation should not collapse into a single scalar too early. If the system converts all evidence into one score, it hides the trade-offs and prevents auditors from understanding why a candidate was selected. Instead, evidence should remain decomposed until the selection policy applies explicit rules.

After evaluation, candidates pass through *filtering*. Filtering applies hard constraints: hazard exclusions, forbidden elements, maximum cost constraints, minimum feasibility thresholds, and any other non-negotiable rules. Filtering should be deterministic given the candidate and the constraint registry. Determinism is important for reproducibility: if filtering depends on stochastic or opaque processes, the pipeline becomes difficult to audit. Candidates that fail constraints are discarded, and the reasons are logged. The distribution of constraint failures is again a monitoring signal: if many candidates fail a particular constraint, it may indicate misconditioning, an overly permissive generator, or misaligned objectives.

The remaining candidates then enter *ranking and selection*. Ranking is optional and should be treated carefully. If ranking is used, it should be accompanied by uncertainty-aware methods (for example, sorting by conservative lower confidence bounds rather than by point estimates). Selection applies quotas and diversity controls, ensuring that the final set of candidates is not merely the top scorers but a portfolio of plausible options. This is the point at which the system produces its main operational outputs: (i) a candidate set for further computational evaluation, (ii) a candidate set recommended for expert review, and (iii) a candidate set proposed for experimental validation, each with evidence records.

Finally, the pipeline may *iterate*. Iteration can take multiple forms. It can mean resampling from the generator with updated conditioning based on what was learned. It can mean fine-tuning predictors based on new data. It can mean narrowing or widening exploration bounds. Iteration is powerful because it turns a one-shot generator into a search process. It is also dangerous because iteration applies optimization pressure that can exploit proxies. Therefore, iteration should be controlled: the system should have stop conditions, monitoring triggers, and explicit approvals for major changes.

Throughout this flow, there is a crucial meta-flow: *logging and provenance*. Every stage should emit artifacts that allow reconstruction. The log should include: conditioning c, generator version θ, sampling parameters, canonicalization identifiers, validity outcomes, evaluation outputs, filtering outcomes, selection decisions, and iteration updates. In a governed pipeline, the log is not merely a record. It is the mechanism by which the organization proves that decisions were constrained and evidence-based. Without it, the pipeline may still produce discoveries, but it cannot be defended.

1.4.3 Interaction or control loops

Generative discovery becomes strategically meaningful when it moves from one-shot generation to iterative loops. These loops can improve performance and efficiency, but they also create new modes of instability. The technical foundations must therefore describe not only the static architecture but also the dynamic control loops that govern behavior over time.

The first loop is the *model-internal iterative sampling loop*. Diffusion models, by design, generate samples through iterative denoising. Autoregressive models generate sequences step-by-step. These internal loops matter because they create control levers: guidance strength, step counts, sampling

temperature, and constraint enforcement during sampling. From a governance perspective, these levers must be constrained. For example, increasing guidance strength can improve adherence to objectives but can reduce diversity and increase mode collapse. Increasing sampling steps can improve quality but increases compute cost. These are not merely technical choices; they shape the distribution of candidates and therefore the risk profile. Organizations should log these settings and define acceptable ranges.

The second loop is the *pipeline iteration loop*: propose → evaluate → select → refine → propose again. This loop is where the system acts like an optimizer. If refinement uses the best predicted candidates to adjust conditioning or to train the generator, the loop can quickly converge to a narrow region of the space. This can be desirable if the region is well-understood and feasible, but it can also represent proxy collapse. Therefore, refinement strategies should incorporate diversity and uncertainty considerations. A conservative refinement might, for example, include both high-scoring candidates and candidates that are informative for reducing uncertainty. In other words, the loop should behave like a disciplined experimental design process rather than like a greedy score maximization routine.

The third loop is the *active learning loop*. In active learning, the system uses uncertainty to choose which candidates to validate next, with the goal of improving predictors. Active learning is attractive because it offers a principled way to spend scarce validation budget: test candidates that will reduce uncertainty and improve future selection. However, active learning also introduces governance requirements. The system must clearly distinguish between exploration experiments (run to learn) and exploitation experiments (run to achieve performance). It must avoid confusing uncertainty reduction with progress toward deployable outcomes. It must also manage feedback drift: if the system learns only from the candidates it selects, it can create selection bias that distorts the predictor. A governed active learning program therefore needs holdout sets, periodic recalibration, and monitoring of prediction error across chemical families.

The fourth loop is the *lab feedback loop*. When experimental results are available, they can update predictors and sometimes the generator. This feedback loop is where the system touches the physical world. It is also where reproducibility and governance are most likely to fail. If models are updated continuously with new lab data, it becomes difficult to reproduce past decisions: the same conditioning might produce different candidates tomorrow because the model changed today. Therefore, model updates should follow a controlled cadence with versioning, change logs, and validation. The organization should be able to answer: what model version was used for this decision, what data was included, and what performance checks were passed before deployment. In regulated environments, this resembles model risk management for financial models: changes are governed, not improvised.

The fifth loop is the *human-in-the-loop approval loop*. Human review is not a single decision; it is often a sequence of reviews at different stages. For example, a computational chemistry team may review candidates before high-fidelity simulation; a synthesis team may review candidates before

committing to synthesis; a safety team may review hazard flags; and a project lead may approve the final experimental plan. This multi-stage human loop is a governance design choice. It defines where accountability sits. A mature pipeline will explicitly encode these review stages and will produce evidence packets tailored to each reviewer's role. The key is that human review must have authority to stop and must be provided with sufficient evidence to make a meaningful decision. If humans are included only as a ceremonial step, the pipeline is effectively autonomous, regardless of what the organization claims.

The sixth loop is the *control and monitoring loop*. A governed system continuously monitors its own outputs and triggers interventions when risk indicators appear. Relevant indicators include: spikes in invalid candidate rates, increased hazard flag rates, widening gaps between predicted and validated performance, increased novelty beyond defined bounds, and reduced diversity in selected candidates. Monitoring should not be reactive theater; it should connect to control actions: adjust sampling volume, tighten constraints, require additional human review, or pause deployment. In other words, the pipeline should behave like a controlled industrial process, not like an unconstrained research script.

These loops interact. For example, aggressive pipeline iteration can increase proxy exploitation, which can then corrupt lab feedback if the lab tests only proxy-exploiting candidates. That feedback can then distort predictors, which then makes selection worse. This is a runaway failure mode. Conversely, conservative active learning can improve predictor calibration, which can improve selection, which can improve lab outcomes, which can improve trust. The difference is governance: the definition of constraints, the design of selection policies, and the discipline of update management.

The central technical foundation is therefore not any one loop, but the principle that loops must be bounded by stop conditions and evidence gates. In a physical-world domain, a loop that runs unchecked is not an innovation engine; it is a risk engine.

1.4.4 Assumptions and constraints

Every generative discovery pipeline rests on assumptions. In a governance-first treatment, these assumptions must be explicit because they define where the system is reliable and where it is speculative. The most fundamental assumption is that the scoring functions approximate reality sufficiently to support triage. Property predictors and feasibility estimators are surrogates. They are trained on finite datasets that reflect past experiments, past strategies, and past biases. They therefore encode both knowledge and ignorance. The pipeline assumes that the surrogate is informative in the region being explored. When exploration moves beyond that region, the surrogate's outputs become increasingly unreliable.

A second assumption is that uncertainty can be estimated and used. Many pipelines implicitly treat predictor outputs as point estimates. That is rarely defensible. A governed pipeline assumes that it can attach uncertainty estimates to predictions, either through model ensembles, Bayesian

approximations, dropout-based methods, or other techniques. The precise method is less important than the operational behavior: the system must know when it does not know. Uncertainty is not an academic embellishment; it is the mechanism that prevents the pipeline from treating extrapolation as fact.

A third assumption is that the data coverage is sufficient to support the objectives. If the organization's target properties are rare or if the desired region of chemical space is underrepresented in training data, the generator and predictors may behave unpredictably. The pipeline must assume some degree of representational alignment: that the training data includes enough examples of similar structures to guide generation and evaluation. This assumption is often violated in practice, especially when organizations pursue novel mechanisms. Therefore, a mature program includes a governance step that assesses data coverage and defines exploration bounds accordingly.

A fourth assumption is that constraints can be operationalized. Many constraints are easy to state in natural language but hard to encode: "avoid reactive intermediates," "ensure manufacturability," "prefer scalable routes." The pipeline assumes that constraints can be translated into enforceable rules or predictors. Where constraints cannot be operationalized, the pipeline must compensate by tightening scope and increasing human review. A common governance failure is to list constraints in a slide deck and then fail to enforce them in the pipeline. A governed program treats unenforceable constraints as risk events, not as aspirations.

A fifth assumption is that the representation captures what matters. If the generator operates on a representation that omits key physical information, it may produce candidates that look plausible but fail in reality. For example, ignoring stereochemistry or conformational flexibility can mislead property prediction. In materials, ignoring microstructure or processing conditions can produce candidates that are theoretically interesting but practically irrelevant. The pipeline assumes that representation is adequate for the stage of discovery. This is a bounded assumption: early-stage ideation can tolerate coarse representations, but later-stage selection cannot. Therefore, the pipeline should have representation escalation: as candidates move closer to validation, richer representations and higher-fidelity simulations are required.

A sixth assumption concerns independence and error structure. Many pipelines assume that errors are roughly independent across candidates or that averaging over many candidates will wash out noise. Under optimization pressure, this assumption fails. The system selects candidates that share the same error modes: they are the ones that exploit predictor weaknesses. Therefore, the pipeline must assume that selection induces correlated error and must actively counteract it through diversity and adversarial stress tests.

A seventh assumption is governance feasibility itself. The pipeline assumes that the organization can maintain versioning, logging, access controls, and review processes. In reality, these are operational burdens. If the organization cannot sustain them, the system will drift toward informal use, and controls will erode. A realistic program therefore constrains deployment to what governance capacity

can support. This is not pessimism; it is risk management. The pipeline should not be allowed to outrun the institution's ability to supervise it.

Constraints, in this context, are the mechanism by which assumptions are bounded. Constraints define what regions of the space the system is allowed to explore, what candidates are disallowed regardless of predicted performance, and what evidence thresholds are required for escalation. The technical foundation is therefore the explicit linking of assumptions to constraints: when an assumption weakens (for example, coverage is poor), constraints tighten (exploration narrows, uncertainty thresholds increase, human review expands). This linkage is what converts abstract model limitations into operational safety.

1.4.5 Technical bottlenecks

The most important technical bottleneck in modern generative discovery is not candidate generation. It is *reliable evaluation under limited data and under optimization pressure.* This bottleneck expresses itself in several specific challenges, each of which has direct governance implications.

The first bottleneck is *uncertainty calibration.* It is not enough to produce an uncertainty number; the uncertainty must be meaningful. Calibration means that when the model says it is uncertain, it is reliably wrong more often, and when it says it is confident, it is reliably wrong less often. In practice, calibration is difficult in scientific domains because labels are noisy, datasets are biased, and the deployment regime differs from the training regime. Under optimization pressure, calibration can degrade further because the system selects unusual candidates. Without calibrated uncertainty, uncertainty thresholds are meaningless and gates become performative. Therefore, calibration is a bottleneck because it is the prerequisite for disciplined selection.

The second bottleneck is *out-of-distribution detection.* The pipeline must detect when a candidate lies outside the domain where predictors are reliable. This can be approached through similarity metrics, density estimation in representation space, ensemble disagreement, or specialized OOD detectors. But OOD detection is hard because chemical space and materials space are complex and because "novel but valid" is exactly what discovery seeks. The system cannot simply reject everything novel. It must instead distinguish between acceptable novelty and dangerous extrapolation. This requires nuanced controls: novelty caps, staged escalation, and increased human review for candidates that are both high-scoring and highly novel. Implementing these controls robustly is a technical bottleneck.

The third bottleneck is *feasibility and synthesizability estimation.* While heuristics and retrosynthesis planners exist [8, 9], feasibility remains one of the most common failure points. The difficulty is that feasibility depends on context: available reagents, organizational capabilities, process constraints, and cost structures. A candidate that is synthesizable in one lab may be infeasible in another. Therefore, feasibility estimation must be contextualized. This means feasibility modules must accept organizational constraints as inputs, and feasibility decisions must be logged with those constraints.

Achieving this level of contextual feasibility is difficult, making it a bottleneck.

The fourth bottleneck is *hazard screening with low false negatives*. Hazard detection is hard because hazards can arise from subtle structural features, and because the set of hazardous possibilities is broad. Screening systems must be conservative, but excessive false positives can paralyze exploration. Achieving an acceptable balance is a bottleneck, especially in early deployments where hazard rules and models are immature. This bottleneck has governance consequences: if hazard screening is weak, deployment should be deferred or scope should be restricted. If hazard screening is overly aggressive, the organization may be tempted to bypass it. Therefore, hazard screening must be both robust and operationally acceptable.

The fifth bottleneck is *evaluation under limited validation budget*. In many programs, only a small fraction of candidates can be tested. This creates a statistical problem: it becomes difficult to estimate the true performance of the pipeline and to detect drift. If the tested set is biased, the organization may misjudge pipeline quality. Therefore, the pipeline must allocate some validation budget to monitoring: testing candidates that provide information about calibration and drift, not just candidates that appear most promising. Designing this monitoring allocation is technically and organizationally challenging.

The sixth bottleneck is *integration and reproducibility*. Even if the models work, integrating them into a pipeline with stable interfaces, deterministic canonicalization, and robust logging is nontrivial. Scientific code often evolves rapidly; model versions change; datasets update; and pipelines drift. Without disciplined engineering, reproducibility collapses. This is a technical bottleneck because it requires infrastructure, not just modeling skill. It is also a governance bottleneck because without reproducibility, the organization cannot defend decisions.

The seventh bottleneck is *benchmark mismatch*. Benchmarks are useful, but they often fail to capture the constraints and costs that matter in real discovery. A model can score well on validity and novelty metrics while failing to produce candidates that survive feasibility and hazard filters. Therefore, teams must build internal evaluation regimes that reflect their constraints. Creating those regimes requires curated datasets, surrogate validation tasks, and realistic funnel simulations. This is expensive and time-consuming, making it a bottleneck that many teams underestimate.

These bottlenecks collectively explain why the field's frontier challenge is not "better generation." It is "better control." Control means calibrated uncertainty, reliable OOD detection, enforceable feasibility and hazard constraints, and reproducible, auditable integration. Organizations that treat these as secondary will discover that they can generate impressive candidates but cannot translate them into validated outcomes. Organizations that treat these as primary will build pipelines that may look less flashy in demos but produce defensible progress in reality.

In a governance-first view, technical bottlenecks are not merely engineering problems; they are decision gates. If calibration is inadequate, tighten uncertainty thresholds and restrict exploration. If OOD detection is weak, cap novelty and require human review. If feasibility estimation is unreliable,

escalate only candidates with clear synthetic routes and strong expert endorsement. If reproducibility is weak, pause deployment until logging and versioning are stabilized. The technical foundations, therefore, are inseparable from governance: they define what the system can responsibly do today, and they define what must be solved before the system can do more tomorrow.

1.5 Mathematical Foundations

1.5.1 Formal problem framing

A governance-first mathematical framing of generative chemistry and materials design must do two things at once. It must be formal enough to clarify what the pipeline is actually doing, and it must be bounded enough to remain usable by senior practitioners who need the conceptual levers rather than the full research literature. The central move is to treat "generative discovery" not as magic creativity, but as constrained optimization under uncertainty in a structured domain. Once this move is made, the rest follows: proxy failure becomes an expected phenomenon, uncertainty becomes a control variable, and governance becomes the institutional mechanism that constrains the optimization problem.

Let $\mathcal{X}$ denote the candidate design space. An element $x \in \mathcal{X}$ represents a candidate molecule, material, or structure. The space is structured: it is not $\mathbb{R}^d$ in any naive sense, even if models embed it into $\mathbb{R}^d$. It may be a set of strings with grammatical constraints, graphs with chemistry-consistent valences, or 3D configurations with symmetry and invariance properties. The organization seeks candidates with desirable properties. Let $f : \mathcal{X} \to \mathbb{R}^m$ be a vector-valued property function mapping x to m properties $f_1(x), \ldots, f_m(x)$ (e.g., predicted activity, stability, conductivity, selectivity). In the physical world, $f(x)$ is unknown prior to validation; it is defined operationally by experiments or high-fidelity simulation. Let $u : \mathbb{R}^m \to \mathbb{R}$ be a utility function aggregating multiple properties into a scalar value that reflects the organization's priorities and trade-offs.

The organization also has constraints. Let $g : \mathcal{X} \to \mathbb{R}^k$ be a vector of constraint functions $g_1(x), \ldots, g_k(x)$, where the feasible set is

$$\mathcal{F} \;=\; \{x \in \mathcal{X} \;:\; g_i(x) \leq 0 \text{ for } i = 1, \ldots, k\}.$$

Constraints include feasibility (synthesizability/manufacturability), safety exclusions, cost bounds, and provenance requirements. Some constraints are hard and must be enforced before escalation; others are soft and act as penalties.

A first-pass formulation of the design task is then the constrained optimization problem

$$\max_{x \in \mathcal{F}} \; u(f(x))\,. \tag{1.1}$$

This equation is the "ideal" objective, but it is not computable in practice, because $f(x)$ is not accessible at scale. Instead, the pipeline operates with learned or simulated surrogates. Let $\hat{f}(x)$ denote a predictor or a bundle of predictors providing an estimate of $f(x)$, and let $\hat{\sigma}(x)$ denote an uncertainty estimate. Similarly, let $\hat{g}(x)$ denote proxy constraints (e.g., predicted toxicity, predicted

synthetic accessibility). The operative problem is therefore a *surrogate-constrained* optimization:

$$\max_{x \in \hat{\mathcal{F}}} \ \hat{u}\Big(\hat{f}(x), \hat{\sigma}(x)\Big), \quad \hat{\mathcal{F}} \ = \ \{x \in \mathcal{X} : \hat{g}_i(x) \leq 0\}. \tag{1.2}$$

The choice of $\hat{u}$ is a governance decision. A naive organization might set $\hat{u}(\hat{f}, \hat{\sigma}) = u(\hat{f})$, effectively ignoring uncertainty. A governance-first organization introduces risk-sensitive terms that penalize uncertainty and proxy fragility, for example by optimizing a conservative estimate such as a lower confidence bound:

$$\hat{u}\Big(\hat{f}(x), \hat{\sigma}(x)\Big) \ = \ u\Big(\hat{f}(x) - \lambda \hat{\sigma}(x)\Big),$$

with $\lambda > 0$ encoding risk aversion and validation scarcity. This is not a purely technical tuning parameter; it encodes how expensive it is for the institution to be wrong. A high-λ policy emphasizes candidates that are not only promising but also within the model's reliable regime. A low-λ policy emphasizes aggressive exploration. Both can be rational, but they must be chosen explicitly and defended.

The unit discipline matters even in a high-level exposition. The property vector $f(x)$ may include dimensioned quantities (e.g., energy in eV, solubility in mol/L, conductivity in S/m). The utility u is therefore typically defined on normalized or transformed properties. In practice, pipelines introduce scaled properties $z_j(x)$ defined by

$$z_j(x) \ = \ \frac{f_j(x) - \mu_j}{s_j},$$

where μ_j and s_j are domain-specific reference means and scales. This normalization is not innocent. It defines relative priorities and influences optimization. A governance-first posture requires that normalization choices be documented, because they shape what "high score" means.

Finally, because validation is scarce, the optimization is not truly over all $x \in \mathcal{F}$. It is over a sequence of batches. Let $B_t \subset \mathcal{X}$ be the set of candidates proposed at iteration t, and let $V_t \subset B_t$ be the subset selected for validation. The pipeline is not solving (1.1) directly; it is solving a sequential decision problem: propose candidates, select a small subset, observe outcomes, update models, and repeat. This sequentiality is where many governance failures hide, because the selection policy shapes what data is observed and therefore what the models learn. The mathematical foundations must therefore extend beyond a static optimization statement to a controlled sequential process.

1.5.2 State, action, or representation spaces

The abstract space $\mathcal{X}$ becomes operational only through representations. In modern generative discovery, there is a consistent pattern: the physical object lives in a structured discrete space, but the model operates in a continuous embedding space. Governance requires clarity about this mapping because it is a primary source of fragility and mismatch.

We can represent candidates in at least three canonical ways:

(i) String spaces. Let Σ be an alphabet of tokens (atoms, bonds, branch markers, ring closures, or more abstract symbols). A string representation is $s \in \Sigma^*$, where Σ^* denotes finite-length sequences. Not all strings are valid; validity is a grammar constraint. Let $\mathcal{S} \subset \Sigma^*$ be the set of valid strings. Then $\mathcal{X}$ can be identified with $\mathcal{S}$ modulo canonicalization (because multiple strings can encode the same structure). The generator, if autoregressive, defines a distribution $p_\theta(s \mid c)$ over sequences.

(ii) Graph spaces. Let a molecular graph be $G = (V, E, \ell)$, where V are nodes (atoms), E edges (bonds), and ℓ label functions assigning atom and bond types. Let $\mathcal{G}$ be the set of graphs satisfying chemical rules (valence, charge consistency, allowed elements). Graph-based generators define distributions over $\mathcal{G}$, either by sequential construction or by latent-variable mechanisms.

(iii) Geometric (3D) spaces. For 3D conformations, a candidate can be represented as $x = (G, r)$ where $G \in \mathcal{G}$ is the connectivity graph and $r \in \mathbb{R}^{3|V|}$ are 3D coordinates. Physical meaning requires invariance to rigid motions: translation and rotation. A 3D generator therefore should define a distribution over (G, r) that is equivariant or invariant under the Euclidean group, which is a technical point but also a governance point: if the representation violates symmetries, the model may learn artifacts that do not correspond to physical reality.

In materials design, analogous representations exist: compositions can be treated as vectors in a simplex, crystal structures as graphs or periodic lattices, and microstructures as fields or images. The core point is that $\mathcal{X}$ is structured and constrained.

The pipeline typically introduces an embedding map $\phi : \mathcal{X} \to \mathbb{R}^d$ that maps a structured candidate into a continuous vector space. Predictors often operate as $\hat{f}(x) = h(\phi(x))$. Generators may also operate in latent spaces. For example, a latent-variable model introduces a latent $z \in \mathbb{R}^d$ and a decoder $D_\theta(z, c)$ that maps latent variables and conditions into candidates. Diffusion models, in one perspective, define a Markov process in a continuous space that gradually denoises toward samples that correspond to valid structured objects.

Conditioning c can itself be formalized. Let $c \in \mathcal{C}$ where $\mathcal{C}$ is a conditioning space containing property targets, constraint descriptors, and metadata. For example, c may include a target property vector $y \in \mathbb{R}^m$, constraint parameters $\kappa \in \mathbb{R}^k$, and a text embedding $t \in \mathbb{R}^{d_t}$. A general conditional generator then defines

$$x \sim p_\theta(x \mid c). \tag{1.3}$$

This is the formal representation of the "proposal mechanism" concept. Governance enters because c is not arbitrary. There must be an allowed set $\mathcal{C}_{\text{allowed}} \subseteq \mathcal{C}$ defining permitted objectives and constraints. Inputs outside $\mathcal{C}_{\text{allowed}}$ are not merely invalid; they are risk events.

To connect to sequential operation, we can frame the pipeline as a controlled process. At iteration t, the pipeline has a state S_t that includes (i) model parameters θ_t for the generator and ψ_t for predictors, (ii) a dataset $\mathcal{D}_t$ of past validated outcomes, (iii) a constraint registry $\mathcal{R}_t$, and (iv) resource budgets (simulation and lab capacity) C_t. The pipeline chooses actions A_t that include the conditioning c_t, the sampling plan (how many candidates, what diversity settings), and the selection

policy parameters. The generator produces a batch B_t, the evaluator produces scores, and the pipeline selects V_t for validation. Validation yields observations Y_t that update $\mathcal{D}_t$ and potentially update ψ_t and θ_t. This framing is not meant to turn the chapter into a control theory treatise; it is meant to make explicit that the pipeline is a sequential decision system, not a static generator.

The key takeaway is that representation choices are part of governance. The map from physical reality to representation to embedding to model outputs is where many silent failures originate. If the representation omits critical physical constraints, the model will exploit that omission. If the conditioning space is unconstrained, the system becomes vulnerable to misuse and mis-specification. Formalizing these spaces is therefore not pedantry; it is how an institution makes clear what the system can and cannot responsibly do.

1.5.3 Objective functions and constraints

In realistic discovery programs, objectives are multi-dimensional and constraints are layered. A governance-first mathematical treatment emphasizes two principles: (i) objective specification must be explicit and complete enough to prevent accidental optimization of the wrong proxy, and (ii) constraints must be enforceable as gates, not merely stated as intentions.

Let $f(x) \in \mathbb{R}^m$ be the property vector. A simple multi-objective formulation is

$$\max_{x \in \mathcal{F}} \; w^\top f(x), \tag{1.4}$$

where $w \in \mathbb{R}^m$ is a weight vector encoding priorities. This is common but fragile. It assumes that properties are commensurate after scaling and that trade-offs can be expressed linearly. In practice, organizations often impose threshold constraints on some properties and optimize others. For example, require toxicity proxy below a threshold and optimize efficacy proxy. This yields a constrained form:

$$\max_{x \in \mathcal{X}} \; f_{\text{perf}}(x) \quad \text{s.t.} \quad f_{\text{tox}}(x) \le \tau, \; f_{\text{cost}}(x) \le \gamma, \; g_i(x) \le 0.$$

This structure is governance-friendly because it separates non-negotiables (thresholds) from optimizables (performance). The challenge, again, is that f is unknown and proxies are used.

A more robust formalism uses *risk-sensitive* objectives under predictive uncertainty. Suppose the predictor provides $\hat{f}(x)$ and an uncertainty estimate $\hat{\sigma}(x)$. A conservative utility might be based on a lower confidence bound:

$$\text{LCB}(x) \;=\; \hat{f}_{\text{perf}}(x) - \lambda \hat{\sigma}_{\text{perf}}(x), \tag{1.5}$$

and then the pipeline selects candidates maximizing $\text{LCB}(x)$ subject to proxy constraints. Alternatively, the pipeline may optimize an acquisition function that trades off exploration and exploitation,

as in Bayesian optimization:

$$\alpha(x) \;=\; \hat{\mu}(x) + \beta\hat{\sigma}(x),$$

where $\beta > 0$ encourages exploration. The governance interpretation is straightforward: λ and β encode institutional risk posture and the value of learning relative to the value of immediate performance. An aggressive exploration posture is not wrong if the program is early-stage and the organization can tolerate failures; it is wrong if the program is near deployment and failures carry high cost. Therefore, these parameters must be tied to stage-gates.

Constraints can be written as a mix of hard and soft forms. Hard constraints define feasibility:

$$\mathcal{F} \;=\; \{x \in \mathcal{X} : g_i(x) \le 0 \; \forall i\}.$$

Soft constraints can be folded into the utility via penalty terms:

$$\max_{x \in \mathcal{X}} \; w^\top \hat{f}(x) - \sum_{i=1}^{k} \rho_i \max\{0, \hat{g}_i(x)\},$$

where $\rho_i > 0$ penalize constraint violations. From a governance standpoint, penalty formulations are dangerous if the constraint is truly non-negotiable. If a constraint is a safety exclusion, it should be hard, not soft. Penalties are appropriate for preferences, not for prohibitions.

A central concept in governance-first design is *specification completeness*. In mathematical terms, incomplete specification means that the true objective is $u(f(x))$ but the pipeline optimizes $\tilde{u}(\hat{f}(x))$ where $\tilde{u}$ omits relevant dimensions. The omitted dimensions become free variables that the optimization will exploit. For example, optimizing efficacy without including toxicity and feasibility effectively allows the system to trade toxicity and infeasibility for efficacy, because nothing in the objective penalizes those trade-offs. The governance remedy is to make critical dimensions explicit either as constraints or as penalties with hard thresholds.

This is also where provenance and hazard constraints enter. Let $h(x)$ be a hazard indicator function and $p(x)$ a provenance compliance indicator. Then a governed feasibility set includes:

$$\mathcal{F}_{\text{gov}} \;=\; \{x \in \mathcal{X} : g_i(x) \le 0,\; h(x) = 0,\; p(x) = 1\}.$$

In practice, h and p are often proxies. This further reinforces the need for conservative screening and human approval gates when proxies are imperfect.

Finally, the selection policy operates on finite batches. Given a batch $B_t = \{x_1, \ldots, x_N\}$, the policy chooses a subset $V_t \subseteq B_t$ of size at most M, where M is the validation budget. This is a combinatorial selection problem:

$$\max_{V \subseteq B_t,\, |V| \le M} \; \sum_{x \in V} \hat{u}(x) \quad \text{s.t. diversity and safety constraints.}$$

Diversity constraints can be formalized, for example, by requiring that pairwise similarity be below a threshold or by maximizing a submodular coverage function. The details vary, but the governance point is stable: selection is a policy, not an afterthought. It is the mathematical embodiment of "how we decide what to test."

1.5.4 Sources of instability or fragility

The mathematical framing clarifies why instability is not an edge case but a default risk. The first source of fragility is *predictor exploitation*, which can be formalized as a divergence between true and surrogate objectives. Suppose the predictor error is $\varepsilon(x) = \hat{f}(x) - f(x)$. If the pipeline selects candidates maximizing $\hat{f}(x)$, it effectively selects candidates maximizing $f(x) + \varepsilon(x)$. Even if $\varepsilon(x)$ has mean zero over the data distribution, selection shifts the distribution toward regions where $\varepsilon(x)$ is positive. As N (the number of sampled candidates) grows, the maximum of $\varepsilon(x)$ over the sampled set grows as well, purely by extreme-value effects. This yields a structural phenomenon: the best predicted candidates are increasingly likely to be those with the most favorable errors, not those with the best true properties. This is the core mathematical reason proxy hacking is inevitable under naive selection.

Uncertainty penalties mitigate but do not eliminate this effect, because uncertainty estimates can themselves be miscalibrated. If the system underestimates uncertainty in some region, it will still exploit that region. Therefore, calibration is part of stability.

The second source of fragility is *distribution shift*. Predictors and generators are trained on data from a historical distribution $P_{\text{train}}(x)$. The pipeline then uses the generator and selection policy to produce candidates from a different distribution $P_{\text{gen}}(x)$. Under optimization pressure, P_{gen} can move far from P_{train}. The error of predictors typically grows with this shift. Formally, generalization bounds often depend on divergence measures between the training and deployment distributions. The exact theorems are not needed here; the operational implication is enough: the more the system pushes into new regions, the less reliable its evaluation becomes. Therefore, the pipeline must monitor and control the distance between P_{gen} and P_{train}, using novelty measures and OOD detection.

The third source of fragility is *optimization pressure* created by iteration. A one-shot generator samples candidates. An iterative pipeline uses selection outcomes to refine generation. This refinement acts like a gradient step in distribution space: it increases probability mass in regions that scored well. If scores are proxy-driven, the pipeline concentrates probability mass in proxy-exploiting regions. This concentration can happen quickly because the generator is flexible and because selection is strong. The result is mode collapse not only in the generative sense but also in the scientific sense: the pipeline explores a narrow family of candidates that look excellent under proxies but are brittle in reality.

The fourth source of fragility is *compounding selection bias*. The pipeline learns from validated

outcomes, but those outcomes are not a random sample from $\mathcal{X}$. They are selected by the policy. This creates a biased dataset $\mathcal{D}_t$ that over-represents certain regions. If predictors are retrained on $\mathcal{D}_t$, they may become highly accurate in the selected regions but may degrade elsewhere. This can create a feedback loop that narrows exploration and makes the system blind to alternative candidates. It also complicates evaluation: the system's apparent accuracy on its own selected data may improve even as its real ability to generalize declines. Mathematically, this is a form of adaptive data collection bias, a well-known phenomenon in sequential decision systems.

The fifth source of fragility is *constraint proxy fragility*. Many constraints are enforced via learned proxies: predicted toxicity, predicted synthesizability, predicted stability. If these proxies are exploited or miscalibrated, the pipeline can produce candidates that pass constraints on paper but fail in reality. This is particularly dangerous for safety constraints. A false negative in hazard screening can have severe consequences. Therefore, safety constraints require conservative proxying and human review.

The sixth source of fragility is *metric collapse*. If the organization measures pipeline performance primarily through proxy metrics (benchmark scores, predicted property improvements, number of candidates generated), the selection policy may be tuned to maximize those metrics rather than real outcomes. This is a socio-mathematical failure mode: the objective function being optimized becomes the KPI function, not the scientific utility function. In formal terms, the organization changes $\hat{u}$ to maximize internal metrics, drifting away from the true utility u. Governance must therefore define metrics that are aligned with validated outcomes and that penalize proxy gaps.

The seventh source of fragility is *non-determinism and version drift*. If the generator and predictors change over time without versioning, the mapping from conditioning c to candidate distribution $p_\theta(\cdot \mid c)$ changes. Decisions become non-reproducible, making auditing and learning difficult. This is a stability failure at the system level rather than at the model level. Mathematically, it means the pipeline is operating with a moving objective and moving constraints, making it hard to attribute outcomes.

These sources of instability are not independent; they interact. Distribution shift increases predictor exploitation; exploitation increases selection bias; selection bias distorts retraining; retraining changes distributions further. The pipeline can therefore drift into a regime where it appears productive but is systematically wrong. The mathematical foundations provide the vocabulary to see this drift as a structural risk, not as bad luck.

1.5.5 What theory does not guarantee

A rigorous governance stance requires stating what the mathematics *does not* guarantee, because senior decision-makers are routinely tempted by a fallacy: if the formulation is crisp, the outcome must be reliable. In generative discovery, crisp formulations coexist with stubborn uncertainty.

First, theory does not guarantee *real-world synthesizability or manufacturability.* Even if a candidate is valid in representation and even if it passes a feasibility proxy, it may be infeasible given real constraints: reagent availability, process conditions, scale-up limitations, regulatory limits, or unmodeled chemistry. The mapping from structure to practical synthesis is complex and context-dependent. Retrosynthesis planning can help, but it is not a proof of feasibility. Therefore, no mathematical objective optimized over $\hat{g}(x)$ can guarantee real synthesizability.

Second, theory does not guarantee *safety.* Safety constraints rely on hazard screening, which is imperfect. Even high-quality toxicity predictors can have blind spots, and hazards can arise from rare structural motifs or from interactions not captured in training data. Moreover, safety is not only a property of the candidate; it is a property of the process by which the candidate is handled, synthesized, and tested. Mathematical constraints can reduce risk, but they cannot certify safety. This is why human oversight and conservative gating remain necessary.

Third, theory does not guarantee that *novelty implies usefulness.* The optimization problem can generate candidates that are far from known data, but distance in representation space does not imply scientific value. It may imply instability, infeasibility, or irrelevant novelty. A generator can produce novel artifacts easily; the bottleneck is proving that novelty corresponds to an improvement in real utility. Mathematical novelty metrics cannot substitute for evidence.

Fourth, theory does not guarantee that optimizing a surrogate objective improves the true objective. This is a central limitation. The surrogate objective $\hat{u}(\hat{f}, \hat{\sigma})$ is only as good as the predictors and uncertainty estimates. Under optimization pressure, surrogate objectives can diverge from reality. This means that even if the pipeline "converges" in the sense of producing higher surrogate scores over time, it may not converge toward better real candidates. Convergence in proxy space is not convergence in scientific space.

Fifth, theory does not guarantee that benchmark performance transfers to deployment. Benchmarks such as GuacaMol and MOSES provide controlled evaluation settings [6, 7], but real deployment adds constraints: feasibility, hazard screening, IP provenance, and organizational process. A model can excel on benchmarks and still fail in practice because the benchmark does not measure what matters. Therefore, benchmark improvements should be treated as component-level evidence, not as deployment readiness.

Sixth, theory does not guarantee that active learning or iterative feedback improves performance. Active learning can improve predictors if uncertainty estimates are meaningful and if selection avoids bias collapse. But if uncertainty is miscalibrated, active learning can focus on uninformative regions. If selection is biased, retraining can degrade generalization. Therefore, iterative loops can either improve or destabilize the system. The mathematics describes the possibility, not the guarantee.

Seventh, theory does not guarantee organizational accountability. A pipeline can be mathematically well-posed and still be operationally irresponsible if it lacks traceability, versioning, and human approval gates. Governance is not derived automatically from optimization. It must be built as

process and infrastructure.

The disciplined conclusion is that mathematical framing is a tool for clarity, not a promise of safety or success. It clarifies what is being optimized, where proxies enter, how constraints shape feasibility, and why instability is expected under optimization pressure. But it cannot guarantee real-world outcomes. That gap between mathematical certainty and physical uncertainty is exactly why governance must be first-class. A system that treats the gap as a minor detail will turn frontier capability into frontier risk. A system that treats the gap as the central design constraint will build a discovery pipeline that is slower in the short term but vastly more defensible over time.

1.6 Evaluation and Validation

1.6.1 Why naive metrics fail

Evaluation is the point where generative chemistry and materials design most reliably deceives well-intentioned teams. The deception is not malicious; it is structural. When a system can generate plausible candidates at scale, it becomes easy to measure the wrong thing and to feel confident about progress that is not real. In this domain, the phrase "the model produced valid molecules" is not a milestone; it is the beginning of the work. Validity is representational well-formedness. Scientific validity is empirical defensibility. Confusing the two is the fastest way to build a pipeline that looks productive while quietly converting validation budgets into disappointment.

The most common naive metric is *syntactic validity*: the fraction of generated outputs that parse correctly or satisfy basic chemical grammar. Validity matters because invalid outputs waste compute and indicate generator misalignment with the representation. But validity is a minimal constraint. A generator that produces 99% valid outputs can still be useless if those outputs are infeasible, unsafe, or irrelevant. Validity is a necessary condition for downstream evaluation; it is not evidence of scientific quality.

A second naive metric is *uniqueness* and *novelty* relative to a dataset. These metrics are attractive because they are easy to compute and because they reward the generative model for producing diversity rather than memorization. However, novelty relative to a dataset does not imply novelty relative to chemistry or materials science, and it does not imply value. A candidate can be novel because it is unstable, nonsensical, or hazardous. In other words, novelty without feasibility is not innovation; it is noise. If an organization treats novelty as a KPI, it risks optimizing for distance from known examples rather than for improvements that survive real constraints.

A third naive metric is *distribution matching* against a reference dataset. Many evaluation frameworks ask whether generated candidates resemble the training distribution. This can be useful for ensuring the generator has learned a plausible manifold, but it is not aligned with discovery goals. Discovery aims to find candidates that outperform what is known, which often means moving beyond the training distribution. If the evaluation metric rewards distribution matching, it implicitly discourages the very novelty the organization seeks. Conversely, if the organization pushes for novelty without controlling uncertainty, it increases the risk of out-of-distribution failure. The correct posture is not to pick one side. It is to treat distribution distance as a risk indicator and to stage-gate novelty based on uncertainty and validation capacity.

A fourth naive metric is *predicted property score*. If a pipeline reports that it can generate candidates with very high predicted activity or very high predicted conductivity, the story is seductive: the model is optimizing the objective. But this is precisely where proxy exploitation becomes likely. If the predictor is noisy, then maximizing its output over a large sample set tends to select candidates that exploit predictor error. Under repeated selection, the system can produce candidates whose

predicted scores climb dramatically while real outcomes do not. Therefore, predicted score is not a measure of progress unless it is paired with calibrated uncertainty, end-to-end validation statistics, and monitoring of the predicted-versus-observed gap.

A fifth naive metric is *benchmark leaderboard performance*. Benchmarks such as those used in molecular generation are valuable for research comparability, but they are incomplete as deployment criteria [6, 7]. They tend to emphasize representational properties (validity, uniqueness, novelty) and proxy property objectives, and they rarely incorporate feasibility, hazard screening, provenance, or real experimental outcomes. An organization that equates benchmark performance with readiness is optimizing for external legibility rather than internal truth. Benchmarks are unit tests for components; they are not acceptance tests for pipelines that touch physical reality.

The underlying reason naive metrics fail can be stated crisply: *they do not price the cost of being wrong.* In physical-world discovery, being wrong is expensive. It consumes scarce validation capacity, misallocates expert attention, and can create safety exposure. Metrics that ignore feasibility, uncertainty, and constraint adherence systematically under-measure risk. They reward the system for producing impressive artifacts rather than for producing candidates that survive the institution's real stage gates.

A governance-first evaluation regime therefore begins with a redefinition of success. Success is not "generate many valid candidates." Success is "generate candidates that survive conservative filters, maintain calibrated uncertainty under novelty, and translate into validated improvements at an acceptable cost per validated gain." Anything else is a demonstration, not a capability.

1.6.2 Appropriate evaluation units

Appropriate evaluation units shift the measurement focus from model-centric metrics to pipeline-centric outcomes. The unit of accountability is not a single generated molecule. It is the end-to-end process by which candidates are proposed, triaged, escalated, and validated. This shift mirrors Chapter 1's argument about trajectory evaluation: in domains where systems act over time and over loops, output-level evaluation collapses into false confidence. Generative discovery is an iterative system. It must be evaluated as such.

The first appropriate unit is *funnel performance*. A governed pipeline is a funnel with stages: generation → validity → feasibility → safety screening → property scoring → selection → higher-fidelity evaluation → validation. Each stage has a pass rate. Evaluating the pipeline means tracking pass rates and their stability over time. If the generator produces many candidates but the feasibility pass rate is near zero, the pipeline is not productive. If hazard flags spike when novelty increases, the exploration regime is unsafe. Funnel metrics are governance-relevant because they show where the system creates waste and where constraints bite.

The second unit is *calibration and uncertainty-vs-error*. The pipeline should measure how prediction

error relates to predicted uncertainty. This is the operational meaning of calibration. For a set of validated candidates $\{x_i\}$, one can compare predicted values $\hat{f}(x_i)$ with observed outcomes $f(x_i)$ and analyze whether higher uncertainty corresponds to larger errors. The exact statistical measures can vary, but the governance principle is stable: uncertainty must be meaningful enough to gate escalation. If uncertainty is not meaningful, the pipeline is effectively flying blind in novel regions.

The third unit is *out-of-distribution detection effectiveness*. OOD detection should be evaluated by measuring how often OOD flags correspond to large predictive errors or feasibility failures. Again, the precise metrics can vary, but the pipeline should have a clear picture of whether its novelty signals are informative. In a mature deployment, novelty is not a celebration metric; it is a risk metric. The evaluation unit therefore includes the detection rate of risky novelty and the false-negative rate of novelty-based screens.

The fourth unit is *predicted-versus-validated gap*. This is perhaps the most important operational metric for controlling proxy exploitation. The pipeline should track the difference between predicted scores and validated outcomes, not just in mean but across the distribution. If the pipeline consistently selects candidates that are predicted to be excellent but validate as mediocre, the system is over-optimizing the predictor. A widening gap over time is a red flag that sampling volume or optimization pressure has outpaced evaluation reliability. A governance-first program treats this gap as a control dashboard item with explicit thresholds that trigger interventions.

The fifth unit is *cost per validated improvement*. In executive terms, this is the metric that connects scientific ambition to resource reality. Validation is expensive, and improvements are rare. Therefore, the pipeline must be evaluated in terms of how much computational and experimental budget is consumed per validated gain. This can be framed relative to a baseline: how does the governed generative pipeline compare to a traditional heuristic-driven pipeline in terms of validated hit rate, time-to-hit, or improvement magnitude? This is where the business case lives. It is also where hype dies, because the metric forces the system to pay for its claims in real evidence.

The sixth unit is *decision traceability completeness*. This is often ignored in technical evaluation but is central to governance. For each validated candidate, can the organization reconstruct: the conditioning inputs used, the model versions, the scoring outputs, the filter decisions, the selection rationale, and the approvals? Traceability is an evaluation unit because an untraceable success is organizationally fragile. If success cannot be explained and reproduced, it cannot be scaled responsibly.

The seventh unit is *constraint adherence under pressure*. Many pipelines behave well in calm regimes and break under pressure—for example, when objectives are tightened, when novelty is increased, or when sampling volume is expanded. Evaluation must therefore include monitoring of constraint adherence as these pressures change. If hazard flags or feasibility failures increase disproportionately under tighter objectives, the system is not robust. Constraint adherence is therefore a core evaluation unit, not a compliance footnote.

These units share a theme: they are end-to-end and evidence-linked. They treat the pipeline as a system that produces candidates under constraints and under uncertainty, and they measure whether the system's outputs can be trusted enough to justify validation expenditure. That is the correct evaluation posture for a domain where the cost of being wrong is physical.

1.6.3 Robustness and stress testing

Robustness in generative discovery is not about whether the model can produce pretty samples. It is about whether the pipeline remains disciplined when conditions change. Stress testing therefore should be designed to simulate the ways real programs fail: by drifting into out-of-distribution regions, by over-optimizing proxies, and by misinterpreting scores as evidence.

A first robustness test is *held-out family evaluation*. Instead of random splits, the pipeline should evaluate predictors and generation quality on held-out chemical families or materials classes. This is a better proxy for real novelty: the model must generalize across families, not just across examples. When the pipeline is used for discovery, it will encounter families that are underrepresented or absent in training data. Held-out family stress tests measure how quickly predictor reliability collapses as the system moves into new regions. They also provide a baseline for deciding how aggressive exploration should be.

A second robustness test is *sensitivity to constraint changes*. In real programs, constraints evolve. Safety criteria tighten, cost constraints change, feasibility requirements shift with manufacturing realities. The pipeline should therefore be stress tested by varying constraint thresholds and observing how candidate distributions and pass rates respond. If small changes in constraints produce massive changes in selected candidates, the pipeline may be brittle. If hazard screening becomes ineffective when novelty increases slightly, controls are inadequate. Sensitivity analysis is not an academic exercise; it is the rehearsal for governance change management.

A third robustness test is *predictor noise injection*. Since predictors are proxies, the pipeline should be evaluated under perturbations: add noise to predictor outputs, alter model parameters slightly, or swap in alternative predictor ensembles. If selection decisions change radically under minor predictor variations, the pipeline is unstable and likely overfit to a specific proxy. A well-governed pipeline should show some stability: the top candidates may vary, but the overall selection should remain within a plausible region and should remain constraint-adherent. Noise sensitivity exposes proxy fragility.

A fourth robustness test is *adversarial optimization checks*. The pipeline should deliberately attempt to optimize predictors to extremes and then examine whether those extreme candidates are plausible under feasibility and hazard constraints. This test operationalizes the proxy hacking concern. If the pipeline can easily produce candidates with extreme predicted scores that nevertheless fail feasibility screens or trigger hazard flags, then naive selection would be dangerous. The purpose of the test is not to demonstrate that hacking is possible—it almost always is—but to quantify how quickly

it emerges as sampling volume and optimization strength increase. This quantification can then inform governance limits on sampling volume and objective tightening.

A fifth robustness test is *distribution shift simulation*. The pipeline can simulate shift by conditioning on targets outside the usual range or by restricting training data to a subset and evaluating on another. The goal is to measure how novelty and uncertainty measures behave under controlled shift. If OOD detectors fail to flag shifted regimes, then uncertainty gating is unreliable. If uncertainty estimates remain low even when error increases, calibration is broken. Shift tests therefore directly support governance decisions about exploration bounds.

A sixth robustness test is *diversity collapse detection*. Many pipelines, under iterative selection, collapse into narrow families. Stress testing should therefore include multi-iteration runs in which selection is repeated and the distribution of candidates is tracked over iterations. If diversity collapses quickly, the pipeline may be converging to a proxy-driven corner rather than exploring meaningful alternatives. Controls such as diversity quotas or submodular selection can be evaluated by whether they prevent collapse without destroying performance. This is a robustness question because diversity collapse reduces resilience and increases correlated failure risk.

A seventh robustness test is *human review robustness*. This is often omitted, but it matters. If the pipeline presents evidence to human reviewers, the interface and evidence packaging can bias decisions. Stress tests should include presenting reviewers with candidates that have similar predicted scores but different uncertainty and feasibility profiles and observing whether reviewers can distinguish them. If reviewers consistently choose high point-estimate candidates despite high uncertainty, the system is effectively incentivizing risk. The governance response is to redesign evidence presentation to foreground uncertainty and constraint adherence.

A final robustness test is *audit replay*. Given a prior run, can the pipeline be replayed to reproduce candidate generation, scoring, and selection? Replayability is a robustness property because it measures whether the system is stable enough to be investigated when something goes wrong. In physical domains, incident reconstruction is not optional. A pipeline that cannot be replayed cannot be governed.

Robustness testing therefore extends beyond model evaluation. It evaluates the pipeline under realistic perturbations: changing constraints, changing predictors, increasing sampling, iterating selection, and introducing distribution shift. The goal is to identify where the pipeline's controls fail, and to treat those failure points as governance-imposed boundaries on deployment.

1.6.4 Failure taxonomies

A practical evaluation regime must include a failure taxonomy that is tied to control actions. Taxonomies are not lists for their own sake; they are maps from observed behavior to interventions. In generative discovery, the main failure modes cluster into predictable categories.

The first category is *predictor gaming and proxy failure*. Symptoms include: extremely high predicted scores with poor validation, widening predicted-versus-observed gaps, and sensitivity of top candidates to minor predictor changes. This failure indicates that optimization pressure has outpaced predictor reliability. Control actions include: increase uncertainty penalties, cap sampling volume, require ensemble agreement, add adversarial stress tests, and allocate validation budget to calibration monitoring.

The second category is *feasibility failure*. Symptoms include: high validity but low feasibility pass rates, frequent synthesis route failures, and high complexity candidates that pass proxy feasibility but fail in practice. Control actions include: tighten feasibility gates, incorporate more realistic retrosynthesis checks, add context-specific constraints (reagent availability, process constraints), and increase expert review at feasibility choke points. Feasibility failure often indicates that the generator is not adequately constrained or that feasibility proxies are miscalibrated.

The third category is *overfitting to dataset distribution*. Symptoms include: strong benchmark performance, low novelty beyond training-like candidates, and poor generalization to held-out families. Another symptom is diversity collapse into narrow scaffolds or materials families. Control actions include: enforce diversity constraints, widen training coverage where possible, explicitly allocate exploration budget to novel regions, and evaluate on held-out families. Overfitting failure is often masked by naive metrics that reward distribution matching.

The fourth category is *unsafe generation pathways*. Symptoms include: hazard flags among generated candidates, generation of candidates in prohibited classes, or inability to explain why hazard screening failed. Control actions include: strengthen hazard exclusion rules, implement conservative safety screening early, restrict conditioning objectives, add access controls, and require human approval for any candidate near hazard boundaries. In severe cases, deployment should be deferred until safety filters are defensible.

The fifth category is *selection bias and feedback drift*. Symptoms include: improving predictor accuracy on selected data but worsening performance on broader evaluation sets, narrowing exploration over time, and instability after model updates. Control actions include: maintain stable holdout sets, control update cadence, log model versions, and introduce counterfactual evaluation by scoring random candidate samples rather than only selected candidates. Feedback drift is a system-level failure that emerges over time; it is not visible in one-shot evaluations.

The sixth category is *traceability and audit failure*. Symptoms include: inability to reconstruct conditioning inputs, missing logs, unclear model versions, or unexplained selection decisions. Control actions are infrastructural: implement immutable logging, enforce run manifests, require evidence packets for escalation, and prohibit pipeline stages that do not emit auditable artifacts. Traceability failures are governance failures regardless of whether the science succeeds.

The seventh category is *human-factor failure*. Symptoms include: over-trust in rankings, anchoring on top candidates despite high uncertainty, or bypassing controls due to perceived time pressure. Control

actions include: redesign interfaces to show uncertainty and constraints, implement role separation, and enforce approval gates that cannot be bypassed. Human-factor failures are predictable; they arise because the system produces persuasive quantitative artifacts.

A good taxonomy is operational. It ties each failure to measurable signals and to specific interventions. The evaluation regime should not merely detect failures; it should define what the pipeline does when failures appear. This is where evaluation becomes governance: the metrics are not passive indicators, they are triggers for control.

1.6.5 Limits of current benchmarks

Benchmarks play an important role in generative discovery, but they are structurally limited relative to real-world deployment. Their limitations matter because they shape what teams optimize and what executives believe. A governance-first chapter must therefore make the limitations explicit and treat them as reasons to build additional evaluation layers, not as reasons to dismiss benchmarks entirely.

First, most benchmarks are *offline*. They evaluate generation quality relative to a fixed dataset and a fixed set of proxy objectives. Offline evaluation cannot capture the dynamics of iterative optimization under uncertainty. It cannot capture how the pipeline behaves when it repeatedly selects top-scoring candidates and feeds back results. Proxy hacking, selection bias, and feedback drift are time-dependent phenomena. Offline benchmarks therefore systematically under-measure the risks that matter most in deployment.

Second, benchmarks often emphasize *representational validity and diversity* rather than feasibility and safety. Validity and diversity are important, but they do not capture whether candidates can be synthesized, whether they are stable, or whether they fall into hazard classes. Some benchmarks include simple property predictors or synthetic accessibility proxies, but these are rarely calibrated to a specific organization's constraints. Therefore, a model can perform well on benchmark metrics and still fail in the first week of real lab triage because the feasibility gate in reality is harsher than the feasibility proxy in the benchmark.

Third, benchmarks rarely incorporate *provenance and IP constraints*. In real deployments, provenance matters: what data trained the model, what constraints applied, and what rights exist over outputs. Benchmarks abstract away these considerations. This is not a flaw from a research perspective; it is a gap from a governance perspective. An institution cannot adopt a benchmark-winning model without understanding its provenance and output implications. Therefore, benchmark success must be supplemented by governance review.

Fourth, benchmarks have limited capacity to capture *hazard screening and misuse risks*. The dual-use nature of generative discovery means that a pipeline must prevent certain outputs. Benchmarks generally do not evaluate whether a model avoids hazardous classes or whether conditioning can

be misused to generate problematic candidates. Therefore, benchmark success can coexist with severe safety failures. Governance controls must therefore be evaluated independently of benchmark metrics.

Fifth, benchmarks often fail to represent *organizational cost structures*. In real discovery, the cost of false positives is high: it consumes lab cycles and delays progress. Benchmarks rarely price this cost. They may reward high-scoring candidates without penalizing the downstream burden of validating them. Therefore, a benchmark can encourage aggressive exploration strategies that are inefficient in practice. A governed program needs metrics tied to cost per validated improvement, not merely proxy score improvements.

Sixth, benchmark datasets can encode *historical biases* that are misaligned with a specific program's goals. A model trained and evaluated on common public datasets may excel at generating drug-like molecules in well-studied families but may be weak in a specialized domain such as novel catalysts or materials with specific processing constraints. Benchmark performance therefore does not necessarily transfer. The correct response is to build internal benchmarks and evaluation datasets aligned with the organization's objectives and constraints.

The disciplined conclusion is that benchmarks are necessary but insufficient. They can tell you whether a generator learns a plausible manifold, whether it produces valid structures, and whether it matches certain proxy objectives. They cannot tell you whether the pipeline is deployable in a governed environment. Deployability requires end-to-end evaluation with feasibility, hazard screening, uncertainty calibration, and traceability. Therefore, a mature program treats benchmarks as component tests and builds an additional evaluation layer that resembles a controlled rehearsal of the real discovery funnel.

In this sense, evaluation and validation are the true bottleneck of modern generative discovery. Generation is now abundant. What differentiates a serious institution from a demo-driven one is whether it can measure the right things, detect proxy-driven failures early, and maintain conservative gates that prevent optimistic artifacts from becoming expensive commitments. The frontier is not merely generating candidates; it is governing the process by which candidates become evidence.

1.7 Implementation Considerations

1.7.1 Minimal viable implementation patterns

A minimal viable implementation in generative chemistry and materials design is not the smallest model that can generate plausible candidates. It is the smallest *governable* pipeline that can propose candidates under explicit constraints, quantify uncertainty well enough to triage, and produce artifacts that a reviewer can interrogate. In this domain, "minimal viable" means "minimal viable governance," because an implementation that cannot be supervised is not merely immature; it is structurally unsafe and operationally misleading.

The first pattern is *constrained generation first, optimization later.* Early deployments should resist the temptation to build an aggressive objective-driven search loop. Instead, start with a conservative generator configuration that is explicitly bounded: allowed element sets, size bounds, known-safe scaffolds or composition ranges, and representation validity checks. The generator should be treated as a sampler from a bounded region of chemical or materials space. Conditioning should be limited to objectives that are understood and defensible at the organization's current maturity. If a program cannot articulate, in operational terms, what it means to "optimize" a property without violating feasibility or safety constraints, it should not implement iterative optimization. It should implement sampling plus filtering.

The second pattern is *filters as first-class citizens.* A minimal viable system should have explicit filter stages for: validity, deduplication, hazard exclusions, feasibility proxies, and basic novelty bounds. These filters should be applied early, and they should be deterministic given candidate and constraint registry. The goal is to avoid spending predictor and simulation budget on candidates that are obviously disallowed. Filters should not live as scattered heuristics in notebooks; they should live as a versioned constraint registry with named rules, owners, and justifications. This design choice is both technical and governance-driven: it makes it possible to audit what the system forbids, and it prevents silent changes to constraints.

The third pattern is *calibrated predictors with uncertainty thresholds.* A minimal viable pipeline can use relatively simple predictors, but it must treat them as probabilistic instruments. For each property estimate, the pipeline should produce an uncertainty signal and should use explicit thresholds to gate escalation. A practical pattern is to enforce "no escalation without confidence": candidates that are predicted to be excellent but carry high uncertainty are not moved forward until additional evidence is gathered (for example, agreement from an ensemble, additional cheap simulations, or expert review). This pattern directly combats proxy exploitation, because proxy exploitation tends to appear first in high-score, high-uncertainty outliers.

The fourth pattern is *portfolio selection rather than greedy top-k.* In early stages, organizations often select the top k candidates by predicted score. This is an almost perfect recipe for spending validation budget on predictor error. A better minimal pattern is to treat candidate selection

as a portfolio problem: select a set that balances predicted promise, feasibility, diversity, and uncertainty. Practically, this means setting quotas: some candidates with high conservative scores, some candidates that are diverse but plausible, and some candidates that are informative for calibration. Even without sophisticated selection algorithms, simple rules (for example, cap similarity, cap novelty, require minimum feasibility) can reduce correlated failure risk dramatically.

The fifth pattern is *explicit stage gates with human approvals*. The minimal viable system should define at least two escalation gates: (i) escalation to expensive simulation or deeper computational evaluation, and (ii) escalation to any form of lab-facing action (including synthesis planning that commits resources). Each gate should have an explicit approval role and an evidence packet. Evidence packets should include: conditioning specification, candidate representation and canonical identifier, predicted properties with uncertainty, feasibility signals, hazard screening results, novelty and similarity measures, and reasons the candidate was selected. The system should not permit lab escalation without documented human sign-off. This is not bureaucracy. It is the mechanism that prevents "model suggestion" from becoming de facto authority.

Finally, minimal viable implementations should avoid premature complexity. It is common to see teams attempt to build an end-to-end active learning loop with continuous retraining, automated lab execution, and multi-objective optimization in the first iteration. This is often counterproductive. Each added loop increases instability and reduces auditability. A disciplined progression is: (1) constrained generation + filters + logging, (2) calibrated prediction + uncertainty gating, (3) staged simulation escalation, (4) limited experimental validation with monitoring, and only then (5) controlled active learning and generator updates. Minimal viable, in this sense, is minimal stable governance.

1.7.2 Reproducibility and determinism

Reproducibility is not a courtesy to future researchers; it is the foundation of accountability in a physical-world pipeline. If a candidate fails in validation, the organization must be able to reconstruct why it was selected. If a candidate succeeds, the organization must be able to reproduce the process that led to it. Without reproducibility, the program becomes a collection of anecdotes and cannot scale responsibly.

The first reproducibility requirement is *versioning of everything that matters*. This includes: datasets (with hashes and provenance metadata), model checkpoints (generator and predictors), constraint registries (the exact rules in effect), and code versions (commit hashes). In a governed pipeline, the unit of work is a "run" that has a stable identity. Each run should record what versions were used so that the run can be replayed.

The second requirement is *controlled randomness*. Generative systems are stochastic. That is normal. But stochasticity must be bounded and reproducible. The pipeline should record random seeds for generation, and it should record sampling parameters. When a run is replayed with the same

seed and same versions, it should reproduce the same candidate set or at least reproduce the same distributional behavior depending on the generator. Full determinism may not always be possible due to hardware nondeterminism, but the principle is to minimize avoidable nondeterminism and to record what cannot be controlled.

The third requirement is *immutable logs and run manifests*. Every run should produce a manifest that includes: timestamps, environment fingerprint (software versions, hardware notes where feasible), data hashes, model checkpoint identifiers, constraint registry version, sampling parameters, and selection policy parameters. This manifest is the audit spine. It should be written automatically, stored immutably, and referenced by downstream artifacts such as candidate lists and validation reports. In mature programs, manifests become as standard as lab notebooks.

The fourth requirement is *checkpointing and rollback*. Models and constraints will change over time. The pipeline must therefore support rollback: the ability to revert to a prior version if a new update introduces drift or failures. This requires that old checkpoints and old constraint registries are preserved, not overwritten. It also requires that evaluation metrics are tracked across versions so that regressions can be detected. Without rollback, the program accumulates risk: each update is a one-way door.

The fifth requirement is *reproducible selection*. Even if candidate generation is stochastic, selection should be reproducible given candidates and evidence. If selection uses random tie-breaking or nondeterministic heuristics, decisions become harder to audit. Selection policies should therefore be deterministic functions of the evidence record, or they should record any randomness explicitly. This matters because selection is where the pipeline commits validation budget. An organization must be able to justify why a candidate was selected, and reproducibility is part of justification.

Reproducibility is often framed as a technical discipline. In this domain, it is also a risk control. It limits the damage of mistakes by enabling root-cause analysis, and it enables scaling success by enabling repeatable processes.

1.7.3 Logging and traceability

Logging is the difference between a pipeline that can be governed and one that cannot. Traceability is not achieved by writing occasional notes; it is achieved by designing the pipeline so that trace artifacts are emitted automatically at every stage. The goal is simple: any candidate that reaches expensive evaluation or lab validation should be accompanied by a complete “evidence trail” that can be audited and replayed.

A practical traceability design begins with *structured records per candidate*. Each candidate should receive a canonical identifier at the moment it is generated and canonicalized. That identifier should be stable across runs if the same candidate is regenerated, enabling deduplication and longitudinal tracking. The candidate record should include the raw representation produced by the generator,

the canonical representation after normalization, and the transformation steps applied. This is important because representation discrepancies can cause evaluation mismatches.

Next, the pipeline should log *conditioning context*. Conditioning inputs should be stored exactly as used, including numeric targets, constraint parameters, and any text conditioning (stored as text and, if applicable, embeddings). Conditioning is part of intent; therefore, it must be traceable. In some organizations, conditioning may contain sensitive strategic information. Traceability therefore must be designed with access control and redaction policies, but the record must exist.

The pipeline should then log *evaluation outputs as decomposed evidence*. For each candidate, store predicted properties, uncertainty estimates, feasibility scores, novelty and similarity measures, and hazard flags. Store the model versions used to produce these scores. Store any auxiliary information such as ensemble disagreement or OOD indicators. The key is to avoid collapsing evidence into a single score. Auditors and reviewers need to see why the candidate was considered promising and what risks were known at the time.

Filtering decisions must be logged as *explicit pass/fail events*. Each filter should record whether the candidate passed, which rule triggered a failure, and what threshold was applied. These events should reference the constraint registry version. This design supports accountability: if a candidate later triggers a safety incident, the organization can examine whether filters were misconfigured, bypassed, or inadequate.

Selection decisions require a higher standard: *selection rationale*. The pipeline should record why a candidate was selected for escalation. This does not require prose. It can be structured: the selection policy rule that triggered selection, the ranking position under a specified metric, the diversity constraints satisfied, and the uncertainty thresholds met. The record should also include the decision authority: which human approved escalation, when, and under what evidence packet. This is where traceability becomes governance: it links technical decisions to accountable roles.

Finally, traceability extends beyond candidate selection to *validation outcomes*. When a candidate is validated (in simulation or in lab), the results must be linked back to the candidate identifier and run manifest. This linkage enables performance monitoring: predicted-versus-observed gaps, calibration tracking, and drift detection. Without this linkage, the pipeline cannot learn responsibly, because it cannot connect outcomes to the decisions that produced them.

A mature traceability system also supports *audit replay*. An auditor should be able to pick a candidate and reconstruct the entire chain: conditioning $\rightarrow$ generation parameters $\rightarrow$ candidate output $\rightarrow$ scores $\rightarrow$ filter outcomes $\rightarrow$ selection $\rightarrow$ approvals $\rightarrow$ validation results. This is the standard that physical-world pipelines should aim for, because the cost of not being able to reconstruct is high: the organization cannot defend itself, cannot learn systematically, and cannot scale.

1.7.4 Cost, latency, and scaling issues

In generative discovery, cost and latency dynamics are counterintuitive for organizations accustomed to digital products. Generation is cheap; validation is expensive. Compute may be expensive relative to other software workloads, but it is typically cheap relative to lab work. The bottleneck therefore shifts away from generation and toward evaluation and validation. The practical consequence is that scaling the generator does not scale the program. It scales the queue of candidates waiting for evidence.

A minimal cost model is a staged funnel. Let N be the number of candidates generated. Let c_{gen} be the marginal cost of generation per candidate, c_{pred} the cost of predictor evaluation, c_{sim} the cost of higher-fidelity simulation, and c_{lab} the cost of lab validation. Typically,

$$c_{\text{gen}} \ll c_{\text{pred}} \ll c_{\text{sim}} \ll c_{\text{lab}}.$$

Even if c_{pred} is nontrivial (ensembles, complex predictors), it remains orders of magnitude below lab validation. Therefore, the pipeline's economic success depends on how effectively early-stage filters and predictors reduce N down to a small set M that enters expensive stages. The key scaling metric is not candidates generated per hour; it is candidates validated per unit cost with a positive hit rate.

Latency matters because it determines iteration speed, but iteration speed must be interpreted carefully. Faster iteration is valuable only if the loop is learning, not merely consuming resources. If the pipeline cycles quickly through proxy-exploiting candidates, it can waste budget faster. Therefore, organizations should treat acceleration as a controlled resource. Early-stage deployments often benefit from slower, more reflective loops that emphasize calibration and constraint tuning. Speed becomes valuable later, once controls are stable.

Compute constraints also interact with uncertainty. Ensembles and calibration methods increase compute load. Some teams respond by simplifying uncertainty estimation, which can increase risk. A governance-first approach treats uncertainty computation as a cost worth paying because it reduces expensive mistakes downstream. The right trade-off is often to spend more compute to reduce lab waste.

Scaling issues also appear in *throughput bottlenecks*. High-throughput generation can overwhelm human reviewers. Therefore, selection policies must be designed to output reviewable candidate sets, not endless lists. The pipeline should have quotas and caps that reflect human bandwidth. It should also have automation in evidence packaging: reviewers should not be manually assembling evidence packets. Review processes must be designed as part of the pipeline.

Finally, scaling must consider organizational resource constraints. Lab capacity is finite. Synthesis teams have queues. Safety reviews take time. Procurement and compliance processes add friction. A pipeline that ignores these realities will either stall or bypass controls. A mature program therefore

treats the entire pipeline as a resource allocation system, where the goal is to maximize validated improvement under real throughput constraints. This is not only a technical design problem; it is a program management problem.

1.7.5 Integration risks

Integration risks are where promising technical prototypes fail in the real world. The most common integration risk is *misinterpretation*: treating AI-suggested candidates as validated. This risk emerges because model outputs are quantitative and persuasive. When a pipeline outputs a ranked list with predicted properties, humans naturally infer that the ranking reflects evidence. If uncertainty and constraints are not foregrounded, the system encourages overconfidence. The integration mitigation is evidence discipline: every candidate must carry uncertainty, feasibility, and hazard context, and escalation must be gated by explicit approvals.

A second risk is *incentive drift*. Teams are often rewarded for producing outputs that look impressive: high predicted scores, high novelty, rapid iteration. These incentives can diverge from the true objective of validated progress. Incentive drift is a governance failure because it turns internal metrics into de facto objectives. The mitigation is to align performance measurement with end-to-end outcomes: validated hit rates, cost per validated improvement, calibration stability, and compliance with traceability requirements. If teams are not rewarded for governance discipline, discipline will erode.

A third risk is *organizational handoff failure*. Generative pipelines often sit between computational teams and experimental teams. If the handoff is informal—a spreadsheet of candidates with little context—experimental teams may waste time or reject suggestions, leading to mistrust. Conversely, if computational teams push candidates into experimental queues without respecting lab constraints, they create friction and potentially safety issues. The mitigation is to formalize the handoff: define evidence packets, define acceptance criteria for lab consideration, and define communication loops where experimental feedback is logged and used to improve the pipeline.

A fourth risk is *control bypass*. Under time pressure, users may bypass hazard screening, bypass uncertainty gates, or manually alter candidate lists. This is a predictable organizational behavior. The mitigation is to design controls that are hard to bypass: enforce gating in the pipeline itself, use role-based permissions, and require that any override is logged with justification and approval. Overrides should be treated as risk events, not as routine.

A fifth risk is *model update drift*. If models are updated frequently without controlled release processes, downstream teams may experience inconsistent behavior. A candidate that would have been selected last week may not be selected this week, not because the science changed, but because the model did. This undermines trust and makes auditing difficult. The mitigation is a release cadence with validation checks, clear versioning, and change logs. In many organizations, this resembles the discipline used for deploying financial risk models: changes are reviewed, tested, and

approved.

A sixth risk is *security and dual-use exposure*. Even when a program is intended for benign discovery, the same system may be misused. Integration therefore includes access control, monitoring, and restrictions on objectives and exports. This is especially important when conditioning interfaces accept free-form text or when the system can generate large candidate sets quickly. Security is not separate from integration; it is part of it.

The overarching integration lesson is that generative discovery systems fail not because the models are weak, but because the organization treats them as suggestion engines without building the governance scaffolding that turns suggestions into defensible actions. Implementation considerations therefore converge on a single principle: build the pipeline so that the path from candidate generation to lab action is narrow, evidence-rich, and auditable. The frontier advantage is not producing more candidates. It is producing candidates that can be advanced responsibly, with clear accountability for each escalation decision.

1.8 Impact and Opportunity

1.8.1 New capabilities unlocked

The most important capability unlocked by generative foundation models in chemistry and materials is not that they can "invent molecules." The deeper shift is that they make *structured exploration cheap*. In domains where the search space is combinatorially vast and human attention is finite, the ability to propose thousands of plausible candidates under specified conditions changes the tempo of discovery. The system becomes an ideation engine that can explore beyond the immediate reach of human intuition and beyond the limitations of hand-curated libraries.

The first unlocked capability is *rapid exploration of structured spaces*. Traditional discovery often proceeds through incremental modifications: change a substituent, adjust a composition, tweak a synthesis condition. Generative models can sample broadly within the constraints they are given, producing candidates across a wide region of chemical or materials space. This does not guarantee that the candidates are good, but it does guarantee that the organization can traverse the space faster than a purely manual workflow. The strategic implication is that the "frontier" is no longer defined by how quickly you can imagine candidates; it is defined by how quickly you can evaluate them credibly.

The second capability is *conditioning and targeted ideation*. In earlier paradigms, guiding search toward specific property profiles required hand-crafted heuristics and expert steering. Modern generators can be conditioned on property targets, constraints, or textual specifications, enabling more direct exploration of design goals. Properly governed conditioning makes the generator a controllable tool: it can be asked to propose candidates that satisfy a particular constraint regime, or that lie within a particular design envelope. This is especially valuable in materials contexts where objectives are multi-dimensional and where trade-offs are central.

The third capability is *diversity at scale*. Even if a team can conceive a handful of candidate families, it is difficult to systematically generate diverse candidates that remain plausible. Generative models can produce candidate sets that span multiple families, increasing the chance that the pipeline explores distinct mechanisms rather than variations of the same theme. Diversity is not merely a research benefit; it is a risk control. When validation budgets are scarce, spending them on correlated candidates increases the probability of correlated failure. Diverse candidate portfolios can improve the information gained per experiment.

The fourth capability is *compression of early-stage iteration*. A common pattern in R&D is that early ideation is slow and then accelerates once a promising family is found. Generative tools can shorten the early stage by quickly proposing plausible candidates and by enabling faster triage with predictors. This can be particularly valuable in programs where time is strategic: rapid identification of promising families can change competitive timelines. However, this acceleration is valuable only if it is governed; otherwise, it accelerates false positives.

The fifth capability is *integration with computational screening.* Generative pipelines can be naturally paired with simulation and predictive screening. The generator proposes candidates, predictors triage them, higher-fidelity simulations refine the shortlist, and experiments validate. This layered approach creates a computational discovery funnel that can be tuned to the organization's constraints. The value is not that the generator replaces experiments; the value is that it can allocate experimental effort more intelligently by reducing the space before lab action.

Taken together, these capabilities redefine what "early discovery" means. They shift early discovery from a human-limited ideation bottleneck to an evaluation-limited governance bottleneck. This is why the opportunity is inseparable from validation and oversight. The capability is real, but it changes where the organization must invest discipline.

1.8.2 Organizational implications

If generative discovery is adopted seriously, it changes organizational structure. The naive assumption is that AI is a tool that sits alongside existing workflows. The more accurate assumption is that AI becomes a *pipeline* that reshapes ownership, roles, and accountability boundaries. The key governance question is: who is responsible for the decisions that move candidates forward?

First, R&D becomes more computational by default. This does not mean that labs disappear. It means that computation becomes the front door of the funnel. Candidate volumes become too large for purely manual ideation, and the pipeline's triage logic becomes a material driver of where experimental effort goes. As a result, organizations need roles that did not previously exist at scale: model risk managers for scientific ML, data stewards for chemical and materials datasets, and pipeline engineers who can maintain reproducible, auditable systems.

Second, organizations must define *approval gate ownership.* In a governed deployment, candidates do not progress because a model ranked them highly. They progress because a responsible party approved them under specified evidence thresholds. This creates a need for explicit gate owners. In practice, different gates may have different owners: computational chemistry owns the gate to simulation; synthesis planning owns the gate to lab feasibility; safety owns the gate to hazard clearance; and project leadership owns the gate to resource commitment. Without explicit ownership, the system will drift toward implicit authority: the model output will become the de facto gate.

Third, accountability boundaries must be clarified. When a model proposes a candidate, who is accountable for its consequences? The model developer? The scientist who accepted it? The manager who approved the budget? In well-governed environments, accountability should be layered and documented: the model is an instrument, the selection policy is the decision logic, and humans approve escalation. The organization must also define what counts as "use" of the system. Is it acceptable to generate candidates privately without logging? If so, how does governance hold? A mature program defines that any candidate intended for escalation must originate from a logged, versioned run with an evidence trail.

Fourth, the introduction of generative pipelines often reveals a cultural tension between computational teams and experimental teams. Computational teams may view the generator as a breakthrough; experimental teams may view it as an additional source of poorly grounded suggestions. The resolution is not persuasion. It is evidence. If the pipeline improves validated hit rates or reduces wasted experiments, trust grows. If it produces impressive predicted scores with poor validation, trust collapses. Therefore, organizational adoption depends on designing feedback loops that incorporate experimental outcomes into pipeline evaluation and on treating experimental teams as co-owners of acceptance criteria.

Fifth, governance introduces program management overhead that must be resourced. Versioning, logging, audit trails, and review gates require time. Organizations that treat this overhead as optional will either bypass it or create brittle processes that fail under pressure. The correct framing is that governance is part of the cost of discovery in this regime, just as safety protocols and quality control are part of the cost of lab work. The opportunity is not cheap discovery; it is faster and more disciplined discovery under constraints.

In short, the organizational implication is that generative discovery is not a plug-in. It is a new R&D operating model. Institutions that succeed will not merely buy a model; they will redesign their discovery pipeline around controlled generation, structured evidence, and explicit accountability.

1.8.3 Potential new industries or markets

The emergence of generative discovery at scale is creating the conditions for new markets, not only new tools. The most visible market is *AI-native discovery platforms*. These platforms aim to provide an integrated stack: candidate generation, property prediction, feasibility screening, and pipeline management, often with user interfaces tailored to domain scientists. The platform opportunity exists because few organizations want to build the entire stack from scratch, and because the stack requires specialized expertise across ML, cheminformatics, simulation, and governance infrastructure.

A second market is *lab automation feedback loops*. When robotic experimentation and automated measurement are integrated with generative pipelines, the system becomes a closed-loop discovery engine: propose, test, learn, propose again. This creates a market for "discovery ops" infrastructure: workflow orchestration, experiment scheduling, data capture, and integration between computational outputs and lab execution. The differentiator will not be model sophistication alone. It will be the quality of the integration and the governance controls that prevent the system from running ahead of safety and evidence.

A third market is what might be called *discovery factories*. In this vision, discovery becomes more like manufacturing: standardized pipelines that can be applied to different targets or materials classes with configurable objectives and constraints. The factory metaphor is attractive because it suggests repeatability and scale. But it also implies that governance becomes industrial: controls, audits, role separation, and incident response become standard operating procedures. A discovery

factory without governance is simply a factory for producing unvalidated candidates.

A fourth market is *specialized validation services*. If generation is abundant, validation becomes the scarce resource. This creates opportunities for organizations that can provide high-throughput, high-quality validation: specialized simulation services, automated synthesis services, or contract labs that can execute standardized validation protocols. In a mature ecosystem, the value chain may shift: those who can validate efficiently and reliably may capture significant value, because they control the conversion of model outputs into evidence.

A fifth market is *governance and compliance infrastructure for scientific AI*. As organizations deploy generative discovery systems, they will face internal and external demands for traceability, provenance, and safety assurance. This creates demand for tooling that looks, structurally, like model risk management in finance: run manifests, audit trails, change management dashboards, access control, and monitoring of drift. The market is not only for discovery models; it is for the control systems that make discovery models deployable.

The strategic implication for executives is that the opportunity is not limited to "better R&D." It includes platform ecosystems and organizational capability building. However, the same warning applies: the market will reward those who can turn candidate generation into validated outcomes under accountability. Platform value will not be measured by how many candidates it can generate, but by how effectively it can support disciplined selection and validation.

1.8.4 Second-order effects

Every capability shift produces second-order effects. In generative discovery, the most important second-order effect is the *overproduction of unvalidated candidates*. When candidate generation becomes cheap, organizations can easily generate far more candidates than they can evaluate. This creates a new kind of operational risk: attention dilution. Teams can spend time triaging candidate lists, discussing predicted scores, and producing presentations about pipeline output, while validated progress remains limited. The result can be an "activity trap" where the organization appears productive but accumulates little real evidence.

A second second-order effect is *concentration risk*. If a field converges on a small number of foundation models, datasets, or platform providers, then many organizations may end up exploring similar candidate spaces and making similar mistakes. This concentration can reduce diversity of approaches and increase systemic vulnerability: a shared blind spot in a widely used model can propagate across the ecosystem. In a competitive environment, it can also reduce differentiation. Organizations that rely heavily on the same models may find that their discovery output becomes less distinctive than expected.

A third effect is *provenance ambiguity*. Generative models are trained on data from multiple sources: public datasets, proprietary datasets, literature-derived structures, and sometimes vendor data.

Outputs can therefore have unclear lineage. Even when a generated candidate is "novel" in a representational sense, questions may arise: is it derived from a known patented structure? Is it a close analog of something in the training data? What rights exist to use it? Provenance ambiguity can lead to IP disputes and can slow commercialization even if the science works. This is why provenance tracking is not an academic exercise; it is a strategic risk control.

A fourth effect is *metric-induced behavior*. As generative discovery becomes more common, organizations may begin to compare themselves via easy metrics: number of candidates generated, novelty scores, benchmark performance. This can distort incentives and push teams toward optimizing what is legible rather than what is meaningful. The second-order effect is a drift toward performative discovery, where the pipeline is tuned for dashboards rather than for validated outcomes.

A fifth effect is *organizational boundary tension*. If discovery becomes more computational, the balance of influence between computational teams and experimental teams shifts. This can create tension over resource allocation and decision authority. If computational teams dominate without respecting experimental constraints, the program can become disconnected from reality. If experimental teams reject computational inputs wholesale, the program loses the opportunity. The second-order risk is cultural: an organization can fail not because the technology is weak, but because it cannot integrate new decision flows and accountability structures.

A sixth effect is *safety externalities*. Even benign discovery tools can have misuse pathways. Increased availability of generative design systems can lower the barrier to exploring hazardous spaces. This is not a claim about intent; it is a claim about capability diffusion. The second-order effect is that institutions may face scrutiny not only for what they do intentionally, but for what their systems could enable. This increases the need for access controls, monitoring, and responsible disclosure practices.

These second-order effects are not reasons to avoid the opportunity. They are reasons to treat opportunity as a systems problem. The organizations that capture value will be those that understand that scaling generation scales not only possibility but also confusion, risk, and governance burden.

1.8.5 Overstated expectations

The frontier narrative around generative discovery often implies that AI will replace experimentation or that discovery will become primarily computational. This expectation is overstated in ways that matter for strategy and budgeting. A governance-first view emphasizes that experiments remain essential, winners remain rare, and progress must be measured in risk-adjusted evidence, not in candidate volume.

First, experiments remain essential because models cannot certify reality. Predictors and simulators can triage, but they cannot replace empirical validation. Even high-fidelity simulations depend on assumptions and approximations. The physical world contains failure modes that are absent

from datasets and models: impurities, process variations, scale effects, and unexpected interactions. Therefore, any deployment narrative that reduces the role of experiments is not merely optimistic; it is operationally dangerous.

Second, winners remain rare. Discovery is inherently a low hit-rate activity. Generative models can improve the efficiency of search, but they do not change the fundamental scarcity of truly high-performing, feasible, safe candidates. In fact, they can create the illusion of abundance by producing many plausible candidates. The organization must resist conflating plausible with successful. The correct measure is not how many candidates the pipeline produces, but how many validated improvements it delivers per unit time and cost.

Third, progress must be risk-adjusted. A candidate that appears promising but carries high uncertainty and high feasibility risk is not equivalent to a candidate that is slightly less promising but far more reliable. In high-accountability environments, the institution's objective is not to maximize the expected best-case outcome; it is to maximize progress subject to constraints and acceptable tail risk. This is why uncertainty gating and conservative selection are not merely technical details; they are strategic choices.

Fourth, generative discovery does not eliminate the need for domain expertise; it changes how expertise is used. Experts move from inventing candidates to defining constraints, interpreting evidence, and approving escalation. This is a meaningful shift, but it is not a reduction in expertise demand. It is often an increase, because experts must now supervise a system that can produce far more options than a human could generate alone.

Finally, the most important expectation to reset is the meaning of "speed." Faster iteration is valuable only if it produces validated learning. A pipeline that generates candidates quickly but validates slowly, or validates unsuccessfully, is not accelerating discovery; it is accelerating waste. The organizations that win will be those that invest in validation infrastructure, uncertainty calibration, and governance controls that keep the pipeline aligned with evidence.

The disciplined opportunity statement is therefore modest but powerful: generative foundation models can expand the proposal space and accelerate early triage, enabling organizations to allocate validation resources more intelligently. They can create new platform markets and new discovery operating models. But they do not repeal the fundamental constraints of physical-world discovery. They make those constraints more visible by shifting the bottleneck from ideation to validation. Executives should treat the opportunity as a governance and systems design problem: the value is unlocked not by generating more candidates, but by building pipelines that convert abundant proposals into scarce, defensible evidence.

1.9 Governance, Risk, and Control

1.9.1 New risk categories

Generative chemistry and materials design changes the risk profile of AI deployment because the outputs can cross the boundary from text to matter. In purely informational domains, a wrong answer is embarrassing or costly; in physical-world domains, a wrong candidate can waste months of work, compromise safety, and create liability. Governance in this section is therefore not a generic compliance posture. It is the institutional machinery that keeps a proposal engine from becoming an unaccountable decision engine.

The first new risk category is *safety risk from hazardous candidates*. This includes accidental generation of candidates that are hazardous to handle, hazardous to synthesize, or hazardous in downstream use. The risk is not limited to explicit toxicity; it includes reactive intermediates, unstable compounds, dangerous byproducts, and material hazards associated with processing conditions. Safety risk also includes the possibility that the pipeline's constraints under-specify safety, allowing candidates that satisfy property objectives while hiding risks in unmodeled dimensions. A distinctive feature of generative systems is that they can produce candidates that fall outside routine experience. That novelty is part of the opportunity, but it also means the system can surface hazards that historical screening heuristics have not encountered.

The second risk category is *validation risk*: the mistaken belief that predicted properties are real. In practice, this is often the largest operational risk. The system produces candidates with high predicted performance, and organizational incentives push teams to treat those predictions as evidence. Validation risk includes proxy exploitation (selecting candidates that exploit predictor error), miscalibrated uncertainty (treating uncertain predictions as reliable), and benchmark overreach (treating offline metrics as deployment readiness). This risk is amplified by the persuasive nature of quantitative outputs. A ranked list with predicted scores looks like disciplined decision-making even when it is not.

The third risk category is *misuse pathways and dual-use exposure*. The same generative tools that support benign discovery can, in principle, be repurposed to explore harmful spaces. The governance question is not whether an institution intends harm; it is whether the institution's controls prevent inappropriate use, unauthorized conditioning, and unmonitored export of candidate lists. Misuse risk includes internal misuse (employees using tools outside approved scope), external misuse (compromised access), and downstream misuse (shared models or outputs used beyond intended contexts). Dual-use risk is not hypothetical in physical domains; it is a recognized feature of capability diffusion.

The fourth risk category is *IP and provenance risk*. Generative models learn from data. Outputs can be derivative in ways that are not obvious. Even when a candidate appears novel relative to a dataset, it may be an analog of a patented structure or may embed patterns from proprietary

sources. Provenance risk includes unclear rights to training data, unclear rights to outputs, and inability to demonstrate the lineage of candidates. This becomes an acute business risk when an organization attempts to commercialize a discovery and faces IP challenges. Provenance also intersects with scientific integrity: if a model's training data is not well-curated, the pipeline can reproduce historical biases and blind spots.

The fifth risk category is *liability surface expansion*. When AI outputs influence physical experimentation or product development, the chain of responsibility becomes more complex. If a candidate causes harm, who is liable? The institution that deployed the pipeline, the team that approved the candidate, the vendor that provided the model, or the individuals who executed the experiment? Liability risk is not purely legal; it is operational. It forces the institution to demonstrate that it exercised reasonable controls, that decisions were made with evidence, and that hazardous pathways were excluded. This is why auditability is not optional; it is part of defensibility.

The sixth risk category is *organizational decision laundering*. Decision laundering occurs when an institution uses AI outputs to justify choices that are effectively unreviewed, shifting responsibility onto the system. In discovery settings, this can appear as: "the model selected these candidates," or "the platform recommended this route." Even if humans sign off, the signature can become ceremonial if the evidence is not interpretable or if the pipeline overwhelms reviewers. Decision laundering is a governance risk because it erodes accountability while preserving the appearance of process.

The seventh risk category is *security and data leakage risk*. Discovery pipelines often use sensitive data: proprietary structures, experimental outcomes, supplier constraints, and strategic targets. If the system is integrated with external services or shared environments, data leakage becomes a risk. Additionally, if candidate outputs are exported without controls, they can reveal strategic directions or proprietary insights. Security in this domain is therefore not only about model misuse; it is about protecting sensitive R&D information.

These risk categories are intertwined. Validation failures can lead to unsafe experimentation. Provenance failures can lead to commercialization disputes. Misuse pathways can create reputational and regulatory exposure. Governance is the discipline that treats these risks as system properties, not as after-the-fact surprises.

1.9.2 Control points and safeguards

Controls in physical-world generative pipelines must be layered. No single safeguard is sufficient because the system can fail in multiple ways: by generating hazardous candidates, by exploiting predictors, by drifting out of distribution, or by being misused. Layered controls ensure that failures are caught early and that escalation requires increasingly strong evidence.

A first control point is the *constraint registry*. This is a versioned, explicit list of constraints that

define what the system is allowed to generate and escalate. Constraints should include feasibility requirements, hazard exclusions, provenance requirements, and operational limits (such as maximum novelty beyond training coverage without special approval). The registry should have named owners for each constraint category: safety owners for hazard exclusions, scientific owners for feasibility constraints, and legal/compliance owners for provenance rules. The registry transforms what is often an informal set of expectations into an enforceable control surface.

A second control point is *hard constraint enforcement in the pipeline*. Constraints that represent prohibitions (hazard exclusions, forbidden element classes, prohibited objectives) must be enforced as hard gates, not as penalties in a scoring function. Hard constraints should be applied early, preferably before expensive evaluation, and should be deterministic given candidate and constraint version. Hard constraints should also be enforced at the conditioning layer: if an objective or conditioning request is outside approved scope, the system should reject it. This prevents the pipeline from being driven into disallowed regimes by user inputs.

A third control point is *hazard exclusion classes and conservative safety screening*. Hazard screening should combine rule-based exclusions with learned screening where appropriate, but it should be designed to minimize false negatives. In safety-critical domains, false negatives are more dangerous than false positives, though excessive false positives can pressure users to bypass controls. Therefore, safety screening should be staged: early conservative screens remove obvious hazards, and later expert review examines borderline cases. Importantly, safety screening should be integrated with access controls: users should not be able to disable it.

A fourth control point is *uncertainty-gated escalation*. Predicted properties are proxies; therefore, escalation decisions should depend on uncertainty and out-of-distribution signals. A practical safeguard is to require that any candidate escalated to expensive validation meets uncertainty thresholds and, ideally, agreement thresholds across multiple predictors or ensembles. This reduces proxy exploitation by preventing the system from escalating extreme predictions that are likely driven by error. Uncertainty thresholds should be calibrated and reviewed; they are not arbitrary technical parameters but risk controls.

A fifth control point is *multi-stage approval gates*. The pipeline should enforce stage gates for: escalation to higher-fidelity simulation, escalation to synthesis planning that commits resources, and escalation to any experiment execution. Each gate should require explicit approval by a designated role and should be accompanied by an evidence packet. The evidence packet should include: constraint checks, hazard screening results, predicted properties with uncertainty, feasibility assessments, and novelty indicators. The approval process should be designed so that the approver can reject candidates without penalty and can request additional evidence. This keeps human oversight substantive.

A sixth control point is *model risk management for scientific predictors*. Predictors are critical components. They require the same discipline as financial risk models in regulated settings:

calibration checks, performance monitoring, drift detection, and controlled updates. Model updates should not be ad hoc. They should follow a release process that includes evaluation on stable holdout sets, stress tests on held-out families, and monitoring of predicted-versus-validated gaps. This is particularly important because the pipeline's selection behavior can create feedback drift. Without model risk management, the system can quietly degrade while appearing productive.

A seventh control point is *rate limits and quotas*. Because candidate generation is cheap, pipelines can generate far more than can be reviewed or validated. Rate limits and quotas are therefore safety controls. They prevent the system from flooding downstream stages and they reduce the probability of discovering hazardous or proxy-exploiting outliers simply through volume. Quotas can be tied to maturity: early-stage pilots may allow limited generation volumes under close supervision; later-stage systems may allow larger volumes but only with robust monitoring and audit.

An eighth control point is *role-based access control (RBAC)* and segregation of duties. Not every user should be able to condition the generator on arbitrary objectives, export candidate lists, or approve escalation. Roles should be defined: proposal roles can generate candidates; validation roles can run simulations; approval roles can authorize lab action; execution roles can perform experiments. This separation reduces decision laundering and creates accountability. It also limits misuse pathways because access to powerful capabilities is restricted.

A ninth control point is *secure artifact handling*. Candidate lists, model checkpoints, and experimental outcomes are sensitive. Controls should specify where artifacts are stored, who can access them, and how exports are monitored. In some programs, outputs should be watermarked or tagged with provenance metadata so that they can be traced if leaked. Secure handling is a governance requirement because the outputs are not merely ideas; they can represent valuable IP and potential safety exposure.

The central principle is that controls must be enforceable in software and defensible in audit. A policy that exists only as a PDF is not a control. A control is a gate that the pipeline cannot bypass without generating an auditable exception.

1.9.3 Human oversight boundaries

Human oversight is often discussed as a slogan: "human in the loop." In physical-world discovery, slogans are insufficient. The oversight boundaries must be explicit: what decisions must humans make, what evidence must they see, and what they are not permitted to delegate to the system.

The first boundary is that *no candidate may be escalated to real-world synthesis or experimentation without human review and approval*. This is a non-negotiable principle for governed deployment. The pipeline can propose and triage, but it cannot authorize physical action. Human review must be substantive, meaning that the reviewer has access to decomposed evidence, including uncertainty and constraints, not merely a ranked score.

The second boundary is *separation of proposer, validator, and approver roles.* The person or team that configures conditioning and generation should not be the sole authority approving escalation. This reduces incentive bias: proposers may be motivated to demonstrate model success and may unintentionally over-trust outputs. Validators (those who run simulations or assess feasibility) provide a different perspective. Approvers (those who commit resources and accept risk) should be accountable and independent. This separation mirrors best practices in high-accountability settings: the same person does not propose a trade, validate the model, and approve the risk limit. Discovery pipelines should adopt the same discipline.

The third boundary is *human ownership of constraint changes.* The system may suggest that relaxing a constraint would produce higher predicted scores. It may be tempting to adjust constraints dynamically. But constraint changes are governance decisions because they change the allowable risk surface. Therefore, constraints should be changed only through a controlled process with documented rationale and approvals. The model should not silently adapt constraints. If constraints drift silently, governance collapses.

The fourth boundary is *human ownership of model updates.* Retraining predictors or updating generators with new data can improve performance, but it also introduces drift and can create non-reproducibility. Therefore, updates should be approved and released through controlled processes. Humans must decide when to update, what data is included, and what validation tests must be passed. The pipeline can assist by reporting drift metrics, but it should not self-update in ways that change decision behavior without oversight.

The fifth boundary is *explicit escalation thresholds.* Humans should not be asked to review thousands of candidates. Oversight must be applied at choke points. Therefore, the pipeline should produce small, reviewable sets that meet minimum evidence thresholds. Humans review candidates above a threshold rather than reviewing everything. The oversight boundary is: the system may reduce the universe, but humans decide what leaves the computational world and enters the physical world.

The sixth boundary is *no silent substitution of evidence.* A subtle oversight failure occurs when the pipeline changes how evidence is computed (new predictor version, new feasibility heuristic) without making that change visible to reviewers. Humans cannot provide meaningful oversight if the evidence basis shifts silently. Therefore, evidence packets must include version identifiers and must highlight changes across runs.

Properly defined oversight boundaries preserve human accountability while still allowing the system to accelerate early exploration. The goal is not to slow discovery; it is to ensure that acceleration does not bypass responsibility.

1.9.4 Monitoring and auditability

Monitoring is the operational counterpart to governance design. Even well-designed controls degrade over time through drift, workarounds, and changing conditions. Monitoring ensures that the organization remains aware of how the pipeline behaves, and auditability ensures that when something goes wrong, the organization can reconstruct decisions and demonstrate control.

A core monitoring requirement is *generation volume tracking*. The organization should track how many candidates are generated per unit time, per project, and per user role. Spikes in volume are risk signals because they increase the probability of encountering proxy-exploiting outliers and because they can overwhelm review capacity. Volume tracking also supports misuse detection: unusual generation patterns may indicate unauthorized exploration.

A second monitoring requirement is *funnel pass rates*. For each stage—validity, hazard screening, feasibility screening, uncertainty thresholds, selection for escalation—track pass rates and their trends. Sudden changes in pass rates can indicate drift, constraint misconfiguration, or changes in conditioning behavior. For example, if hazard flags suddenly drop to near zero while novelty increases, this could indicate that hazard screening is being bypassed or that its inputs have changed.

A third monitoring requirement is *outlier pattern detection*. Outliers include candidates with extreme predicted scores, candidates with extreme novelty, candidates with high disagreement across predictors, and candidates that repeatedly appear across runs. Outlier monitoring supports early detection of proxy hacking. If the top-scoring candidates are also the most uncertain and most novel, the selection policy is likely vulnerable.

A fourth monitoring requirement is *predicted-versus-validated gap tracking*. This is a central governance metric. The organization should maintain dashboards that compare predictions to validated outcomes and that track calibration over time. Widening gaps should trigger interventions: tighten uncertainty gates, reduce sampling volume, retrain predictors, or restrict exploration. The purpose is to prevent the organization from spending months validating proxy-exploiting candidates before realizing the problem.

A fifth monitoring requirement is *drift monitoring*. Drift can occur in data distributions, in model behavior, and in user behavior. Drift monitoring includes: distributional distance between generated candidates and training data, changes in predictor calibration, changes in constraint failure patterns, and changes in the types of objectives users request. Drift matters because it changes reliability. A pipeline that is safe in one regime may become unsafe in another.

Auditability requires *immutable audit trails*. For any candidate escalated beyond early screening, the organization must be able to reconstruct: conditioning inputs, model versions, sampling parameters, filter decisions, selection rationale, and approvals. Audit trails must be complete and tamper-resistant. They must also be usable: auditors should not need to reconstruct evidence from scattered logs. A structured artifact bundle per run is a practical pattern.

Auditability also requires *decision reconstruction*. When a candidate fails in validation, the organization should be able to answer: why did we believe this candidate was promising, what evidence did we rely on, and what controls were in effect? When a candidate succeeds, the organization should be able to reproduce the process to confirm that success is not a one-off. Decision reconstruction is the difference between learning and repeating mistakes.

Finally, monitoring and auditability must be tied to *control actions*. Dashboards without interventions are theater. A governed pipeline defines what happens when metrics cross thresholds: who is notified, what gates tighten, whether exploration pauses, whether model updates are frozen, and whether incidents are reviewed. Monitoring is part of control, not merely observation.

1.9.5 Deployment deferral criteria

A governance-first posture must include explicit criteria for when deployment should be restricted or deferred. In physical-world domains, the cost of premature deployment is high, and the reputational damage from uncontrolled failures can exceed the cost of delay. Deferral criteria are therefore not pessimism; they are maturity gating.

The first deferral criterion is: *if safety filters cannot be defined and enforced, do not deploy beyond constrained pilots*. If the organization cannot articulate hazard exclusion classes, cannot implement them as enforceable gates, or cannot demonstrate that they are not bypassed, then any deployment that can reach physical action is irresponsible. The correct response is to restrict scope: limit generation to known-safe classes, limit conditioning objectives, limit exports, and require heightened human review. If even restricted scope cannot be defended, defer.

The second criterion is: *if uncertainty cannot be calibrated well enough to gate escalation, restrict exploration to in-distribution regimes*. Without calibrated uncertainty, the pipeline cannot distinguish reliable predictions from speculative ones. In that case, the pipeline should not be used for aggressive novelty or for high-stakes selection. It may still be used as an ideation tool within known families, but it should not be used to justify escalation of novel candidates. If the organization intends novelty-based discovery but lacks uncertainty control, deployment should be deferred until calibration and OOD detection are credible.

The third criterion is: *if the predicted-versus-validated gap is unstable or widening, pause optimization and reduce sampling pressure*. A widening gap is evidence that the pipeline is optimizing predictor error. Continuing in that regime wastes validation budget and can create false confidence. The correct response is to tighten controls, allocate validation to calibration monitoring, and potentially retrain predictors. If the organization cannot stabilize the gap, deployment should remain in research mode rather than operational mode.

The fourth criterion is: *if traceability is incomplete, prohibit escalation*. If the system cannot produce immutable logs and evidence packets that support audit, it should not be allowed to influence

physical action. This is a hard governance rule. Without traceability, the institution cannot defend its decisions or learn systematically. In regulated or high-accountability environments, incomplete traceability is itself a deployment blocker.

The fifth criterion is: *if role separation and approval gates cannot be enforced, do not deploy as a decision-support system.* If the same team can generate candidates and directly send them to the lab without independent review, the system is effectively autonomous. If the institution cannot enforce segregation of duties due to size or process limitations, it must compensate by restricting scope and increasing oversight. If it cannot, defer.

The sixth criterion is: *if provenance and IP constraints cannot be managed, restrict use to internal research and do not commercialize outputs.* Provenance ambiguity may not block internal exploration, but it can block commercialization. If the institution cannot document training data lineage and cannot assess output novelty relative to protected spaces, it risks IP disputes. This is a governance deferral criterion for downstream deployment, not necessarily for research.

Deferral criteria should be documented as part of the program charter, not invented during crisis. They should also be tied to clear remediation paths: what must be built, measured, or proven to lift restrictions. This makes governance constructive rather than merely prohibitive.

Risk & Control Notes

Governance minimums for physical-world generative pipelines.

- Explicit constraint registry (feasibility, hazard, provenance) with named owners.
- Calibrated uncertainty thresholds that gate escalation to higher-cost validation stages.
- Immutable logging of conditioning inputs, generated candidates, filter outcomes, and selection rationales.
- Role separation: proposal, validation, approval, and experiment execution are independently accountable.
- Documented deferral rules when controls cannot be defended under audit.

1.10 Outlook and Open Questions

1.10.1 Near-term research questions

The near-term research agenda for generative chemistry and materials design is not primarily about making generators more expressive. Expressivity is already sufficient to produce an abundance of plausible candidates. The pressing frontier is to make the entire pipeline more *reliably constrained*: to generate candidates that satisfy explicit requirements without collapsing into proxy exploitation, to quantify uncertainty in ways that survive optimization pressure, and to integrate lab feedback without destabilizing the system. These are not glamorous research questions compared to new architectures, but they are the ones that determine whether generative discovery becomes a deployable capability rather than a perpetual demo.

The first near-term question is how to build *uncertainty estimation that remains calibrated in the regimes discovery actually visits*. Calibration in i.i.d. test settings is not enough. The pipeline will select candidates precisely because they are unusual and promising. That selection shifts the distribution and concentrates error. The challenge is therefore to produce uncertainty measures that remain informative under adaptive sampling and repeated selection. Practical directions include ensemble methods, Bayesian approximations, and hybrid uncertainty signals that combine model-based uncertainty with data-coverage indicators. The deeper research challenge is to understand how to calibrate uncertainty when the evaluation distribution is not a static population but a moving target shaped by the pipeline's own policy.

The second near-term question is *constraint-aware generation that is stable and auditable*. Many systems enforce constraints after the fact, generating candidates and then filtering. This wastes compute and can produce brittle behavior when constraints tighten. A more robust approach is to incorporate constraints into the generation process itself: to shape sampling so that prohibited classes are avoided and feasibility constraints are respected. However, constraint-aware generation is fragile because constraints can interact in non-linear ways and because enforcing them inside the generator can reduce diversity and create mode collapse. The near-term research challenge is to design methods that satisfy constraints while maintaining controlled exploration, and to make those methods auditable so that institutions can defend what is being enforced.

A third question is how to *stabilize lab-feedback integration*. In principle, closing the loop between generation and experimental validation is the source of compounding advantage: models can learn what works and refine their proposal distribution. In practice, feedback loops can destabilize both predictors and generators because the data is small, biased, and noisy, and because updates can move the system in unpredictable ways. Near-term work must therefore focus on update discipline: when to update, how to prevent catastrophic drift, how to preserve reproducibility, and how to incorporate new evidence without erasing previous knowledge. The problem is as much about process control as it is about machine learning.

A fourth question is *adversarial robustness under optimization*. Proxy hacking is not a rare accident; it is the expected result of optimizing imperfect predictors over large candidate sets. Near-term research should treat this as an adversarial problem: can we design predictors and selection policies that are robust to being optimized? This includes stress-testing predictors with deliberately optimized candidates, building adversarial training regimes, and designing selection policies that penalize extreme outliers unless supported by multiple independent evidence sources. The goal is not to eliminate proxy exploitation entirely, but to make it harder and to detect it earlier.

A fifth question is *representation-level alignment with physical constraints*. Especially in materials contexts, the representation may omit critical aspects of reality: processing conditions, microstructure, defects, and kinetics. Near-term work should focus on integrating these factors into representations and conditioning schemes, or at least on defining how coarse representations can be safely used at early stages without being misinterpreted as definitive. The practical goal is a staged representation strategy: coarse, fast representations for ideation and triage; richer, physics-aligned representations for escalation.

Collectively, these near-term research questions aim at one outcome: reducing the gap between "generative validity" and "deployable credibility." The frontier is not whether we can generate. It is whether we can generate under governance constraints and still produce validated progress.

1.10.2 Technical unknowns

Even if near-term research improves uncertainty, constraint enforcement, and feedback loops, there remain deep technical unknowns that shape the long-run ceiling of generative discovery. These unknowns matter for executive strategy because they determine what timelines and what risk profiles are realistic.

The first unknown is *how generative models generalize to truly novel chemical and materials families*. Models trained on existing data learn the structure of known manifolds. Discovery often seeks candidates that sit at the edges of those manifolds or outside them. There is no general guarantee that the model's learned representation supports meaningful extrapolation. In practice, models may generate plausible-looking candidates that are structurally inconsistent with deeper physical constraints, or they may simply fail to propose candidates that represent genuinely new mechanisms. The unknown is not whether models can produce novelty, but whether the novelty is the right kind: novelty that aligns with feasible chemistry or materials science rather than novelty that exploits representational artifacts.

The second unknown is *predictor robustness under optimization pressure*. Predictors can appear accurate on standard test sets and still fail catastrophically when optimized against. This is not simply an engineering bug; it reflects a deeper issue: predictive models are trained to minimize average error over a distribution, not to resist being gamed by adversarial search. In generative discovery, the pipeline itself is the adversary. It searches for high scores. The unknown is how far

we can push predictor robustness without requiring prohibitive amounts of validated data. There may be regimes where robust prediction is unattainable with available data, forcing institutions to rely more heavily on conservative gating and expensive simulations.

A third unknown is *the reliability of feasibility and synthesizability proxies*. Feasibility depends on context, and context changes. A model that predicts synthetic accessibility in one environment may be inaccurate in another. Retrosynthesis planners can provide route suggestions, but route suggestions are not route guarantees. The unknown is whether we can develop feasibility proxies with sufficiently low false negatives to act as reliable gates, or whether feasibility will remain an expert-driven bottleneck. This matters because feasibility failures are among the most expensive: they can consume substantial time before a candidate is even testable.

A fourth unknown is *error accumulation and compounding across pipeline stages*. Each stage introduces approximations: generator approximations, predictor approximations, screening approximations, selection approximations. The pipeline's overall reliability depends on how these errors interact. In some cases, errors may cancel; in others, they may compound. A system might, for example, systematically favor candidates that are both high-scoring and difficult to synthesize, because the feasibility proxy is weak and the scoring proxy is strong. The unknown is how to characterize and control these interactions at scale. This is an area where system-level evaluation may be more important than model-level improvement.

A fifth unknown is *the stability of iterative improvement*. Many narratives assume that closed-loop discovery naturally improves over time: more data, better models, better candidates. In practice, iterative loops can plateau or destabilize. Small datasets, selection bias, and shifting objectives can produce cycles where the system chases proxy artifacts rather than accumulating robust knowledge. The unknown is whether iterative loops can be made reliably monotonic in validated progress, or whether the best we can do is to manage them as stochastic processes with occasional gains.

These technical unknowns imply that organizations should be cautious about committing to aggressive roadmaps that assume smooth scaling. The technology is powerful, but the system-level reliability is still an open engineering and scientific challenge. A governance-first organization treats these unknowns as reasons to stage-gate deployment and to invest in validation infrastructure rather than betting everything on model improvements.

1.10.3 Governance unknowns

Governance in generative discovery is not mature. The technical systems are advancing faster than the institutional frameworks that constrain them. This creates governance unknowns that matter for adoption: what standards will be expected, what liabilities will attach, and what evidence practices will become normal.

The first governance unknown is *standards for provenance documentation*. Organizations will

increasingly be asked: what data trained the model, what rights exist over that data, and what is the lineage of outputs? Today, practices vary widely. Some programs treat provenance as informal; others attempt structured registries. A major unknown is whether the field will converge on standard provenance artifacts: dataset datasheets, model cards tailored to scientific domains, and output lineage reports that connect candidates back to training sources in defensible ways. Without such standards, institutions face uncertainty in IP risk and in partner trust.

The second unknown is *standards for validation documentation*. In regulated or safety-sensitive contexts, it is not enough to say "we validated the candidate." The question becomes: validated how, under what protocol, with what controls, and with what reproducibility? The industry lacks uniform norms for documenting the progression from generative proposal to validated evidence. A likely trajectory is the emergence of standardized evidence packets: run manifests, candidate records, screening results, simulation outputs, lab protocols, and approval signatures. The unknown is how quickly these standards will emerge and whether they will be imposed externally (by regulators or partners) or adopted internally (as best practice).

The third unknown is *liability and accountability frameworks*. When AI influences physical experimentation, liability becomes layered. If a hazardous candidate is generated and mishandled, was the harm due to the model, the pipeline, the user, or the institution's controls? In many industries, liability frameworks will evolve through precedent rather than proactive design. The governance unknown is what level of control and documentation will be considered "reasonable." This matters because institutions may need to demonstrate not only that they had controls, but that they followed industry-standard controls. The standard is not yet defined.

The fourth unknown is *audit norms for scientific AI*. Finance has mature audit practices for models; physical-world AI discovery does not. A governance-first program must decide what to log, how to store it, how long to retain it, and how to support incident reconstruction. The unknown is whether there will be convergence on audit expectations analogous to model risk management in finance. If so, early adopters who build robust audit infrastructure will have an advantage. If not, institutions may underinvest until a failure forces a shift.

The fifth unknown is *controls for dual-use risk*. Some institutions may face expectations to demonstrate that their systems cannot be easily repurposed for harmful exploration. What constitutes an adequate control set? Access control alone may not suffice. Monitoring, constraint enforcement, and restriction of conditioning objectives may be expected. The unknown is what the baseline will be, and how institutions will balance openness (for scientific collaboration) against restriction (for safety). This tension will shape industry norms.

Governance unknowns therefore suggest a strategic posture: build controls and documentation that would stand up under scrutiny even if standards tighten. In other words, treat governance not as meeting today's minimum expectations, but as building defensible practices that will remain acceptable as expectations evolve.

1.10.4 What would change the assessment

The assessment of generative discovery would change meaningfully if several technical and institutional developments occur. These developments are not mere incremental improvements; they would alter the risk-reward profile and would justify broader deployment.

The first development would be *reliable feasibility predictors with demonstrated low false negatives*. If organizations could trust that feasibility screens rarely pass infeasible candidates, the validation funnel would become far more efficient. This would reduce wasted lab cycles and would make the pipeline's outputs more credible. Importantly, feasibility predictors would need to be contextual: aware of organizational capabilities and constraints. Feasibility that is true in principle but false in practice would not be sufficient.

The second development would be *high-quality safety screens with low false negatives and defensible scope*. Safety screening is a deployment blocker when it is weak. If safety screens become reliable enough to define enforceable hazard exclusion classes with acceptable operational burden, institutions could expand the exploration regime. This would make the opportunity larger because it would allow the pipeline to explore more broadly without creating unacceptable safety exposure. Such screens would likely combine rule-based exclusions, learned hazard models, and conservative escalation policies.

The third development would be *validated benchmarks with wet-lab outcomes*. The field currently relies heavily on offline benchmarks. If standardized benchmarks existed that included end-to-end pipelines and measured success in terms of validated outcomes under realistic constraints, organizations could compare approaches more meaningfully. Such benchmarks would not eliminate the need for institution-specific evaluation, but they would provide a stronger baseline than current proxy-heavy metrics. Benchmarks with real validation outcomes would also pressure the field toward more honest measurement.

The fourth development would be *robust uncertainty calibration under adaptive selection*. If uncertainty estimates became reliably informative even under optimization pressure and distribution shift, governance gates could be tightened or loosened with more confidence. This would reduce proxy exploitation and would make iterative loops safer. It would also enable more principled active learning, improving the efficiency of validation spending.

The fifth development would be *standardized governance artifacts and audit norms*. If the industry converged on standard run manifests, provenance registries, candidate evidence packets, and change management practices, institutions could implement governance more efficiently and could defend their practices more credibly. This would reduce friction in partnerships, procurement, and regulatory interactions. It would also reduce the variance in quality across deployments, which is currently high.

If these developments occur, the assessment shifts from "powerful but governance-limited" to "scalable

under disciplined controls." Until then, the opportunity remains real but bounded: institutions can use generative tools as proposal engines within constrained regimes, but they must invest heavily in validation and governance to avoid expensive failure modes.

1.10.5 Link to subsequent papers

This paper's core lesson is that when AI outputs touch the physical world, governance must move from being a policy layer to being a system architecture layer. The subsequent paper in this book shifts from chemistry and materials to physics and simulation surrogates. The transition is natural: both domains share a common pattern of risk. In chemistry and materials, the system proposes candidates that must be validated. In physics surrogates, the system replaces expensive simulation with fast approximations that can be rolled out over time. In both cases, naive evaluation fails, error can accumulate silently, and governance depends on end-to-end validation, uncertainty control, and rollback mechanisms.

The bridge is therefore a general template for physical-world AI deployment: treat models as components of pipelines; define constraints as enforceable gates; treat uncertainty as a control variable; evaluate end-to-end, not by isolated benchmarks; and maintain auditability and reproducibility as non-negotiable. The next chapter will apply this template to surrogate modeling, where the central failure mode is not the generation of hazardous candidates but the silent compounding of approximation error when surrogates are rolled forward in time. The surface domain changes, but the governance logic remains: capability without disciplined evaluability is not progress; it is unmanaged risk.

Bibliography

[1] E. Hoogeboom, V. G. Satorras, C. Vignac, and M. Welling. Equivariant diffusion for molecule generation in 3D. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*, PMLR **162**, 8867–8887, 2022.

[2] M. Xu, L. Yu, Y. Song, C. Shi, S. Ermon, and J. Tang. GeoDiff: A geometric diffusion model for molecular conformation generation. In *International Conference on Learning Representations (ICLR)*, 2022.

[3] R. Gómez-Bombarelli, J. N. Wei, D. Duvenaud, J. M. Hernández-Lobato, B. Sánchez-Lengeling, D. Sheberla, J. Aguilera-Iparraguirre, T. D. Hirzel, R. P. Adams, and A. Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS Central Science*, **4**(2), 268–276, 2018.

[4] W. Jin, R. Barzilay, and T. Jaakkola. Junction tree variational autoencoder for molecular graph generation. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, PMLR **80**, 2323–2332, 2018.

[5] N. De Cao and T. Kipf. MolGAN: An implicit generative model for small molecular graphs. *arXiv preprint* arXiv:1805.11973, 2018.

[6] N. Brown, M. Fiscato, M. H. S. Segler, and A. C. Vaucher. GuacaMol: Benchmarking models for de novo molecular design. *Journal of Chemical Information and Modeling*, **59**(2), 644–651, 2019.

[7] D. Polykovskiy, A. Zhebrak, B. Sanchez-Lengeling, S. Golovanov, O. Tatanov, S. Belyaev, A. Kurbanov, A. Artamonov, V. Aladinskiy, M. Veselov, and A. Aspuru-Guzik. Molecular Sets (MOSES): A benchmarking platform for molecular generation models. *Frontiers in Pharmacology*, **11**, 565644, 2020.

[8] P. Ertl and A. Schuffenhauer. Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. *Journal of Cheminformatics*, **1**, Article 8, 2009.

[9] M. H. S. Segler, M. Preuss, and M. P. Waller. Planning chemical syntheses with deep neural networks and symbolic AI. *Nature*, **555**, 604–610, 2018.

[10] F. Urbina, F. Lentzos, C. Invernizzi, and S. Ekins. Dual use of artificial-intelligence-powered drug discovery. *Nature Machine Intelligence*, **4**(3), 189–191, 2022.

Chapter 2

Chapter 7 — Physics Surrogates and Simulation Under Constraint

Abstract. Physics and engineering organizations increasingly treat simulation as the core engine of design, validation, and iteration. AI surrogates promise to compress that engine: replacing expensive solver calls with rapid learned approximations for PDEs, CFD, N-body dynamics, and detector pipelines. The opportunity is strategic rather than cosmetic. When inference becomes cheap, teams can explore larger design spaces, run richer scenario analyses, and shift from slow batch iteration to interactive experimentation. Yet the same speed introduces a new class of institutional risk: surrogate outputs are easy to operationalize at scale, even when their validity is fragile, their uncertainty is uncalibrated, or their behavior under rollout is unstable. The central governance problem is therefore not whether surrogates can be accurate on held-out test cases, but whether they remain decision-grade under distribution shift, regime boundaries, and long-horizon compounding error.

This chapter frames surrogate modeling as operator learning: approximating a mapping from initial and boundary conditions and parameters to evolving solution fields. It explains why naive metrics such as mean squared error are insufficient, why conservation and stability violations can be catastrophic while appearing visually plausible, and why the organization must define explicit operating envelopes with enforceable fallback to trusted solvers or experimental checks. The chapter also details implementation patterns that preserve accountability—hybrid surrogate-plus-solver workflows, regime-sliced validation, uncertainty thresholds, and monitoring for drift—so speed does not outrun auditability. The companion Colab notebook demonstrates these ideas using synthetic spatiotemporal data, emphasizing rollout evaluation, constraint checks, and governance artifacts that make “fast physics” defensible in practice.

2.1 Orientation and Scope

2.1.1 Motivation and context

Physics and engineering organizations already run on simulation. Whether the artifact is a new airframe, a battery chemistry, a satellite thermal stack, a fusion-relevant plasma experiment, or a detector pipeline in a high-energy physics lab, the core operational pattern is the same: propose a design, simulate its behavior under parameterized conditions, interpret the results, and iterate. The expensive part of that loop is the solver call—often not because a single evaluation is exotic, but because the institutional workflow requires thousands to millions of them across parameter sweeps, uncertainty studies, design-of-experiments grids, and optimization cycles. This is why the idea of an AI surrogate lands so forcefully with executives: it promises to compress the time constant of the organization. If a surrogate can produce high-fidelity approximations at a fraction of the cost of computational fluid dynamics (CFD), finite element analysis (FEA), N-body simulation, Monte Carlo transport, or detector emulation, then what was previously a quarterly design cadence can become weekly; what was previously a limited exploration of alternatives can become a broad search over a design space; what was previously a conservative engineering choice can become an innovation bet that is still plausibly testable.

But the same property that makes surrogates valuable—cheap inference—creates their most dangerous failure mode: the organization can start treating the surrogate output as if it were the solver output. Simulation has always been a model of reality, but it is a model with an unusually strong accountability chain: it is defined by equations, discretizations, and numerical stability properties that are at least in principle inspectable and bounded by decades of verification and validation (V&V) practice. By contrast, the surrogate is a learned approximation whose validity is a function of training coverage, inductive bias, and calibration discipline. The executive risk is not that the surrogate is wrong; every model is wrong. The executive risk is that the surrogate can be wrong in ways that are hard to see, hard to diagnose, and easy to operationalize at scale. A solver is slow and therefore naturally rate-limited; a surrogate is fast and therefore naturally multiplicative. When a surrogate is used to accelerate the design loop, error is no longer an isolated technical defect but a propagation mechanism: wrong predictions inform wrong downstream decisions, which in turn shape the next round of designs, experiments, and capital allocation. In that sense, surrogates are not just models; they are organizational accelerants, and accelerants amplify both competence and negligence.

This is why a governance-first framing is not a moral preference but an operational necessity. In mature simulation cultures—especially those influenced by aerospace, energy, and defense practice—simulation results are rarely accepted as a single number. They are accepted as an evidentiary object: a set of assumptions, discretization choices, boundary conditions, convergence tests, sensitivity checks, and comparative validations against known baselines. A surrogate challenges this evidentiary posture because it invites a different habit: to accept a prediction because it is

available. That is not a technical problem; it is a process problem. The institution must decide what constitutes admissible surrogate evidence, what uncertainty must be demonstrated, what failure modes are unacceptable, and how the surrogate is constrained to operate inside an explicitly defined envelope. Without those decisions, the surrogate does not merely speed up simulation; it speeds up the creation of plausible-looking errors.

The motivating context therefore has two layers. The first is the business case: expensive solvers are bottlenecks in design, manufacturing, and science. The second is the governance case: the bottleneck moves from compute to trust. Once inference is cheap, the scarce resource is the institution's ability to certify that surrogate outputs are safe to act upon. That certification is not a single approval event. It is an ongoing discipline of calibration, monitoring, envelope enforcement, and escalation. If these controls are absent, then the surrogate becomes a liability generator: it creates a trail of decisions that may be difficult to defend after a failure, because the institution cannot reconstruct why it trusted the output or whether the output was produced inside a validated regime.

This chapter takes that tension seriously. It treats AI surrogates as a frontier opportunity precisely because they can transform the economics of iteration, but it also treats them as a frontier risk precisely because they can erode the institutional habit of evidentiary discipline. The goal is not to argue for or against surrogates. The goal is to define what it means to adopt them responsibly: to harvest speed without losing auditability, and to gain coverage without surrendering to false confidence.

2.1.2 Why this topic is at a frontier moment

The reason this topic is arriving now, as a distinct frontier rather than a niche technique, is that the representational tools of modern AI have begun to match the structure of physical simulation tasks. For years, most machine learning surrogates were narrow function approximators. They were trained for a specific geometry, a specific parameterization, a specific operating regime, and a specific output variable. They worked when the problem was tightly scoped and the institution could guarantee that the surrogate would be used only within the narrow conditions it had seen. That still matters, and those classical surrogates will remain useful. The frontier shift is that the field is now building models that attempt to learn operators: mappings from initial conditions, boundary conditions, and parameters to entire solution fields over time. This is a conceptual escalation. Learning an operator is not learning a scalar response surface; it is learning a family of dynamics. It aligns more naturally with the object that engineers care about: a field solution, not a single number.

At the same time, foundation-model ideas have migrated from text and images into the world of spatiotemporal fields. The phrase "foundation surrogate" is not a marketing flourish; it refers to an ambition: to create reusable architectures that can be conditioned on problem descriptors and generalize across families of PDEs, geometries, or parameter regimes. In practice, the current state is uneven. Many claims of broad generality collapse under careful testing, and the gap

between demonstrations and deployable systems remains large. Nevertheless, the trajectory is clear enough to create executive pressure. When leaders see surrogates produce skillful medium-range weather forecasts, emulate expensive CFD-like dynamics, or generate high-resolution spatiotemporal predictions at speeds that would have been implausible even a few years ago, they correctly infer that the economics of simulation may change. They also correctly infer that their competitors will attempt to exploit that change. The frontier moment is not merely technical; it is strategic.

This strategic pressure is amplified by the structure of modern product and research pipelines. In many sectors, the number of design alternatives is no longer constrained by human creativity but by the cost of evaluation. If evaluation becomes cheaper, the organization can explore more of the design space, and exploration itself becomes a differentiator. In materials, one sees this in candidate screening; in engineering, in topology optimization and rapid iteration; in climate and weather, in ensemble forecasting and scenario analysis; in high-energy physics, in detector simulation and reconstruction. Surrogates can act as the evaluation engine in these loops, turning what was previously a small-batch process into a large-batch process. That is not a minor acceleration; it changes what problems are feasible to attempt.

Yet the same shift creates a governance cliff. When the organization moves from "a solver run is expensive, so we treat it as a significant evidentiary event" to "a surrogate run is cheap, so we can run it everywhere," it changes the epistemic culture. Decisions become easier to justify with numbers because numbers are abundant. The frontier moment is therefore defined by a paradox: the organization becomes more capable of exploring possibilities while becoming less capable of knowing which of those possibilities are real. The bottleneck migrates from compute to evaluation and validation. This is exactly the pattern already seen in other frontier chapters of this book: acceleration produces a new dependency on reliable evaluation, because the cost of being wrong scales with speed.

Finally, there is a second frontier pressure: surrogates are beginning to be integrated into control loops, not just offline design. When a surrogate approximates a dynamical system in real time, it can be used to propose actions, stabilize systems, or optimize performance on the fly. This can occur in robotics, process control, energy systems, and adaptive experimentation. Once the surrogate is in the loop, errors are not just wrong answers; they are wrong actions. This elevates governance requirements. Institutions must specify when the surrogate is allowed to act, what thresholds trigger a fallback controller, and how uncertainty is used as a control signal. In other words, the frontier moment is not that surrogates exist; it is that they are being promoted from analysis tools to decision components.

2.1.3 What has changed recently

Several technical developments have converged to make this frontier credible. The most visible is the rise of neural operator methods—architectures explicitly designed to learn mappings between

function spaces rather than between finite-dimensional vectors. These methods attempt to represent solution operators in a way that is less tied to a particular discretization and more tied to the underlying physics. The importance of this shift is not that it magically solves generalization; it is that it creates a language for generalization that matches the problem. When the object is an operator, the natural question becomes: over what family of conditions is the learned operator reliable? That question is closer to how simulation teams already think: they care about envelopes, regimes, and boundary cases. Operator learning gives them a technical handle to connect model performance with those regime descriptions.

A second change is scalable spatiotemporal modeling. The community has adapted transformers, convolutional backbones, and diffusion-like generative mechanisms to fields evolving over time. In domains like weather forecasting, the results have been sufficiently strong to matter, not because they eliminate numerical methods but because they compete with them on speed and, in some settings, on skill. This matters for the AI 2026 thesis because it demonstrates that learned surrogates can become institutional infrastructure. When a model can produce a global forecast quickly, it becomes a platform capability. Platform capabilities invite widespread reuse, and widespread reuse is precisely where governance must be explicit. A model used by one research group is a local risk; a model used across a value chain is a systemic risk.

A third change is conditioning and parameterization. Earlier surrogates often failed because they were brittle: they could not gracefully incorporate varying boundary conditions, forcing terms, geometries, or control parameters. Newer systems increasingly represent these conditions explicitly and train models to respond to them. This does not guarantee robustness, but it changes the design space. The model is no longer merely interpolating within a fixed simulation environment; it is learning to modulate its outputs based on the stated conditions. That increases flexibility and, in practice, increases temptation: organizations begin to use the model under new conditions because the interface suggests it can handle them. This is a governance hazard disguised as a capability. The model may accept a boundary condition input even when it has never seen that regime; the user may interpret acceptance as competence. Conditioning therefore must be paired with explicit regime tracking: it is not enough that the model can be conditioned; the institution must know which condition-space regions are validated.

A fourth change is the maturation of hybrid patterns: surrogate plus solver correction, residual-based refinement, and active learning loops. In these patterns, the surrogate is not treated as a replacement but as a proposal generator. The solver remains the authority when correctness matters, while the surrogate accelerates exploration and reduces the number of expensive solves needed. This is a governance-aligned pattern because it preserves an audit trail: the surrogate suggests; the solver verifies; the institution signs off on verified results. It also naturally produces the data needed to monitor surrogate performance over time, because the surrogate predictions can be compared against solver outputs in the regimes where decisions are being made. The hybrid approach is not always feasible, especially when the solver is too slow or unavailable in production, but when it is

feasible it is often the only defensible path for high-stakes use.

A fifth change is that uncertainty quantification and out-of-distribution detection are increasingly treated as first-class concerns, even if the state of the art remains imperfect. The scientific machine learning community has recognized that raw predictive accuracy is not enough for deployment. Calibration, confidence estimation, and OOD alerts are part of the operational envelope definition. This matters because it aligns with executive accountability: leaders do not need a model that is always right; they need a model that knows when it might be wrong and that triggers a safe process when uncertainty is high. The shift from "predict well" to "predict and know when not to be trusted" is foundational for governed adoption. It is also technically difficult in high-dimensional field settings, which is why this remains frontier: the most important requirements are the hardest ones.

Taken together, these changes mean that AI surrogates are no longer isolated research curiosities. They are becoming system components: trained on large simulated datasets, integrated into design loops, and sometimes embedded into real-time systems. The organization's challenge is therefore not merely technical adoption but institutional adaptation: defining envelopes, validating rollouts, monitoring drift, and making sure that speed does not outrun accountability.

2.1.4 Explicit exclusions and non-goals

This chapter is not a numerical analysis textbook. It will not attempt to teach finite difference schemes, finite volume methods, FEM formulations, Riemann solvers, turbulence modeling, or the deep mathematical theory of PDE well-posedness. Those topics are essential, and they remain the foundation for trustworthy simulation, but they are not the audience's bottleneck here. The bottleneck for senior practitioners is deciding how to use AI surrogates without producing an accountability gap. That is a governance and evaluation question that sits above the technical details of any particular solver. The chapter assumes the reader recognizes that solvers exist, that they embody trade-offs, and that their outputs are used as evidence. The chapter's job is to show what changes when the evidence is learned rather than computed.

This chapter is also not a safety certification manual. While it will discuss safety risks and the kinds of controls that safety-minded institutions should apply, it will not provide a regulatory compliance recipe. Safety certification is domain-specific and often governed by standards, regulatory frameworks, and institutional practices that differ across aerospace, automotive, energy, medical devices, and scientific labs. What the chapter will do is articulate the control points that are common across domains: operating envelopes, uncertainty thresholds, fallback procedures, traceability, and audit logs. These controls are necessary but not sufficient. Any claim that a surrogate can be "certified" in general without domain-specific validation is itself a governance failure.

A third exclusion is the claim that AI replaces experiments. In physical science and engineering, experiments are not merely validation; they are the ultimate anchor of truth. Surrogates can

accelerate hypothesis generation, candidate screening, and scenario analysis, but they do not abolish the need to test reality. Indeed, a well-governed surrogate program often increases the importance of experimentation because it enables the organization to explore more hypotheses, which must then be pruned by empirical checks. The non-goal is therefore not to argue that experiments are obsolete, but to clarify how surrogates can be used to allocate experimental budgets more intelligently without pretending that experimental truth can be inferred from model confidence.

A fourth exclusion is an implicit one: this chapter does not treat "foundation surrogates" as inherently superior. The chapter does not assume that larger models or more general architectures are automatically safer or more valuable. Many organizations will find that narrow surrogates with well-defined envelopes are preferable to general surrogates with unclear validity. The chapter's stance is intentionally conservative: adoption is justified when the institution can define what the surrogate is for, where it is valid, how it fails, and what happens when it is uncertain. If those questions cannot be answered, the surrogate may still be used for exploratory research, but not as decision-grade evidence.

Finally, the chapter does not treat speed as the primary metric of success. Speed is the temptation, but the governance-first criterion is stability under rollout and reliability under regime variation. A surrogate that is fast but unstable is not an asset; it is a mechanism for manufacturing plausible errors. The non-goal is therefore to chase speedups for their own sake. The goal is to define when speedups translate into strategic advantage because they are paired with controls that preserve institutional trust.

2.1.5 Role of this chapter in AI 2026

This chapter occupies a deliberate position in the structure of *AI 2026: Frontier Awareness Without the Hype*. It follows Chapter 6, which examined generative models in chemistry and materials design, and it precedes Chapter 8, which examines frontier AI in finance under constraint. That placement is not accidental. Chapter 6 introduced a core pattern: when AI can generate candidates cheaply, the bottleneck becomes validation—especially when optimizing proxies that do not fully capture real-world feasibility. Physics surrogates generalize that pattern beyond chemistry. They show that even when the output is a physically interpretable field, the validation problem remains, and in some respects becomes harder because errors can compound over time and across coupled variables. The chapter therefore continues the book's central thesis: the frontier is governed capability, and governed capability depends on evaluation that matches the system's temporal and structural complexity.

The bridge to Chapter 8 is equally important. Finance is not physics, but decision-making under uncertainty has a shared structure. In finance, models are judged not only by average accuracy but by tail behavior, stability under regime change, and the ability to explain and audit decisions. Surrogates in physics have analogous requirements: average field error is not enough; one must track

worst-case failures, regime boundaries, and out-of-distribution use. The organizational pattern also repeats: once inference is cheap, the institution risks producing an abundance of outputs without an abundance of understanding. In finance this becomes model risk management; in physics it becomes surrogate risk management. The difference is that physics often has a solver or experiment available as an oracle for spot checks, while finance often does not. That contrast is pedagogically valuable: it clarifies what governance can borrow from scientific validation practices and where finance faces unique constraints.

Within the broader arc of the book, Chapter 7 also reinforces a recurring control principle: *one-step accuracy is not enough.* In Chapter 1, evaluating an agent's output at a single step is insufficient; one must evaluate trajectories. In Chapter 2, deeper reasoning introduces delayed failures; one must evaluate the reasoning process and its failure modes. In Chapter 4, more context introduces provenance and relevance failures; one must evaluate what evidence the system used. In Chapter 5, planning and search amplify weak objectives; one must evaluate the plan's constraints and stop conditions. Chapter 7 shows the scientific analog: a surrogate may be accurate for one step or one snapshot, but when rolled out, its errors can compound into physically meaningless trajectories. The governance implication is direct: evaluation must be aligned to the way the model will be used, especially when the model is iterated over time.

Finally, this chapter sets up Chapter 10's system-of-systems perspective. Multimodal vision–language–action systems are governed at the system level because their failure modes arise from integration and feedback. Physics surrogates anticipate that governance challenge: the surrogate is rarely used alone. It is embedded in a pipeline—an optimizer, a controller, a design loop, a human review process. The failure modes often arise not from a single prediction but from repeated use, implicit trust, and pipeline coupling. Chapter 7 therefore helps train the reader to think in terms of operational envelopes, monitoring, and fallback, which are precisely the concepts needed when AI systems graduate from producing information to influencing actions.

In short, Chapter 7 is the book's case study in "fast physics" as a governed capability. It shows how frontier AI can compress the time constants of scientific and engineering organizations, and it shows why that compression makes validation and uncertainty the true bottleneck. The promise is real, but the constraint is not optional: speed without evaluability is not progress; it is an institutional way of getting lost faster.

2.2 Conceptual Abstraction

2.2.1 Core abstraction

The conceptual heart of surrogate simulation is not "a model that predicts a number." It is a model that stands in for a procedure. In classical simulation, an organization encodes its physical assumptions into a solver: a set of governing equations, boundary conditions, discretization rules, numerical stability constraints, and convergence criteria. When a surrogate is introduced, the organization is not merely learning an output mapping; it is learning an approximation to the solver's *operator*. That distinction matters because an operator is defined over families of inputs and returns structured objects—fields evolving over space and time—rather than scalar outputs. In the language of scientific machine learning, the surrogate attempts to learn a map of the form

$$\mathcal{G} : (u_0, \theta, \mathrm{BC}) \mapsto u(\cdot, t),$$

where u_0 is an initial condition (often itself a field), θ denotes parameters (material properties, Reynolds number, forcing amplitudes, geometry descriptors, control inputs), and BC denotes boundary and forcing conditions. The output $u(\cdot, t)$ is not a single value; it is the solution field over a domain, potentially for multiple times $t \in [0, T]$. Even when a surrogate is used to produce derived observables—lift coefficients, stress maxima, heat flux, detector hit distributions—the underlying conceptual task is field prediction or operator approximation. This is why "foundation surrogates" are even imaginable: if one can learn operators rather than bespoke response surfaces, then architectures might become reusable across families of PDEs or geometries, at least within some controlled envelope.

The governance relevance follows directly: once the surrogate is framed as an operator, the key question becomes *over what domain of inputs is the learned operator reliable enough to support decisions*? This is not a philosophical question; it is the operational definition of the surrogate's admissibility. A decision-grade surrogate must be valid for the regimes where the institution will use it, which requires an explicit description of those regimes. In practice, the input space is enormous. Initial conditions are high-dimensional, boundary conditions can vary, and parameter changes can induce regime transitions (laminar to turbulent, subsonic to transonic, stable to unstable plasma modes). Therefore the chapter's conceptual move is to treat "the surrogate" not as a monolithic predictor but as a *governed operator with an operating envelope*. The operating envelope is the institution's explicit statement of what the surrogate is allowed to do, where it is expected to behave well, and what triggers escalation when it may not.

This framing also clarifies why a surrogate is not simply "faster simulation." Simulation is evidence production; the surrogate is evidence approximation. If the organization uses surrogate outputs to choose designs, allocate experiments, tune controllers, or justify claims to regulators or customers, then the surrogate is participating in decision-making. The core abstraction therefore includes not

only the learned operator $\mathcal{G}_\phi$ (parameterized by weights ϕ) but also the decision context in which its outputs are interpreted. A surrogate with small average error may still be unacceptable if it fails at boundary regimes relevant to safety margins. Conversely, a surrogate with moderate average error may be valuable if it is well-calibrated and is used only for exploratory narrowing of candidate sets, with solver verification for final decisions. The operator view forces the institution to specify *use* alongside *accuracy*.

There is a second conceptual layer: most surrogates are not used once; they are used iteratively. They are rolled out over time (predicting u_{t+1} from u_t), embedded in optimizers (which query the surrogate repeatedly under varying parameters), or used in active learning loops (which decide what new simulations to run). In those settings, the surrogate becomes part of a dynamical system. The abstraction must therefore include the feedback loop: the surrogate does not merely produce outputs; it shapes the sequence of future inputs. That is why evaluation must be trajectory-aligned and why governance must be pipeline-aligned. If the surrogate's outputs steer the next decisions, then its errors can steer the organization into unvalidated regions. The operator abstraction makes this visible: the surrogate approximates $\mathcal{G}$, but the institution's workflow composes $\mathcal{G}_\phi$ with itself and with decision rules. Composition is where fragility often appears.

Finally, the operator abstraction clarifies the legitimate ambition and the legitimate limit of the technology. The ambition is to learn families of operators that generalize beyond narrow training conditions, enabling reuse and speed. The limit is that physics is not a dataset. Physical systems have regime boundaries, discontinuities, and constraints that are unforgiving. Therefore the "core abstraction" of this chapter is best stated as follows: *a surrogate is a learned operator deployed inside a governed envelope, intended to produce decision-sufficient evidence under explicit constraints, with escalation when conditions are out-of-envelope.* Everything else in the chapter—mathematical framing, evaluation methodology, implementation patterns, and governance controls—follows from this abstraction.

2.2.2 Key entities and interactions

Once we adopt the operator-and-envelope view, the key entities in a surrogate-enabled simulation organization become legible. The first entity is the *target system*: the physical process or simulated phenomenon of interest, described by governing equations and domain specifications. The second entity is the *authoritative evaluator*: typically a trusted numerical solver and, when available, experiments or high-fidelity measurements. The solver is expensive but conceptually authoritative because it is anchored in explicit numerical methods and has established V&V practices. The third entity is the *surrogate model*: an approximator trained on solver outputs (and possibly augmented with observational data) to provide fast predictions within some regime. The fourth entity is the *decision process*: the organizational pipeline that consumes simulation outputs to make choices—design selection, parameter tuning, safety margin estimation, scheduling of experiments, or

operational control.

Crucially, governance introduces additional entities that are often missing from purely technical descriptions. One is the *operating envelope artifact*: a documented, versioned statement of the surrogate's valid input regimes, known limitations, required preprocessing, and evaluation criteria. Another is the *calibration artifact*: evidence that uncertainty estimates are meaningful and that predictive confidence is aligned with observed error rates (at least within regimes). A third is the *fallback policy*: the explicit rule that determines when the surrogate is insufficient and the workflow must revert to solver evaluation or human escalation. A fourth is the *monitoring and audit log*: the record of where the surrogate was used, under what conditions, with what confidence, and what downstream decisions it influenced. These governance entities are not "paperwork." They are the mechanism by which the institution preserves defensibility when speed increases and outputs proliferate.

The interaction pattern across these entities can be described as a loop with several modes. The canonical mode is *propose* $\rightarrow$ *evaluate* $\rightarrow$ *decide* $\rightarrow$ *refine*. In a classical setting, evaluate means solver-run. In a surrogate setting, evaluate can mean surrogate-run, solver-run, or hybrid. The loop begins with a proposal: a design candidate, a boundary condition scenario, a parameter set, or an initial condition sample. The proposal is fed to the surrogate (or solver), producing a predicted field $\hat{u}$ and derived observables $\hat{y}$. The decision process then applies rules: rank candidates, check constraints, compute margins, or generate recommendations. Refinement occurs as the decision process proposes the next batch of candidates, often driven by optimization heuristics, Bayesian search, or human intuition.

Where the human sits in this loop is the central governance question. There are at least three plausible configurations. In a *human-in-the-loop* configuration, the surrogate provides exploratory evidence and humans decide what to trust, often with solver confirmation for final decisions. In a *human-on-the-loop* configuration, the surrogate and optimizer run largely autonomously, but humans supervise via monitoring dashboards and intervene when alerts fire. In a *human-out-of-the-loop* configuration, the surrogate runs continuously and decisions are automatically enacted (typical in some control applications). This chapter treats the third configuration as high-stakes and generally inappropriate without strong controls, because it couples surrogate error directly to action. For most organizational contexts, the governed pattern is either human-in-the-loop with solver verification, or human-on-the-loop with conservative fallback.

The hybrid interaction deserves special attention because it is often the only defensible pattern for early adoption. In a hybrid pattern, the surrogate is used to generate candidate fields and observables quickly, and the solver is used selectively to verify the most consequential predictions or to correct errors in regions where uncertainty is high. This can be implemented as residual-based correction, where the surrogate's prediction is used as an initial guess for the solver, reducing solve time, or as selective sampling, where solver runs are allocated to regions of the input space where the surrogate is uncertain or where decision impact is high. The important point is that the solver

is not merely a training-data generator; it is a governance backstop. The organization keeps a foot in the world of explicit numerical authority while enjoying the speed of learned inference.

Another interaction pattern is active learning. Here the surrogate is not static. It informs the choice of new simulation points to run, with the explicit goal of improving coverage and reducing uncertainty in regions that matter. Active learning is attractive because it aligns cost with value: the organization spends solver budget where it improves decision confidence. But it also introduces a new governance hazard: the surrogate's current beliefs shape the data collection process, which can create blind spots if the selection strategy is naive. Governance must therefore include not only "what the surrogate predicts" but "how the surrogate decides what to learn next." In other words, the data pipeline becomes part of the control surface.

A final interaction to highlight is downstream coupling. Surrogate outputs rarely end at a report. They feed into CAD revisions, manufacturing constraints, procurement decisions, safety cases, and sometimes regulatory submissions. This means that errors are not merely scientific; they are contractual, financial, and reputational. Therefore the governance entities—envelope, calibration, fallback, logs—must be integrated with the organization's decision documentation. If a design is approved based partly on surrogate evidence, the approval record must capture whether the evidence was within envelope, whether solver verification occurred, and what uncertainty thresholds were met. This is what makes surrogate adoption institutionally legible rather than a black-box acceleration that no one can later explain.

2.2.3 What is being optimized or controlled

In technical papers, surrogate modeling is often described as optimizing predictive accuracy: minimize an error norm between predicted and true fields. That is a correct mathematical statement but an incomplete control statement for deployment. In a governed institution, what is being optimized is not simply prediction; it is *decision-sufficient reliability.* The goal is to make the surrogate safe to use for a defined purpose, under defined conditions, with defined escalation behavior. This implies at least four coupled objectives: accuracy, stability, calibration, and detectability of invalid use.

Accuracy is the obvious objective: reduce the discrepancy between $\hat{u}$ and u, or between derived observables $\hat{y}$ and y. But even within accuracy, the choice of metric must reflect the decision context. A uniform L^2 field error may be a poor proxy for safety margins if rare spikes or boundary-layer errors matter disproportionately. In high-energy physics detector simulation, a small bias in a tail distribution can distort downstream inference even if average error is low. Therefore the accuracy objective must be weighted toward the quantities that drive decisions: constraint satisfaction, extreme values, integral invariants, or regime transition indicators. Conceptually, accuracy is not "how close is the field everywhere"; it is "how close is the evidence where it matters."

Stability is the second objective and is often more important than raw accuracy. Stability refers to how errors behave under rollout and composition. If the surrogate is used iteratively, then a small

one-step error can amplify into a qualitatively wrong trajectory. Many institutions discover this too late because standard offline test sets measure one-step or one-shot error. But deployment composes the model repeatedly. Stability must therefore be controlled explicitly. This can be done by training on multi-step losses, enforcing constraints that prevent drift, using implicit time-stepping structures, or designing hybrid correction loops. However it is not enough to implement stability techniques; the organization must declare stability criteria that reflect use. If the surrogate is used for 100-step rollouts in a control setting, stability criteria must cover that horizon. If it is used for single-step forecasting, the criteria differ. Stability is therefore a governance contract: "we will use the model this way; we require stability to this extent."

Calibration is the third objective and is the most directly governance-relevant. Calibration means that the surrogate's stated confidence corresponds to observed error rates. In a decision-making organization, uncertainty is a control signal. It determines when to trust an output, when to fall back to a solver, and when to escalate to human review. A surrogate that is accurate but overconfident is dangerous because it suppresses escalation. A surrogate that is underconfident may be safe but economically useless because it triggers fallback too often. Calibration is therefore not a cosmetic add-on; it is the interface between the model and the institution's controls. When calibration is reliable, governance can be algorithmic: thresholds can be set, and the system can enforce discipline. When calibration is unreliable, governance must be manual and conservative, which limits the value of the surrogate.

Detectability of invalid use is the fourth objective and is often neglected. Surrogates fail most dangerously when used out of distribution: new geometries, new parameter regimes, unmodeled physics, or unanticipated boundary conditions. A governed system must detect when inputs are out-of-envelope and must not silently produce outputs that appear authoritative. This detectability can be implemented via OOD detectors, embedding distance measures, constraint checks, or ensemble disagreement. But conceptually, the objective is to minimize the rate of *silent failures*: cases where the surrogate is wrong and the system does not realize it. In high-accountability contexts, the institution may prefer to accept more false alarms (triggering solver fallback) rather than tolerate silent failures. This is a policy choice: the false-negative rate for invalid-use detection should be governed like a safety parameter.

Taken together, these objectives imply that the surrogate's optimization problem is multi-criteria and institution-specific. The organization is not optimizing "best benchmark score." It is optimizing an operational trade-off curve: speed versus reliability, coverage versus conservatism, automation versus auditability. This is why the chapter emphasizes decision-sufficient reliability. A surrogate that is slightly less accurate but well-calibrated and stable may be more valuable than a surrogate that is highly accurate on average but brittle at regime boundaries. The institution is optimizing the *quality of evidence* under constraints.

It is also important to recognize that the optimization is not only inside the model; it is inside the workflow. When the organization introduces a surrogate, it must decide what fraction of evaluations

to verify with the solver, how to allocate solver budget for monitoring, how to update the surrogate over time, and how to version and document changes. These are control decisions. They shape the effective reliability of the system more than the marginal improvement of the model architecture. In other words, the governed system's performance is a function of the surrogate $\mathcal{G}_\phi$ and the policy π that decides when to trust it:

$$\text{Outcome} = \text{DecisionPipeline}(\mathcal{G}_\phi, \pi, \text{Envelope}, \text{Fallback}, \text{Logs}).$$

This expression is not meant as formal mathematics but as a reminder: institutions deploy systems, not models. The optimization target is the system's defensible performance, not the model's benchmark number.

2.2.4 Distinction from prior paradigms

To govern surrogates intelligently, leaders must understand what is genuinely new and what is a rebranding of older ideas. Several paradigms precede modern surrogates, and each provides useful intuitions and misleading analogies.

The first prior paradigm is reduced-order modeling (ROM). ROMs aim to compress the state space of a physical system into a low-dimensional representation—often via proper orthogonal decomposition (POD), Galerkin projection, or related techniques—so that dynamics can be simulated cheaply in the reduced space. ROMs are attractive because they preserve a clear link to the governing equations and can sometimes provide interpretable modes and error bounds. However ROMs can be fragile under regime transitions, and their construction often depends on careful selection of basis functions and training trajectories. The governance-relevant lesson from ROMs is that *compression creates envelope dependence.* A ROM is valid where its basis is valid. This directly anticipates the surrogate envelope concept. The misleading analogy is to assume that because ROMs are equation-derived, they are automatically safer. ROMs can also fail catastrophically when used outside their regime; they just fail in a way that is more traceable to the basis choice.

The second paradigm is classical statistical surrogate modeling: Gaussian processes, polynomial chaos expansions, kriging, response surfaces, and related emulators. These methods typically target scalar or low-dimensional outputs and provide uncertainty estimates. They are often used in design-of-experiments workflows and uncertainty quantification. Their governance lesson is the value of explicit uncertainty and the discipline of specifying priors and kernels. Their limitation is scalability: high-dimensional fields and complex geometries strain these methods, and their inductive biases may be too rigid for many modern simulation tasks. The misleading analogy is to assume that uncertainty is "solved" because a method outputs a variance. Uncertainty is only useful if it is calibrated and decision-relevant, and high-dimensional field settings make calibration harder.

The third paradigm is classical deep learning surrogates: convolutional networks trained to map

parameter vectors to field outputs on fixed grids, or recurrent networks trained for short rollouts. These surrogates can be fast and accurate in narrow settings, but they often depend heavily on fixed discretizations and struggle with generalization across geometries or regimes. Their governance lesson is the danger of silent overfitting: a model that performs well on a test set constructed from the same distribution may still fail under operational shift. Their misleading analogy is to treat "deep learning generalization" as a single property. In physics, generalization is regime-dependent, and the relevant shifts are often structured (boundary conditions, geometry, forcing), not i.i.d. noise.

What distinguishes the current frontier is the ambition to learn operators and to scale architectures to handle spatiotemporal fields and conditioning. Neural operator methods, transformer-like spatiotemporal models, and diffusion-based generative field models attempt to represent families of dynamics more flexibly than earlier surrogates. They also encourage the idea of reuse: train once, apply broadly. This is both the promise and the governance hazard. The more reusable a surrogate is claimed to be, the more urgently the institution needs envelope definition and monitoring. Narrow surrogates can be governed by local knowledge; broad surrogates require explicit institutional controls because no single team can anticipate all uses.

Another distinction is that modern "foundation" approaches often rely on massive simulated datasets and large-scale training. This changes the risk profile in two ways. First, it increases the temptation to treat the trained model as infrastructure, because sunk cost and impressive training effort create organizational commitment. Second, it makes provenance and versioning more complex: datasets are large, preprocessing pipelines are intricate, and training procedures include many hyperparameters. Governance must therefore treat the surrogate like any other critical system component: version it, document it, and define change control. Older paradigms often lived in research teams; foundation surrogates aspire to live in production pipelines.

Finally, the transparency trade-off becomes sharper. ROMs and many classical emulators have interpretable structures. Modern deep surrogates can be harder to explain, especially when they produce plausible-looking fields. The chapter's governance stance is not that interpretability is mandatory, but that *auditability is mandatory*. If the model is not interpretable, the institution must compensate with stronger empirical validation, logging, and fallback mechanisms. In other words, the distinction from prior paradigms is not merely architectural; it is institutional. Modern surrogates demand system-level governance because they are fast, reusable, and easily embedded into decision loops.

2.2.5 Conceptual failure modes

A governance-first abstraction is only useful if it names the ways things go wrong. The most dangerous surrogate failures are not obvious crashes. They are failures that produce outputs that look reasonable, satisfy superficial checks, and are therefore adopted into the organization's decision stream. Several conceptual failure modes recur across domains.

The first is distribution shift, or more precisely *regime shift*. In physical systems, shifts are often structured and abrupt: the onset of turbulence, the formation of shocks, a phase transition, a change in boundary layer separation, or the appearance of instabilities in coupled dynamics. A surrogate trained mostly in one regime may interpolate smoothly across parameter changes, producing fields that look plausible but are physically inconsistent with the true regime. This is why regime-sliced evaluation is a governance requirement: performance must be measured across parameter slices and near regime boundaries, not just in aggregate.

The second failure mode is conservation or constraint violation. Physical systems are constrained by conservation laws, positivity constraints, and stability conditions. A surrogate can violate these constraints while still minimizing average error. Worse, it can violate them in localized ways that are visually subtle but operationally catastrophic: a small mass conservation drift that accumulates over rollout, a negative density artifact, an energy injection that destabilizes a control loop, or a detector simulation that violates known symmetries. Constraint violations are particularly dangerous because they can be masked by low mean error. Therefore governance requires explicit constraint metrics and mandatory checks: conservation residuals, stability indicators, and invariance tests. If the institution cannot cheaply compute these checks, it cannot cheaply trust the surrogate.

The third failure mode is rollout compounding. In many deployments, the surrogate is used iteratively: predict next state, feed it back, predict again. Even if one-step error is small, the compounding can cause divergence. This is not a corner case; it is a structural property of dynamical systems approximation. The error at time t becomes part of the input at time $t+1$. If the surrogate's errors have systematic bias or if its dynamics are slightly mis-specified, the rollout can drift into regions the model never saw during training, accelerating failure. This is why one-step benchmarks are governance traps. The institution must evaluate long-horizon behavior aligned to intended use and must include stop conditions or re-anchoring mechanisms (e.g., periodic solver correction) when drift is detected.

The fourth failure mode is false confidence. A surrogate may output high-confidence predictions even when it is out of envelope. This can happen because the model is deterministic, because its uncertainty estimation is uncalibrated, or because the OOD detector is weak. False confidence is arguably the most dangerous failure mode because it disables governance controls that rely on uncertainty thresholds. In a governed system, uncertainty is not just an informative statistic; it is a gating mechanism. If uncertainty is untrustworthy, then gating becomes either too permissive (leading to silent failures) or too conservative (destroying the surrogate's value). Therefore calibration and OOD detection are not optional enhancements; they are the control plane.

The fifth failure mode is "smooth but wrong." This deserves explicit naming because it repeatedly misleads human reviewers. Learned models, especially those trained with mean-squared losses and smooth architectures, tend to produce smooth outputs. Smoothness is aesthetically convincing. It can also be physically wrong, particularly in systems where the true solution has sharp gradients, discontinuities, turbulence-like structures, or rare localized phenomena. In CFD, a surrogate may

smear shocks. In materials, it may smooth microstructure features that determine failure. In detectors, it may smooth tail events that determine discovery significance. Humans are vulnerable to this because smoothness reads as stability. Governance must therefore resist visual plausibility as evidence. Visual inspection can be a sanity check, but it cannot be the acceptance criterion.

A sixth failure mode is scope creep, which is organizational rather than mathematical. Because surrogates are easy to run, teams will naturally expand their use: new geometries, new boundary conditions, new optimization objectives, new downstream pipelines. Each expansion is a new distribution shift. Without explicit envelope enforcement, scope creep is inevitable. The surrogate becomes a general-purpose tool by default, and governance becomes reactive rather than preventive. Scope creep is not solved by telling teams to "be careful." It is solved by implementing envelope checks that are enforced by the tooling and by requiring solver verification when operating outside validated zones.

A seventh failure mode is feedback-induced bias. When a surrogate is used in an optimization or active learning loop, it shapes what the organization explores. If the surrogate has systematic biases, the optimizer may exploit them, producing designs that look good under the surrogate but fail under the solver or in reality. This is a scientific analog of the broader phenomenon seen in AI planning and search: optimization amplifies objective weaknesses. Here the "objective weakness" is not only a mis-specified goal but a mis-specified evaluator. If the evaluator is approximate, the optimizer will find the cracks. Governance must therefore treat surrogate-driven optimization as adversarial by default: any system that searches a design space against an approximate evaluator must include verification gates and should expect exploitation of surrogate errors.

These failure modes are not arguments against surrogates. They are arguments for explicit control. A surrogate can be a strategic advantage when it is treated as a governed approximation: used within envelope, monitored for drift, calibrated for uncertainty, and backed by solver or experimental verification when stakes are high. The conceptual abstraction of this chapter is designed to make that governance posture unavoidable. If leaders treat surrogates as "fast physics," they will adopt them as replacements. If leaders treat surrogates as "fast approximations with explicit envelopes and escalation," they can adopt them as accelerators without surrendering accountability. The difference is not rhetorical; it is the difference between capability and governed capability.

2.3 Historical and Technical Lineage

2.3.1 Preceding approaches

The modern enthusiasm for AI surrogates can look, from a distance, like a sudden revolution: neural networks appear, solvers disappear, and "fast physics" arrives on schedule to rescue every engineering timeline. The actual lineage is less theatrical and more instructive. Surrogates are not a new desire. They are the latest entry in a long tradition of attempts to reduce the cost of evaluating physical models while preserving enough fidelity to support decisions. Understanding that lineage matters because it reveals what has repeatedly worked, what has repeatedly failed, and what the current wave truly changes. It also prevents an institutional mistake that appears in every technology cycle: treating a new tool as if it abolishes old constraints, when in fact it changes where those constraints bind.

The oldest and still dominant preceding approach is straightforward: numerical solvers. Finite difference, finite volume, finite element, spectral methods, Monte Carlo transport, and N-body integrators are not merely technical instruments; they are institutional objects. They have validation cultures, standards, documentation practices, and established meaning inside organizations. A solver run is interpreted as evidence because it is anchored in equations and numerical analysis. It may be wrong because the model is wrong, the boundary conditions are wrong, or the discretization is inadequate, but its failure modes are at least conceptually attributable to explicit choices. Solvers also impose healthy friction: they are expensive enough that teams treat them with respect. That friction is a hidden governance feature. It forces prioritization, encourages sensitivity studies, and limits the proliferation of speculative results. In many organizations, the solver's slowness is not only a cost; it is a barrier against undisciplined overuse.

Alongside full solvers, reduced-order modeling (ROM) emerged as a pragmatic compromise. Rather than solving the full high-dimensional system, ROM compresses the dynamics into a low-dimensional subspace. Techniques like proper orthogonal decomposition (POD), balanced truncation, Galerkin projection, and various modal decompositions create a basis in which the dominant modes of variation are represented. ROMs excel when dynamics are well-captured by a small number of coherent structures and when the operating regime is stable. They were—and remain—powerful in control settings where real-time computation is necessary. Their limitations are equally instructive: ROMs can become unstable under regime shifts, their bases may fail when the system explores new regions, and their error behavior can be hard to bound without careful analysis. Yet ROMs contributed two governance-relevant ideas that survive into modern surrogates. First, *validity is regime-bound*: a model is only as good as the subspace it was built from. Second, *stability is a first-class property*: a model that is accurate but unstable is unusable in rollout.

A third set of preceding approaches comes from classical surrogate modeling and uncertainty quantification. Gaussian process regression, kriging, polynomial response surfaces, radial basis

function interpolants, polynomial chaos expansions, and related emulators were developed to approximate expensive simulations with uncertainty estimates. These methods typically target low-dimensional outputs or carefully engineered features and are often deployed in design-of-experiments workflows. Their strength is methodological discipline: they force teams to think about sampling strategies, priors, smoothness assumptions, and predictive uncertainty. Their weakness is scalability and representational capacity for high-dimensional fields, complex geometries, and strongly nonlinear dynamics. But institutionally, these methods set the template for what "responsible approximation" looks like: approximate, quantify uncertainty, and validate where it matters. That template is precisely what many organizations forgot when deep learning surrogates arrived, because deep learning made approximation easy while making uncertainty less explicit.

Finally, there were classical machine learning surrogates prior to the current operator-learning wave. Early neural network emulators, kernel methods, and spatiotemporal convolutional architectures were used to map parameters to outputs or to predict short-term dynamics. They demonstrated that learned models could accelerate simulation-like tasks, but they were typically bespoke: tied to fixed discretizations, fixed geometries, narrow regimes, and limited horizons. They worked well when used as local accelerators and poorly when treated as general replacements. Their key contribution was operational proof that learning-based surrogates can reduce costs, and their key governance lesson was negative: when the surrogate is treated as a plug-and-play substitute for a solver, failure is often silent, and the organization loses the ability to attribute error. These preceding approaches, taken together, form the foundation of today's frontier. They also expose the persistent tension: approximation is valuable only when its limits are known and enforced.

2.3.2 Key inflection points

The current wave did not begin with "AI" in general; it began with a set of concrete inflection points that changed what surrogates could plausibly represent. The first inflection point was the move from scalar emulation to field emulation at scale. This shift was enabled by advances in deep learning for high-dimensional structured data: convolutional networks for grids, graph networks for meshes, attention mechanisms for flexible conditioning, and training practices that allowed models to learn from large corpora of simulation outputs. Once networks could represent complex spatial patterns, the surrogate problem stopped being "predict a few outputs" and became "predict a full state."

The second inflection point was the emergence of physics-informed approaches, particularly physics-informed neural networks (PINNs) and related constraint-aware learning. The conceptual move of PINNs is to incorporate the governing equations into training by penalizing residuals of the PDE, enforcing boundary conditions, or combining data losses with physics losses. The practical performance of PINNs varies widely by domain and setup, and they are not a universal solution. Nevertheless, the inflection point matters because it changed the conversation: it made constraints

explicit as part of learning rather than as an afterthought. It also introduced a key governance idea: the surrogate should be judged not only by fit to data but by adherence to physical structure. In institutional terms, PINNs pushed teams to ask, "What constraints must never be violated?" even if the implementation details remain domain-specific.

The third and arguably most important inflection point was operator learning. Neural operator methods—such as Fourier neural operators, DeepONets, and related architectures—reframed the surrogate task as learning a mapping between function spaces. This matters because it aligns with the true object of simulation: an operator from conditions to solutions. Operator learning also suggests a path to discretization robustness: if the model represents the operator in a way that is not strictly tied to a particular grid, it may generalize across resolutions or meshes more gracefully. That ambition is not fully realized in all settings, but it provides the conceptual architecture for "reusable surrogates." More importantly for governance, operator learning gives language for defining envelopes: the operator is valid over a family of conditions, and that family can be characterized and tested.

The fourth inflection point was the adaptation of spatiotemporal deep learning to dynamical systems at scale. Weather forecasting is the most visible example: learning models that predict global atmospheric fields over time. The significance is not only scientific. It is institutional. Weather forecasting demonstrates that learned models can become part of operational infrastructure, not just research prototypes. Once a surrogate becomes infrastructure, the stakes of its failure change. It is reused, integrated, and trusted by downstream systems. This forces a governance posture that many engineering teams are not accustomed to applying to machine learning systems: version control, change management, monitoring, and incident response.

The fifth inflection point was the migration of foundation-model patterns—transformers and diffusion models—into field modeling. Transformers offer flexible conditioning and the ability to represent long-range dependencies; diffusion models offer a generative mechanism that can capture complex distributions over fields. These approaches opened new possibilities: generative surrogates that produce ensembles, conditional generation under varied boundary conditions, and representations that can handle heterogeneity. They also introduced new governance hazards: generative models can produce plausible samples that are not physically valid, and their uncertainty is often epistemic in name but not calibrated in operation. The inflection point is therefore double-edged. It expands capability while intensifying the need for validation discipline.

A final inflection point was the rise of hybrid workflows as a deliberate design choice rather than a compromise. Historically, surrogates were often used either as replacements or as cheap approximations with little structure. Increasingly, practitioners have recognized that the safest and most valuable deployments use surrogates to accelerate exploration while reserving solvers for verification. In these hybrid workflows, the surrogate is not merely an endpoint; it is a proposal mechanism embedded in a governed pipeline. This reframing is an inflection point because it shifts emphasis from "model performance" to "system performance." It also aligns with executive

accountability: the institution can claim speed gains while still producing solver-verified evidence for high-impact decisions. Hybrid workflows are not glamorous, but they are often the difference between deployable value and impressive demonstrations that cannot be trusted.

2.3.3 What persisted vs what broke

When new techniques arrive, organizations often assume that the old constraints are obsolete. In simulation, that assumption is particularly dangerous. Many things persisted, and those persistent features are precisely what make governance non-negotiable.

The most important persistence is that physics imposes constraints that do not yield to clever representation. Conservation laws, stability requirements, boundary condition consistency, and regime-dependent behaviors remain. A surrogate may approximate them, but the underlying system does not forgive violations. Therefore, while learning has improved the ability to approximate complex fields, it has not changed the fact that physically invalid predictions can be catastrophic. This is why constraint checks and residual metrics remain central, even if the surrogate is learned. The solver's culture of demanding physical plausibility persists as a requirement; it cannot be replaced by benchmark scores.

A second persistence is the dominance of stability and error propagation. Numerical analysis has long taught that average accuracy is not enough; stability determines whether errors remain bounded under iteration. That lesson persists and becomes more acute in surrogate rollouts. In many surrogate deployments, the surrogate is iterated as a time-stepper or used repeatedly inside an optimizer. The relevant question is not "does it predict well on average?" but "do its errors remain bounded when composed?" This is a direct inheritance from numerical methods, and it remains the central technical determinant of whether a surrogate can move from offline evaluation to operational use.

A third persistence is the institutional need for verification and validation. Even when the surrogate is accurate, organizations must maintain a chain of evidence. Standards and practices like V&V in CFD exist because simulation outputs are used in safety-critical decisions. The introduction of surrogates does not remove that need; it changes what must be verified. Instead of verifying a solver's discretization and convergence behavior, the institution must verify the surrogate's operating envelope, calibration, and drift behavior. In other words, the V&V mindset persists, but the object of V&V expands from numerical methods to learned systems.

What broke, at least partially, is the assumption that every surrogate must be handcrafted per system. Historically, building a surrogate often meant building a local emulator: a specific mapping for a specific geometry and a specific output. Operator learning and foundation-like approaches challenge that assumption by suggesting that architectures and even trained weights can transfer across tasks. This is not a clean break; it is partial and regime-dependent. But it is enough to create a new organizational reality: teams can be tempted to reuse surrogates across projects, to

treat them as platforms, and to treat them as general-purpose evaluators. This is a profound change because it moves surrogate risk from local to systemic. When a surrogate is reused broadly, a failure mode can propagate across products or business units, creating correlated risk.

Another thing that broke is the old friction barrier. Solver cost provided natural governance through scarcity. Surrogates remove scarcity and therefore remove that accidental governance. Organizations must replace it with deliberate governance: gating, thresholds, envelope enforcement, and monitoring. This is why "fast physics" is not merely faster computation; it is an institutional transformation. Speed breaks the old assumption that results are scarce and therefore reviewed. With surrogates, results are abundant and must be filtered by policy.

A final partial break is the belief that uncertainty is naturally available. Classical emulators like Gaussian processes provided uncertainty estimates as part of the model. Deep learning surrogates often do not, or provide uncertainty that is difficult to interpret. As a result, institutions can lose uncertainty discipline exactly when they need it most. The emerging emphasis on calibration and ensemble methods is a response to this break, but it is not yet a universally reliable fix. The persistence is that decisions require uncertainty; the break is that many modern surrogates do not naturally provide it in a usable form. Governance must therefore force uncertainty back into the workflow as a first-class deliverable.

2.3.4 Why older intuitions fail

Even when leaders understand the lineage, they can still apply the wrong intuitions. Several older intuitions—imported from mainstream machine learning or from simplified views of simulation—fail in the surrogate setting, and the failures are predictable.

The first failing intuition is the standard ML idea that good test performance implies reliable generalization. In many ML domains, the test set is assumed to approximate the deployment distribution, and generalization is measured as the gap between training and test error. In PDE and physics settings, deployment shift is not a nuisance; it is the norm. The relevant shifts are structured, and they can be qualitatively transformative. A small change in a parameter can cause a shock to form; a slight geometry change can produce separation; a minor forcing change can trigger instability. These are not "outliers" in a statistical sense; they are regime boundaries. Therefore the intuition that a held-out test set is sufficient is wrong. The evaluation must be regime-aware and must focus on boundaries and tail behavior. This is why the chapter emphasizes envelope definition: the institution must specify where it intends to operate and evaluate specifically there.

The second failing intuition is that smooth interpolation is safe. Many classical surrogates, and many deep models, are excellent interpolators. In physical systems, interpolation across regimes can be physically invalid. The model may smoothly blend between two qualitatively different behaviors, producing a solution that is neither. This is especially dangerous when the output is a field, because a smooth field can look plausible. The human reviewer sees continuity and infers correctness. The

physics, however, may require discontinuity. Thus the intuition that "smooth predictions are stable predictions" fails. Governance must compensate with constraint checks and regime indicators that are not fooled by visual plausibility.

The third failing intuition is that constraints learned during training will automatically generalize. Physics-informed losses or conservation penalties can improve behavior within the training distribution, but they do not guarantee constraint satisfaction outside it. A model can learn to satisfy constraints in the regimes it has seen and still violate them in new regimes. Worse, the penalty may interact with the model's inductive bias in ways that produce brittle constraint satisfaction: the model satisfies the constraint by learning a shortcut that breaks when conditions change. Therefore, the intuition that adding a physics penalty "solves" physical validity is wrong. Constraints must be monitored during deployment, not merely encouraged during training.

The fourth failing intuition is that uncertainty estimates—if present—are trustworthy by default. In many organizational contexts, an uncertainty number is treated as a control signal without calibration evidence. In surrogate modeling, uncertainty is difficult precisely because outputs are high-dimensional, errors can be structured, and OOD regimes can be adversarially "confident." If the institution treats uncertainty as a truth indicator without calibration, it may create a false sense of safety. This is why calibration diagnostics must be part of the governance artifacts, and why uncertainty thresholds must be tuned to policy objectives (e.g., minimizing silent failures).

The fifth failing intuition is that faster evaluation simply improves optimization. In design loops, optimization is often limited by evaluation cost, so faster evaluation appears to imply better designs. But optimization against an approximate evaluator can be dangerous: the optimizer will exploit evaluator weaknesses. This is a well-known phenomenon in many settings—Goodhart-like effects, adversarial optimization, reward hacking—and it appears here in a scientific form. The surrogate becomes the objective landscape, and the optimizer finds the surrogate's blind spots. Therefore, the intuition that "the surrogate will help us find better designs faster" is only true if verification gates exist. Without verification, the surrogate may help the organization find designs that exploit model error rather than physics.

In summary, older intuitions fail because physical systems are regime-structured, because rollouts compound errors, and because decision loops amplify approximations. The governance implication is direct: evaluation must be aligned with operational use, and "average performance" is an insufficient proxy for reliability.

2.3.5 Inherited lessons

The purpose of reviewing lineage is not nostalgia; it is to inherit the lessons that remain true and to apply them to modern systems. Several lessons from numerical analysis, UQ, and safety practice are particularly important for governed surrogate adoption.

From numerical analysis, the central inherited lesson is that *stability dominates*. A method that is accurate but unstable is not deployable, because errors grow under iteration. This lesson maps directly onto surrogate rollouts. The institution must treat long-horizon behavior as a first-class metric, and it must test stability under the same conditions in which the surrogate will be used. This also means that the institution should prefer conservative rollout strategies—periodic re-anchoring to solver states, hybrid correction, and explicit stop conditions—when stability is uncertain. Numerical analysis also teaches that discretization matters. Even if an operator-learning surrogate claims discretization invariance, the institution must test across resolutions and mesh variations relevant to deployment. "Works on the training grid" is not a sufficient claim.

From classical uncertainty quantification, the key lesson is that *uncertainty is a deliverable, not an ornament*. In design-of-experiments culture, uncertainty drives sampling and decision-making. Surrogates should be evaluated not only on prediction error but on calibration quality: do confidence intervals behave as expected; do uncertainty estimates increase near boundaries; does the system flag OOD conditions. UQ practice also teaches that sampling should be purposeful. It is not enough to collect large datasets; one must ensure coverage of regimes that matter. This motivates active learning and regime-focused data generation rather than naive random sampling.

From safety practice, especially in domains where simulation supports certification, the inherited lesson is that *tail risk matters more than average performance*. Safety failures are rarely average-case. They occur at boundaries, under rare combinations of conditions, and under compounding effects. Therefore evaluation must explicitly target worst-case regimes, stress tests, and adversarial scenarios. Safety practice also emphasizes traceability: the ability to reconstruct what evidence supported a decision. In surrogate deployment, traceability requires logging the input conditions, model version, uncertainty outputs, and whether fallback verification occurred. Without this, an organization cannot credibly defend its decisions after an incident.

Another safety-derived lesson is the value of independent checks. In engineering practice, no single artifact is trusted alone; cross-checks and redundancy are built into workflows. Surrogate deployment should follow the same principle. Constraint checks provide one kind of independent validation; solver spot checks provide another; experimental checks provide the ultimate anchor. The institution should design its surrogate program so that no single model output can silently drive a high-impact decision without passing through at least one independent validation gate.

Finally, the lineage teaches a cultural lesson: *fast results are not the same as trustworthy results*. Solvers forced a culture of patience because compute was expensive. Surrogates will tempt organizations toward impatience because compute is cheap. Governance must therefore reintroduce deliberate friction where it matters: mandatory documentation of envelope use, required verification for high-impact decisions, and explicit escalation protocols. This friction is not bureaucracy; it is the mechanism that keeps speed from becoming recklessness.

In the context of AI 2026, these inherited lessons support a broader thesis: the frontier is not simply

new capability, but new capability that remains governable. Physics surrogates represent a frontier precisely because they challenge institutions to preserve old disciplines—stability, uncertainty, validation, and traceability—under conditions of unprecedented speed and scale. The technical lineage shows that these disciplines were learned the hard way. The governance challenge now is to keep them intact while adopting the new tools that threaten to make them feel optional.

2.4 Technical Foundations

2.4.1 System or architectural components

A governed surrogate system is not a single neural network placed next to a solver. It is an engineered stack that takes structured physical conditions as input, produces structured predictions as output, and exposes enough intermediate artifacts to support validation, monitoring, and escalation. The technical foundations therefore begin with a decomposition: what are the components that must exist for a surrogate to be useful *and* governable?

The first component is the *problem specification layer*: the representation of the physical scenario to be evaluated. In simulation, this specification includes at minimum (i) a geometry or domain description, (ii) initial conditions u_0, (iii) boundary conditions and forcing terms BC, and (iv) parameter settings θ (material properties, dimensionless numbers, control parameters). In practice, this layer also includes discretization choices (grid resolution, mesh topology, time step), because learned models can be sensitive to these even when they claim invariance. A disciplined surrogate system treats the problem specification as a first-class object: it is versioned, validated for consistency (units, ranges, completeness), and logged.

The second component is the *encoding layer*, which maps the problem specification into a representation suitable for a model backbone. This is deceptively important. Many surrogate failures originate not in the backbone but in the encoding, because the encoding defines what the model can condition on and how it "sees" boundary conditions and geometry. Encoding can take several forms:

- **Grid-based encodings** represent fields on regular lattices. These are common in atmospheric modeling, many CFD surrogates with structured meshes, and PDE benchmarks. They allow convolutional and Fourier-based operators.
- **Mesh-based encodings** represent fields on unstructured meshes typical in complex geometries. These often require graph neural networks, mesh attention, or coordinate-based networks. The challenge is preserving invariances and handling varying mesh topology.
- **Coordinate-based encodings** represent the solution as a function $u(x, t)$ approximated by networks that take coordinates and conditions as input. This can support continuous representations and transfer across discretizations, but can be harder to train and can hide numerical instability behind smoothness.
- **Hybrid encodings** combine a low-dimensional parameter vector θ with high-dimensional field descriptors, often using separate encoders and fusion layers. This is common when a few parameters control a family of dynamics but the initial state is complex.

For governance, the encoding layer must also enforce basic input validity: range checks, unit normalization, and explicit flags when conditions are outside known training distributions. A model that accepts an input is not necessarily competent on it; encoding must therefore include

envelope-aware validation.

The third component is the *model backbone*, which can be broadly grouped into operator networks, transformer-like spatiotemporal models, diffusion/generative field models, and classical spatiotemporal CNNs. Each class has different strengths and governance implications.

Operator networks (e.g., neural operators) are designed to approximate mappings between function spaces. They often include mechanisms like Fourier transforms or learned kernel integrations. Their promise is that they can represent operator structure more naturally and can generalize across resolutions in some settings. For governance, operator backbones can make it easier to define what "input function space" is being represented, but they do not eliminate the need for explicit envelope definition.

Transformers and attention-based models can represent long-range dependencies and can condition flexibly on parameters, boundary tokens, or geometry descriptors. Their strength is flexible context integration; their weakness is that they can appear to generalize while relying on subtle correlations that break under shift. For governance, transformer-like surrogates often need stronger OOD detection and more careful monitoring because their internal representations can be hard to interpret.

Diffusion and generative models for fields are relevant when the goal is not a single prediction but a distribution: ensembles of plausible trajectories, stochastic dynamics, or uncertainty-aware sampling. Their strength is expressive distribution modeling; their weakness is that they can produce physically implausible samples that look realistic. Governance therefore requires hard constraint checks or projection operators on generated samples.

Classical spatiotemporal CNNs and recurrent models remain important because they are often easier to train and deploy and can be strong in narrow regimes. Their governance advantage is simplicity; their governance risk is brittleness and overconfidence when used outside a narrow scope.

The fourth component is the *constraint and projection layer*. This layer exists because a learned backbone rarely guarantees physical validity. Constraints can be implemented as *soft* penalties during training (physics-informed losses, residual penalties, divergence-free regularizers) or as *hard* projections during inference (enforcing positivity, projecting onto divergence-free subspaces, normalizing conserved quantities, applying boundary condition enforcement). From a governance perspective, hard projections are appealing because they provide explicit guarantees on specific constraints, but they can also introduce artifacts or hide model deficiencies by forcing outputs to satisfy a constraint without being dynamically consistent. Soft constraints are flexible but provide no guarantee. A well-governed system typically uses both: soft constraints to encourage physically structured learning and hard checks/projections to ensure that certain violations are never passed downstream.

The fifth component is the *decoder and observable computation layer*. Decision-makers rarely consume raw fields. They consume derived metrics: lift and drag, maximum stress, energy spectra,

flux integrals, detection efficiencies, or risk indicators. The decoder maps predicted fields to these observables and also computes diagnostics for governance: residual norms, conservation errors, stability indicators, and uncertainty summaries. This layer is where "what the organization cares about" becomes explicit. If the decoder computes only business metrics and not physical diagnostics, the surrogate system becomes a liability: it is optimized for decisions without monitoring physical validity.

The sixth component is the *uncertainty and OOD layer*. This may be implemented via ensembles, dropout-based approximations, Bayesian neural approximations, conformal calibration, energy-based OOD detectors, embedding-distance measures, or hybrid solver comparisons. The specific method matters less than the governance principle: the surrogate must emit a signal that can gate its own usage. Without uncertainty and OOD signals, the institution has no automated way to enforce envelope boundaries, and every use becomes a human judgment call. That is not scalable, and it defeats the point of "fast physics."

The seventh component is *orchestration and logging*. The surrogate system must integrate with simulation pipelines, optimizers, and design tools. It must log inputs, model versions, outputs, uncertainty, diagnostics, and whether fallback was triggered. This logging is not a mere engineering convenience; it is the basis for incident reconstruction and accountability. If the institution cannot replay what the surrogate did, it cannot audit decisions that depended on it.

Finally, there is the *fallback and authority layer*: the solver, experimental measurement, or higher-fidelity evaluator that is invoked when the surrogate is uncertain, out-of-envelope, or used for high-stakes decisions. A surrogate system without a fallback is not necessarily invalid, but it is inherently riskier because it relies entirely on the surrogate's self-assessment. In most high-accountability contexts, a fallback is the difference between exploratory use and decision-grade use.

These components together define what "technical foundations" means in this chapter: not architecture in isolation, but architecture embedded in a governed system that produces evidence and preserves traceability.

2.4.2 Information flow

Once the components are named, the next question is how information moves through the system and where governance signals are produced. The canonical information flow begins with conditions and ends with decision-relevant observables, but there are multiple variants depending on whether the surrogate predicts one-shot trajectories or rolls out step-by-step.

At the start, the system ingests a condition object c that includes $(u_0, \theta, \mathrm{BC}, \mathcal{D})$, where $\mathcal{D}$ denotes domain and discretization descriptors. The encoding layer maps c into a latent representation z. This mapping is not neutral: it defines invariances and inductive biases. For example, a Fourier-based

operator implicitly assumes periodic structure or at least global spectral representation; a mesh graph encoder emphasizes connectivity and local neighborhoods. Governance requires that the encoding be documented because changes in encoding can change envelope behavior even if the backbone is unchanged.

From z, the backbone produces either a *one-shot* prediction of the full trajectory $\hat{u}(\cdot, t)$ for $t \in [0, T]$, or a *stepwise* prediction that yields $\hat{u}_{t+1} = f_\phi(\hat{u}_t, \theta, \mathrm{BC})$. One-shot prediction is attractive because it avoids compounding errors from self-feeding, but it can be harder to train and can hide temporal inconsistencies. Stepwise prediction aligns with the structure of many solvers and can be simpler, but it raises the compounding-error problem. Many operational systems use a hybrid: the model predicts in chunks, or predicts a coarse trajectory that is refined.

After the backbone prediction, the constraint/projection layer applies checks and potentially projections. Importantly, governance distinguishes *checks* from *projections*. Checks produce diagnostics: residual norms, conservation errors, boundary condition satisfaction, positivity constraints. Projections modify the output to satisfy constraints. In a high-accountability setting, checks must always be recorded, and projections must be explicitly marked, because projections can create a false appearance of validity. A projected output may satisfy conservation, but it may be dynamically inconsistent with the learned evolution. Decision owners must know whether they are seeing a raw model output or a corrected one.

Next, the system computes observables $\hat{y}$ and governance signals g. Observables might include performance metrics, risk margins, or scientific summary statistics. Governance signals include uncertainty estimates $\hat{\sigma}$, OOD scores s_{ood}, constraint violations v, and stability indicators κ. The key point is that the system must output both $\hat{y}$ and g. If it outputs only $\hat{y}$, it is an accelerator without brakes.

Finally, the orchestration layer applies a policy π that decides how to proceed. A minimal policy might be:

- If s_{ood} exceeds threshold τ_{ood}, do not accept surrogate output; invoke fallback.
- If uncertainty $\hat{\sigma}$ exceeds threshold τ_σ for decision-critical observables, invoke fallback.
- If constraint violation v exceeds tolerance, reject output and escalate.
- Otherwise, accept output for the intended use and log the decision.

This policy is part of the information flow because it determines whether the surrogate's output becomes evidence or remains a proposal. In a governed system, π is documented, versioned, and reviewed like any other control. It encodes the institution's risk tolerance.

The distinction between one-shot and rollout has direct consequences for information flow. In a rollout setting, the system repeats the encode-predict-check loop at each step or at each chunk. Governance signals must be tracked over time. A model may be within envelope at early steps and drift later. Therefore the system should compute time-indexed diagnostics: $v(t)$, $s_{\mathrm{ood}}(t)$, $\hat{\sigma}(t)$, and

stability indicators over the horizon. This is analogous to trajectory evaluation in agentic systems: the evaluation unit is not a single output but the behavior over time.

The information flow must also accommodate integration into external loops: optimization, active learning, and control. In those settings, the surrogate's outputs are consumed not by a human immediately but by another algorithm that proposes new inputs. Therefore governance signals must be machine-readable and gating must be enforceable. A warning that a human might ignore is insufficient; the system must be able to block downstream consumption when conditions are out of envelope.

In summary, the technical foundation is not merely that information flows from inputs to outputs. It is that governance information flows alongside predictions and that policies use those governance signals to determine admissibility. The surrogate becomes a controlled instrument rather than a free-running generator of plausible fields.

2.4.3 Interaction or control loops

Surrogates are most valuable when embedded in loops: design iteration, optimization, adaptive sampling, and sometimes operational control. These loops are where the institution gains leverage—and where risk amplifies. A surrogate that is safe in isolation can become unsafe when used repeatedly by an optimizer that searches for edge cases. Therefore the technical foundations must treat loops as first-class objects.

The first and most common loop is the *design exploration loop*. Teams propose candidates, evaluate them, and refine. A surrogate accelerates evaluation, enabling broader exploration. Governance requires that exploration be separated from commitment. A mature pattern is to use the surrogate to narrow the candidate set and then use the solver to verify finalists. This is a technical workflow pattern with a governance rationale: it preserves speed while ensuring decision-grade evidence is solver-verified. The technical design challenge is to structure the interface so that solver verification is not optional. In practice, this often means embedding verification steps in pipeline tooling rather than relying on human discipline.

The second loop is *optimization against the surrogate*. In this loop, an optimizer—gradient-based, evolutionary, Bayesian, or heuristic—queries the surrogate many times to find designs that maximize an objective. This loop is a classic amplification mechanism. The optimizer will push the surrogate into regions where it is uncertain or wrong, because those regions may appear to offer high objective values. This is not an exotic adversarial attack; it is the natural behavior of search. Therefore, a governed surrogate system must include optimizer-aware controls. At minimum, the optimizer should receive penalty terms for uncertainty and OOD scores, making it expensive to propose designs in unvalidated regions. More robustly, the optimizer should be constrained to remain within an explicit envelope, and any candidate outside that envelope should be either rejected or automatically routed to solver evaluation. The key technical point is that governance cannot be layered on top

after the fact. It must be integrated into the objective and constraints of the loop.

The third loop is *active learning and data acquisition*. Here the surrogate's uncertainty guides the selection of new solver runs. This is a powerful way to spend solver budget efficiently: simulate where it matters. But it requires technical care. If uncertainty estimation is uncalibrated, active learning can waste budget or reinforce blind spots. If selection criteria focus only on high uncertainty, they may miss rare but critical boundary regimes where uncertainty is falsely low. Therefore a governed active learning loop typically combines multiple criteria: uncertainty, diversity coverage, boundary exploration, and decision impact. It also logs selection rationales, because the data pipeline becomes part of the governance story: the institution must be able to explain why certain regimes were sampled and others were not.

The fourth loop is *hybrid surrogate-plus-solver correction*. In this pattern, the surrogate provides an initial guess or a coarse approximation, and the solver refines or corrects it. Technically, this can reduce solver iterations, speed up convergence, or enable multilevel methods. Governance-wise, it provides a backstop: the final evidence is solver-consistent. The design challenge is to ensure that the surrogate does not introduce biases that the solver fails to correct, particularly if the correction is approximate. For example, if a solver is used with early stopping because the surrogate's initial guess appears good, the institution may unknowingly accept a biased result. Therefore hybrid correction must itself be validated, and correction budgets must be governed. The surrogate should accelerate without reducing the solver's reliability guarantees beyond acceptable tolerance.

The fifth loop is *uncertainty-triggered gating and fallback*. This loop is central to governance. The surrogate runs by default, but when uncertainty or OOD signals exceed thresholds, the system triggers fallback: a solver run, an experimental check, or human review. Technically, this requires that uncertainty and OOD signals be produced cheaply and reliably. It also requires that fallback be accessible within workflow latency constraints. If fallback is too slow, teams will bypass it. Governance therefore includes an operational design requirement: fallback must be feasible enough that it is actually used. In many organizations, this implies tiered fallback: cheap approximate solvers for quick checks, high-fidelity solvers for final sign-off, and experimental checks for selected cases.

The sixth loop, in some domains, is *control-in-the-loop*. The surrogate is used as part of a controller, model predictive control, or adaptive experimentation. This loop is the most sensitive because errors can translate into actions. A surrogate-based controller must therefore be conservative: it must include safety constraints, robust fallback controllers, and explicit stop conditions. The technical foundations here overlap with safe control: constraints must be enforced, and uncertainty must be treated as a reason to act less aggressively, not more. This chapter does not aim to teach control theory, but the governance principle is clear: if the surrogate influences action, then the system must be designed to fail safe.

Across all loops, the recurring theme is that surrogates are not merely predictors. They are

evaluators embedded in search, sampling, and control. The technical foundations must therefore include loop-aware validation: test the surrogate not only on static test sets but within the loop dynamics that will use it. This is analogous to evaluating an agent in the environment it will act in, not only on static questions. For surrogates, the "environment" is the optimization and decision pipeline.

2.4.4 Assumptions and constraints

Every surrogate system rests on assumptions. Governance begins when those assumptions are made explicit and translated into enforceable constraints. Several assumptions are common and must be confronted directly.

The first assumption is *regime coverage*. A surrogate learns from data. If training data does not cover relevant regimes, the surrogate cannot be trusted in them. This seems obvious, but organizations frequently assume coverage because datasets are large. Size is not coverage. Coverage requires deliberate sampling across parameter spaces, geometries, boundary conditions, and initial condition families. Governance therefore requires that the institution define what regimes matter for decision-making and ensure that training and validation include them. This is where envelope definition begins: the envelope is not merely a description of valid inputs; it is a statement that training and validation evidence exists for those inputs.

The second assumption is *discretization compatibility*. Many surrogates are trained on specific grids or meshes. Even operator-learning methods that claim discretization robustness often have practical sensitivities. Therefore the institution must decide whether the surrogate is tied to a discretization or can handle multiple. If it is tied, then the envelope must include discretization constraints. If it can handle multiple, then the institution must validate that claim across resolutions and mesh families. Governance requires that discretization choices be logged because changes in mesh can be a hidden form of distribution shift.

The third assumption is *physical completeness*. Simulations often simplify physics: turbulence models, reduced chemistry, neglected couplings, idealized boundary conditions. A surrogate trained on these simulations inherits their simplifications. This is a governance issue because decision-makers may forget what is missing. If the surrogate is used to support decisions outside the simulation's validity, the institution can commit errors of false certainty. Therefore governance artifacts must include a statement of physics scope: what equations, what couplings, what approximations, and what is explicitly not modeled. This mirrors model risk management in finance: the model is only valid for what it represents.

The fourth assumption is *stationarity of usage*. Many surrogates are trained and validated under assumptions about how they will be used: certain parameter distributions, certain geometry families, certain horizons. In deployment, usage drifts. Teams explore new designs; products evolve; operating conditions change. Therefore the system must assume that shift will happen and must include

monitoring. Governance here is not a one-time certification but an ongoing regime-tracking practice. The envelope must be enforced at runtime, and drift must trigger revalidation or retraining.

Constraints arise as the operational expression of these assumptions. A governed surrogate system includes both hard and soft constraints.

Hard constraints are enforced rules that prevent the surrogate from being used outside envelope or from producing outputs that violate non-negotiable conditions. Examples include rejecting inputs outside parameter bounds, rejecting outputs that violate positivity, rejecting trajectories that exceed stability thresholds, and blocking downstream consumption when OOD scores exceed limits. Hard constraints reduce flexibility but provide safety.

Soft constraints are encouraged behaviors, often implemented as loss terms or penalties: conservation residual penalties, smoothness regularization, spectral constraints, or stability-promoting losses. Soft constraints can improve behavior but do not guarantee it. Governance therefore treats soft constraints as evidence of intent, not evidence of compliance. Compliance must be checked explicitly.

A critical governance choice is deciding which constraints must be hard. In many physical domains, some constraints are truly non-negotiable: mass conservation within tolerance, positivity of density, adherence to boundary conditions. Others may be negotiable within bounds, especially for exploratory use. The institution must decide which are which and encode that decision in tooling. If everything is soft, then the surrogate's validity is a matter of hope. If everything is hard, the surrogate may become unusable. The point is not maximal constraint but governed constraint aligned with decision risk.

Another assumption concerns the relationship between surrogate outputs and decision thresholds. Many decisions rely on safety margins. If the surrogate has bias near the margin, it can flip decisions. Therefore the institution must treat margin regions as special regimes. Constraints and validation should be stricter near decision boundaries. This is a direct analog to finance: models must be reliable near risk limits, not only in average conditions. In physics surrogates, the "risk limit" may be a stress threshold, a temperature limit, or a stability boundary. Governance requires targeted evaluation there.

2.4.5 Technical bottlenecks

Despite the rapid progress in architectures, several bottlenecks remain stubborn, and they are precisely the bottlenecks that determine whether surrogates can be trusted. These bottlenecks are technical, but they are also governance bottlenecks because they limit the strength of control signals.

The first bottleneck is *long-horizon stability*. Many surrogates can predict accurately for short horizons or produce plausible one-shot outputs. The difficulty is maintaining fidelity under rollout, especially in chaotic or turbulent systems. Small errors can grow, and the surrogate can drift into

unphysical attractors. This bottleneck matters because many real uses involve repeated predictions: control, scenario simulation, iterative design loops. If stability cannot be demonstrated over the relevant horizon, the surrogate must be restricted to short-horizon or exploratory roles. Governance should treat "horizon of validity" as part of the envelope: a surrogate may be valid for $T \leq T_{\max}$ and invalid beyond. This is a concrete, enforceable control.

The second bottleneck is *calibration of uncertainty for high-dimensional fields.* Producing a single uncertainty scalar for a complex field is not enough; uncertainty is spatially structured and regime-dependent. Calibration becomes difficult because errors can be localized, correlated, and non-Gaussian. Many UQ methods scale poorly or produce uncertainty that is hard to interpret. Yet without reliable uncertainty, gating policies are weak. This is why calibration remains a frontier bottleneck: it is the interface between the surrogate and governance.

The third bottleneck is *out-of-distribution detection in structured physics spaces.* OOD in physics is not like OOD in images. It can be subtle: a boundary condition pattern that is slightly different, a geometry modification that changes flow separation, a parameter combination that triggers a new regime. Embedding distances may fail because the model maps novel regimes into familiar-looking latents. Energy-based methods and ensembles can help, but none are perfect. Governance must therefore assume that OOD detection will have false negatives and must compensate by conservative policies, periodic solver spot checks, and stress testing.

The fourth bottleneck is *transfer across geometries and meshes.* Many engineering problems involve complex geometries, and mesh changes are routine. A surrogate that cannot handle geometry variation is limited to narrow use cases. Geometry transfer requires representations that respect symmetries, handle varying topology, and generalize across discretizations. This remains technically hard, and it is a major barrier to treating surrogates as reusable infrastructure. Governance implications follow: if geometry transfer is weak, then envelope enforcement must be strict, and reuse must be limited.

The fifth bottleneck is *regime boundary coverage and rare-event fidelity.* Many physically important phenomena are rare or localized: shocks, instabilities, fracture initiation, tail events in detector responses. These are precisely the events that drive safety and risk. Surrogates trained on average distributions may underrepresent them. Even when trained, loss functions may smooth them. Therefore the technical bottleneck is not just modeling capacity but data and objective design: how to ensure the surrogate preserves rare but critical structure. Governance must demand explicit evaluation on these phenomena and should treat them as special regimes requiring solver verification.

The sixth bottleneck is *integration robustness.* Even a good surrogate can fail when integrated into pipelines: preprocessing mismatches, unit errors, misapplied boundary conditions, or misinterpreted outputs. These are classic "system integration" risks that become more acute when the surrogate is fast and widely used. The technical foundation therefore includes careful interface design: typed condition objects, validation checks, and consistent metadata. Governance treats integration as part

of the risk surface, not as an afterthought.

A final bottleneck is *organizational dependency and change management.* Technically, surrogates can be updated, retrained, and improved. Institutionally, updates are risky because they can change behavior in subtle ways. This is a technical bottleneck because it requires robust evaluation harnesses: regression tests, envelope checks, stability tests, and calibration comparisons across versions. Without such harnesses, the organization cannot safely improve the surrogate. Governance therefore links directly to technical capability: the ability to update responsibly is part of the system's maturity.

In aggregate, these bottlenecks explain why "fast physics" is frontier rather than solved. The field can produce impressive demos, but deployment requires stability, calibration, OOD detection, and controlled transfer—precisely the areas that are hardest. This is the chapter's core message at the technical level: architectures matter, but governed deployment is bottlenecked by the reliability of signals that determine when to trust the surrogate. If those signals are weak, the system must be conservative. If those signals improve, the institution can widen the envelope responsibly. That is the path from novelty to defensible infrastructure.

2.5 Mathematical Foundations

2.5.1 Formal problem framing

A governed discussion of surrogates must eventually cash out in formal terms, because the institution's ability to define operating envelopes, validation criteria, and fallbacks depends on what exactly is being approximated. The mathematical object of interest in surrogate simulation is not merely a function $f : \mathbb{R}^d \to \mathbb{R}^k$. It is an *operator* that maps problem conditions—often themselves functions—to a solution field. We begin by defining the elements of a generic physics simulation task and then state the surrogate learning problem in a form that makes constraints and deployment usage explicit.

Let $\Omega \subset \mathbb{R}^p$ denote the spatial domain (with $p \in \{1, 2, 3\}$ in typical applications), and let $t \in [0, T]$ denote time. Let $u : \Omega \times [0, T] \to \mathbb{R}^m$ denote the state field, where m may be the number of physical quantities (e.g., velocity components, pressure, temperature, density, concentrations). A wide class of simulation problems can be written abstractly as a (possibly nonlinear) partial differential equation (PDE) or integro-differential system

$$\mathcal{F}(u; \theta) = 0 \quad \text{on } \Omega \times (0, T], \tag{2.1}$$

$$u(\cdot, 0) = u_0 \quad \text{on } \Omega, \tag{2.2}$$

$$\mathcal{B}(u; \theta) = 0 \quad \text{on } \partial\Omega \times (0, T], \tag{2.3}$$

where $\theta \in \Theta \subset \mathbb{R}^q$ denotes parameters (material properties, forcing coefficients, dimensionless numbers, geometry parameters), $\mathcal{F}$ encodes the governing equations, and $\mathcal{B}$ encodes boundary/forcing conditions. The *condition object* c is the tuple

$$c := (u_0, \theta, \mathcal{B}, \Omega).$$

In many practical settings, $\mathcal{B}$ itself can be parameterized and included as part of θ; we separate it to emphasize that boundary conditions are a distinct driver of regime behavior and a common source of shift.

Assuming the PDE is well-posed for c in the sense that a solution exists and is unique (or that we have selected a particular admissible solution concept), there is an induced solution operator

$$\mathcal{G} : \mathcal{C} \to \mathcal{U}, \qquad u = \mathcal{G}(c),$$

where $\mathcal{C}$ denotes the space of admissible conditions and $\mathcal{U}$ denotes the space of solution fields. The surrogate's objective is to approximate $\mathcal{G}$ by a parameterized operator $\mathcal{G}_\phi$ (weights ϕ) such that

$$\hat{u} = \mathcal{G}_\phi(c) \approx \mathcal{G}(c) = u.$$

Importantly, $\mathcal{G}$ is defined relative to an authoritative evaluator. In many organizations, $\mathcal{G}$ is not the true physics operator but the *solver operator*: the mapping implemented by a numerical method under specific discretization choices, turbulence closures, and modeling assumptions. This matters for governance. The surrogate inherits not only physics but also the solver's modeling scope.

Training data typically consists of pairs $\{(c_i, u_i)\}_{i=1}^{N}$, where $u_i = \mathcal{G}(c_i)$ is obtained by solver runs or measurements. The core learning problem can then be written as an expected risk minimization:

$$\min_{\phi} \; \mathbb{E}_{c\sim\mathcal{D}}\big[\ell(\mathcal{G}_\phi(c), \mathcal{G}(c))\big]\,, \tag{2.4}$$

where $\mathcal{D}$ is the deployment-relevant distribution over conditions and ℓ is a loss functional on fields. In practice, $\mathcal{D}$ is unknown and must be approximated by a training distribution $\mathcal{D}_{\text{train}}$. Governance risk arises precisely when $\mathcal{D}_{\text{train}}$ diverges from the true operational distribution $\mathcal{D}_{\text{op}}$. Therefore, the formal framing must explicitly distinguish these distributions, because the institution's envelope is, in effect, the subset of $\mathcal{C}$ where it believes $\mathcal{D}_{\text{train}}$ provides sufficient coverage.

To incorporate physical constraints, we often write an augmented objective:

$$\min_{\phi} \; \mathbb{E}_{c\sim\mathcal{D}_{\text{train}}}\big[\ell(\mathcal{G}_\phi(c), u) + \lambda\, \mathcal{R}(\mathcal{G}_\phi(c); c)\big]\,, \tag{2.5}$$

where $\mathcal{R}$ penalizes violations of PDE residuals, conservation laws, boundary inconsistencies, or other invariances, and $\lambda \geq 0$ controls the trade-off. This makes the constraint story explicit: constraints are encoded in $\mathcal{R}$, but their enforcement strength depends on λ and on training coverage.

The formal framing also depends on usage. If the surrogate is used for one-shot trajectory prediction, then $\mathcal{G}_\phi(c)$ outputs an entire field $u(\cdot, t)$ for $t \in [0, T]$. If it is used for stepwise rollout, then the surrogate approximates a transition operator:

$$\mathcal{T} : \; (u_t, \theta, \mathcal{B}, \Omega) \mapsto u_{t+\Delta t},$$

with learned approximation $\mathcal{T}_\phi$. The full trajectory surrogate is then the composition

$$\hat{u}_{k+1} = \mathcal{T}_\phi(\hat{u}_k, \theta, \mathcal{B}, \Omega), \quad \hat{u}_0 = u_0,$$

and the implied operator is $\mathcal{G}_\phi = \mathcal{T}_\phi^{(K)}$ for $K = T/\Delta t$. This compositional structure is the source of rollout instability and compounding error; it is therefore central to any governance-aware mathematical treatment.

Finally, decision-making typically uses observables rather than full fields. Let $\mathcal{H} : \mathcal{U} \to \mathbb{R}^r$ map fields to observables $y = \mathcal{H}(u)$ (e.g., lift, drag, stress maxima, flux integrals). Then the institution's decisions depend on $\hat{y} = \mathcal{H}(\hat{u})$. A surrogate may be acceptable if $\hat{y}$ is accurate and stable even if $\hat{u}$ has localized errors. Therefore, the formal objective and validation should align with $\mathcal{H}$ when decisions are the target. This is a governance point: you govern the surrogate relative to the evidence

it produces, not relative to an abstract field norm.

2.5.2 State, action, or representation spaces

The surrogate problem is often described as "learning from high-dimensional data." The mathematical content of that phrase is that both inputs and outputs live in discretizations of function spaces. Making this explicit helps clarify why generalization is hard and why stability and envelope definition dominate.

Let $\mathcal{U}$ denote a Banach or Hilbert space of admissible states (for example, $\mathcal{U} = L^2(\Omega; \mathbb{R}^m)$ or a Sobolev space $H^s(\Omega; \mathbb{R}^m)$ depending on regularity). Similarly, the condition space $\mathcal{C}$ includes $u_0 \in \mathcal{U}$, parameters $\theta \in \Theta$, and boundary condition objects $\mathcal{B}$ belonging to an appropriate boundary data space $\mathcal{B} \in \mathcal{V}$. The operator $\mathcal{G}$ maps between these spaces.

In computation, these function spaces are discretized. Let $\{x_j\}_{j=1}^n$ denote grid points or mesh nodes in Ω. The discretized state at time t is represented as a vector

$$u(t) \approx \mathbf{u}(t) \in \mathbb{R}^{n \times m},$$

often flattened to $\mathbb{R}^{nm}$. For time-discrete settings with steps $t_k = k\Delta t$, we have $\mathbf{u}_k \in \mathbb{R}^{nm}$. Similarly, boundary conditions and forcing may be discretized on $\partial\Omega$ or represented by parameter vectors; geometry may be represented explicitly via meshes or implicitly via signed distance functions.

The surrogate's representation space depends on architecture. In grid-based encodings, the surrogate takes tensors $\mathbf{u}_0 \in \mathbb{R}^{n_1 \times n_2 \times m}$ (2D) or $\mathbb{R}^{n_1 \times n_2 \times n_3 \times m}$ (3D). In mesh-based settings, the state is a set of node features on a graph $G = (V, E)$ with $|V| = n$. In coordinate-based settings, the surrogate approximates a continuous function by a network $f_\phi(x, t; c)$ whose values are evaluated at requested points.

This multiplicity of representations matters because it shapes inductive bias and governance risk. A model trained on a fixed grid implicitly assumes a fixed discretization; changes in mesh can act as distribution shift even if physical conditions are unchanged. A mesh-based model may generalize across discretizations but can be sensitive to graph connectivity and quality. A coordinate-based model may generalize across resolutions but can produce smooth approximations that miss sharp features. Therefore the "representation space" is not a neutral choice; it defines the surrogate's envelope in practice.

Where does "action" enter? In pure surrogate modeling, there is no action variable as in control. However, in rollout settings the surrogate defines a transition map, and in optimization settings the surrogate is queried by a policy that selects new parameters θ. In those contexts, we can treat the surrogate system as a dynamical system with state $\mathbf{u}_k$ and control/parameter inputs θ and $\mathcal{B}$. The

surrogate then defines an approximate evolution

$$\mathbf{u}_{k+1} = F_\phi(\mathbf{u}_k, \theta, \mathcal{B}, \Omega),$$

where F_ϕ is a discrete-time operator. Stability and compounding errors then become properties of this induced dynamical system. The institution's decision loop acts like an outer controller selecting θ, and the surrogate's outputs influence those selections. This is why surrogate risk resembles agentic risk: the system's behavior emerges from repeated interaction, not from a single prediction.

Finally, the mapping to observables $\mathcal{H}$ defines a smaller representation space $\mathbb{R}^r$ that is often the true target for decision-making. Governance should distinguish between field-level validity and observable-level validity. Some applications require field correctness (e.g., safety-critical stress distributions), while others primarily require accurate observables with bounded error. The mathematical framework supports both by treating $\mathcal{H}$ explicitly.

2.5.3 Objective functions and constraints

The choice of objective function is where governance intent becomes mathematical. It encodes what errors are tolerated, what physical violations are penalized, and how trade-offs are made. We consider three layers: field mismatch losses, physics constraint penalties, and hard projection or admissibility constraints.

Field mismatch losses. A common choice is an L^2 loss over space-time:

$$\ell_{\text{field}}(\hat{u}, u) = \int_0^T \int_\Omega \|\hat{u}(x,t) - u(x,t)\|^2 \, dx \, dt.$$

In discretized form, this becomes a weighted sum over grid points and time steps:

$$\ell_{\text{field}}(\hat{\mathbf{u}}, \mathbf{u}) = \sum_{k=0}^{K} w_k \sum_{j=1}^{n} \|\hat{\mathbf{u}}_k(x_j) - \mathbf{u}_k(x_j)\|^2.$$

While convenient, this loss can hide rare but critical errors. Governance-driven losses often include weights that emphasize boundary layers, regions of high gradient, or quantities of interest. One may also define losses directly on observables:

$$\ell_{\text{obs}}(\hat{u}, u) = \|\mathcal{H}(\hat{u}) - \mathcal{H}(u)\|^2,$$

or a combined loss $\ell = \ell_{\text{field}} + \alpha \ell_{\text{obs}}$.

Physics constraint penalties. Constraints can be encoded as residual penalties. Let $\mathcal{F}$ be the

PDE operator from (2.1). A physics-informed penalty might be

$$\mathcal{R}_{\text{PDE}}(\hat{u}; c) = \int_0^T \int_\Omega \|\mathcal{F}(\hat{u}; \theta)\|^2 \, dx \, dt,$$

plus boundary penalties

$$\mathcal{R}_{\text{BC}}(\hat{u}; c) = \int_0^T \int_{\partial\Omega} \|\mathcal{B}(\hat{u}; \theta)\|^2 \, dS \, dt.$$

Conservation laws can be encoded similarly. For example, if mass conservation implies $\nabla \cdot v = 0$ for velocity field v, one can add

$$\mathcal{R}_{\text{div}}(\hat{v}) = \int_0^T \int_\Omega \|\nabla \cdot \hat{v}(x, t)\|^2 \, dx \, dt.$$

Energy constraints, positivity constraints, and symmetry constraints can likewise be expressed as penalties. The governance caveat is that penalties are *soft*. They encourage compliance within the training distribution, but they do not guarantee compliance under shift.

Hard projections and admissibility constraints. Because soft penalties are insufficient for high-stakes deployment, many systems impose hard constraints at inference. Mathematically, this can be represented as projecting the raw output $\tilde{u} = \mathcal{G}_\phi(c)$ into an admissible set $\mathcal{A} \subset \mathcal{U}$:

$$\hat{u} = \Pi_{\mathcal{A}}(\tilde{u}),$$

where $\Pi_{\mathcal{A}}$ is a projection operator. The admissible set may encode positivity, conservation, boundary conditions, or other invariants. In discretized settings, projections can be explicit normalizations or constrained optimization solves.

Hard constraints can also appear as inequality constraints in training:

$$\min_\phi \ \mathbb{E}_{c \sim \mathcal{D}_{\text{train}}}[\ell(\mathcal{G}_\phi(c), u)] \quad \text{s.t.} \quad g(\mathcal{G}_\phi(c); c) \leq 0,$$

where g encodes constraint violations. In practice, such constraints are often relaxed into penalties.

The key governance trade-off is that projections can *mask* model deficiencies. If a projection enforces conservation, a downstream user may believe the model is physically correct, when in fact the projection is correcting an otherwise inconsistent output. Therefore a governed system must log projections and record both pre- and post-projection diagnostics. The institution must know whether "validity" is learned or enforced.

Another governance-relevant objective is rollout-aware loss. If the surrogate is used in rollout, training purely on one-step losses can produce models that drift. A rollout-aware objective might minimize cumulative error over a horizon:

$$\ell_{\text{roll}}(\phi) = \mathbb{E}_{c \sim \mathcal{D}_{\text{train}}} \left[\sum_{k=0}^{K} \|\hat{\mathbf{u}}_k(\phi) - \mathbf{u}_k\|^2 \right],$$

where $\hat{\mathbf{u}}_{k+1}(\phi) = F_\phi(\hat{\mathbf{u}}_k(\phi), \theta, \mathcal{B}, \Omega)$. This explicitly accounts for compounding. It is harder to optimize but aligns training with deployment. Governance should treat this alignment as a requirement when the surrogate will be rolled out in use.

Finally, decision-aware objectives incorporate uncertainty and OOD penalties. For example, one can penalize overconfident errors by adding calibration losses, or incorporate risk-sensitive losses that emphasize tail behavior:

$$\ell_{\text{risk}} = \mathbb{E}[\ell] + \beta\, \text{CVaR}_\alpha(\ell),$$

where CVaR_α emphasizes high-loss tail events. This is directly aligned with governance: tail risk matters more than average error in high-stakes settings.

2.5.4 Sources of instability or fragility

The formalism above makes clear why instability and fragility are structural rather than accidental. Several mechanisms produce fragility in surrogate systems, and they can be expressed mathematically.

Rollout error amplification. In rollout settings, the surrogate defines a dynamical system $\hat{\mathbf{u}}_{k+1} = F_\phi(\hat{\mathbf{u}}_k, \cdot)$. Let $\mathbf{u}_{k+1} = F(\mathbf{u}_k, \cdot)$ denote the true (solver) update. Define the error $e_k = \hat{\mathbf{u}}_k - \mathbf{u}_k$. A first-order expansion yields

$$e_{k+1} = F_\phi(\hat{\mathbf{u}}_k, \cdot) - F(\mathbf{u}_k, \cdot) \approx \underbrace{\big(F_\phi(\mathbf{u}_k, \cdot) - F(\mathbf{u}_k, \cdot)\big)}_{\text{model bias}} + \underbrace{J_k\, e_k}_{\text{propagation}},$$

where J_k is a Jacobian-like sensitivity of the dynamics with respect to the state. Even if the model bias term is small, the propagation term can amplify errors when $\|J_k\|$ is large, which is common in chaotic or stiff systems. This is why one-step accuracy does not imply rollout stability. Stability requires controlling the effective amplification factors over the horizon, either through architecture, training, or correction.

Regime discontinuities. Many systems exhibit qualitative changes as parameters vary. Mathematically, the solution operator $\mathcal{G}(c)$ may be continuous in some regimes and sharply changing in others; bifurcations, shocks, and instabilities can make the mapping highly sensitive. In such settings, a surrogate trained on smooth losses may learn an averaged, smoothed approximation that is wrong near discontinuities. The fragility is not merely that the model lacks capacity; it is that the mapping itself may be poorly approximated by smooth interpolators in regions of interest.

Mesh dependence and representation mismatch. Discretization enters the problem through the representation of u. If a surrogate is trained on a discretization-specific representation $\mathbf{u} \in \mathbb{R}^{nm}$, then the learned mapping effectively approximates a discretized operator $\mathcal{G}^{(h)}$ at mesh size h. When the mesh changes, the operator changes. Even if the physical operator is the same, the discretized object differs. Unless the surrogate is designed to be discretization-invariant and validated as such, mesh changes act as distribution shift. This is why transfer across meshes is a bottleneck: the model

must represent operator structure rather than discrete artifacts.

Compounding under iterative deployment and optimization. When a surrogate is embedded in an optimizer, the optimizer selects inputs c that maximize or minimize an objective based on surrogate outputs. If the surrogate has regions of systematic error, the optimizer will tend to exploit them. Formally, if the optimizer solves

$$c_\phi^\star = \arg\max_c \; J(\mathcal{G}_\phi(c)),$$

then the selected $c_\phi^\star$ may lie in a region where $\mathcal{G}_\phi$ overestimates performance. This produces a selection bias. The fragility is amplified because the optimization procedure concentrates probability mass on regions where the surrogate is least reliable. This is an adversarial phenomenon created by search itself, even without malicious intent.

Constraint mismatch and projection artifacts. If constraints are enforced via projections $\Pi_{\mathcal{A}}$, the effective dynamics become $\hat{u} = \Pi_{\mathcal{A}}(\tilde{u})$. Projections can introduce non-smooth behavior and can create artifacts that interact with rollout. A projection may correct a violation at one step but introduce subtle distortions that accumulate. Moreover, the projection may hide the underlying model's tendency to violate constraints, giving a false sense of learned validity. This creates fragility at the governance level: decision owners may trust the model more than warranted because constraint compliance is enforced rather than learned.

Uncertainty fragility. If uncertainty estimates $\hat{\sigma}(c)$ are used for gating, then errors in $\hat{\sigma}$ directly translate into governance failures. A false negative (low uncertainty when the model is wrong) produces silent failure. A false positive (high uncertainty when the model is correct) produces unnecessary fallback and erodes value. In high-dimensional systems, calibration is hard and can drift over time. Therefore uncertainty is itself a fragile component, and governance must treat it as such: periodically recalibrate, monitor drift, and avoid overreliance on a single uncertainty signal.

These fragilities are structural. They arise from composition, regime sensitivity, discretization, and the adversarial nature of optimization. The practical implication is that stability and governance controls cannot be added after training. They must be designed into the system: rollout-aligned evaluation, envelope enforcement, hybrid correction, and conservative gating.

2.5.5 What theory does not guarantee

A mature governance posture requires knowing what cannot be guaranteed, even with elegant mathematics. There are several non-guarantees that are especially important for executives and system owners because they define where human oversight and conservative policy remain necessary.

No guarantee of rollout stability. Even if a surrogate achieves low one-step error or low one-shot error on held-out data, there is no general theorem that guarantees stable behavior under long rollouts in complex nonlinear systems. Stability depends on the learned dynamics, the sensitivity of

the underlying system, and the operating regime. Without explicit stability analysis and horizon-aligned evaluation, the institution cannot assume that errors remain bounded. Therefore, "good test performance" is not a certificate of safe rollout.

No guarantee that constraints generalize. Physics-informed penalties and constraint-aware training can improve compliance within the training distribution, but they do not guarantee constraint satisfaction outside it. Constraint generalization is itself a generalization problem and can fail under distribution shift. Moreover, the surrogate can satisfy constraints in aggregate while violating them locally. Thus, constraints must be monitored at runtime. The institution should treat constraint checks as mandatory diagnostics, not optional training enhancements.

No guarantee of reliable uncertainty without explicit calibration. Uncertainty estimates produced by deep models are not reliable by default. Even Bayesian-inspired approximations can be miscalibrated in high-dimensional settings. Reliable uncertainty requires explicit calibration procedures, validation on relevant regimes, and ongoing monitoring. Without this, uncertainty thresholds are not defensible governance controls; they are heuristics. A governed institution must treat uncertainty as a continuously validated component of the system.

No guarantee of safe generalization across geometries and discretizations. Claims of transfer across meshes or geometries are often contingent on training data diversity and architecture design. There is no universal guarantee that a surrogate trained on one set of geometries will behave correctly on a new family, especially near regime boundaries. Therefore, envelope definition must include geometry scope, and new geometries should trigger revalidation. Reuse is valuable, but it must be earned with evidence.

No guarantee against optimization exploitation. When a surrogate is embedded in a search loop, it becomes part of an optimization landscape. There is no general guarantee that the optimizer will not find and exploit surrogate errors. Therefore, any surrogate-driven optimization must include verification gates. This is not pessimism; it is a known structural effect of optimization against imperfect evaluators.

No guarantee that the surrogate approximates reality rather than the solver. Even if the surrogate perfectly approximates the solver operator $\mathcal{G}$, that does not mean it approximates real physics. The solver itself may be biased or incomplete. Therefore, governance must treat the surrogate as inheriting solver assumptions. Experimental validation remains the ultimate anchor. The surrogate can accelerate exploration, but it cannot eliminate epistemic humility.

These non-guarantees define the boundary between technical ambition and institutional responsibility. The mathematical foundations show what is being learned and why constraints and stability are hard. Governance is the discipline of acting as if these non-guarantees are real, because they are. A surrogate program that ignores them will eventually discover them through failure. A governed surrogate program incorporates them upfront through envelope limits, fallbacks, monitoring, and explicit decision boundaries.

2.6 Evaluation and Validation

2.6.1 Why naive metrics fail

In simulation organizations, evaluation is not a ceremonial step at the end of a model build. It is the mechanism by which evidence becomes admissible for decisions. Surrogates change the cost structure of evaluation, but they do not change the institutional requirement: if you cannot evaluate the surrogate in the way you will use it, you cannot responsibly deploy it. The primary reason naive metrics fail is that they evaluate the wrong object. They evaluate snapshots and averages, while operational use depends on regimes, constraints, and trajectories.

Mean squared error (MSE) is the canonical naive metric. It is convenient, differentiable, and easy to report. It also systematically understates risk in precisely the settings where surrogate governance matters most. First, MSE aggregates error across space (and often time). A localized but critical error—a boundary-layer misprediction that drives stress, a shock position error that flips heating loads, a negative density artifact that triggers instability—can be "washed out" by the volume of the domain. Second, MSE privileges smoothness: models that smear sharp features can lower MSE even when they destroy physically important structure. Third, MSE does not respect conservation and invariance. A model can have low MSE while slowly violating mass conservation, accumulating error over rollout until the trajectory becomes physically meaningless. In other words, MSE can reward the wrong kind of approximation: it rewards visual plausibility and average closeness, not physical admissibility and decision-grade reliability.

Offline test sets introduce a second naive failure. Many surrogate evaluations use random splits: train on one subset of simulated conditions and test on another subset drawn from the same distribution. This is not wrong as a learning diagnostic, but it is often irrelevant to deployment. In physics and engineering, the most consequential failures occur at regime boundaries and under distribution shift: new geometries, new boundary conditions, parameter combinations that trigger different dynamics. Random splits rarely emphasize those regions. They create a comforting illusion: a model looks strong on "held-out" data while remaining fragile where the organization will actually push it. This is particularly pernicious in surrogate-enabled optimization loops, where the organization will systematically explore extremes, not average conditions. An offline test set sampled from historical simulations is therefore often the opposite of what is needed: it tests the model where it is most comfortable.

Averages mask catastrophic tails, and this is the governance-critical point. Institutions do not fail because a model is wrong on average. They fail because the model is wrong in a way that crosses a threshold: a safety margin is violated, a stability boundary is crossed, a manufacturing constraint is breached, a rare event becomes plausible. These are tail events in the space of conditions, and they are often tail events in the space of errors. A surrogate can have excellent average performance and still have a nontrivial rate of catastrophic outliers. If those outliers are rare enough, they will not

be visible in headline metrics and may not even appear in typical test sets. But when surrogates are used at scale—thousands of evaluations per day—rare outliers become inevitable. Governance therefore demands tail-aware evaluation: the institution must estimate the rate and character of worst-case failures, not only mean performance.

Finally, naive metrics fail because they do not match the evaluation unit of deployment. If a surrogate will be rolled out for 100 steps, evaluating one-step error is a category mistake. If a surrogate will be used to rank candidates in an optimization loop, evaluating it only on random samples misses exploitation behavior. If a surrogate will be used under specific parameter slices (e.g., near operating limits), evaluating it on global averages dilutes the relevant regime. The underlying principle is simple: the evaluation unit must be aligned to the use case. In governed deployment, this is non-negotiable. A surrogate that cannot be evaluated in the operational mode should be treated as exploratory only.

2.6.2 Appropriate evaluation units

If naive metrics fail because they measure the wrong thing, then appropriate evaluation begins by defining the evaluation unit correctly. In surrogate simulation, there are at least five units that should appear in any governed evaluation program: regime slices, constraint metrics, residual norms, stability indicators, and long-horizon error growth.

Regime slices. Regime-sliced evaluation treats the condition space as structured. Instead of reporting a single test score, the institution evaluates performance over slices of the parameter space Θ, boundary condition families, and geometry classes. For example, in CFD-like settings, slices might correspond to Reynolds number bands, Mach number bands, or angle-of-attack ranges. In structural analysis, slices might correspond to load magnitude regimes and material property regimes. In climate-like settings, slices might correspond to seasonal regimes or forcing scenarios. The governance goal is to identify where the surrogate is reliable and where it is not, and to define the operating envelope accordingly. Slice evaluation also enables targeted improvement: if performance collapses in a certain regime, active learning or additional solver runs can be focused there. Importantly, regime slicing should include boundary regions where regime transitions occur. A surrogate that is "good on average" but fails near transitions is not decision-grade.

Constraint metrics. Constraint metrics evaluate whether the surrogate respects physical invariants and admissibility conditions. These metrics are often more important than field error because they correlate with catastrophic failure. Examples include mass conservation error, divergence constraints for incompressibility, energy drift, boundary condition satisfaction, symmetry violations, positivity constraints, and known integral invariants. In governance terms, these are "red-line metrics." Institutions can set tolerance thresholds beyond which outputs are rejected or escalated. Constraint metrics should be computed for every prediction in deployment, not only in offline evaluation, because constraint violations often spike under shift. A surrogate can look accurate in

benign regimes and violate constraints in extreme regimes; constraint monitoring is therefore the first line of defense.

Residual norms. Residual evaluation measures how well the surrogate satisfies the governing equations, even when ground truth is not available. This is particularly valuable when solver comparisons are costly. If $\mathcal{F}(u;\theta) = 0$ is the PDE, then a residual norm $\|\mathcal{F}(\hat{u};\theta)\|$ can serve as a physics-consistency diagnostic. Residual norms are not perfect: they depend on discretization and can be fooled by models that learn to reduce residuals without capturing correct dynamics. Nevertheless, residuals can be a useful governance tool, especially when combined with other checks. The key is not to treat residuals as proof, but as a screening metric that can trigger fallback when residuals are anomalously high.

Stability indicators. Stability evaluation targets rollout behavior and sensitivity. In rollout settings, institutions should measure error growth rates, divergence tendencies, and stability margins over the relevant horizon. One can define stability indicators such as the growth of $\|\hat{u}_k - u_k\|$ with k, the growth of constraint violations over time, or the sensitivity of outputs to small perturbations in initial conditions. In chaotic systems, sensitivity is expected, but a surrogate may exaggerate or distort it. Therefore stability indicators should be compared against solver behavior, not against an ideal of "no growth." The governance question is whether the surrogate's rollout remains within acceptable error bounds for the intended use and whether it signals when it is drifting.

Long-horizon error growth. Long-horizon evaluation is the most direct alignment with deployment when surrogates are iterated. Rather than evaluating one-step error, the institution evaluates trajectories: does the surrogate remain accurate over T seconds or K time steps, and how do errors accumulate? This can be summarized by horizon-specific metrics: error at k, integrated error over horizon, maximum deviation, and time-to-failure (the time until some constraint violation or threshold is crossed). Time-to-failure is particularly governance-relevant because it connects directly to operational restrictions. If the surrogate is reliable for 20 steps but not for 200, the envelope should specify that horizon. This is a concrete, enforceable limit.

These units should be paired with decision-relevant observables. If the organization uses $\mathcal{H}(u)$ to produce lift, stress, or other quantities, then evaluation should include observable error and calibration specifically on those. A surrogate may be acceptable if observables are reliable even if local field errors exist, but only if those field errors do not undermine safety or constraints. Therefore the evaluation must be multi-layered: field-level, constraint-level, observable-level, and trajectory-level.

A final requirement is *versioned evaluation*. A surrogate is not static: it will be retrained, updated, and deployed in new contexts. Evaluation must be repeatable and comparable across versions. This implies fixed evaluation suites aligned to operational regimes, with regression tests that detect degradation. In governance terms, the evaluation harness is part of the control system. Without it, change management becomes guesswork.

2.6.3 Robustness and stress testing

If slice evaluation and constraint metrics define a baseline, robustness testing defines whether the surrogate can be trusted under the kinds of perturbations and extremes that operational use will create. Stress testing is not optional; it is the only way to estimate tail risk and to detect failure modes that do not appear in nominal conditions.

A first class of stress tests involves **boundary condition perturbations**. Boundary conditions and forcing terms are major drivers of regime behavior. Small changes can produce large effects, especially near transitions. Therefore robustness evaluation should include controlled perturbations: vary inlet profiles, forcing amplitudes, boundary temperature conditions, or external fields. The purpose is twofold: (i) measure sensitivity relative to solver behavior, and (ii) detect when the surrogate behaves pathologically, such as producing nonphysical amplification or suppressing legitimate dynamics. Perturbations should be designed to mimic real operational variability, not just random noise. Governance is concerned with realistic stress, because realistic stress is what produces real failures.

A second class involves **longer horizons**. Many surrogates are evaluated over short rollouts because it is cheaper and because performance looks better. Governance requires the opposite: evaluate longer than you plan to use, and evaluate until failure. This is analogous to stress testing a financial model beyond normal market conditions. If a surrogate will be used for 50 steps, evaluate it for 100 or 200 and measure time-to-failure distributions. This reveals whether errors accumulate slowly, whether they spike, and whether the surrogate has stable attractors or drift into unphysical states. It also reveals whether uncertainty signals increase appropriately as the model drifts.

A third class is **worst-case regime testing**. This is not the same as random sampling. Worst-case regimes include parameter combinations that are physically extreme but operationally relevant: high loads, high Reynolds numbers, near-stall angles, extreme thermal gradients, stiff reaction regimes, or high-energy tail events in detector simulations. In many organizations, these regimes are precisely where decisions matter most because safety margins are thin. Surrogates must be evaluated there even if data is scarce. If data is scarce, that scarcity is itself a governance signal: the organization should not rely heavily on surrogate outputs in regimes where it lacks training and validation evidence.

A fourth class is **out-of-distribution evaluation**. OOD testing should not be treated as a single test; it should be structured. One can define OOD along several axes: geometry shift (new shapes), parameter shift (values outside training ranges), boundary condition shift (new patterns), and discretization shift (new meshes). Each axis should be tested separately and in combination, because combined shifts are common in practice. The evaluation should measure not only error but also *detection*: does the system flag the input as out-of-envelope? A model that is wrong but uncertain is less dangerous than a model that is wrong and confident. Therefore OOD stress testing must include calibration of OOD scores and uncertainty signals.

A fifth class is **adversarial parameter sweeps**. This recognizes that optimization behaves like an

adversary: it searches for extreme values. Institutions can simulate this by explicitly searching for conditions that maximize surrogate error or constraint violation. One can do this by optimizing the difference between surrogate and solver outputs over parameters, subject to plausibility constraints. The result is not necessarily physically likely, but it reveals vulnerabilities. This is a governance-oriented analog of adversarial testing in security: find the cracks before the optimizer or workflow finds them in production. Adversarial sweeps are particularly valuable when the surrogate is used in design optimization, because they approximate the exploitation behavior of the real pipeline.

Robustness testing must also include **pipeline perturbations**. Many failures arise not from the model but from integration: unit mismatches, normalization drift, discretization inconsistencies. Therefore evaluation should include end-to-end tests that run the full pipeline with realistic metadata and validate that outputs remain stable under small interface variations. This is often neglected in research settings but becomes critical in deployment. A surrogate system is only as reliable as its weakest interface.

Finally, robustness testing must be connected to **fallback behavior**. A governed surrogate system is not required to succeed under all stress. It is required to fail safely. Therefore stress testing should include scenarios designed to trigger fallback, and the evaluation should measure whether fallback triggers correctly and whether the system behaves as intended: does it route to solver verification; does it block downstream consumption; does it log the event. This turns robustness testing into a governance test: not only "does the model predict," but "does the system enforce its own limits."

2.6.4 Failure taxonomies

Evaluation is not complete until the institution can name and classify failures. A failure taxonomy is not academic labeling; it is the basis for monitoring, incident response, and improvement. In surrogate systems, several failure classes recur across domains, and a governed program should track them explicitly.

Drift in long rollouts. The surrogate's trajectory gradually diverges from solver behavior, sometimes without obvious discontinuity. Drift can manifest as slow energy gain/loss, phase errors, or gradual deformation of structures. Drift is dangerous because it can remain visually plausible while becoming physically wrong. Drift is detected by long-horizon error growth metrics and by monitoring constraint residuals over time.

Constraint violation spikes. Even when average constraint error is small, violations can spike under certain conditions. These spikes often correlate with regime boundaries, sharp gradients, or extreme parameter combinations. Constraint spikes are governance-critical because they can indicate a transition from "approximate but admissible" to "physically invalid." Monitoring should treat spikes as escalation events, not as noise.

Silent failure. The surrogate produces outputs that look plausible and do not trigger alarms, yet

are wrong in a decision-relevant way. Silent failures are the most dangerous because they bypass governance controls. They often occur when uncertainty is miscalibrated, when OOD detection fails, or when evaluation lacks coverage of the relevant regime. A governed program should measure silent failure rate by comparing surrogate outputs to solver outputs on random spot checks and on targeted stress tests, and by tracking cases where decisions based on surrogate outputs would differ from decisions based on solver outputs.

Overconfidence under OOD. This is a specific silent failure mode: the surrogate is used outside envelope, but reports low uncertainty. Overconfidence can arise because the model is deterministic, because uncertainty methods are weak, or because embeddings map novel regimes into familiar regions. This failure mode directly undermines uncertainty-triggered gating. It must be measured explicitly in OOD tests: the calibration question is not only "is uncertainty correlated with error" but "does uncertainty rise when the input is out-of-envelope."

Plausible-but-invalid outputs. Many learned models produce smooth fields that appear physically reasonable but violate subtle physical laws: wrong spectra, wrong invariants, wrong discontinuity structure. This failure mode is not captured by visual inspection and may not be captured by MSE. It requires domain-specific diagnostics: spectral checks, invariant checks, residual checks, and in some cases downstream inference checks (e.g., detector outputs affecting reconstruction). This class is especially important because it exploits human trust in plausibility.

Geometry and discretization brittleness. The surrogate performs well on trained geometries and discretizations but fails when meshes change or geometry varies. This failure mode is common in engineering because geometry variation is routine. It must be tracked separately because it implies envelope limitations: a surrogate may be valid for a geometry family but not for general geometry.

Optimization exploitation. When embedded in an optimizer, the surrogate is driven into regions where it is wrong. The failure manifests as designs that perform well under the surrogate but poorly under solver verification. This is a pipeline-level failure, not a static prediction failure. It requires evaluation within optimization loops and the inclusion of verification gates.

Integration and preprocessing errors. Unit mismatches, normalization drift, incorrect boundary condition encoding, and metadata errors can create failures that appear like model error but are actually pipeline defects. A failure taxonomy must include these because the operational response is different: you do not retrain the model to fix a unit mismatch; you fix the interface and add validation checks.

A useful taxonomy also distinguishes failures by *governance severity*. Some failures are acceptable in exploratory contexts but unacceptable in decision-grade contexts. For example, moderate field error may be acceptable if the surrogate is used only to rank candidates before solver verification. The same error may be unacceptable if the surrogate output is used as final evidence. Therefore each failure class should be mapped to permitted uses and escalation rules. This is how the taxonomy

becomes operational: it informs policy and monitoring.

2.6.5 Limits of current benchmarks

The final evaluation challenge is structural: many existing benchmarks and community evaluation practices are misaligned with governance needs. This is not a criticism of the research community; benchmarks serve different goals. But institutions must recognize that leaderboard performance is not a deployment certificate.

Most scientific machine learning benchmarks emphasize speed and average accuracy on curated datasets. They often use fixed train/test splits, fixed discretizations, and limited horizon rollouts. They rarely prioritize regime boundary coverage, long-horizon stability under rollout, constraint enforcement under shift, or fail-safe behavior. In other words, benchmarks are optimized for demonstrating capability, not for demonstrating governability. A model can "win" a benchmark and still be unsuitable for high-accountability deployment because it fails silently under conditions that benchmarks do not test.

Benchmarks also often underrepresent the complexity of real institutional pipelines. Real systems involve geometry changes, coupled physics, messy boundary conditions, and integration with decision loops and optimizers. Benchmarks frequently abstract these away for tractability. As a result, benchmark performance can overestimate operational robustness. Institutions must therefore build internal evaluation harnesses that reflect their real use cases. This is expensive, but it is unavoidable. The cost of building evaluation is part of the cost of adopting surrogates. This is consistent with the central thesis of AI 2026: as capability accelerates, evaluation becomes the bottleneck.

Another limitation is that benchmarks rarely measure uncertainty calibration in a governance-relevant way. They may report probabilistic scores or ensemble spread, but they often do not test the link between uncertainty and decision gating: whether uncertainty thresholds reliably prevent silent failures. Calibration must be evaluated relative to policy objectives: low false negatives for out-of-envelope detection, acceptable false positives for fallback. Benchmarks seldom encode these operational trade-offs.

Similarly, OOD testing is often shallow. Benchmarks may include a nominal distribution and a slightly shifted distribution, but real OOD in physics often involves regime transitions and geometry variations that are qualitatively different. Without strong OOD testing, institutions may deploy models that look robust on paper but fail when confronted with real novelty.

Finally, benchmarks seldom incorporate the most important question for governance: *what happens when the model is wrong*? A governed system is defined not by its success cases but by its failure behavior. Does it flag uncertainty, does it trigger fallback, does it remain stable enough to avoid catastrophic divergence, does it preserve constraints. Current benchmarks rarely score these properties. Therefore, institutions must treat benchmarks as preliminary evidence at best, useful

for selecting candidate architectures but insufficient for deployment decisions.

The practical conclusion is that evaluation and validation for physics surrogates must be treated as a program, not a metric. The program includes regime-sliced evaluation, constraint diagnostics, long-horizon rollout tests, OOD and adversarial stress testing, failure taxonomy tracking, and system-level verification of fallback behavior. This is not optional overhead. It is the operational core of governed "fast physics." If the institution invests in surrogates but not in evaluation, it is buying speed without buying confidence. In a high-accountability environment, that is not innovation; it is accumulated liability.

2.7 Implementation Considerations

2.7.1 Minimal viable implementation patterns

Implementation is where the chapter's thesis becomes operational: "fast physics" is not primarily a model selection problem; it is a system design problem under accountability constraints. A minimal viable surrogate program should therefore begin with patterns that preserve institutional admissibility of results. The goal is not to deploy the most advanced architecture; it is to deploy a workflow in which the surrogate produces value without becoming an un-auditable source of error.

The first minimal pattern is the **hybrid exploration–verification split**. In this pattern, the surrogate is explicitly designated as an exploration tool, and the solver remains the sign-off instrument. The surrogate proposes, ranks, filters, or approximates; the solver verifies the finalist designs, the safety-critical scenarios, or any case that crosses a governance threshold. The organization should treat this split as policy, not as an informal norm. In practice, this means the pipeline must make it hard to "forget" verification: solver runs should be triggered automatically for outputs that will be used in high-impact decisions, and surrogate-only results should be labeled as exploratory in logs and reports.

A key implementation detail is that hybrid workflows should be *asymmetric*. The surrogate should be allowed to be wrong in ways that do not harm the institution because the solver catches those errors before commitment. The surrogate's value is speed in broad search; the solver's value is evidence for decisions. In this sense, the surrogate can tolerate higher average error if it maintains high *ranking fidelity* or *directional correctness*. For example, in design optimization, if the surrogate reliably identifies promising candidates, solver verification can select the true optimum among them. The surrogate does not need to be perfect; it needs to be useful and governable.

The second minimal pattern is **tiered criticality modes**. Institutions rarely have one usage mode. They have exploratory research, engineering iteration, and formal sign-off. Each mode has different tolerance for error and different governance requirements. A mature implementation therefore exposes explicit "modes":

- **Mode A — Exploratory:** surrogate outputs are used to generate hypotheses and narrow search. Minimal constraints and faster runtime; uncertainty and OOD are still logged, but fallback is optional.
- **Mode B — Engineering iteration:** surrogate outputs can drive design iteration, but must pass constraint checks and must trigger solver fallback under uncertainty or OOD. Outputs are versioned and reproducible.
- **Mode C — Decision-grade:** surrogate outputs cannot be used as final evidence; solver verification is mandatory. The surrogate may still accelerate by warm-starting solver runs or narrowing candidate sets.

The critical point is that these modes must be encoded in tooling. If mode selection is an informal human decision, it will drift under time pressure. If mode selection is encoded (with default conservative settings), governance becomes enforceable.

The third minimal pattern is **uncertainty-triggered gating with mandatory fallback**. This is the simplest deployable "control loop" for surrogates. The surrogate runs first; it produces prediction and diagnostics. If uncertainty exceeds a threshold or OOD score exceeds a threshold, the system routes the case to a solver run and records that fallback was invoked. This pattern yields two governance benefits: it creates a clear operational envelope, and it generates a continuous stream of solver comparisons that can be used for monitoring and recalibration.

The fourth minimal pattern is **surrogate warm-starting and residual correction**. Even when the solver remains the final authority, the surrogate can provide value by reducing solver iterations. Warm-starting uses the surrogate's field prediction as the initial guess for iterative solvers. Residual correction uses the surrogate to predict a coarse field, then corrects it using solver residual iterations or multilevel methods. This pattern is often safer than full replacement because the solver's structure remains in control of convergence. However, the institution must validate that warm-starting does not introduce systematic bias or failure modes (e.g., convergence to the wrong attractor). Warm-starting should be treated as a performance optimization under governance, not as a free upgrade.

The fifth minimal pattern is **envelope-first deployment**. Before deployment, the organization defines an operating envelope $\mathcal{E}$ in parameter space, geometry space, and horizon. This envelope is not merely a statement; it is implemented as input validators and runtime checks. Any input outside $\mathcal{E}$ is automatically treated as high-risk and routed to solver verification (or rejected, depending on policy). Envelope-first deployment is the operational translation of the chapter's core message: the surrogate is valuable when its limits are enforced.

A minimal viable program should resist the temptation to "start with full replacement." Full replacement is sometimes possible in low-stakes settings, but it should not be the first pattern in high-accountability contexts. The correct initial deployment is a hybrid pipeline that produces measurable value while generating a steady stream of evidence about where the surrogate is reliable. Over time, the envelope can expand as evidence accumulates. That is what maturity looks like.

2.7.2 Reproducibility and determinism

Reproducibility is not a research virtue; it is a governance requirement. If the institution cannot reproduce a surrogate result that influenced a decision, it cannot defend that decision under audit, incident review, or regulatory scrutiny. In physics surrogates, reproducibility has several layers: dataset reproducibility, training reproducibility, inference reproducibility, and pipeline reproducibility.

Dataset and simulation reproducibility. The training dataset is itself an artifact that must be versioned. This includes not only the raw simulated fields but the specification of how they were generated: solver version, numerical settings, discretization choices, boundary condition distributions, geometry definitions, and any preprocessing steps. A common failure mode is "dataset drift": the training set is regenerated or expanded without proper control, leading to subtle differences that change model behavior. Therefore, a governed program should treat dataset creation as a controlled process with:

- dataset identifiers and immutable snapshots;
- explicit parameter distributions used to sample training conditions;
- solver configuration files and version hashes;
- mesh generation scripts and mesh versioning;
- documentation of any filtering, downsampling, or normalization.

If synthetic datasets are generated for training (common in benchmarks), the random seeds and sampling code must be logged. If real solver runs are used, the solver input decks and run manifests must be stored.

Training determinism and environment capture. Training deep models can be nondeterministic due to hardware parallelism, floating-point differences, and nondeterministic kernels. A governance-first program should not promise perfect determinism if it is not achievable. Instead, it should define a reproducibility standard that is sufficient for audit: the ability to reproduce a model within a tolerance and to reproduce evaluation outcomes and envelope claims. This typically requires:

- fixed random seeds (where feasible) and recorded seed values;
- pinned library versions and containerized environments;
- recorded hardware/driver versions when relevant;
- deterministic settings enabled where available (at cost to performance);
- training run manifests that record hyperparameters, data splits, and checkpoints.

The practical governance deliverable is a *training manifest* that can be replayed. Even if exact weight reproduction is impossible, the institution must be able to reproduce the training procedure and the evaluation results that justified deployment.

Inference determinism. Inference is often easier to make deterministic than training. If the surrogate is deterministic, inference should be bitwise reproducible given the same environment. If the surrogate is stochastic (e.g., diffusion-based ensembles), reproducibility requires fixing inference seeds and recording them. A governed program should treat stochasticity as a first-class deployment decision: stochastic surrogates can be valuable for uncertainty estimation, but they introduce complexity in audit. Therefore, the system should log whether an output is deterministic or sampled, and if sampled, it should log the seed and sampling configuration.

Discretization and preprocessing determinism. A major reproducibility failure mode is that the same physical condition c is encoded differently in different pipeline versions. For example, boundary condition discretization changes, interpolation methods change, or normalization constants change. These changes can alter outputs even when the model weights are unchanged. Therefore, preprocessing code must be versioned and its version must be recorded in every inference log. The institution should treat preprocessing as part of the model, not as a peripheral detail.

Audit-grade environment fingerprints. A reproducible surrogate program should produce an environment fingerprint for each training and deployment run: OS version, Python version, library hashes, git commit IDs, container image digest, and hardware identifiers (where relevant). The purpose is not to satisfy curiosity; it is to enable incident reconstruction. When something goes wrong, the institution must answer: what model ran, on what environment, with what inputs, producing what outputs. Without environment fingerprints, this is impossible at scale.

Reproducibility also interacts with governance policy: if the institution cannot reproduce a result, that result should not be admissible for decision-grade use. This is a hard boundary, but it is one that prevents a subtle failure mode: "ephemeral truth," where results exist only in transient runs and cannot be defended later. In high-accountability environments, ephemeral truth is unacceptable.

2.7.3 Logging and traceability

Logging is the mechanism by which a surrogate system becomes auditable. In simulation-heavy organizations, solvers often already produce logs, but surrogate systems require richer logging because the failure modes are different: they involve data distributions, learned approximations, and drift. A governed surrogate program should log at three levels: training logs, evaluation logs, and deployment logs. Each level must support traceability back to the artifacts that justified use.

Training regime logs. Training logs should include the regime coverage specification: what parameter ranges were used, what boundary condition families were sampled, what geometry classes were included, and what discretizations were represented. It should also include constraint settings: physics penalties, projection methods, and calibration methods. The goal is to be able to answer: *what did we train the model to do?* and *what evidence do we have that it can do it?*

Calibration and validation diagnostics. If uncertainty estimates are used for gating, calibration diagnostics are not optional. The system must store calibration curves, coverage tests, and OOD detection performance metrics. These diagnostics should be versioned because calibration can drift across model versions. Governance requires that the institution can show that uncertainty thresholds were chosen based on evidence rather than intuition.

Operating envelope artifacts. The envelope is the centerpiece of governable deployment. It should be recorded as an artifact that is human-readable and machine-enforceable. Practically, this often resembles a "model card," but for physics surrogates it must include more structure:

- parameter ranges and regime definitions (with explicit boundaries);
- geometry scope (families included and excluded);
- discretization scope (supported meshes/resolutions);
- horizon limits for rollout validity;
- constraint tolerances and which are hard vs soft;
- required fallback rules and gating thresholds;
- known failure modes and prohibited uses.

The envelope artifact is a governance contract between model builders and model users. Without it, the surrogate will be used outside scope because "outside scope" will not be legible.

Deployment trace logs. Every surrogate inference used in production should generate a trace record. At minimum, this record should include: input condition identifiers, model version, preprocessing version, output identifiers, uncertainty and OOD scores, constraint diagnostics, and whether fallback was invoked. If projections or post-processing were applied, that should be logged explicitly. The trace should be sufficient to reconstruct the exact decision context: what the model saw, what it produced, and how the system responded.

A common organizational failure is to log only the final output. That is not traceability. Traceability requires logging the governance signals that justified acceptance. If a surrogate output was accepted because uncertainty was below a threshold, the recorded evidence must include the uncertainty value and the threshold. If fallback was invoked, the record must include that decision and the solver run identifier. This is the difference between "we ran a model" and "we ran a governed system."

Linking logs to decision artifacts. In high-accountability environments, surrogate outputs are not ends in themselves; they feed into design documents, safety cases, or operational decisions. Therefore logs must link to downstream decision artifacts. This can be done via unique run IDs that propagate through the pipeline. The result is a chain of custody: from condition specification to surrogate run to solver fallback (if any) to final decision document. Without this chain, audit becomes a scavenger hunt.

Finally, traceability must support incident reconstruction. If a physical system fails or a design proves flawed, the organization must be able to reconstruct whether surrogate outputs contributed and whether controls behaved as intended. This is why logs must be structured, searchable, and retained according to policy. Governance is not satisfied by having logs somewhere; it requires being able to use them.

2.7.4 Cost, latency, and scaling issues

Surrogates change the economics of simulation, but only if implemented with clear break-even logic. A common mistake is to view surrogate adoption as a pure engineering upgrade: training is expensive, inference is cheap, therefore value is automatic. In reality, value accrues only when the

surrogate meaningfully replaces or accelerates expensive solver calls in workflows that operate at scale, and when governance controls do not erase the speed advantage through excessive fallback.

Training cost versus inference savings. Training a high-quality surrogate can be expensive in two ways: compute cost for training, and solver cost to generate training data. The institution should explicitly model break-even: how many solver calls must be avoided (or accelerated) for the surrogate program to pay for itself. Let C_{solver} be the cost per solver run, C_{train} be the combined cost of data generation and training, and C_{inf} be the cost per surrogate inference (typically small). If the surrogate replaces N solver runs, the break-even condition is roughly:

$$N \cdot (C_{\text{solver}} - C_{\text{inf}}) \gtrsim C_{\text{train}}.$$

In practice, the condition must be adjusted for fallback rate. If a fraction p of cases trigger solver fallback, then the effective savings per evaluation is reduced. The break-even logic becomes:

$$N \cdot \big((1-p)C_{\text{solver}} - C_{\text{inf}} + p \cdot \Delta\big) \gtrsim C_{\text{train}},$$

where Δ captures any solver acceleration due to warm-starting or reduced iteration count. The governance implication is immediate: a surrogate that triggers fallback too often may be safe but economically unattractive, and teams may be tempted to relax controls. Therefore the institution must design the workflow so that safety does not rely on bypassable controls.

Latency and workflow integration. Surrogates provide value when their latency is meaningfully lower than solver latency. In engineering pipelines, latency matters not only in absolute terms but in human workflow terms. If a solver run takes hours and a surrogate takes seconds, the workflow changes: engineers can iterate rapidly. But if governance requires solver fallback frequently and fallback takes hours, then the workflow can become fragmented: fast cycles interrupted by slow gates. A mature implementation uses tiered solvers or tiered verification: cheap approximate solvers for quick checks and high-fidelity solvers for final sign-off. This preserves the surrogate's interactive value while maintaining evidence quality.

Scaling and model serving. Many surrogate deployments involve high-throughput inference: thousands of evaluations per day in optimization loops. This introduces serving concerns: GPU availability, batching, concurrency, and reliability. Governance adds additional overhead: computing diagnostics, uncertainty estimates (often via ensembles), and logging. The institution must budget for this overhead. If governance diagnostics make inference too slow, teams may disable them. Therefore diagnostics must be engineered efficiently and prioritized: constraint checks and basic OOD detection often provide the most value per compute.

Value accrual locations. Surrogates do not create value everywhere. They create value where solver calls are numerous, expensive, and part of iterative loops: design optimization, uncertainty propagation, Monte Carlo studies, parameter sweeps, and real-time approximations. They create less value where solver calls are rare or where decisions require full solver evidence anyway. A

governance-first strategy is to deploy surrogates first in workflows where they can be safely used for exploration and where solver verification is already part of practice. This yields immediate value without creating new compliance burdens.

Hidden costs: evaluation infrastructure. The largest hidden cost of surrogate deployment is often not training. It is evaluation infrastructure: building regime-sliced test suites, stress tests, monitoring dashboards, and incident reconstruction tooling. This is not optional. It is the price of governability. Institutions that budget for training but not for evaluation will create a brittle program that cannot be defended. The correct framing is that surrogates are not a model project; they are a capability that requires ongoing evaluation operations.

2.7.5 Integration risks

Integration risk is where surrogates become institutionally dangerous. The technical model may be strong, but the organization can still fail because of how the surrogate is embedded in decision processes. Several integration risks recur and should be treated as explicit hazards.

Organizational dependency and model monoculture. As surrogates become fast and convenient, teams can become dependent. They stop running the solver except when forced. Over time, solver expertise atrophies, and the institution loses the ability to recognize when the surrogate is wrong. This creates monoculture risk: one surrogate model becomes the default lens through which physics is interpreted. If that model has a systematic bias, the organization can propagate that bias across many products or decisions. Governance should therefore mandate periodic solver use even when not strictly required, both to maintain calibration and to maintain institutional competence.

Downstream error propagation. Surrogate outputs often feed into downstream models, optimizers, or design rules. A small surrogate bias can propagate into large downstream impacts. For example, a systematic underprediction of stress could lead to under-designed components; a systematic overprediction of performance could lead to unrealistic product expectations. Integration risk therefore includes *coupling risk*: the surrogate is not isolated; it is part of a chain. The institution must map downstream dependencies and identify where surrogate errors have high leverage. Those points should have stronger verification gates.

Simulator replacement complacency. The most common integration failure is cultural: teams treat surrogate outputs as "the simulation," not as an approximation. This is especially likely when outputs look smooth and plausible and when surrogate runtime makes it feel like a solver. The organization then begins to skip verification because the surrogate's convenience creates a false sense of certainty. Governance must counteract this by explicit labeling, training, and enforced workflow gates. Technically, the system should watermark or annotate outputs with model version and confidence indicators, making it hard to mistake a surrogate result for solver evidence.

Scope creep beyond validated regimes. Surrogates invite reuse. A team trains a surrogate for

one geometry family or parameter range, and another team uses it for a slightly different problem because "it's close." Over time, "slightly different" becomes "different enough to fail," and failures become silent because the model still produces plausible outputs. Scope creep is therefore an integration risk that must be addressed with enforcement: envelope validators, OOD detection, and organizational policy that prohibits use outside envelope without revalidation. This is not bureaucracy; it is the prevention of untracked drift.

Incentives to bypass controls. If governance controls—fallback triggers, constraint checks, logging—add friction or delay, teams under deadline pressure will bypass them. This is not a moral failing; it is predictable incentive behavior. Therefore the implementation must minimize the temptation to bypass by making controls efficient, by integrating them into tooling, and by ensuring that bypassing is visible. For example, if the system logs when diagnostics are disabled, and if decisions require those logs, then bypass becomes institutionally costly. Controls must be designed as socio-technical mechanisms, not merely technical features.

Misinterpretation of uncertainty and diagnostics. Even when uncertainty signals exist, users may misinterpret them. A low uncertainty estimate does not mean correctness if calibration is poor; a high uncertainty estimate does not necessarily mean uselessness if fallback is available. Therefore integration requires user education and decision protocols: how to interpret signals, when to escalate, and what thresholds mean. Governance demands that these protocols be documented and trained, not assumed.

Change management risk. Surrogates will be updated. Updates can improve average performance while degrading rare-event behavior, stability, or calibration. If the organization lacks regression tests and evaluation harnesses, it cannot safely update. This creates a perverse outcome: either the institution never updates (stagnation) or it updates without control (drift). Change management must therefore be treated as part of integration: versioned deployment, staged rollouts, and mandatory evaluation before promotion to higher criticality modes.

In total, implementation considerations are not "engineering details." They are the practical governance architecture of surrogate deployment. A minimal viable program begins with hybrid patterns, enforces tiered criticality, treats reproducibility as admissibility, logs for traceability, models break-even under fallback, and anticipates organizational dependency. Surrogates are powerful because they compress time. Governance is required because compressing time also compresses the opportunity to notice that something is wrong. A well-implemented surrogate program buys speed *and* preserves the institution's ability to remain accountable for what it builds.

2.8 Impact and Opportunity

2.8.1 New capabilities unlocked

The central practical promise of physics surrogates is not that they make simulation "better." It is that they change the time constant of the design-and-validate loop. When a solver call takes hours, organizations structure themselves around scarcity: scarce runs, scarce iteration, scarce exploration, and—too often—scarce learning. When a surrogate produces approximations in seconds, the organization can shift from scarcity to abundance. That shift unlocks capabilities that are difficult or impossible under traditional solver economics.

The most immediate capability is **faster design loops**. Engineering design is a sequential decision process: propose a design, evaluate it, modify it, repeat. The speed of evaluation sets the cadence of innovation. With a surrogate, iteration becomes interactive. Engineers can explore parameter changes in real time, test hypotheses quickly, and identify promising regions of design space without waiting for long solver queues. This matters not only for productivity; it changes the quality of thinking. When the cost of evaluation is low, teams can test more counterfactuals, challenge assumptions earlier, and detect sensitivity to boundary conditions before committing to expensive prototypes.

A second capability is **broader exploration of design spaces**. Many design problems are combinatorial or high-dimensional: shapes, materials, boundary conditions, and control parameters interact. Under solver-only workflows, organizations often explore only a narrow region because each evaluation is expensive. With surrogates, one can run large parameter sweeps, Monte Carlo uncertainty propagation, and global sensitivity analyses that were previously impractical. This allows teams to quantify uncertainty more honestly: rather than presenting a single "best estimate," they can present distributions over outcomes. In high-accountability contexts, the ability to quantify variability and sensitivity is as valuable as the ability to optimize.

A third capability is **real-time approximations for control and monitoring**. In many physical systems, control decisions must be made faster than a solver can run: adaptive control in manufacturing, real-time monitoring of complex systems, interactive training simulators, or embedded diagnostics. Surrogates can provide approximate state estimates or predicted responses quickly enough to be used in operational loops. This unlocks new categories of applications: digital twins that respond in real time, model predictive control with richer dynamics, and interactive decision support tools that incorporate physical reasoning without requiring full simulation.

A fourth capability is **interactive scientific exploration**. In scientific settings, surrogates can act as "fast hypothesis engines," enabling researchers to explore families of scenarios rapidly. For example, in plasma physics or astrophysics, running high-fidelity simulations across large parameter spaces is often prohibitive. Surrogates can provide an initial map of the landscape, highlighting interesting regions for solver validation. This shifts the role of expensive simulation from exploration

to confirmation. It is an inversion of the traditional order, and it can accelerate discovery when handled responsibly.

A fifth capability, often overlooked, is **amortized solver expertise**. High-quality solvers encode decades of numerical methods and domain knowledge. Surrogates effectively distill some of that expertise into a reusable evaluator. When deployed carefully, a surrogate can propagate best-practice simulation behavior into workflows that otherwise would not have access to solver experts. This can raise the baseline quality of engineering iteration. However, this capability is only valuable when the surrogate's envelope is explicit and enforced; otherwise, it becomes a mechanism for propagating error at scale.

Finally, surrogates unlock **new evaluation patterns** that were previously too expensive. For instance, one can use ensembles of surrogate predictions to approximate uncertainty, run many cheap "what-if" checks, and detect sensitivity to boundary condition variation. In governance terms, this can strengthen the institution's risk posture by making it easier to stress test designs. This is an important inversion: surrogates can accelerate both capability and validation, if the organization uses them that way. The frontier opportunity is therefore not merely "faster engineering." It is "faster engineering with better visibility into uncertainty," provided the organization invests in the evaluation harness that makes this visibility real.

2.8.2 Organizational implications

Technological change matters in institutions only insofar as it changes roles, incentives, and decision rights. Physics surrogates do all three. If a solver becomes less central as a day-to-day tool, the simulation team's identity and responsibilities must change. The most successful organizations will be those that treat this as a deliberate governance transition, not as an informal drift.

One major implication is that **simulation teams evolve from execution to governance**. In solver-centric organizations, simulation specialists spend much of their time running simulations, tuning meshes, managing convergence, and delivering results. As surrogates take over routine evaluations, the simulation team's comparative advantage shifts. Their new value is not in producing a single run result. It is in defining regimes, validating surrogate behavior, calibrating uncertainty, and setting operating envelopes. The simulation team becomes the steward of admissibility: they define what can be trusted and under what conditions. This is not a downgrade; it is an elevation into a role that is closer to institutional risk management.

This shift requires explicit recognition in organizational design. If simulation teams are evaluated solely on throughput of runs, they will resist surrogates or will use them without governance to maintain perceived productivity. If simulation teams are evaluated on *governed confidence*—the quality of envelopes, the robustness of validation, the ability to reconstruct decisions—then surrogate adoption becomes a pathway to higher institutional value.

A second implication is the creation of **new roles centered on calibration and envelope management**. In finance, model risk management exists because institutions learned that models require oversight. Scientific surrogates create a similar need. A mature surrogate program will have designated owners for:

- **Operating envelope definition:** specifying regimes, geometries, discretizations, and horizons where surrogate outputs are admissible.
- **Calibration and uncertainty governance:** ensuring uncertainty estimates remain meaningful, and updating gating thresholds based on evidence.
- **Monitoring and drift detection:** tracking whether the surrogate is used outside envelope and whether performance changes over time.
- **Change management:** managing version upgrades, regression testing, and controlled rollouts.

These roles can be housed in simulation teams, platform teams, or risk functions, depending on the institution's structure. The governance principle is that they must exist and have authority. Without authority, envelope definitions become "guidelines," which are inevitably ignored under deadline pressure.

A third implication is **a shift in decision rights**. When solver results were scarce and expensive, decisions were often bottlenecked on simulation availability. Surrogates reduce that bottleneck. More people can access simulation-like outputs. This democratizes capability, but it also risks democratizing error. Therefore institutions must decide who is allowed to use surrogate outputs for which purposes. This often becomes a tiered access model: broad access for exploratory use, restricted access for decision-grade use, and mandatory solver verification for sign-off. Implementing this requires both policy and tooling. If policy exists without tooling, it will not scale.

A fourth implication is **new training requirements**. Engineers and managers must learn to interpret surrogate outputs differently from solver outputs. They must understand uncertainty, OOD signals, and the meaning of envelope boundaries. They must be trained to treat surrogate results as conditional evidence, not ground truth. This is a cultural shift. Without training, people will do what is natural: treat a smooth visualization as truth. Governance requires countering that natural tendency with disciplined protocols.

A fifth implication is **platformization**. Once surrogates exist, organizations often build platforms: shared inference services, libraries of pretrained models, standardized logging, and dashboards. This can create enormous value by reducing duplication. But it can also create concentrated risk: a platform surrogate used across multiple products can become a single point of failure. Therefore platform teams must coordinate closely with governance teams. Platformization without governance is the organizational equivalent of monoculture: efficient, scalable, and catastrophically correlated.

In sum, the organizational opportunity is not only to accelerate design. It is to build a new institutional function: simulation governance. Organizations that do this well will gain speed

without losing accountability. Organizations that do it poorly will gain speed and lose the ability to explain their own decisions.

2.8.3 Potential new industries or markets

When a capability becomes reusable and standardized, markets form around it. Physics surrogates are on a trajectory toward commoditization in certain domains, but commoditization will be governed by trust. The most plausible new markets are therefore not "surrogate models" in the abstract. They are *surrogate services with documented operating envelopes and validation evidence.*

One emerging category is **surrogate platforms**. These are managed services that integrate data pipelines, training infrastructure, inference APIs, logging, and monitoring. Their value is not merely that they provide a model. It is that they provide the scaffolding for governed deployment: version control, audit trails, and evaluation harnesses. In regulated or safety-critical industries, platform value will hinge on the quality of governance features. The platform that can show "here is the envelope, here is the calibration, here is the drift monitoring, here is the fallback integration" will be more valuable than the platform that only shows "here is a faster model."

A second market is **calibrated operator libraries**. In some physics domains, operators are reusable across institutions: certain PDE families, certain flow regimes, certain structural approximations. One can imagine libraries of pretrained surrogate operators with clear documentation of regimes and expected error bounds. These would function like numerical libraries today, but with a crucial difference: they would need to ship with governance artifacts. A "calibrated operator" is not a black box; it is an artifact with evidence. The market may reward vendors who provide not only weights but also validation suites, stress tests, and envelope definitions.

A third market is **marketplaces for surrogate components**. Rather than buying a full surrogate, organizations may buy components: geometry encoders, boundary condition embeddings, uncertainty modules, or constraint projection layers. These components could be assembled into domain-specific pipelines. This resembles the ecosystem around MLOps today, but with physics-specific constraints. Again, governance will be a differentiator: components that come with transparent diagnostics and integration guidance will be preferred.

A fourth potential market is **verification-as-a-service**. If surrogates become widespread, there will be demand for independent verification and auditing. External auditors could validate that a surrogate's envelope claims are supported by evidence, that calibration is sound, and that monitoring is adequate. This is analogous to model validation in finance but specialized for scientific surrogates. Such a market would likely emerge first in safety-critical industries where liability pressure forces independent review.

A fifth market is **hybrid solver–surrogate toolchains**. Solver vendors may integrate surrogates directly into their products: surrogates that warm-start solver runs, guide mesh refinement, or

propose reduced-order approximations. This is a natural evolution because solvers already have distribution and trust. Surrogates may become a feature of solver ecosystems rather than a competitor. In such integrated systems, the governance challenge becomes even more important: the line between "solver output" and "surrogate-assisted output" must be explicit.

Finally, there is a potential market in **domain-specific digital twins**. Digital twins require fast simulation-like behavior in real time. Surrogates can enable that. The market will be in manufacturing, energy, infrastructure, and aerospace, where real-time monitoring and predictive control have high value. However, these digital twin markets will be constrained by governance: the twin must be trusted. Trust will depend on envelope documentation and fail-safe behavior.

These markets are plausible but not guaranteed. Their success depends on whether the community can standardize evaluation and documentation enough to make trust portable. Without standards, every surrogate becomes a bespoke risk, and markets remain fragmented.

2.8.4 Second-order effects

Frontier technologies have predictable second-order effects: they increase scale, reduce marginal costs, and therefore increase the volume of outputs and the volume of mistakes. Physics surrogates are no exception. A governance-first analysis must therefore consider not only direct benefits but systemic risks that arise when "fast physics" becomes ubiquitous.

The first second-order effect is **cheap-but-wrong proliferation**. When simulation-like outputs are cheap, they will be produced in large quantities. This can improve exploration, but it can also flood organizations with plausible results that are not physically valid. If decision processes are not adapted, the organization may begin to treat quantity as confidence: "we ran thousands of scenarios, therefore we are safe." This is epistemic inflation. The number of runs does not matter if the evaluator is wrong in correlated ways. Surrogates can therefore create a false sense of robustness: massive scenario counts with unmeasured bias.

The second second-order effect is **monoculture risk**. If a small number of surrogate architectures or pretrained models become standard, many organizations may rely on the same approximations. This creates correlated failure risk across firms and supply chains. If a widely used surrogate has a systematic blind spot, it can propagate the same error into many products. This is not hypothetical; it is the standard pattern of platform dependencies in software. In physics, the consequence is that entire sectors could share the same modeling blind spot, which is dangerous because failures will not be diversified.

The third effect is **optimization amplification**. As surrogates become cheap, optimization loops become more aggressive. Organizations will search more, optimize more, and push designs closer to boundaries. This increases the value of surrogates, but it also increases exposure to surrogate error because optimization concentrates on extremes. In a solver-only world, the cost of evaluation

naturally limits how aggressively teams optimize. In a surrogate world, that natural brake disappears. Governance must replace it with explicit constraints and verification gates. Otherwise, surrogates will systematically encourage boundary-pushing behavior without adequate evidence.

The fourth effect is **skill atrophy**. If engineers rely heavily on surrogates, solver literacy can decline. Over time, fewer people will understand mesh sensitivity, convergence behavior, and numerical stability. This creates organizational fragility. When surrogate outputs are wrong, fewer people will recognize the warning signs. Skill atrophy also reduces the institution's ability to validate surrogates because validation requires understanding solver behavior. Therefore a governed program should include deliberate "skill preservation" practices: periodic solver use, solver training, and cross-validation exercises.

The fifth effect is **shifting liability surfaces**. When solver outputs are used, liability is anchored in well-understood numerical methods and established engineering practices. When surrogates influence decisions, the liability surface expands: training data, model versioning, drift monitoring, and calibration become relevant in incident reviews. If organizations do not implement traceability and envelopes, they may find themselves unable to defend decisions. The second-order effect is therefore not only technical risk but legal and reputational risk. This is one reason why governance artifacts—run logs, model cards, envelope definitions—are not bureaucratic overhead. They are liability controls.

The sixth effect is **cascading failures in coupled systems**. Surrogates are often used as components inside larger pipelines: multi-physics coupling, integrated design environments, or multi-stage decision chains. A small surrogate error can propagate through coupling and produce large downstream differences. When multiple surrogates are chained, errors can compound in unexpected ways. This resembles the compounding error problem in rollout dynamics, but at the system level. Governance must therefore consider system-level validation: it is not enough to validate each surrogate in isolation if they are used together.

The main lesson of these second-order effects is that the benefits of cheap simulation are inseparable from the risks of cheap error. Surrogates change not only what is possible, but what is tempting. Governance must anticipate those temptations.

2.8.5 Overstated expectations

The final part of impact analysis is to identify expectations that are likely to be overstated—because overstated expectations are a governance hazard. They lead organizations to deploy surrogates in inappropriate contexts, to reduce verification prematurely, and to confuse speed with truth.

The most common overstated expectation is **"surrogates remove the need for physical validation."** They do not. At best, surrogates shift where physical validation occurs in the pipeline. They can reduce the number of experiments by narrowing candidate sets, but they cannot eliminate

experimental anchoring, because both solvers and surrogates are models. The institution still needs periodic experimental checks to ensure that the solver—and therefore the surrogate—remains aligned with reality. In high-stakes systems, experimental validation remains the ultimate authority.

A second overstated expectation is **"foundation surrogates generalize broadly."** Broad generalization is possible in some structured domains, but physics is full of regime boundaries and qualitative transitions. Claims of broad generalization often hold within a family of regimes and fail at transitions. Governance requires that generalization claims be translated into envelope statements supported by evidence. "It generalizes" is not an admissible claim. "It is validated for these parameter bands, these geometries, these horizons" is admissible.

A third overstated expectation is **"uncertainty estimates make it safe."** Uncertainty is a control signal only if it is calibrated and monitored. Uncalibrated uncertainty can be worse than no uncertainty because it provides a false sense of safety. Moreover, uncertainty methods can have failure modes: overconfidence under OOD, underconfidence that triggers too much fallback, or instability across versions. Governance must treat uncertainty as a component that requires validation, not as a magic safety wrapper.

A fourth overstated expectation is **"speed automatically translates into value."** Speed translates into value only when it changes decision-making. If organizations cannot integrate surrogates into workflows with proper gates, they may not capture value. Conversely, if they integrate surrogates without gates, they may capture speed at the cost of liability. The value of surrogates accrues where they reduce cycle time *without* undermining admissibility. This is why hybrid patterns often dominate early deployments: they capture value while preserving sign-off integrity.

A fifth overstated expectation is **"a surrogate can replace a simulator as a drop-in."** Drop-in replacement is rare in high-accountability contexts because solvers provide more than outputs; they provide interpretability through numerical diagnostics, convergence behavior, and decades of institutional trust. Surrogates can approximate outputs, but they do not automatically provide solver-like evidence. Therefore replacement must be earned by building surrogate-specific evidence systems: constraint checks, residual monitoring, OOD detection, and traceability. The implementation burden is substantial. Organizations that underestimate it will either fail to deploy or will deploy unsafely.

Finally, there is the fundamental expectation that must be corrected: **"fast physics is physics."** Fast physics is only useful when error is bounded and known. The organization does not need perfect prediction; it needs decision-sufficient reliability. But decision-sufficient reliability is not a performance number; it is a governance state. It requires an envelope, validation, monitoring, and fallbacks. Without these, fast physics becomes fast plausibility. And plausibility, when mistaken for truth, is the most expensive output a model can produce.

The opportunity of physics surrogates is real. They can compress iteration cycles, enable broader exploration, and unlock new operational applications. But the frontier value is not in capability alone. It is in *governed capability*: speed coupled to explicit limits, calibrated confidence, and

enforceable verification. Organizations that internalize that distinction will treat surrogates as strategic infrastructure. Organizations that do not will treat surrogates as shortcuts—and will eventually discover that shortcuts in physics are rarely shortcuts in outcomes.

2.9 Governance, Risk, and Control

2.9.1 New risk categories

A physics surrogate is not merely a faster simulator. It is an institutional commitment to accept learned approximations as inputs to decisions that may ultimately touch safety, product integrity, capital allocation, and liability. The governance question is therefore not "is the model accurate?" but "what new categories of failure does this system introduce, and can the institution remain accountable when those failures occur?" In physics settings, the risk surface is distinctive because it couples three difficult properties: high-dimensional outputs, regime boundaries with qualitative shifts, and deployment workflows that often embed the surrogate inside iterative search.

The first risk category is **safety risk from silent physical invalidity**. In many physical domains, the most dangerous failure is not a visibly absurd output; it is a plausible output that violates constraints in ways that matter. A stress field that looks smooth but violates equilibrium locally; a flow field that "resembles" turbulence but violates mass conservation in ways that change lift; a thermal distribution that is near-correct on average but misplaces hotspots. These are failures that can cross safety margins while passing superficial review. They are also failures that can be amplified by downstream decision processes that trust the surrogate as a simulation-like authority. The key governance point is that physical invalidity is often silent. It is not a user-interface problem; it is a structural property of learned approximations under regime shift.

The second risk category is **accountability risk from evidence ambiguity**. Solver outputs come with a cultural and procedural infrastructure: convergence checks, mesh refinement studies, numerical diagnostics, and established norms about what counts as evidence. Surrogates can produce outputs that look like solver outputs but are not accompanied by solver-style evidence. If the organization does not build new evidence standards—constraint diagnostics, envelope checks, calibration audits—then surrogate outputs become evidentiary orphans: used in decisions without defensible provenance. Accountability risk emerges when a decision must be defended and the organization cannot explain why a surrogate output was accepted, what regime it was in, or what controls were applied.

The third risk category is **scope creep and envelope erosion**. Surrogates are highly reusable artifacts. Once one exists, there is a strong incentive to use it beyond its validated domain: a slightly different geometry, a slightly different boundary condition family, a slightly longer horizon. This is rarely malicious; it is operational convenience. But scope creep is precisely how silent failures enter. The envelope is eroded one "small" extension at a time until the surrogate is operating in conditions where validation evidence is thin. This risk is compounded by organizational memory: new teams inherit a surrogate and treat it as established infrastructure, without understanding its original assumptions.

The fourth risk category is **latent liability from optimization exploitation**. When surrogates are

used in optimization loops, they become targets of search. Even without adversaries, optimization pushes toward extreme designs and parameter combinations. If the surrogate has regions where it is systematically wrong in favorable directions, the optimizer will find them. The organization can then commit to designs that appear optimal under the surrogate but fail under the solver or in reality. This creates latent liability: the design pipeline has produced confident but false evidence. Unlike ordinary model error, exploitation-induced failures can be hard to detect because the surrogate's outputs are self-consistent and because the optimizer reinforces them.

The fifth risk category is **organizational dependency and competence atrophy**. A successful surrogate makes itself indispensable. That is the point. But dependency creates a second-order risk: solver literacy and numerical skepticism decline. Engineers may stop asking "what is the mesh sensitivity?" because no mesh is visible. Teams may stop running solver checks because they are "slow." Over time, the institution loses the tacit knowledge needed to notice when surrogate outputs are wrong. This is governance-relevant because it reduces the institution's ability to perform independent validation. If the surrogate becomes the only lens, the organization becomes brittle.

The sixth risk category is **correlated failure and monoculture**. If many teams within a firm—or across firms—use the same surrogate family, the same architectures, or the same pretrained operator libraries, failures become correlated. This matters in supply chains: multiple vendors may independently validate using the same surrogate approach and therefore share blind spots. Correlated failure is a systemic risk because it defeats diversification. In high-stakes engineering, monoculture is not merely an IT risk; it is a physical risk.

The seventh risk category is **data provenance and solver-assumption risk**. Surrogates inherit the assumptions of their training data. If training data comes from solvers, the surrogate inherits solver biases, closures, and numerical artifacts. If training data comes from experiments, it inherits measurement noise and experimental scope. In either case, provenance matters. The risk is not only that the surrogate is wrong; it is that the surrogate is wrong in a way that is hard to detect because it matches the training authority. This can create a closed loop of self-confirmation: the surrogate matches solver behavior, the organization treats solver behavior as truth, and drift from physical reality remains unnoticed until costly failure.

These categories are not theoretical. They are the natural consequences of placing a learned operator into a workflow that historically relied on numerical evidence and physical validation. Governance exists to ensure that new speed does not become new opacity.

2.9.2 Control points and safeguards

Control is the practical translation of governance. A surrogate program is governable only if it has enforceable control points—places in the workflow where the system can prevent misuse, trigger verification, and generate audit evidence. The controls must be designed as a system, not as independent "best practices," because weaknesses in one control will be exploited by workflow

pressure or by optimization dynamics.

The first control point is the **operating envelope** as a machine-enforceable artifact. The envelope is the institution's explicit statement of where surrogate outputs are admissible. It should include parameter ranges, boundary condition families, geometry classes, discretization assumptions, and horizon limits. Most importantly, it must be enforced at runtime. This implies input validators that check whether a requested condition c lies in $\mathcal{E}$. If $c \notin \mathcal{E}$, the system should refuse surrogate-only evaluation and either trigger fallback or require explicit escalation approval.

Envelope enforcement also includes "soft" envelope indicators: similarity metrics in embedding space, density estimators, or OOD scores. These can catch cases that are formally in range but semantically novel (e.g., boundary condition patterns never seen). Governance requires both: hard checks for explicit ranges and soft checks for novelty.

The second control point is **uncertainty thresholds that trigger fallback**. If the surrogate provides uncertainty estimates—through ensembles, stochastic sampling, or Bayesian approximations—then the institution can define thresholds beyond which outputs are not admissible without solver verification. This turns uncertainty into a gating mechanism. However, governance requires that thresholds be chosen empirically and that uncertainty be calibrated. A control that is not calibrated is not a control; it is a placebo. Therefore the safeguard includes a calibration requirement: uncertainty thresholds must be supported by validation evidence that demonstrates acceptable false negative rates for silent failure.

The third control point is **mandatory constraint checks**. Constraint checks are the simplest and often most powerful safeguard because they target physical admissibility. Every surrogate output should be accompanied by constraint diagnostics: conservation errors, residual norms, boundary condition satisfaction measures, and any domain-specific invariants. The institution defines tolerances. Outputs that violate tolerances are rejected or escalated automatically. Constraint checks should be mandatory even in exploratory modes, because they prevent the most dangerous failure class: plausible but physically invalid outputs being treated as evidence.

Constraint checks can also be layered. A cheap first-pass check can be run on every output; more expensive checks can be run on outputs that will influence high-impact decisions. This preserves speed while maintaining safety.

The fourth control point is **fallback to solver as a required pathway, not an optional escape hatch**. Fallback is not merely "run the solver sometimes." It must be engineered as part of the workflow. If fallback is painful—slow, manual, or politically difficult—teams will avoid it. Therefore a governed surrogate system integrates fallback so that it is automatic, tracked, and operationally feasible. Fallback should also be tiered: cheap lower-fidelity solvers for quick verification and high-fidelity solvers for sign-off. The system should record which level of fallback was used, and decision-grade use should require the appropriate level.

The fifth control point is **approval gates tied to criticality**. Different uses have different stakes.

Governance requires that the system recognize this. For example, using a surrogate to rank design candidates is lower risk than using it to certify safety margins. Therefore the pipeline should include explicit criticality tagging and corresponding gates. High-criticality uses require solver verification and human sign-off. Medium-criticality uses require constraint checks and calibrated uncertainty gating. Low-criticality exploratory uses may rely primarily on diagnostics and labeling. The essential point is that controls scale with risk.

The sixth control point is **version control and change management**. Models will change. Controls must ensure that a new surrogate version cannot silently replace an old one without revalidation. This implies: pinned model versions in production, staged rollouts, regression test suites, and explicit promotion criteria. Governance should treat surrogate updates as controlled releases, not as routine retraining. A model that is "slightly better" on average may be worse in tails, worse in calibration, or worse in stability. Therefore promotion must be evidence-based.

The seventh control point is **decision output labeling and provenance watermarking**. A practical safeguard is to make it difficult for users to confuse surrogate outputs with solver outputs. Reports and visualizations should include model version, envelope status, uncertainty status, and whether solver verification was performed. This is not merely UI decoration. It is a governance mechanism that reduces accidental misuse and creates a clear chain of custody for evidence.

Together these safeguards form a control stack: envelope enforcement, uncertainty gating, constraint checks, fallback integration, criticality gates, version control, and provenance labeling. A surrogate program lacking any of these is not necessarily doomed, but it is structurally vulnerable to the most common failure pathways: scope creep, silent invalidity, and bypass under pressure.

2.9.3 Human oversight boundaries

Human oversight is often invoked as a solution to model risk, but in practice it fails unless boundaries are explicit. "Human in the loop" becomes theater when scale is high and review is informal. For physics surrogates, effective oversight requires defining: who reviews, what they review, when review is mandatory, and what authority review has.

The first boundary is **mandatory human review for high-impact uses**. High-impact uses include: any decision that affects safety margins, regulatory submissions, certification, significant capital expenditure, or irreversible physical builds. For these decisions, surrogate outputs may be used as supporting information, but solver verification and human review must be mandatory. The human reviewer must have access not only to the surrogate output but to its diagnostics: envelope status, constraint checks, uncertainty signals, and fallback history. The reviewer's role is to assess admissibility, not to judge whether the field "looks right."

The second boundary is **separation of builder and decision owner**. The team that builds the surrogate should not be the sole authority on whether its outputs can be used to justify decisions.

This is a classic governance principle: avoid conflicts of interest. In practice, this can be implemented through independent validation teams, a model risk function, or a cross-functional review board that includes simulation experts, safety engineers, and domain owners. The goal is not to slow down innovation; it is to ensure that envelope claims and calibration claims are scrutinized by someone whose incentives are aligned with decision integrity.

The third boundary is **explicit sign-off discipline**. In solver-centric workflows, sign-off often depends on established practices: mesh independence studies, convergence criteria, and documentation. Surrogate workflows require analogous sign-off artifacts: envelope definition, validation results, calibration diagnostics, stress test outcomes, and documented failure modes. Human sign-off must attach to these artifacts, not merely to a prediction. The sign-off event should be recorded, including who approved, what version was approved, and what conditions were assumed. Without this, accountability is diffuse.

The fourth boundary is **limits on interpretive freedom**. Surrogate outputs can invite interpretation: "the flow looks stable," "the hotspot is small," "the system seems safe." Governance requires that interpretive claims be tied to metrics and constraints. Humans should not be permitted to "explain away" constraint violations because the plot looks plausible. Therefore oversight protocols should specify which diagnostics are binding. For example: if mass conservation error exceeds tolerance, the output is not admissible regardless of visual plausibility. This prevents the most common human failure mode: trusting the story the output tells.

The fifth boundary is **review workload realism**. If a surrogate program produces thousands of outputs per day, humans cannot review them all. Therefore oversight must be selective and policy-driven. Humans review: (i) the model and envelope before deployment, (ii) a sample of outputs for monitoring and spot checks, and (iii) any output that triggers escalation (OOD, high uncertainty, constraint violation near threshold). This is how human oversight scales: not by reviewing everything, but by reviewing the right things.

The sixth boundary is **escalation authority and stop-the-line power**. Oversight is ineffective if reviewers cannot halt deployment or force fallback. A governed program must grant reviewers authority: to restrict envelope, to mandate additional validation, or to suspend a surrogate version if drift is detected. This is culturally difficult in fast-moving organizations, but it is necessary. Surrogates compress time; governance must preserve the ability to stop when evidence is insufficient.

In summary, human oversight is a boundary system. It defines when humans must intervene, what evidence they must see, and what authority they have. Without explicit boundaries, "human oversight" becomes a rhetorical cover for ungoverned deployment.

2.9.4 Monitoring and auditability

Deployment is not the end of governance. It is the beginning of monitoring. A surrogate's validity is not static because operational distributions drift, usage expands, and pipelines change. Monitoring and auditability ensure that the institution can detect drift, enforce envelopes, recalibrate uncertainty, and reconstruct incidents when something goes wrong.

The first monitoring requirement is **regime drift tracking**. The organization must measure whether the surrogate is being used within its validated envelope. This means tracking the distribution of inputs c encountered in production and comparing it to the training and validation regimes. Drift can be explicit (parameters outside range) or subtle (boundary condition patterns changing, geometry families shifting). Monitoring should produce dashboards that show: what fraction of usage is in-envelope, what fraction triggers soft OOD signals, and where drift is concentrated. This turns scope creep from an invisible cultural phenomenon into a measurable operational signal.

The second requirement is **continuous calibration monitoring**. If uncertainty estimates are used for gating, calibration cannot be assumed stable. The institution should periodically compare surrogate predictions against solver runs or experimental measurements and evaluate whether uncertainty still correlates with error. This can be done through scheduled solver spot checks: randomly sample a subset of production queries for solver verification and measure calibration metrics. The sampling strategy should be risk-weighted: more sampling near envelope boundaries and near decision thresholds. Calibration drift should trigger policy adjustments: tighter thresholds, expanded fallback, or retraining.

The third requirement is **periodic solver and experimental spot checks**. Solver spot checks are a minimum; experimental spot checks are the gold standard when feasible. The governance principle is that a surrogate program must maintain an external anchor to reality. If the surrogate is trained on solver outputs and the solver is not periodically checked against experiments, the institution risks drifting into a closed loop of self-confirmation. Spot checks prevent complacency and provide evidence for envelope validity claims.

The fourth requirement is **incident reconstruction capability**. When a physical system fails or a design proves flawed, the institution must be able to reconstruct the decision chain. This requires audit logs that record: inputs, model version, preprocessing version, diagnostics, envelope status, uncertainty values, constraint checks, fallback decisions, and human approvals. Logs must be structured and retained. Reconstruction must be feasible within operational timelines. "We could reconstruct it if we spent a month searching logs" is not auditability; it is fiction.

The fifth requirement is **traceability across pipeline dependencies**. Surrogate outputs often feed into other systems: optimizers, multi-physics couplers, decision dashboards. Auditability requires that surrogate run IDs propagate downstream so that any decision artifact can be traced back to the surrogate outputs and their diagnostics. This is a chain-of-custody requirement. Without it, the institution cannot answer basic questions: which surrogate outputs influenced this decision, under

what envelope status, with what uncertainty.

The sixth requirement is **monitoring for bypass and control degradation**. Controls can be disabled. Teams can bypass fallback. Logging can be reduced. Governance therefore requires monitoring not only of model performance but of control adherence. For example: track whether constraint checks are running, whether fallback rates change suddenly (indicating bypass or drift), whether uncertainty thresholds were modified, and whether users are invoking "override" modes. This is socio-technical monitoring: it treats humans as part of the system, because they are.

The seventh requirement is **regression testing and model lifecycle audit**. When models are updated, auditability requires the ability to compare versions. The institution should maintain a history of evaluation results, calibration curves, stress test outcomes, and known failure modes per version. This supports two critical governance functions: (i) deciding whether an update is safe, and (ii) explaining, after an incident, whether an update contributed.

Monitoring and auditability are the operational backbone of governable deployment. Without them, controls decay, scope creeps, and liability becomes unbounded.

2.9.5 Deployment deferral criteria

The most mature governance decision is sometimes "not yet." Deployment deferral is not failure; it is disciplined risk management. Because surrogates can produce plausible outputs even when wrong, the threshold for deployment should be higher than "it seems to work." Deferral criteria make that threshold explicit and enforceable.

The first deferral criterion is **inability to define an operating envelope**. If the institution cannot specify where the surrogate is valid—parameter ranges, geometry families, boundary condition scope, horizon limits—then deployment should be deferred for decision-grade uses. The surrogate may still be used for exploratory research, but without an envelope, it cannot be governed. An undefined envelope means the organization cannot meaningfully detect misuse. In high-accountability contexts, this is a hard stop.

The second deferral criterion is **absence of a credible fallback mechanism**. If the pipeline cannot route uncertain or OOD cases to solver verification in a timely and reliable way, the surrogate cannot be safely embedded in decision processes. Without fallback, the organization has no safety valve. In such settings, use should be restricted to exploratory contexts with clear labeling, and decision-grade deployment should be deferred until fallback integration exists.

The third deferral criterion is **unstable or uncalibrated uncertainty**. If uncertainty estimates are used for gating but calibration is poor—high false negatives for silent failure—or calibration drifts unpredictably, then uncertainty gating cannot be trusted. In that case, the institution must either (i) defer deployment, (ii) restrict the envelope dramatically, or (iii) rely on mandatory solver verification for all high-impact uses. The key principle is that uncalibrated uncertainty cannot be

treated as a control.

The fourth deferral criterion is **failure under stress tests at regime boundaries**. If robustness testing reveals frequent silent failures or constraint violations near regime transitions that are operationally relevant, deployment should be deferred or restricted. This is particularly important because regime boundaries are where catastrophic failures concentrate. An institution that deploys despite known boundary fragility is knowingly accepting tail risk.

The fifth deferral criterion is **insufficient traceability and audit infrastructure**. If the organization cannot log inputs, outputs, diagnostics, versions, and approvals, then it cannot reconstruct incidents. In high-accountability environments, that alone is sufficient reason to defer decision-grade deployment. Traceability is not optional; it is part of liability control.

The sixth deferral criterion is **organizational readiness failure**. Even if the model is strong, deployment should be deferred if the organization lacks trained users, clear decision protocols, or enforcement of criticality modes. A surrogate program can fail because of culture and incentives, not because of math. Therefore governance should include readiness checks: have users been trained, are sign-off protocols defined, is there an owner for envelope management, is there authority to enforce gates.

The seventh deferral criterion is **mismatch between surrogate outputs and decision requirements**. Some decisions require evidence that surrogates cannot provide: formal certification, legally defensible safety cases, or highly sensitive physical builds. In such cases, surrogates may still be valuable for exploration, but they should not be used as primary evidence. Deferral here is not about model weakness; it is about evidentiary requirements. Governance requires aligning tools to what decisions demand.

Deployment deferral criteria, when explicit, protect the institution from the most common failure mode in frontier AI adoption: moving faster than the evidence system can support. Physics surrogates are compelling precisely because they are fast. Governance exists to ensure that speed does not outrun accountability. If the institution cannot enforce envelopes, cannot fall back, cannot calibrate uncertainty, cannot log decisions, and cannot reconstruct incidents, then the correct decision is to defer. That decision is not conservative; it is rational.

2.10 Outlook and Open Questions

2.10.1 Near-term research questions

The near-term research agenda for physics and simulation surrogates is not primarily about inventing ever-larger architectures. It is about converting capability into governable reliability. In this domain, the frontier is not "can we predict complex fields?"—that question is increasingly answered in selective settings—but "can we know when the prediction is trustworthy, and can we enforce physical admissibility when the model is used the way organizations actually use it?" The most valuable research directions, therefore, sit at the intersection of uncertainty quantification, out-of-distribution detection, constraint enforcement, and hybrid verification.

The first cluster of research questions concerns **uncertainty quantification (UQ) for fields**. Scalar prediction problems already struggle with calibration; field prediction makes calibration harder because error is structured in space and time, and because the stakes often depend on localized features rather than global norms. Near-term research must focus on UQ methods that produce actionable signals for governance: not merely a variance map, but a calibrated indication of whether key observables and constraint metrics are within acceptable error bounds. This includes developing uncertainty measures that remain meaningful under distribution shift and that can be tied to envelope enforcement. In practical terms, a surrogate program needs a UQ output that can answer: "Should we trust this output enough to proceed without a solver run?" That is a decision-aligned question, and UQ research must be evaluated against it.

Closely related is **OOD detection for high-dimensional physical states**. Traditional OOD methods often rely on density in embedding space or classifier confidence, but physics settings introduce structured novelty: new geometries, new boundary condition patterns, new regimes, and new discretizations. Near-term research should aim at OOD detectors that are *factorized by novelty source*: geometry novelty, parameter novelty, boundary novelty, and discretization novelty. This decomposition is governance-relevant because each novelty type has different remediation. Parameter novelty may be addressed by targeted simulation; geometry novelty may require new encoders; discretization novelty may require architecture changes. OOD research that collapses all novelty into one score may be less useful than research that yields diagnostic, actionable OOD signals.

A third near-term cluster is **stable constraint enforcement under rollout**. Constraint penalties during training are often effective in-distribution and fragile out-of-distribution. Hard projections can enforce constraints but may introduce artifacts that destabilize rollout. What is needed are enforcement mechanisms that remain stable when outputs are iterated, especially in dynamical systems where small errors amplify. This includes research on constrained architectures, invariant-preserving discretizations inside networks, and post-processing methods that correct without destabilizing. The key is not simply to reduce constraint error at a single step; it is to prevent constraint drift over

time, which is precisely where silent failures accumulate.

A fourth cluster concerns **hybrid verification methods that scale**. In high-accountability organizations, the solver remains an authority. But the cost advantage of surrogates is lost if the solver must verify everything. Near-term research should focus on hybrid strategies that reduce solver workload while preserving evidence quality: surrogate warm-starting, residual-based refinement, multi-fidelity verification, and adaptive sampling that concentrates solver calls where uncertainty is highest. The goal is to design verification processes that are proportional to risk: cheap verification for low-risk regions and expensive verification for boundary regions. This is effectively an "evaluation budget allocation" problem, and it echoes earlier chapters in AI 2026: evaluation is the bottleneck, so you must allocate it strategically.

Finally, near-term research should address **decision-aligned evaluation metrics**. Much of the field still optimizes for average error. Governance requires tail-aware metrics, time-to-failure metrics, and decision fidelity metrics in optimization loops. Research that produces new benchmarks and evaluation protocols aligned to stability and tail risk may have more practical impact than research that marginally improves MSE on existing datasets. If the evaluation target changes, the incentives for architecture design change, and the field can move toward more governable surrogates.

2.10.2 Technical unknowns

Even with strong near-term progress, several technical unknowns remain unresolved because they are structural properties of physics and of learned approximation. These unknowns define the boundary between what surrogates can plausibly be trusted for in the next few years and what will remain solver- or experiment-dominated.

The first unknown is **transfer across geometries**. Many surrogates perform well on fixed domains or families of similar geometries. Transfer to substantially new geometries is hard because geometry changes the operator itself: boundary layers form differently, flow separation points move, stress concentrations emerge in new locations. Encoding geometry in a way that supports reliable operator generalization remains an open technical problem. Mesh-based methods, coordinate-based methods, and transformer-like architectures for irregular domains each offer partial solutions, but none provide a general guarantee of stable transfer. For governance, this means geometry scope will remain a major envelope constraint.

The second unknown is **transfer across discretizations and numerical settings**. Real engineering workflows change meshes, resolutions, and solver settings frequently. A surrogate trained on one discretization may fail when the discretization changes, even if the physical conditions are the same. Some architectures aim for discretization invariance, but achieving it robustly—especially in 3D, on complex meshes, and across multi-physics couplings—remains uncertain. This unknown matters because discretization drift is an operational reality. If discretization transfer remains fragile, institutions will need either strict discretization controls or continuous revalidation.

The third unknown is **behavior in extremes and at regime boundaries**. Many systems exhibit qualitative transitions: turbulence onset, shock formation, material yielding, phase change, chemical ignition, or rare-event detector behaviors. These are exactly the regimes where decisions are most consequential and where surrogates are most likely to fail. It remains uncertain how broadly surrogates can be made reliable at such boundaries without extensive training data and specialized architectures. This is not merely a data problem. It is a representation and stability problem: learning discontinuous or sharply sensitive operators is intrinsically hard.

The fourth unknown is **long-horizon robustness**. Rollout stability is a core unresolved issue. Even when surrogates are accurate at short horizons, errors can compound over long rollouts. In chaotic systems, long-horizon prediction is fundamentally limited, but organizations may still need long-horizon statistics or stable attractor properties. It remains unclear what classes of long-horizon properties surrogates can reliably preserve (invariants, statistics, coarse observables) and what classes will remain unreliable. This unknown should push institutions toward conservative horizon envelopes and hybrid correction strategies.

The fifth unknown is **multi-physics coupling**. Many real systems are not single PDEs. They involve coupled processes: fluid-structure interaction, thermo-mechanical coupling, chemical reactions with transport, electromagnetics with thermal effects, or complex detector pipelines. Building surrogates for each subsystem is not enough; the coupled system can amplify errors and create emergent failure modes. It is technically uncertain how to build surrogate systems that remain stable when coupled—especially when each component surrogate has its own uncertainty and bias. Coupling is where "system-of-systems" risk appears in physics, and it will remain a hard frontier.

The sixth unknown is **robust constraint enforcement under compositional use**. Even if a surrogate enforces constraints locally, constraints can be violated when outputs feed into downstream computations, optimizers, or control loops. For example, a surrogate that produces physically plausible fields might still create biased gradients that mislead an optimizer. The unknown here is not simply "constraint satisfaction," but constraint satisfaction under the compositional use patterns that organizations employ. This is a system-level unknown that will likely require system-level evaluation.

These unknowns imply that surrogates will advance unevenly: highly effective in well-characterized regimes and less reliable at boundaries, geometry changes, and coupled extremes. Governance should embrace that unevenness by building envelope-based deployment rather than aiming for universal replacement.

2.10.3 Governance unknowns

The governance unknowns for physics surrogates are not secondary to technical unknowns. They are central, because institutional adoption is constrained by liability, certification, and the ability to defend decisions. Several governance questions remain unresolved because the field lacks standardized

norms.

The first governance unknown is **standards for surrogate certification**. In many industries, numerical solvers and engineering analyses have established certification practices. Surrogates do not. What constitutes acceptable evidence that a surrogate is valid within an envelope? How large must validation sets be? What stress tests are required? What calibration diagnostics are mandatory? Without standards, each organization invents its own practice, and evidence quality becomes inconsistent. Standardization is hard because physics domains vary, but minimal standards—envelope documentation, uncertainty calibration, constraint checks, and drift monitoring—are plausible. The unknown is whether industry and regulators will converge on such standards and how quickly.

The second unknown is **documentation norms and audit artifacts**. In finance, model documentation is extensive because regulators demand it. In engineering, solver documentation exists, but surrogate documentation is not yet standardized. The field needs norms for "surrogate model cards" that include operating envelopes, training provenance, evaluation results, known failure modes, and approved uses. The unknown is how detailed and how prescriptive these artifacts must be for high-accountability deployment, and whether organizations will invest in them before being forced to.

The third unknown is **liability allocation**. When a surrogate contributes to a design that fails, who is responsible? The model builder, the decision owner, the platform team, the vendor, or the institution as a whole? Liability frameworks in engineering typically assume determinism and traceability. Surrogates introduce learned behavior, training data dependencies, and drift. It is unclear how liability will be apportioned, especially when vendor models are used or when surrogate components are integrated into solver toolchains. This uncertainty will slow adoption in regulated contexts unless governance artifacts can create defensible accountability.

The fourth unknown is **audit norms for surrogate-in-the-loop decisions**. Auditors will need to know what evidence to request and how to evaluate it. Should audits examine model weights? Probably not. Should they examine evaluation harnesses, envelope enforcement, and logging? Almost certainly. But norms are immature. The unknown is what audits will look like and what level of transparency will be required, particularly for proprietary surrogates. This is not merely a compliance issue; it shapes market structure. If audits require deep transparency, vendor models may be constrained. If audits focus on system-level controls, vendor models may be more viable.

The fifth unknown is **governance of continuous updating**. Surrogates may be updated frequently as new simulation data becomes available. Continuous improvement is attractive, but it conflicts with certification and stability. How should organizations govern frequent updates? What is the acceptable cadence? How should regression testing be structured? The unknown here is the maturity of lifecycle governance in scientific ML. Until robust lifecycle practices are widespread, institutions may either freeze models (reducing value) or update unsafely (increasing risk).

The governance frontier is therefore not only technical. It is institutional. The chapter's argument is

that value at the frontier is dominated by validation and control. Governance unknowns determine whether that validation can be standardized, audited, and defended.

2.10.4 What would change the assessment

The assessment of physics surrogates in this book is deliberately cautious: surrogates are powerful, but their value depends on evidence and governability. Several developments would materially improve that assessment, not by making surrogates more impressive, but by making them more defensible.

First, **verified operating envelopes with calibrated uncertainty** would change the adoption calculus. If organizations can reliably define envelopes and enforce them with low false negatives—meaning OOD and high-risk cases are rarely missed—then surrogates become safer to use broadly. Calibration here is not cosmetic. It must be tied to operational thresholds and must be monitored over time. A surrogate program that can demonstrate stable calibration under drift and version updates is a fundamentally more governable program.

Second, **benchmarks that emphasize stability and tail risk** would change incentives across the field. If the community's standard evaluation includes long-horizon rollout stability, constraint preservation, time-to-failure distributions, and OOD detection performance, then models will be designed to optimize those properties. Today's benchmark incentives often reward average accuracy and speed. A shift toward governance-relevant benchmarks would accelerate progress on the properties that institutions actually need.

Third, **hybrid verification tooling that scales operationally** would unlock deployment. If toolchains make it easy to integrate surrogates with solver fallbacks, residual-based refinement, and multi-fidelity validation, then organizations can capture speed without sacrificing evidence quality. The critical improvement would be making fallback cheap enough that it is not bypassed and structured enough that it produces audit evidence automatically.

Fourth, **standardized documentation and audit artifacts** would reduce institutional friction. If there are widely accepted templates for surrogate model cards, envelope specifications, calibration reports, and monitoring dashboards, then adoption becomes less bespoke. This is not glamorous research, but it is what makes capability portable and defensible. In high-accountability settings, portability of governance is as important as portability of weights.

Fifth, **evidence of robust transfer across geometry families and discretizations** would expand envelopes. If surrogates can be shown, with credible validation, to transfer across common geometry variations and mesh changes, then their operational usefulness increases dramatically. This is a technical improvement, but its impact is governance: it reduces the number of "scope creep" incidents because the scope is legitimately larger.

Finally, **clear liability and certification pathways** would change institutional posture. If

regulators, standards bodies, or industry consortia define what counts as acceptable surrogate use and how accountability is assigned, organizations can adopt with clearer risk pricing. Without that clarity, adoption remains cautious and uneven.

In short, what would change the assessment is not a single breakthrough model. It is a convergence of calibrated uncertainty, stability-oriented benchmarks, scalable hybrid verification, standardized governance artifacts, and clearer institutional norms. Together, these would move surrogates from "promising but risky" to "strategically deployable."

2.10.5 Link to subsequent chapters

This chapter's position in the book is not accidental. It completes the transition from the scientific application theme of Chapter 6 (chemistry and materials) into a broader thesis that will dominate the remainder of the volume: at the frontier, *validation dominates value*. Physics surrogates make that thesis concrete. They show how a capability that is genuinely transformative—massively accelerated simulation—becomes institutionally dangerous if evaluation, envelopes, and audit trails are weak.

The bridge to Chapter 8 (finance under constraint) is direct. Finance has long lived in a world where models are powerful and failure is expensive. Model risk management, operating limits, stress testing, and auditability are not optional in finance; they are existential. Physics surrogates are increasingly similar. They are models used in decision loops, susceptible to drift, vulnerable to tail risks, and capable of creating unpriced liabilities when trusted blindly. The governance patterns that finance developed—envelopes, escalation rules, independent validation—are precisely the patterns physics surrogate programs must adopt. In this sense, Chapter 7 is the scientific mirror of Chapter 8: different domain, same governance logic.

The bridge to Chapter 10 (multimodal vision–language–action) is also clear. Multimodal systems are system-of-systems technologies: perception feeds into reasoning, which feeds into action. Surrogate simulation systems are already system-of-systems when embedded in optimization loops and coupled pipelines. Both domains share the same governance requirement: model-level evaluation is insufficient. What matters is system-level behavior under rollout, under perturbation, and under real operational constraints. The lesson of physics surrogates—evaluate trajectories, enforce envelopes, build fallbacks, log for reconstruction—will reappear in Chapter 10 as a general governance pattern for AI systems that act in the world.

Finally, this chapter serves as a pivot in the book's narrative tone. Up to this point, the book has built a conceptual ladder: agents amplify risk, reasoning deepens hidden failures, representation engineering introduces fragile controls, and memory architectures create provenance risks. Physics surrogates add a new dimension: the physical world is unforgiving, and when AI is used to approximate it, mistakes are not rhetorical—they can become material. The outlook, therefore, is not that surrogates will be rejected. It is that surrogates will be adopted where governance makes

them defensible, and they will be constrained where governance cannot yet price the risk. That is "frontier awareness without the hype" in its most concrete form: speed is real, but so is the bill when the speed outruns the evidence.

Bibliography

[1] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. *Fourier Neural Operator for Parametric Partial Differential Equations*. International Conference on Learning Representations (ICLR), 2021. arXiv:2010.08895.

[2] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. *Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators*. Nature Machine Intelligence, 3:218–229, 2021. doi:10.1038/s42256-021-00302-5.

[3] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. *Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations*. Journal of Computational Physics, 378:686–707, 2019. doi:10.1016/j.jcp.2018.10.045.

[4] Tobias Kurth, Junqi Yin, Shubham Chhaparia, Markus Jarosik, Brian W. Hall, Heather Nguyen, et al. *FourCastNet: Accelerating Global High-Resolution Weather Forecasting using Adaptive Fourier Neural Operators*. In *Proceedings of the Platform for Advanced Scientific Computing Conference (PASC)*, 2023. doi:10.1145/3592979.3593412.

[5] Rémi Lam, Alvaro Sanchez-Gonzalez, Matthew Willson, Peter Wirnsberger, Meire Fortunato, Ferran Alet, et al. *Learning skillful medium-range global weather forecasting*. Science, 382:1416–1421, 2023. doi:10.1126/science.adi2336.

[6] Kaifeng Bi, Lingxi Xie, Hengheng Zhang, Xin Chen, Xiaotao Gu, and Qi Tian. *Accurate medium-range global weather forecasting with 3D neural networks*. Nature, 619:533–538, 2023. doi:10.1038/s41586-023-06185-3.

[7] Michael Takamoto, Christopher Knobel, Jannis S. B. Berryman, Matthias Wiewel, and others. *PDEBench: An Extensive Benchmark for Scientific Machine Learning*. Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track, 2022. arXiv:2210.07182.

[8] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. *Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles*. Advances in Neural Information Processing Systems (NeurIPS), 2017. arXiv:1612.01474.

[9] Weitang Liu, Xiaoyun Wang, John D. Owens, and Yixuan Li. *Energy-based Out-of-distribution Detection*. Advances in Neural Information Processing Systems (NeurIPS), 2020. arXiv:2010.03759.

[10] ASME. *ASME V&V 20-2009: Standard for Verification and Validation in Computational Fluid Dynamics and Heat Transfer*. American Society of Mechanical Engineers, 2009. (Standard overview page: ASME Codes & Standards).

Chapter 3

Chapter 8 — Finance Under Constraint: Evaluation, Disclosure, and Tail Risk

Abstract. Finance is where AI hype goes to meet its sworn enemy: accountability. The frontier opportunity is real—foundation models can turn unstructured text into structured evidence, accelerate research workflows, and reduce the labor cost of compliance drafting. But the frontier liability is just as real: systems that sound certain can cause outsized losses, trigger disclosure failures, or quietly shift responsibility from humans to machines in ways that regulators and courts will not find charming.

This chapter frames frontier financial AI as a *constrained decision-support system* rather than an autonomous decision-maker. The central design move is not better prediction; it is stronger control. We formalize the pipeline as information I transformed into a representation z, producing bounded recommendations a subject to explicit policy constraints, escalation rules, and auditable evidence trails. We show why accuracy and backtesting are insufficient: governance failures are often not numerical errors but procedural violations—missing disclosures, suitability lapses, provenance loss, and tool-mediated mistakes inside multi-step workflows.

The chapter then translates this framing into practical evaluation and implementation patterns: case-based testing with evidence and constraints; structured outputs separating facts, assumptions, and open questions; deterministic harnesses for repeatability; and decision-dossier logging that supports post-hoc reconstruction. We close with a governance taxonomy tailored to finance—disclosure and suitability risk, privacy and insider-information handling risk, and monoculture risk—and define deployment deferral criteria that treat reviewability and provenance as prerequisites rather than optional "best practices." The goal is simple: capture the value of frontier AI without outsourcing fiduciary duty to a plausible paragraph generator.

3.1 Orientation and Scope

3.1.1 Motivation and context

Finance is the clearest environment in which to see the difference between *capability* and *permission*. Many domains can tolerate a clever-but-wrong paragraph generator for longer than they should; finance cannot. A small numerical error can cascade into a materially incorrect valuation, an erroneous risk limit breach, an inappropriate client communication, a disclosure omission, or a compliance failure that is expensive long after the spreadsheet has been closed. What makes this domain distinctive is not that it is uniquely complex—healthcare and energy are complex too—but that it is uniquely *accountability-dense*. Decisions are constrained not only by physics and economics, but by fiduciary duty, suitability standards, market conduct rules, internal policies, and the simple, unforgiving fact that people lose money in ways that are measurable, litigable, and reputationally durable.

Frontier AI enters this environment with two apparently contradictory promises. The first is productivity: models that can read filings, contracts, transcripts, research notes, and policies; convert unstructured text into structured tables; extract key risk factors; draft internal memos; and provide decision support in workflows that previously required teams of analysts and reviewers. The second is improved judgment: models that can reason across heterogeneous evidence, handle ambiguity, and propose coherent interpretations under time pressure. Both promises contain truth. They also create a trap: the better a system becomes at producing credible narrative and convenient structure, the easier it is for humans to surrender responsibility without noticing that they have done so.

This chapter is motivated by that trap. The central governance problem in AI-driven finance is not that models will occasionally be wrong. The deeper problem is that models can be wrong *in ways that look right*, inside workflows that are difficult to audit. When AI is used as a drafting tool, errors are visible and corrigible; when AI is used as a decision-support system embedded into a pipeline, errors become procedural and systemic. A model that drafts a memo can be corrected; a model that silently determines which evidence is considered, which policy constraints are triggered, which risks are summarized, and which uncertainties are suppressed can change the organization's behavior while leaving a tidy, plausible paper trail.

The shift from analytics to decision support is therefore not cosmetic. Traditional quantitative finance typically separates modeling from authority. A forecasting model produces a signal; a portfolio manager decides; risk and compliance review; and systems enforce limits. Frontier AI blurs those boundaries. A model can ingest a client email, retrieve policy, summarize constraints, recommend a response, generate suitability language, compute exposures via a tool call, and format the output in a client-ready style. The model did not "place a trade," yet it performed much of the cognitive labor that defines the decision, and it did so in a way that can create the *appearance* of

compliance while weakening the underlying discipline of evidence and review.

This is why the chapter's framing emphasizes decision systems under governance constraints rather than "AI in finance" in the generic sense. The focus is on systems that sit near the edge of authority: research copilots, compliance drafting assistants, risk triage tools, reporting systems, and advisory workflow aides. These tools can generate real value precisely because they reduce friction in the highest-cost parts of the organization: reading, summarizing, drafting, cross-referencing, reconciling, and documenting. But they also create new liability surfaces precisely because they operate where rules, policies, and evidentiary standards are not optional. The goal is not to argue that such systems should not exist. The goal is to state clearly the conditions under which they can be deployed without turning "efficiency" into an unpriced transfer of risk.

A final contextual point matters for senior readers: in finance, AI does not merely affect internal productivity; it affects *market behavior*. If many institutions use similar models, trained on similar corpora, with similar retrieval stacks and similar evaluation shortcuts, errors become correlated. This is a systemic issue, not a vendor selection issue. Frontier AI therefore must be understood as both an organizational tool and a market-wide technology. Governance must consider not only whether a single system is safe, but whether the deployment pattern across firms can amplify herding, monoculture risk, and synchronized failure. In short: finance makes AI real because finance makes consequences real, and consequences are where governance stops being a slogan and becomes an operating discipline.

3.1.2 Why this topic is at a frontier moment

The frontier moment in finance is defined by a change in input and a change in output. The change in input is that the most valuable information in modern financial work is often unstructured: narrative disclosures, management commentary, covenant language, legal terms, risk factor phrasing, footnotes, audit opinions, policy documents, client communications, and internal deliberation. For decades, quantitative systems have excelled when the world is already tabular. Yet much of the most consequential information is not tabular until humans make it so. Foundation models shift the boundary of what can be processed automatically: not because they are magical, but because they can transform unstructured material into structured representations at low marginal cost.

The change in output is more important: the outputs are no longer just predictions or scores. Frontier models can participate in workflows, generate drafts, propose actions, and execute tool calls within constrained environments. This is the point where AI becomes execution-adjacent. Even if the system is explicitly forbidden from trading, it can influence the steps that determine whether a trade is considered, how it is documented, what risks are summarized, and what constraints are emphasized. In governance terms, this is a shift from *model risk* to *workflow risk*. Traditional model risk management assumes a relatively stable mapping: inputs $\rightarrow$ outputs, evaluated by metrics and monitored for drift. Workflow systems involve sequences: retrieval, interpretation, reasoning, tool

use, drafting, and interaction with humans. The relevant unit of evaluation is not the output text; it is the trajectory of actions and intermediate states that produced the outcome.

This is why finance sits at the frontier of the book's broader thesis: capability without evaluability is negligence, and evaluability becomes harder as systems act over time. Finance is already a world of checklists, controls, segregation of duties, audit trails, and approvals. It should be the easiest place to govern AI, and yet it is precisely where frontier AI can most easily create the illusion of governance while undermining its substance. A system can generate outputs that look like compliance artifacts without reliably grounding them in correct evidence. It can cite policy without retrieving the correct version. It can produce disclaimers that satisfy style guides while omitting the substantive constraints that matter. The frontier moment is therefore not simply that models are more capable; it is that they are capable in a way that can bypass the spirit of controls while appearing to follow the letter.

At the frontier, the difference between a helpful system and a liability machine is often not the model; it is the *control layer*. When a model is embedded in finance, the most important design question becomes: what are the allowed actions, under what conditions, with what evidence, and with what escalation and logging? In other words, the system must be treated as a constrained decision-support policy. This is the executive-friendly reframing: do not ask whether the model is "smart." Ask whether the system's authority is bounded, whether it is auditable, and whether the organization can reconstruct why a recommendation was produced and which evidence was used.

The frontier moment is also regulatory and institutional. Supervisors and regulators are increasingly attentive to AI-assisted decision processes, particularly in contexts where conflicts of interest, personalization, suitability, and disclosure risks can arise. Firms are simultaneously under pressure to adopt productivity tools and under pressure to demonstrate control. This creates a predictable failure mode: "pilot forever" systems that drift into production-like usage without production-grade governance. The model is present in the workflow, outputs are relied upon, but documentation and evaluation are not built to a standard commensurate with the risk. The frontier moment is therefore a maturity mismatch: frontier capability meets legacy governance. The gap is not merely technical; it is organizational.

Finally, finance is at a frontier moment because of the interaction between AI and incentives. In a domain where speed matters, systems that reduce friction are adopted quickly. But the costs of failure can be delayed and externalized: a bad decision might only manifest when a position is stressed, when a disclosure is challenged, when a client complains, or when a regulator asks for reconstruction. This creates an asymmetric adoption dynamic: benefits are immediate, costs are contingent. Governance exists to correct this asymmetry by requiring evidence, testability, and accountability before scale. The frontier is precisely where institutions are tempted to treat "useful" as "acceptable." This chapter's purpose is to keep those terms separate.

3.1.3 What has changed recently

Three changes have materially altered the practical landscape of AI in finance: improved ingestion across modalities, increased reliance on inference-time reasoning and tool use, and a reframing of risk management from "quant models" to "AI systems." Each matters because it shifts what is feasible and what must be governed.

First, ingestion. Modern models can process text, tables, and hybrid documents more effectively than prior generations of NLP systems, particularly when combined with retrieval and document parsing pipelines. In finance, this means that core artifacts—10-Ks, 10-Qs, earnings call transcripts, covenant packages, term sheets, policy manuals, research notes, and client communications—can be consumed at scale. The significance is not that models "understand" them in a human sense; it is that models can produce structured representations (entities, numbers, claims, risks, obligations) that are useful enough to accelerate workflows. This is a qualitative shift from earlier approaches that required bespoke parsers, narrow ontologies, and high-friction engineering. The barrier to experimentation has fallen, and experimentation is exactly how systems quietly become operational.

Second, inference-time reasoning and tool use. Frontier systems are increasingly deployed not as static predictors but as orchestrators. They can call calculators, query databases, retrieve documents, and iterate over drafts. This creates a new class of failure modes: the model may choose the wrong tool, mis-handle units, misread a retrieved snippet, or propagate an early mistake through subsequent steps. Traditional evaluation that checks only the final output misses these internal errors. More importantly, tool use can create a false aura of rigor. A system that shows a calculation looks disciplined even if the inputs were wrong. A system that cites a policy looks compliant even if the citation was irrelevant or outdated. The governance challenge becomes ensuring that tool calls are logged, verified, and constrained, and that intermediate artifacts are subject to review in the same way final outputs are.

Third, institutional emphasis on extending model risk management. Historically, financial institutions developed robust practices for governing quantitative models: inventory, validation, performance monitoring, change control, and independent review. Those practices remain valuable, but they do not automatically extend to foundation models and agentic workflows. A forecasting model has a clear target variable and a well-defined validation regime; a decision-support system that drafts and synthesizes does not. Its failure modes include hallucinated facts, missing evidence, improper confidence, and procedural violations. Its outputs are often narrative and context-dependent. This forces a shift: evaluation must become case-based and governance-aware, with explicit measurement of policy violations, provenance integrity, escalation correctness, and reviewability. In practice, this means building evaluation harnesses that look more like audit workpapers than Kaggle leaderboards.

A related change is the increasing availability of long-context systems and retrieval-augmented generation. This improves coverage but introduces subtle risk: more context does not equal better evidence. It can dilute relevance, obscure provenance, and create overconfidence. In finance, where

decisions must be justified, the relevant question is not how much the model can ingest, but what the model chooses to treat as evidence. Retrieval shifts the burden from "model correctness" to "retrieval correctness," which is often less visible and less tested. A system can be accurate on average while failing precisely on the cases that matter: high-stakes edge cases with ambiguous documents and conflicting constraints.

Another practical change is organizational: deployment pathways are faster. Teams can adopt AI tools through SaaS products, browser plugins, internal APIs, and workflow integrations. This ease of adoption interacts with weak governance in predictable ways. A tool begins as "draft-only," then becomes "draft plus suggestion," then becomes "draft plus recommended next step," and eventually becomes "draft plus automated filing template." The authority grows by accretion. The frontier is where authority is accumulated without a single explicit decision to grant it. For finance, this is dangerous because accountability demands explicit decisions, explicit constraints, and explicit auditability.

Finally, the discourse has shifted. There is growing recognition—inside and outside institutions—that narrative competence is not the same as evidentiary correctness. This is a crucial cultural change. It creates space for governance-first design: structured outputs that separate facts, assumptions, and open questions; uncertainty statements that are calibrated and meaningful; and control layers that enforce scope boundaries. The frontier moment is therefore also a pedagogical moment: institutions must learn to treat LLM outputs as drafts and hypotheses, not conclusions, and must build processes that make that discipline operational rather than aspirational.

3.1.4 Explicit exclusions and non-goals

This chapter is deliberately not a guide to trading systems, not a handbook for alpha generation, and not a set of instructions for market exploitation. The book's purpose is frontier awareness without the hype, and in finance the fastest route to hype is to confuse "AI can produce a plausible decision" with "AI can responsibly own the decision." We therefore exclude any instructions or patterns intended to evade regulation, manipulate markets, exploit operational vulnerabilities, or circumvent compliance controls. We also exclude any suggestion that frontier AI provides reliable, generalizable predictive advantage, particularly because many of the most common claims in this direction are artifacts of weak evaluation and survivorship bias.

The chapter also does not attempt to provide legal advice, regulatory interpretations, or jurisdiction-specific compliance guidance. Finance is regulated differently across regions and products, and the correct posture for an executive-level governance book is to emphasize process and control design rather than to pretend to be a regulatory manual. Where we discuss disclosure, suitability, and fiduciary concepts, the intent is to identify risk surfaces and governance requirements, not to adjudicate legal standards. Institutions must consult qualified legal and compliance professionals when designing or deploying systems that touch client-facing outputs or decision authority.

A further exclusion is the temptation to treat "guardrails" as a substitute for governance. Content filters, refusal behaviors, and prompt-level constraints can be helpful, but they are not sufficient. They are often brittle, easily bypassed, and difficult to audit. A governance-first system requires structural controls: defined authority, logged evidence, escalation rules, change control, and independent review. This chapter therefore does not present prompt tricks as compliance solutions. It treats prompt engineering as an operational detail within a broader control regime.

We also do not claim that long context, retrieval, or tool use automatically produces truth. These mechanisms can improve performance and reduce some error types, but they introduce their own failure modes and can increase the risk of misplaced confidence. In finance, the standard is not "better than before"; it is "defensible under scrutiny." The non-goal is therefore to optimize for convenience or narrative quality. The goal is to design systems that remain reviewable, auditable, and appropriately humble about uncertainty.

Another explicit non-goal is the replacement of human judgment. In high-accountability financial settings, the most valuable human work is not typing; it is deciding what is relevant, what is missing, what is uncertain, and what must be escalated. Frontier AI can accelerate the mechanical parts of cognition—summarization, extraction, drafting, cross-referencing—but it cannot be allowed to launder decisions. This chapter therefore treats the human-in-the-loop not as a ceremonial checkbox but as an accountable actor with defined responsibilities. Systems should be designed to make human review easier and more accurate, not to make it psychologically effortless to approve.

Finally, we do not aim to settle open research questions about reasoning, calibration, or interpretability. The chapter is pragmatic: it identifies what executives and governance leaders can do today, with existing tools, while acknowledging that certain technical unknowns remain unresolved. The non-goal is to present a complete technical survey. The goal is to provide a controlled conceptual model and an operational vocabulary for making decisions about AI adoption in finance without mistaking novelty for readiness.

3.1.5 Role of this chapter in AI 2026

Within the book's arc, Chapter 8 serves as the point where the earlier frontier themes—trajectory evaluation, reasoning risk, representation control, memory and retrieval, and planning—are forced into operational discipline by a domain that cannot tolerate ambiguity about responsibility. Chapters 1 through 5 establish the core frontier claim: modern AI systems become dangerous when they act over time, and governance must follow the trajectory, not the output. Finance is where that claim becomes a daily operating constraint. If you cannot reconstruct what a system did, why it did it, and what evidence it used, you cannot safely deploy it in workflows that touch client outcomes, disclosures, or regulated decisions.

This chapter therefore translates the book's research frontiers into a regulated, high-accountability setting. It shows that the value proposition of frontier AI in finance is not autonomous optimization;

it is disciplined decision support. The boundary is crucial. Autonomous optimization is seductive because it promises to convert computation into profit. It is also where liability explodes because autonomy makes it harder to maintain evidentiary standards and human accountability. Decision support, by contrast, can create substantial value by accelerating the work that surrounds decisions: assembling evidence, structuring arguments, drafting documentation, and flagging risks. The chapter's role is to argue that this is not a compromise; it is the realistic frontier.

Chapter 8 also acts as a bridge between "frontier research risks" and "frontier domain risks." In earlier chapters, we discuss reasoning depth as a risk surface, memory as a governance choice, and planning as an amplifier of objectives. Finance provides concrete instantiations: overconfident narratives that resemble research; retrieval stacks that silently privilege some sources over others; planning loops that optimize for speed or client satisfaction at the expense of suitability; and tool-using systems that make calculation errors look like rigor. By grounding these mechanisms in finance, the chapter gives executives a way to recognize the same structural risks when they appear in other domains.

Another role of the chapter is to redefine evaluation for senior stakeholders. Finance teams are accustomed to metrics and backtests. The chapter argues that for frontier AI decision-support systems, evaluation must be case-based, governance-aware, and dossier-driven. The relevant questions become: Did the system separate facts from assumptions? Did it retrieve the correct policy? Did it cite evidence accurately? Did it escalate when evidence was missing or the decision was high impact? Can the organization replay the exact inputs, retrieval results, tool calls, and outputs? These are not academic questions; they are operational prerequisites for defensible deployment. In a sense, the chapter imports audit discipline into AI evaluation.

The chapter also provides a template for institutional design. It emphasizes roles and boundaries: builders versus validators, business owners versus approvers, and the segregation of duties required to prevent decision laundering. It highlights that governance is not only a set of rules but a set of *interfaces*: how systems present uncertainty, how they display evidence, how they trigger escalation, and how they record logs. In finance, poor interface design is a risk control failure. A system that produces polished outputs without showing evidence invites over-trust. A system that cannot explain what it used and what it ignored invites after-the-fact rationalization.

Finally, Chapter 8 prepares the reader for Chapter 9 on interpretive AI. Finance is not only numbers; it is narrative, policy, and interpretation. The discipline required in finance—provenance, citation integrity, claim tracking, and explicit uncertainty—becomes even more essential in knowledge work domains where truth is less directly observable. By showing how evidentiary discipline functions in a high-stakes setting, the chapter equips the reader to recognize and govern epistemic drift in interpretive domains. In the book's structure, finance is the turning point: a domain where frontier capability is valuable, but only when it is constrained, auditable, and subordinate to accountable human judgment.

3.2 Conceptual Abstraction

3.2.1 Core abstraction

The core abstraction of this chapter is deliberately simple: frontier AI in finance should be modeled as a *constrained decision-support system*, not as an autonomous decision-maker. That is not a rhetorical preference; it is a design constraint imposed by governance reality. In a regulated, fiduciary, and litigation-prone domain, the question is never merely whether a system can produce an answer. The question is whether the system can produce a *defensible* answer: one that is grounded in identified evidence, conforms to policies and rules, expresses uncertainty appropriately, and routes high-impact cases to accountable humans with clear escalation.

We can express the system at a high level as a mapping

$$I \longrightarrow z \longrightarrow a,$$

where I denotes inputs (documents, tables, notes, internal policies, synthetic market context in demonstrations), z denotes an internal representation (structured facts, extracted entities, normalized quantities, risk flags, retrieved policy clauses, uncertainty annotations), and a denotes an output action proposal (a recommendation, a draft, a risk summary, a checklist, an escalation, or a refusal). The crucial term is not shown in the arrow, because it is the governance layer: the mapping is subject to *policy constraints* that define what actions are allowed, what claims require evidence, what must be escalated, what must be refused, and what must be logged.

This abstraction matters because it makes explicit a boundary that organizations often leave implicit: who has authority? Frontier models are useful precisely because they can *simulate* authority by producing coherent, confident, and procedurally plausible outputs. But in finance, simulated authority is dangerous. The system must be architected so that the model's role is bounded to decision support, not decision ownership. This boundary can be expressed concretely through constraints on the action space a. If a includes "execute trade" or "advise client to buy security X," the system is in a different risk class than if a includes "draft internal summary," "extract covenant terms," "flag missing disclosures," or "prepare decision dossier for human review." The conceptual abstraction forces the organization to define a in governance terms rather than in product terms.

A further conceptual move is to treat the representation z as a first-class governance artifact. In many deployments, the representation is treated as an internal model state—opaque and transient. In finance, that is precisely what cannot happen. The representation must be made partially explicit: what facts were extracted, from which sources, with what confidence; what assumptions were introduced; what was not found; what policies were retrieved; what tools were invoked; what calculations were performed; and what uncertainties remain. In other words, z is not only a computational intermediary; it is a *dossier-in-progress*. This is how a system becomes auditable.

Finally, the abstraction emphasizes that the most important output is often not the recommendation itself but the *control decision*: whether to proceed, whether to escalate, whether to refuse, and whether to request additional evidence. The system's "intelligence" is therefore measured not by its boldness but by its restraint. A model that always answers is not helpful; it is reckless. A system that reliably distinguishes low-stakes drafting from high-stakes decision support—and escalates accordingly—is what turns frontier capability into institutional value.

3.2.2 Key entities and interactions

To govern a decision-support system, one must name its entities and make their interactions explicit. In finance, ambiguity about roles is itself a control failure. A typical frontier AI decision-support system involves at least six entities: (i) sources of information, (ii) retrieval and parsing components, (iii) a reasoning or generation model, (iv) auxiliary tools (calculators, databases, validators), (v) a constraint and policy layer, and (vi) human roles with explicit approval authority. A seventh entity is often neglected but is non-negotiable: (vii) the audit trail, which must be treated as a product artifact rather than an engineering afterthought.

Information sources include market data, internal research notes, filings, contracts, transcripts, emails, policy manuals, risk limits, and historical decisions. In a pedagogical notebook, these must be synthetic; in real deployment, they are heterogeneous, messy, and unevenly reliable. Governance begins here: sources must be classified by sensitivity, provenance, and allowed use. Not all information is permissible for all tasks. Some sources carry confidentiality risk; others carry insider-information risk; others carry licensing and usage constraints. A system that cannot track source classes cannot be trusted to operate near client-facing outputs.

Retrieval and parsing components determine what the model sees. This is where many governance failures occur because retrieval is often treated as a technical optimization problem rather than a decision about evidence. Retrieval chooses which documents are relevant; parsing chooses which segments are extracted; and chunking determines what context is preserved. These choices shape the model's output as much as the model's parameters do. In a governed system, retrieval decisions must be logged, and the system must be able to show which evidence was retrieved and which was not. Otherwise, the organization cannot distinguish a model error from an evidence selection error.

The model performs interpretation, summarization, reasoning, and drafting. Its role should be defined as "evidence-conditioned synthesis under constraints." This definition matters because it rejects a common failure mode: allowing the model to act as an authority rather than as a synthesizer. The model's outputs should be structured where possible, separating facts, assumptions, and open questions. Free-form narrative is not prohibited, but it must be anchored to identifiable evidence and bounded by explicit uncertainty.

Tools are used to reduce error in computation and verification. Examples include calculators for exposures, parsers for tables, checkers for numeric consistency, and validators that enforce

schema constraints. Tools are not automatically safer than pure language generation; they can propagate wrong inputs with high confidence. But tools offer two governance advantages: they can be deterministic and they can be logged. A system should prefer tool-based computation for quantities and allow narrative generation primarily for interpretation and drafting.

The policy and constraint layer is the conceptual heart of the system. It defines what can be produced, what cannot, and what must be escalated. Constraints include scope limits (e.g., no personalized investment advice), disclosure requirements (e.g., required caveats or mandatory risk statements for certain contexts), suitability boundaries (e.g., when client profile is incomplete), and procedural rules (e.g., human sign-off for client-facing communications). Importantly, constraints must apply not only to final outputs but to intermediate states: retrieval, tool calls, and the kinds of inferences the model is permitted to make.

Human roles must be explicit. In a governed workflow, the model builder is not the approver; the approver is not the business owner; and the business owner is not the risk owner. Segregation of duties exists for a reason. When AI is introduced, roles blur because outputs become easier to produce. A system must support role clarity: who reviews evidence, who approves client-facing language, who signs off on risk implications, and who owns the decision in the event of failure.

The audit trail is the final entity and the one that turns systems into defensible processes. The audit trail should record: input identifiers, retrieval results, policy versions, prompts or instructions used, tool calls and outputs, intermediate structured representations, final outputs, and human approvals. Without this, the organization cannot reconstruct incidents, cannot learn from failures, and cannot demonstrate governance to internal or external stakeholders.

These entities interact through a loop that can be described operationally as:

$$\text{ingest} \rightarrow \text{interpret} \rightarrow \text{evaluate} \rightarrow \text{recommend} \rightarrow \text{review} \rightarrow \text{log}.$$

The loop is not merely a workflow; it is the unit of accountability. Each arrow must be governed. "Interpret" requires evidence discipline. "Evaluate" requires explicit constraints. "Recommend" requires bounded action space. "Review" requires human authority and interface design that supports skepticism. "Log" requires immutability and reconstruction. If any link is weak, the system will eventually find it, because frontier AI systems are optimization engines in disguise: they optimize for plausibility and completion unless constrained otherwise.

3.2.3 What is being optimized or controlled

In finance, it is tempting to describe AI systems as optimizing profit. That temptation is understandable and often misleading. The correct description is that such systems optimize a composite of utility terms under constraints—and that many of those terms are institutional rather than financial. The most valuable AI in finance often optimizes *time-to-decision*, *quality of documentation*, *coverage*

of evidence, and *consistency of process*. Profit is downstream, contingent, and frequently dominated by risk and compliance factors.

We can write a multi-objective utility function for a decision-support system as

$$U(a; z) = \lambda_1 U_{\text{decision}}(a; z) \; + \; \lambda_2 U_{\text{risk}}(a; z) \; + \; \lambda_3 U_{\text{ops}}(a; z) \; + \; \lambda_4 U_{\text{governance}}(a; z),$$

where U_{decision} captures decision usefulness (clarity, relevance, coherence), U_{risk} captures risk-awareness (flagging tail risks, identifying exposures, highlighting missing data), U_{ops} captures operational efficiency (reduced manual effort, consistent formatting, faster triage), and $U_{\text{governance}}$ captures compliance and accountability (provenance integrity, policy adherence, appropriate escalation). The λ_i weights encode institutional priorities. A governance-first organization assigns meaningful weight to $U_{\text{governance}}$ even when it reduces short-term throughput, because the cost of failure is not linear.

The crucial point is that constraints are not merely "penalties" in the objective; they are hard boundaries. A governed system must enforce constraints such as:

$$\text{Policy}(a, z) \leq 0,$$

where Policy can represent a vector of constraints: scope boundaries, disclosure requirements, suitability checks, data privacy restrictions, and escalation rules. Some constraints are absolute (e.g., "do not provide personalized investment advice" in certain contexts). Others are conditional (e.g., "if confidence is below threshold, escalate," or "if evidence is missing, request information rather than conclude").

What, then, is being controlled? The system's control objective is best described as *boundedness*. Boundedness has three dimensions.

Bounded outputs. Outputs must be constrained in content and form: structured schemas that separate facts, assumptions, and open questions; explicit citations or evidence references where feasible; and careful language about uncertainty. The system should avoid categorical claims when evidence is incomplete. It should avoid prescriptive instructions when authority is not granted.

Bounded authority. The system's action space must exclude autonomous decisions in sensitive contexts. Even when the system provides a recommendation, it must do so in a way that reinforces human responsibility rather than eroding it. This is not achieved by disclaimers alone. It is achieved by workflow design: requiring sign-off, forcing evidence display, and making escalation the default for high-impact cases.

Bounded uncertainty. The system must not only express uncertainty but do so meaningfully. In finance, "I may be wrong" is not a control. The system needs calibrated confidence signals tied to evidence quality, not to rhetorical hedging. When uncertainty is high, the system should escalate or request more evidence rather than produce a confident narrative.

A governance-first control objective can be summarized as: *maximize decision usefulness subject to constraints, while minimizing governance violations and maintaining auditability.* This statement appears modest. In practice, it is demanding. It requires building systems that are evaluated not only on helpfulness but on compliance behavior under adversarial pressure, on provenance integrity under document drift, and on escalation correctness under uncertainty.

Finally, it is essential to recognize that frontier AI systems also optimize *human behavior.* This is rarely acknowledged explicitly, yet it is central to governance. A system that produces polished outputs reduces the psychological friction of approval. It can cause humans to accept outputs they would otherwise scrutinize. In that sense, the system is optimizing for reduced resistance. Governance must counteract this by designing interfaces and processes that restore appropriate friction: forcing evidence review, requiring explicit confirmation of assumptions, and logging approvals with accountability.

3.2.4 Distinction from prior paradigms

The contrast with traditional quantitative finance is not merely that frontier AI handles unstructured data. The deeper difference is that traditional paradigms are *narrow, explicit, and measurable*, while frontier AI systems are *broad, implicit, and procedurally complex.* This difference changes how risk must be governed.

Traditional quant systems typically involve structured inputs, explicit model forms, and well-defined outputs. Even when the model is complex, it is usually tied to a numeric target: forecast returns, estimate volatility, compute value-at-risk, score credit risk. Evaluation is correspondingly metric-driven: backtests, out-of-sample performance, stress tests, and scenario analysis. Governance frameworks evolved around this structure. Model risk management inventories models, validates them against known targets, monitors drift, and enforces change control. The system boundary is clear: the model is a component, and decision authority remains with humans and with pre-existing control functions.

Frontier AI in finance breaks this clarity in at least four ways.

First, the tasks are broader. A system may read documents, summarize risks, draft communications, and propose next actions. The output is not a single number but a composite artifact: narrative plus structured claims plus references plus recommendations. The evaluation target is not obvious. There is no single ground truth for “best memo” or “best compliance summary.” As a result, organizations are tempted to use proxy metrics: user satisfaction, time saved, or superficial correctness on easy cases. These proxies systematically underestimate tail risk.

Second, the reasoning is implicit. In traditional models, assumptions are often explicit: factor exposures, parameter estimates, scenario definitions. In frontier systems, assumptions can be latent: the model may infer missing context, interpret ambiguous language, or fill gaps with plausible

content. This is dangerous in finance because plausible is not defensible. A system that silently invents a missing covenant threshold is not "helpful"; it is a liability generator. The governance response must be to force assumption disclosure and to separate extracted facts from inferred content.

Third, the workflow is multi-step and tool-mediated. A system might retrieve policy, call a calculator, and iterate drafts. Each step is a potential point of failure. Traditional governance often focuses on the model component; frontier governance must focus on the sequence. This is why trajectory-level evaluation from earlier chapters becomes operational here. A single correct output does not imply a reliable process; it may be a lucky outcome produced by an unreliable path.

Fourth, frontier AI is interactive. Users can steer the system through prompts, context selection, and iterative refinement. This creates a socio-technical risk surface: the system's behavior depends on user behavior, and users vary. In traditional models, user interaction is limited; in frontier systems, interaction is central. This means governance must include interface design, training, and policy enforcement at the interaction layer. A system that can be coaxed into prohibited advice is not governed merely because it "usually" refuses.

The conceptual distinction can be stated crisply: traditional quant paradigms treat risk as primarily *statistical*—error distributions, model misspecification, regime shifts. Frontier AI introduces risk that is *procedural and epistemic*—errors in evidence selection, hallucinated claims, missing disclosures, persuasive narratives without grounding, and drift of responsibility from humans to systems. Statistical risk is familiar and governable. Procedural and epistemic risk require different controls: provenance, logging, structured outputs, escalation rules, and review processes.

This distinction also clarifies why "MRM for AI" cannot be a rebranding exercise. It must extend the governance object. The object is no longer "the model" but "the system that produces decisions or decision-adjacent artifacts." That includes retrieval, tools, prompts, policies, user interfaces, and human approvals. The goal is not to create a new bureaucracy; it is to match governance to the true source of risk.

3.2.5 Conceptual failure modes

A governance-first conceptual abstraction must name failure modes not as anecdotes but as structural patterns. Frontier AI in finance fails in ways that are predictable given the incentives and the nature of language models. The most dangerous failures are not spectacular "model goes rogue" stories; they are mundane, plausible, and procedurally hidden.

Narrative persuasion mistaken for evidence. Language models are optimized to produce coherent text. Coherence is psychologically compelling, especially under time pressure. In finance, where memos and summaries are central, coherence can be mistaken for correctness. The model can produce a compelling explanation for why a credit spread widened, cite plausible macro factors,

and weave a narrative that feels right—while missing the actual driver in the evidence. This is an epistemic failure: the organization mistakes rhetorical quality for evidentiary grounding. The control response is evidentiary discipline: claims must be tied to sources; uncertainty must be explicit; missing evidence must be highlighted; and the system must be designed to prefer "open question" over invented explanation.

Provenance failures and leakage. Retrieval and memory systems can incorporate stale, irrelevant, or unauthorized content. A system might retrieve an outdated policy clause, a superseded risk limit, or a draft contract rather than the executed version. It might also accidentally expose sensitive information through summarization. Provenance failures are often invisible because the output looks fine. The control response is source governance: classify sources, enforce access controls, log retrieval results, and require evidence display for high-impact outputs.

Decision laundering. This is perhaps the central institutional failure mode. Decision laundering occurs when humans treat model outputs as a justification rather than as a draft. The system produces a recommendation with plausible reasoning; the human approves it with minimal scrutiny; responsibility becomes diffused. Later, when challenged, the organization points to the model output as if it were an external authority. This is a governance failure because accountability must remain human. Controls must therefore be designed to prevent passive approval: require explicit sign-off, require acknowledgement of assumptions, and ensure that the audit trail identifies the human decision-maker and the evidence they reviewed.

Automation bias and over-trust. Even without explicit decision laundering, humans tend to over-trust automated systems, especially when systems appear confident and consistent. Over-trust is amplified when outputs are polished, when time pressure is high, and when the organization rewards speed. This is not a flaw in individual users; it is a predictable human factor. Governance must incorporate countermeasures: interface cues that communicate uncertainty, forced review steps, and training that emphasizes skepticism. The system should not be designed to maximize user delight; it should be designed to maximize defensible correctness.

Tail-risk amplification through workflow automation. Many errors are tolerable at low stakes but catastrophic at high stakes. Frontier systems can amplify tail risk by scaling small errors across many cases quickly. A minor misinterpretation of a disclosure requirement might propagate across hundreds of drafted communications. A subtle retrieval bug might cause systematic omission of a risk factor. The danger is scale. Controls must therefore include monitoring for systematic drift, periodic audits of outputs, and stress testing on adversarial cases designed to reveal rare but catastrophic failures.

Scope creep and hidden authority. A system introduced for internal drafting can quietly become client-facing. A model used to summarize can become a recommender. A recommender can become a quasi-approver when humans begin to rubber-stamp. Authority grows by usage pattern rather than by policy decision. This is a governance failure because risk class changes without

explicit approval. Controls must include clear scope definitions, technical enforcement of usage boundaries, and governance gates for changes in deployment context.

Illusory compliance. Systems can generate compliance-flavored language without satisfying compliance substance. They can produce disclaimers that sound right but omit required elements. They can cite policies without actually applying them. They can create the appearance of suitability analysis without incorporating the necessary client facts. This failure mode is dangerous because it produces artifacts that are plausibly defensible until scrutinized. Governance must therefore require checkable structures: explicit fields for required disclosures, explicit lists of suitability inputs used, and validators that enforce completeness.

Tool-mediated errors disguised as rigor. A tool call can create unwarranted confidence. A system that outputs a table of exposures appears more reliable than one that outputs prose, even if the exposures were computed with wrong assumptions. Tool errors are particularly dangerous because they are systematic and can be replicated across cases. Controls must include unit checks, consistency checks, and independent verification for high-impact calculations.

These failure modes share a theme: the model's greatest risk is not that it is malicious, but that it is *plausible*. In a domain where plausibility is currency, plausibility without grounding is counterfeiting. The conceptual abstraction of this chapter is designed to prevent the institution from being fooled by its own productivity gains. Constrained decision support is not a lesser ambition; it is the only ambition that aligns frontier capability with fiduciary reality.

3.3 Historical and Technical Lineage

3.3.1 Preceding approaches

Before frontier AI, "AI in finance" largely meant one of two things: statistical models applied to structured data, or rules-based systems applied to procedural compliance. Both families were imperfect, sometimes brittle, and often over-trusted, but they shared one property that made them governable: their scope was typically narrow enough that humans could understand what the system was meant to do, and validators could define what success and failure looked like.

On the quantitative side, factor models and time-series forecasting dominated institutional practice for decades. Factor models provided a language for risk decomposition and portfolio construction: exposures to broad drivers, idiosyncratic components, and measurable sensitivities. Forecasting models—ranging from simple autoregressive baselines to more sophisticated econometric and machine learning variants—sought to predict returns, volatility, default probabilities, or liquidity measures. Classical risk models, including scenario analysis and value-at-risk families, were never perfect representations of reality, but they were operationally useful because they were explicit: they could be explained as assumptions, calibrated to known data, stress tested, and monitored.

A crucial institutional point is that these models existed inside a broader machinery of controls. A model's output rarely became a decision without mediation. Risk limits, stop-loss rules, approvals, independent model validation, and oversight committees provided friction. Even in fast-paced environments, there was a separation—at least in principle—between model generation and decision authority. The model proposed; humans and systems constrained.

On the compliance side, rules-based systems and expert workflows provided procedural discipline. Institutions built checklists, decision trees, and policy engines to enforce requirements: suitability constraints, disclosure templates, client categorization, KYC/AML procedures, and reporting obligations. These systems were sometimes crude and often expensive to maintain, but they were legible. If a rule fired, someone could see why. If a checklist step was missed, an auditor could identify the gap. In high-accountability settings, legibility is a form of safety.

In parallel, the industry adopted early natural language processing in constrained ways: extracting entities from filings, classifying news sentiment, mapping topics in transcripts, or searching documents for clauses. These systems were generally narrow, trained for specific tasks, and evaluated on metrics that could be measured. Even when they were imperfect, their failures were often in the category of "missed extraction" rather than "invented claim." That difference matters: omission is often safer than fabrication, because omission triggers human work; fabrication can pass unnoticed.

This is not nostalgia. Many preceding approaches failed spectacularly during regime shifts, suffered from data leakage, and produced false confidence. But their narrowness provided an advantage: institutions could build governance regimes around them. Model risk management emerged precisely because structured models, even when wrong, were wrong in ways that could be analyzed statistically

and monitored. The older world was not safe; it was merely governable enough that large institutions could operate with bounded surprises.

3.3.2 Key inflection points

The historical transition from traditional approaches to frontier AI is not a single leap; it is a sequence of inflection points that progressively lowered friction and expanded scope. Each inflection point added capability, but also introduced a new kind of risk surface. Understanding that lineage matters because governance failures often arise when institutions apply the wrong historical intuition to a new technological regime.

The first inflection point was the expansion of machine learning into core financial decision domains where structured labels existed: credit risk, fraud detection, and operational risk scoring. These systems introduced non-linear models, larger feature sets, and often improved predictive performance. They also introduced opacity, complex interactions, and feedback loops (for example, when model decisions changed the data distribution). Institutions responded by extending model validation practices: challenger models, fairness considerations in some contexts, and stronger change control. The key lesson was that better performance did not remove governance needs; it increased them. When a model has more influence, the cost of error grows.

The second inflection point was the rise of scalable NLP for finance. As digital disclosures expanded, transcripts became ubiquitous, and news became instantly machine-readable, institutions increasingly sought to mine text. Early finance NLP was often brittle—dictionary methods, simple classifiers—but it opened a new frontier: extracting signals from narrative. This was operationally valuable because narrative contains leading indicators of risk and intent that numbers alone do not capture. However, it also introduced interpretive risk. Text is ambiguous, strategic, and context-dependent. Institutions learned to treat NLP outputs as weak signals, not as ground truth, and to be cautious about overfitting to the style of disclosures.

The third inflection point was the maturation of retrieval and document search technologies in enterprise settings. Indexing, semantic search, and knowledge management systems improved, enabling faster access to internal and external corpora. This mattered because it transformed information availability from a human bottleneck to a system feature. But it also created provenance risk: the organization began to rely on what the system retrieved, and retrieval systems can be wrong in systematic ways. Importantly, retrieval failures are often silent. A missing document does not announce itself; it merely disappears. This foreshadows a central governance theme for frontier AI: if you cannot reconstruct what evidence was visible, you cannot defend the decision.

The fourth inflection point, and the one most relevant to this book, was the emergence of foundation models that can generalize across tasks and domains, and that can transform unstructured input into structured output with minimal task-specific engineering. This does not mean they are reliably correct; it means they are broadly capable. In finance, this expands the feasible set dramatically:

summarizing filings, extracting terms, drafting memos, generating compliance language, interpreting policy, and synthesizing multi-document narratives. It also collapses the old boundary between analysis and communication. Previously, analysis required structured modeling and communication required human writing. Frontier models can do both, which makes them attractive in workflows where narrative and numbers must align.

The fifth inflection point is agentic and tool-using behavior: systems that do not merely produce text but call tools, iterate, search, and plan. This turns models into workflow orchestrators. For finance, this is where "copilot" can become "quasi-operator" even without explicit authority to execute trades. A tool-using model can compute exposures, check limits, draft client communications, and propose actions in a sequence that looks like a competent analyst's work. The risk is that this competence is simulated and may break under stress, and that errors in early steps can compound across the trajectory.

A final inflection point is organizational rather than technical: the democratization of deployment. Low-friction APIs, SaaS integrations, and internal "AI platforms" allow teams to embed models into workflows rapidly. This shifts the bottleneck from engineering to governance. The institution can adopt faster than it can evaluate. That mismatch is the hallmark of a frontier moment: capability accelerates, but evaluability lags. The result is a proliferation of systems that are useful enough to be used, but insufficiently governed to be defensible.

3.3.3 What persisted vs what broke

The correct way to interpret frontier AI in finance is not as a replacement of prior paradigms, but as a disruption of certain assumptions while leaving core institutional requirements intact. Many executives underestimate this because technological narratives often focus on novelty. Governance begins by identifying what did not change.

What persisted is the simple fact that accountability does not move. A firm remains responsible for its disclosures, communications, and decisions regardless of whether an AI system helped draft them. Fiduciary duties do not become optional because outputs were generated quickly. Regulators and courts do not accept "the model said so" as an excuse. Internal control functions—risk, compliance, audit—still have to be able to reconstruct how an outcome occurred. This is the persistent backbone: auditability, accountability, and control are not legacy burdens; they are the price of operating in a domain where harm is measurable and responsibility is assignable.

Another persistent element is that finance is fundamentally adversarial and incentive-driven. Market participants optimize. Clients have interests. Firms have conflicts. Any system that influences decisions can be gamed, pressured, or misused. This is why governance cannot assume benign use. A model that drafts communications can be prompted to minimize risk language. A system that summarizes policy can be nudged to omit constraints. A retrieval system can be manipulated through document placement and phrasing. The adversarial nature of finance persists, and it means

that safety must be robust to pressure, not merely to average-case usage.

What broke are several implicit assumptions that traditional governance frameworks relied upon. The first is the assumption that financial AI must be narrow and structured. Traditional model risk management works best when the model has a clear purpose, a measurable target, and a bounded output. Frontier AI systems are often not like that. They generate narratives, synthesize evidence, and participate in workflows. Their success criteria are multi-dimensional and context-dependent. This does not mean governance is impossible; it means governance must evolve from metric validation to case-based, scenario-driven evaluation.

The second broken assumption is that models can be evaluated primarily by output accuracy. In frontier systems, an output can be accurate for the wrong reasons or inaccurate in ways that are not captured by metrics. The critical failure modes involve process: missing evidence, incorrect citations, policy violations, unsuitable recommendations, and inadequate escalation. These are not "accuracy" errors in the traditional sense. They are governance errors. A system can score well on standard benchmarks and still be unacceptable in client-facing workflows because it fails precisely where accountability is highest.

The third broken assumption is that the system boundary is clear. In traditional paradigms, the model is a component embedded in a known pipeline. In frontier AI, the "model" often includes retrieval, memory, prompts, tools, and user interaction. The system is socio-technical. Governance cannot focus only on the weights; it must govern the entire decision-support apparatus. This is why the book repeatedly emphasizes system-level evaluation rather than model-level worship.

The fourth broken assumption is that human review is automatically effective. In legacy workflows, humans review documents they wrote or analyses they produced. With AI, humans review outputs that may be polished and plausible but not grounded. This shifts the cognitive burden: reviewers must check evidence, not just readability. Without interface support and process redesign, "human-in-the-loop" becomes "human rubber-stamp." The presence of a human does not guarantee safety; the design of the review process does.

In short, what persisted is the requirement for accountability and evidence. What broke is the comfortable alignment between model evaluation and institutional defensibility. Frontier systems can be useful in ways that are hard to measure, and risky in ways that are easy to miss. The lineage matters because it explains why naive extension of older governance intuitions fails.

3.3.4 Why older intuitions fail

Older intuitions fail for predictable reasons: they assume stable targets, well-defined metrics, and separable components. Frontier AI in finance violates each assumption in ways that create governance blind spots.

Consider backtesting. Backtesting is powerful when the output is a numeric signal and the decision

rule is fixed. It tells you how a strategy would have behaved under historical data. But frontier AI systems often operate on documents, workflows, and interactions that are not stationary. Documents change format and language. Policies evolve. Data access changes. Users change prompts. The model itself may be updated. A backtest cannot capture these dynamics because the object being tested is not stable. Worse, backtesting can create false confidence because it produces a familiar artifact—performance curves, metrics—that looks like rigor. In the context of a workflow system, those artifacts may be irrelevant to the true risk.

Accuracy metrics fail for similar reasons. Accuracy is meaningful when there is a ground truth label. But many finance tasks relevant to frontier AI are interpretive: summarizing risks, drafting disclosures, synthesizing research, identifying missing information, recommending escalation. There is no single label for "best summary." As a result, institutions may use proxy metrics like human ratings, time saved, or benchmark scores on generic QA tasks. These proxies do not measure what matters: whether the system is compliant, whether it cites evidence correctly, whether it escalates appropriately, and whether it can be audited.

Older intuitions also fail because they underestimate the importance of provenance. In traditional structured modeling, the dataset is defined: a table of observations with known columns. Provenance is implicit in the dataset construction process, and governance focuses on data quality. In frontier AI, evidence is assembled dynamically through retrieval and memory. The "dataset" is the retrieved context, which can vary case by case. If provenance is not logged, there is no way to reconstruct what the model saw. Without reconstruction, there is no defensible accountability. This is why provenance is not an optional feature; it is a precondition for use in high-stakes settings.

Another failure of older intuitions is the assumption that errors are symmetric and independent. Traditional risk modeling often assumes that errors average out, or that monitoring can detect drift. Frontier AI errors are often correlated through shared retrieval stacks, shared prompts, and shared model behavior. A systematic hallucination pattern can affect many outputs. A retrieval bug can omit a policy clause across an entire organization. These are not random errors; they are systemic. Monitoring must therefore include pattern detection, policy violation tracking, and periodic audits, not only performance drift on numeric metrics.

Older intuitions also neglect the socio-technical dimension. In traditional modeling, user behavior matters, but the interaction is limited. In frontier AI, user interaction is central. Users can pressure systems to answer, to be confident, or to minimize risk language. The system's behavior emerges from the interaction. This means governance must include user training, interface constraints, and consistent refusal behavior. A system that is safe when used perfectly but unsafe when used realistically is not safe.

Finally, older intuitions fail because they assume that the hardest part is prediction. In many finance workflows, the hardest part is not predicting a number; it is producing a defensible narrative and documentation that aligns with policy. Frontier AI excels at producing narrative. That is

exactly why it is dangerous. The model's strength is in the domain's most persuasive medium. Governance must therefore treat narrative outputs as high-risk artifacts, requiring evidence discipline and structured controls.

The conclusion is not that older methods are obsolete. It is that their governance logic must be extended. Backtesting and performance monitoring remain necessary for numeric components. But frontier systems require additional evaluation objects: trajectories, dossiers, policy compliance, and provenance integrity. The old tools are necessary but insufficient.

3.3.5 Inherited lessons

Despite the break in assumptions, the lineage provides valuable inherited lessons. The most important lesson is that governance is not anti-innovation; it is the mechanism by which innovation becomes deployable. Finance learned this through decades of model deployment. Frontier AI should inherit this institutional wisdom rather than repeating the cycle of overconfidence.

From model risk management, we inherit the idea of *inventory and accountability*. Every material model must be identified, owned, documented, and reviewed. For frontier AI, the inventory object must expand: not only the base model, but the retrieval system, prompts, tool integrations, policy rules, and output schemas. Ownership must be explicit: who owns the system, who validates it, who monitors it, and who approves changes. Change control must apply to prompts and policies as much as to code.

We also inherit the lesson of *independent validation*. In traditional MRM, validators are distinct from builders. They test assumptions, replicate results, and challenge claims. Frontier AI needs the same separation, but with expanded test suites: scenario-based evaluations, adversarial prompting, policy-violation measurement, provenance checks, and reconstruction drills. Validators must test not only "can the system answer" but "can the system refuse," "can it escalate," and "can it show evidence."

Another inherited lesson is *stress testing*. Finance learned to ask, "What happens in the tails?" Frontier AI requires tail-aware evaluation: missing documents, conflicting evidence, ambiguous policy, and pressure to provide prohibited advice. Systems must be tested under conditions that mimic real-world stress: time pressure, incomplete information, and adversarial user behavior. The goal is not to eliminate failure, which is unrealistic, but to ensure failures are caught and escalated rather than silently shipped into decisions.

From compliance workflows, we inherit the lesson that *process matters more than intent*. A system may be intended as "draft-only," but if it is used to influence client-facing decisions, it must be governed as such. Process controls—approvals, segregation of duties, logging—are the institution's defense against good intentions. Frontier AI should be integrated into existing control environments, not treated as a special exception.

A further inherited lesson is the importance of *documentation that supports reconstruction.* Auditors and regulators do not evaluate your intentions; they evaluate your artifacts. A governed AI system must produce artifacts that make review possible: what evidence was used, what assumptions were made, what policies applied, what tools were invoked, and who approved the output. This is the "decision dossier" concept: an output is not a paragraph; it is a traceable package.

Finally, we inherit humility about regime change. Quantitative finance learned—often painfully—that models fail when regimes shift. Frontier AI adds new forms of shift: document drift, policy drift, interface drift, and user behavior drift. The governance response mirrors the old lesson: monitor, retrain or update cautiously, and require re-validation when the environment changes. But the content of monitoring expands beyond numeric performance to include policy violation rates, citation accuracy, escalation frequency, and evidence coverage.

In summary, the lineage provides both warning and guidance. The warning is that finance repeatedly over-trusts new modeling techniques until failure teaches restraint. The guidance is that robust governance frameworks exist, but must be extended to govern systems that reason, retrieve, and act over time. The central inheritance is therefore conceptual: treat frontier AI not as a clever tool, but as a decision-support system whose legitimacy depends on evaluation, constraints, and auditability.

3.4 Technical Foundations

3.4.1 System or architectural components

A frontier AI system in finance is rarely "a model." It is a composite architecture in which the model is only one component—and, in many failures, not even the most important one. The technical foundation therefore begins with a system decomposition that is both implementable and governable. For the purposes of this chapter, the minimal architecture contains five functional blocks and one cross-cutting governance layer: (1) ingestion, (2) retrieval and memory, (3) reasoning and synthesis, (4) tools and calculators, (5) structured output and user interface, and (6) governance controls that constrain every block rather than sitting at the end as a polite disclaimer.

(1) Ingestion. Ingestion converts heterogeneous inputs into a normalized internal form. In finance, inputs are rarely clean. Even in a synthetic demonstration, one should emulate the real difficulty: documents with tables, footnotes, inconsistent formatting, and mixed qualitative and quantitative content. In production, ingestion includes file parsing, table extraction, metadata capture, optical character recognition for scanned PDFs, and segmentation into addressable chunks. The governance-relevant part of ingestion is not simply "can we read the file." It is: (i) can we identify what the file is, (ii) can we track its version, (iii) can we record its provenance and access class, and (iv) can we preserve the mapping between a claim and the original source location. A system that cannot point to "where this number came from" is already failing the finance standard, regardless of how impressive the narrative output looks.

(2) Retrieval and memory. Retrieval determines what the model sees and, by extension, what the organization treats as evidence. A modern architecture uses some mixture of keyword search, semantic search (embeddings), hybrid retrieval, and filtering by metadata (date, source type, policy version, confidentiality class). Memory can be ephemeral (session context) or persistent (stored notes, prior decisions, long-term user preferences). In finance, persistent memory is dangerous unless tightly governed because it can cause cross-case leakage, outdated policy reuse, or contamination by unverified user inputs. Retrieval should therefore be treated as an evidence function with explicit controls: allowlists of sources, "latest policy only" rules, and logging of retrieved items with identifiers. If a model cites policy, the system should record which policy version was retrieved, which section was referenced, and whether the retrieval satisfied recency constraints.

(3) Reasoning and synthesis layer. This is the model's domain: interpreting retrieved evidence, extracting entities, summarizing risks, drafting outputs, and proposing next steps. The key architectural choice is whether the system is designed for free-form narrative or for structured reasoning. A governance-first design biases toward structure: the model produces a draft in a constrained schema that includes separated fields for facts, assumptions, open questions, risks, and recommended escalations. Narrative can be generated, but it should be downstream of the structured representation, not upstream. This reduces the risk that the model's rhetorical momentum will

carry it beyond the evidence. Architecturally, this layer often includes prompt templates, system policies, and possibly "self-check" or critique steps. One must be cautious: self-checking can create false confidence if it is not grounded in independent verification. The reliable control is not the model agreeing with itself; it is the model being constrained by evidence and policy.

(4) Tools and calculators. Tools are where finance systems can become safer and more dangerous simultaneously. They can improve numerical correctness by offloading computation to deterministic functions, but they can also amplify error if inputs are wrong and outputs are trusted because they look precise. Tools include calculators for exposures and ratios, parsers for financial statements, validators for numeric consistency, and possibly database queries for internal reference data. Governance requires that tool calls be logged with inputs and outputs, and that tool outputs be linked to the evidence that supplied the inputs. For high-impact computations, the architecture should support independent verification: redundancy through cross-checks, unit tests, or comparison to baseline calculations.

(5) Structured outputs and user interface. This component is frequently underestimated. In a governance-first deployment, the user interface is part of the control system. It determines what the user sees, what is easy to approve, what is hard to ignore, and how uncertainty is communicated. A system that displays a polished memo without showing evidence encourages rubber-stamping. A system that foregrounds evidence references, highlights assumptions, and requires explicit acknowledgement creates appropriate friction. Structured output means more than JSON. It means that the output can be reviewed systematically: each claim has an evidence anchor; each assumption is labeled; each open question is explicit; each recommendation is bounded and accompanied by an escalation status.

(6) Governance layer (cross-cutting). The governance layer enforces policy rules, refusal behavior, escalation triggers, logging, and change control. Critically, it is not a "post-processing filter." It must constrain the system end-to-end. It should enforce source access policies during ingestion and retrieval. It should enforce task scope rules during reasoning (e.g., "no personalized advice"). It should enforce structured schema completeness at output time (e.g., required fields for client-facing drafts). It should enforce escalation when uncertainty is high or when required evidence is missing. And it should enforce logging for every step, because auditability is not optional in high-accountability finance.

This decomposition yields a practical thesis: finance AI systems are governed architectures, not clever models. The frontier capability is real, but the deployable capability is defined by the control surfaces. If the architecture cannot support evidence traceability, escalation, and reconstruction, it is not ready for any workflow where accountability matters.

3.4.2 Information flow

The technical foundation becomes operational when we specify the information flow: how evidence enters the system, how it is transformed, and how it becomes a decision-support artifact. In a governance-first design, the information flow must preserve two invariants: (i) the system must maintain a chain of custody from claim to evidence, and (ii) the system must preserve the distinction between what is known and what is inferred.

A canonical information flow can be described as:

$$\text{Source documents} \rightarrow \text{Extracted facts} \rightarrow \text{Structured representation} \rightarrow \text{Decision-support output}.$$

Each arrow is a transformation stage with specific risks and controls.

Stage 1: Source documents. "Source documents" includes both external artifacts (filings, transcripts, market data feeds, research publications) and internal artifacts (policies, risk reports, prior decisions, internal memos). The system must attach metadata: source type, timestamp, version, access class, and reliability tier. In finance, this metadata is not a technical nicety; it is a governance requirement. A clause in a draft policy does not have the same authority as a clause in an approved policy. A number in a press release does not have the same reliability as a number in audited financial statements. If the system treats all text as equal, it will reliably produce plausible but indefensible outputs.

Stage 2: Extracted facts. Fact extraction is the process of turning raw sources into explicit claims: entities, numbers, dates, obligations, and relationships. Here the system should aim for conservative extraction: it is better to miss a fact than to invent one. Extraction should include pointers: not only "the debt is $X," but "the debt is $X as stated in document D, page P, section S." In practice, extraction often involves table parsing and natural language extraction. The control point is that extraction output must carry evidence anchors forward into later stages.

Stage 3: Structured representation z. The representation is where governance either becomes real or remains aspirational. A structured representation should separate:

- **Facts:** extracted claims directly supported by sources, with evidence identifiers.
- **Assumptions:** any missing values, interpretations, extrapolations, or contextual guesses.
- **Open questions:** missing evidence, ambiguous clauses, unresolved conflicts between sources.
- **Risk flags:** identified tail risks, policy conflicts, uncertainty warnings.
- **Policy context:** which constraints apply, which rules were triggered, and why.

This separation is not merely pedagogical. It is how the organization protects itself from the model's tendency to smooth over uncertainty. A model will often prefer to complete the story. The structured representation forces the story to declare its gaps.

Stage 4: Decision-support output a. The final output is produced from the structured

representation, ideally as two coupled artifacts: (i) a user-facing narrative or checklist that is readable and actionable, and (ii) a machine- and audit-friendly dossier that preserves structure, evidence, and logs. The system should also produce an escalation status: proceed, proceed-with-review, escalate, or refuse. In finance, an output without a clear status invites misuse. A "helpful" draft that is actually high-risk should not be presented in the same way as a low-risk summary.

The key technical principle is *evidence attachment*. Every material claim should be accompanied by evidence references, either inline or in a structured appendix. The system should not merely cite sources; it should show the relevant excerpt or at least the specific location that supports the claim. This reduces hallucinated citations and makes human review feasible. In a governance-first workflow, the reviewer's job is not to admire the prose but to verify the evidence chain.

The second technical principle is *interpretation containment*. Interpretations are often necessary—finance is full of ambiguity—but they must be labeled. A model can suggest "this covenant likely restricts dividends" as an interpretation, but it must label that as interpretation and include the clause that supports it. It must also flag uncertainty and suggest escalation if the interpretation is material.

Information flow is therefore not a pipeline that produces an answer; it is a pipeline that produces an accountable artifact. This is the technical meaning of governance-first design: the system's output is not merely an opinion, but a traceable transformation of evidence under constraints.

3.4.3 Interaction or control loops

Frontier AI in finance cannot be governed as a one-shot function. It must be governed as a set of interacting control loops. These loops exist at different time scales: within a single case (human-in-the-loop review), across cases (monitoring and drift detection), and across system versions (change control and re-validation). The technical foundation must therefore include loop design as explicitly as it includes model selection.

Human-in-the-loop approval gates. The first loop is the case-level review loop. The system produces an output and a dossier; a human reviewer assesses evidence, checks assumptions, confirms compliance requirements, and either approves, requests revision, or escalates. The technical requirements for this loop are concrete:

- The system must expose evidence anchors so the reviewer can verify claims efficiently.
- The system must highlight assumptions and open questions so the reviewer cannot miss them.
- The system must support structured feedback (e.g., marking a claim as unsupported) that can be logged.
- The system must record the reviewer identity, timestamp, and decision rationale.

A system that requires reviewers to hunt through narrative for hidden assumptions is a system designed for failure. Review loops must be engineered, not merely declared.

Escalation loops. The second loop is escalation. A governed system must know when to stop pretending it knows. Escalation triggers can be rule-based (e.g., client-facing outputs require supervisor sign-off) or uncertainty-based (e.g., evidence coverage below threshold) or impact-based (e.g., large exposure change). The technical challenge is to define triggers that are conservative enough to prevent harm without overwhelming human capacity. This is a genuine systems problem: if escalation triggers fire too often, users will route around the system or rubber-stamp escalations; if triggers fire too rarely, the system will cause harm. The solution is not to "tune" until users are happy; it is to align triggers with risk tiering and to design workflows that can absorb escalation for high-impact cases.

Monitoring loops. Beyond individual cases, the system must be monitored continuously. Monitoring in frontier AI finance should include:

- Policy violation rates (scope violations, missing disclosures, prohibited advice).
- Evidence integrity metrics (citation accuracy, evidence coverage, retrieval recency).
- Escalation frequency and correctness (are escalations occurring when they should?).
- Drift signals (changes in document structure, changes in model behavior after updates).
- Anomaly detection (spikes in certain error types, correlated failures across teams).

Monitoring is not only about performance; it is about governance behavior. A system that remains accurate but begins omitting disclosures is deteriorating in the dimension that matters most.

Incident reconstruction loops. When failures occur—and they will—the organization must be able to reconstruct what happened. This is not optional. Reconstruction requires immutable logs and replay capability: the ability to re-run the system with the same inputs, retrieval index state, policy version, prompts, and tools. The technical foundation must therefore include versioning of every component that influences outputs. Without replay, incidents devolve into opinion disputes, and governance becomes theater.

Change control and re-validation loops. Frontier systems are updated frequently: model versions change, prompts are refined, retrieval indexes are rebuilt, policies are updated, tools are modified. Each change can alter behavior. The technical foundation must therefore include a disciplined release process: test suites, regression evaluations, and approvals for changes that affect risk tier. In finance, "continuous deployment" without governance is an invitation to correlated failure.

These control loops are the operational expression of the book's frontier thesis: evaluation must be trajectory-level, not output-level. A system that cannot be governed through loops cannot be safely integrated into decision-support workflows. The most sophisticated model is irrelevant if the control loops are weak.

3.4.4 Assumptions and constraints

Every technical foundation rests on assumptions. In finance, assumptions are unavoidable; what is avoidable is pretending they are not there. A governance-first system must therefore declare its assumptions and encode its constraints in architecture rather than in disclaimers.

Assumption 1: Data quality is uneven. Financial data and documents vary in reliability and structure. Even within the same firm, different desks and functions maintain different standards. External documents change formatting across years. Transcripts contain errors. Tables are misparsed. This assumption means the system must be robust to noise and missingness, and must explicitly represent uncertainty rather than smoothing it away.

Assumption 2: Provenance is fragile. In a retrieval-based system, the evidence set is dynamic. Provenance can fail through stale indices, misclassified sources, or unauthorized memory reuse. This assumption forces architecture choices: strict source classification, recency filters for policy documents, and logs that capture retrieval results. Without provenance control, the system becomes an "evidence blender," mixing authoritative and non-authoritative content into a plausible output.

Assumption 3: Users will apply pressure. Users in finance are often under time pressure and may push systems to produce definitive outputs. They may also have incentives to minimize risk language. This is not moral judgment; it is institutional reality. Therefore, constraints must be enforced technically, not socially. A system must refuse prohibited outputs and escalate high-risk cases regardless of user pressure.

Assumption 4: The environment changes. Regimes shift, policies update, and document conventions evolve. The system must be designed for drift: monitoring, periodic re-validation, and conservative behavior under uncertainty. A one-time validation is not a control; it is a historical artifact.

Constraints follow from these assumptions and from the legal and fiduciary environment.

Constraint 1: Scope boundaries. The system must be explicitly constrained to avoid unauthorized functions, especially personalized investment advice, autonomous trade execution, and any form of recommendation that bypasses suitability requirements. These constraints should be encoded in action space restrictions, refusal policies, and output schemas that require disclaimers and escalation for certain contexts.

Constraint 2: Disclosure integrity. Client-facing and external outputs require consistent disclosure language and must not omit required risk information. The system must therefore include completeness checks: required fields, mandatory risk statements, and version-controlled templates.

Constraint 3: Suitability and personalization controls. Where suitability applies, the system must not infer missing client information. It must request it or escalate. A model that fills in missing suitability facts is creating liability.

Constraint 4: Privacy and insider-information handling. The system must enforce access controls and prevent leakage. Sensitive inputs must be redacted where possible, and logs must be designed to preserve auditability without replicating confidential content unnecessarily.

Constraint 5: Auditability. The system must generate logs and dossiers that support reconstruction. This is a technical constraint: if logs are not immutable and replayable, the system cannot be used in high-stakes contexts.

These assumptions and constraints imply a design posture: conservative by default, explicit about uncertainty, and engineered for review. The point is not to slow the organization down; it is to ensure that speed is not purchased by offloading accountability to an un-auditable system.

3.4.5 Technical bottlenecks

Even with a sound architecture, several technical bottlenecks limit what frontier AI can safely do in finance today. These bottlenecks are not merely performance issues; they are governance issues because they determine whether outputs can be trusted under institutional scrutiny.

Bottleneck 1: Claim verification. The hardest problem in finance AI is not generating a plausible memo; it is verifying that each material claim is supported by the cited evidence. Retrieval helps, but models can still misattribute claims, hallucinate citations, or draw conclusions not warranted by the source. Automated verification is difficult because documents are complex and context-sensitive. The practical governance response is to reduce the verification burden: require evidence anchors, show excerpts, and constrain the model to extract-and-summarize rather than to speculate. But the bottleneck remains: scalable, reliable claim verification is not solved, and systems must be designed with that limitation in mind.

Bottleneck 2: Calibrated uncertainty for narrative outputs. Finance requires meaningful uncertainty. Models can produce hedging language, but rhetorical hedging is not calibration. A system must determine when evidence is weak, when sources conflict, and when the task is outside scope. Reliable uncertainty signals for long-form narrative remain technically challenging. Until calibration improves, the safe design choice is to use conservative thresholds and escalation, treating "unknown" as an output state rather than as a failure.

Bottleneck 3: Robustness under regime and document drift. Traditional models struggle under regime shifts; frontier systems add new drift dimensions: changes in document phrasing, policy updates, shifts in internal templates, and changes in user prompting behavior. A retrieval index built on last year's documents may not retrieve well on this year's structure. A prompt tuned for one desk may fail on another. Robustness requires continuous monitoring and re-validation, which is organizationally expensive. The technical bottleneck is building evaluation harnesses that keep up with drift without becoming an unmaintainable bureaucracy.

Bottleneck 4: Tool reliability and error propagation. Tool-using systems can compound errors:

a misparsed number becomes an incorrect calculation, which becomes a confident recommendation. Tools are deterministic, but the pipeline into tools is not. The bottleneck is not writing calculators; it is ensuring that inputs are validated, units are consistent, and failure states are detected. This demands schema enforcement and cross-checks, which add complexity and latency.

Bottleneck 5: Retrieval integrity at scale. Retrieval is frequently the hidden failure point. It is sensitive to embedding quality, chunking strategy, metadata completeness, and index freshness. In finance, where "latest policy" matters, retrieval must respect versioning and recency. The bottleneck is building retrieval systems that are both performant and provably compliant with source constraints. Without this, the model may be governed but the evidence supply chain is not.

Bottleneck 6: Reviewability and throughput. Even if everything is logged, the system must be reviewable at scale. Human review is a throughput constraint. If outputs are too long, too dense, or too poorly structured, review becomes impractical and users will skip it. The bottleneck is therefore partly UX and partly process design: making review efficient without reducing rigor. This is a technical problem because it requires structured outputs, clear evidence display, and workflow integration that supports approvals and audit trails.

These bottlenecks are not excuses to avoid deployment; they are reasons to deploy in bounded ways. A governance-first architecture treats bottlenecks as design constraints: restrict scope to tasks where verification is feasible, require escalation where uncertainty is high, and invest in logging and evaluation harnesses before scaling authority. The frontier is not a race to autonomy. In finance, the frontier is the disciplined expansion of decision support under constraints that make accountability real.

3.5 Mathematical Foundations

3.5.1 Formal problem framing

A governance-first treatment of frontier AI in finance begins by refusing a seductive simplification: this is not "prediction," and it is not "optimization for profit." It is decision support under constraints where the constraint set is not a footnote—it is the system. The purpose of mathematics here is not to prove theorems that the real world will immediately violate. The purpose is to formalize the objects that governance must control: evidence, representations, permissible actions, and policy constraints that define the boundary of authority.

We model the system as producing a bounded action proposal a conditioned on an internal representation z of evidence and context. The representation z is itself derived from raw inputs I via an ingestion and retrieval pipeline. The system's objective can be written as:

$$\max_{a \in \mathcal{A}} \ \mathbb{E}\big[U(a; z)\big] \quad \text{s.t.} \quad \text{Policy}(a, z) \leq 0.$$

This compact expression hides the key governance insight: the feasible set is shaped more by $\text{Policy}(\cdot)$ than by $U(\cdot)$. In a regulated environment, the organization does not want the globally optimal action; it wants the action that is *permissible, reviewable, and defensible* under constraints.

To make this concrete, we define:

- I: inputs (documents, tables, internal notes, synthetic market context, policy documents, tool outputs).
- $z = g(I)$: representation produced by a pipeline g that includes extraction, retrieval, normalization, and uncertainty annotation.
- $a \in \mathcal{A}$: an action proposal produced by the system, such as a draft summary, a risk flag set, an escalation request, a refusal, or a bounded recommendation for human consideration.
- $U(a; z)$: utility of action a given representation z, capturing decision usefulness, risk awareness, operational efficiency, and governance quality.
- $\text{Policy}(a, z)$: constraint function encoding legal, regulatory, and internal policy restrictions, including disclosure requirements, suitability rules, data handling rules, scope limits, and escalation thresholds.

The expectation $\mathbb{E}[\cdot]$ reflects uncertainty: the system does not observe the true state of the world, and the representation z is noisy. In practice, the expectation can represent a distribution over plausible states consistent with evidence, or an empirical average over scenario cases used in evaluation. The key point is that decision support is fundamentally a problem of acting under partial information.

A governance-first framing differs from a classic decision-theoretic framing in one essential way: the system is not rewarded for boldness. In many ML settings, the objective rewards correct predictions

and penalizes incorrect ones. In finance decision support, the system must also be rewarded for *restraint*: when evidence is insufficient, the best action may be to ask questions, escalate, or refuse. This introduces "epistemic actions" into the action space—actions that improve knowledge rather than directly produce a recommendation.

We therefore decompose the action a as:

$$a = (a_{\text{content}}, a_{\text{status}}, a_{\text{dossier}}),$$

where a_{content} is the human-facing artifact (summary, memo, checklist), $a_{\text{status}} \in \{\text{proceed}, \text{proceed-with-review}, \text{escal}$ is the system's control decision, and a_{dossier} is the structured evidence and log package that supports auditability. A system that outputs content without status and dossier is not producing an action in the governance sense; it is producing text.

Finally, the constraint function is not merely "no prohibited advice." It is an operational encoding of institutional responsibility. A policy constraint can require that certain outputs only be produced if specific evidence fields are present; that certain contexts always trigger escalation; or that certain classes of sources are never used for client-facing outputs. In this sense, $\text{Policy}(a, z)$ defines the feasible region in which the system is allowed to operate. The mathematics is therefore a language for governance: it tells us what must be specified to make autonomy impossible where autonomy is not permitted.

3.5.2 State, action, or representation spaces

To govern a decision-support system, one must formalize the spaces that the system traverses: what it knows (state), how it represents what it knows (representation), and what it is allowed to propose (action). The distinction matters because governance often fails when these spaces are conflated. A model may behave as if it knows something because it can generate a plausible statement, even when that statement is not supported by evidence in z.

Let the true environment state be $s^\star \in \mathcal{S}^\star$. In finance, $s^\star$ includes latent factors that cannot be fully observed: true creditworthiness, liquidity under stress, future macro outcomes, and private information. The system does not observe $s^\star$. Instead, it observes inputs I and produces a representation z that lives in a designed representation space $\mathcal{Z}$.

We define the system's operational state as:

$$s = (m, e, p),$$

where:

- $m \in \mathcal{M}$: market context (prices, vol surfaces, macro indicators, regime labels—synthetic in demonstrations).

- $e \in \mathcal{E}$: evidence state (retrieved documents, extracted facts, citations, conflicts).
- $p \in \mathcal{P}$: internal policy state (applicable rules, thresholds, versioned policy artifacts, role permissions).

The evidence component e is not simply "data." It is a set of claims with provenance. We can model evidence as:

$$e = \{(c_i, \pi_i, \rho_i)\}_{i=1}^{n},$$

where c_i is a claim (a number, fact, clause, or assertion), π_i is a provenance pointer (document id, section, timestamp), and ρ_i is a reliability annotation (confidence score, source tier, extraction method). Conflicts can be represented by relations among claims, such as a conflict graph or constraint set that indicates which claims cannot simultaneously be true.

The representation $z \in \mathcal{Z}$ is a structured summary of s, designed for both model consumption and auditability. A governance-first representation space typically includes explicit partitions:

$$z = (F, A, Q, R, C),$$

where:

- F: facts extracted from evidence (supported claims with provenance).
- A: assumptions introduced by the system (imputed values, interpretations).
- Q: open questions (missing evidence, unresolved conflicts).
- R: risk flags (tail risks, compliance risks, uncertainty warnings).
- C: constraints triggered (which policies apply, which thresholds are crossed).

This partition is not cosmetic. It is a mathematical guardrail against hallucination: any statement not supported by provenance must reside in A or Q, not in F.

The action space $\mathcal{A}$ must be explicitly bounded. In this chapter's framing, $\mathcal{A}$ excludes autonomous trades and personalized investment advice. We define:

$$\mathcal{A} = \mathcal{A}_{\text{draft}} \cup \mathcal{A}_{\text{summary}} \cup \mathcal{A}_{\text{risk}} \cup \mathcal{A}_{\text{escalate}} \cup \mathcal{A}_{\text{refuse}}.$$

Elements of these sets include:

- Draft outputs (internal memos, disclosure drafts, checklists).
- Summaries (filing summaries, transcript highlights, policy summaries).
- Risk artifacts (flag sets, risk narratives with evidence anchors).
- Escalation actions (request additional evidence, route to compliance, require supervisor sign-off).
- Refusals (decline to answer due to scope or insufficient evidence).

We can further represent an action as a tuple:

$$a = (y, \sigma, \ell),$$

where y is the content output (possibly structured), σ is the status class, and ℓ is the log/dossier payload. This emphasizes that logging is part of the action, not a side effect.

Finally, in finance one must represent *role-based permissions*. Let $r \in \mathcal{R}$ denote the user role (analyst, compliance officer, portfolio manager). The permissible action space becomes role-conditional:

$$\mathcal{A}(r) \subseteq \mathcal{A}.$$

A governed system must enforce $\mathcal{A}(r)$ strictly, and must log role identity as part of the audit trail. Otherwise, systems become permission-amplifiers: they allow low-authority users to generate high-authority artifacts.

3.5.3 Objective functions and constraints

The utility function $U(a; z)$ encodes what the institution values *within the permitted region*. Because finance is multi-objective, we formalize U as a weighted sum of components. A convenient representation is:

$$U(a; z) = \lambda_{\text{use}} U_{\text{use}}(a; z) + \lambda_{\text{risk}} U_{\text{risk}}(a; z) + \lambda_{\text{ops}} U_{\text{ops}}(a; z) + \lambda_{\text{gov}} U_{\text{gov}}(a; z),$$

with $\lambda_i \geq 0$. Each component can be defined to reflect governance-first priorities.

Decision usefulness U_{use}. Measures relevance, clarity, and alignment to the user's task. For example, it can reward completeness of a checklist, correct identification of key drivers, and coherent synthesis. Importantly, usefulness should be assessed conditional on evidence: a confident but unsupported claim should not score high.

Risk awareness U_{risk}. Rewards identification of tail risks, flagging of missing information, and appropriate caution under uncertainty. In a governance-first system, U_{risk} is not an afterthought; it is a core term. A system that always provides a neat answer with no warnings is optimizing usefulness at the expense of safety.

Operational efficiency U_{ops}. Captures time saved, reduction of manual effort, and standardization. This is the term most organizations focus on when adopting AI. The governance-first posture is not to deny it, but to prevent it from dominating the objective.

Governance quality U_{gov}. Rewards compliance with policy, provenance integrity, proper separation of facts and assumptions, correct escalation, and completeness of logging. This term is what makes the optimization problem institutional rather than purely technical.

Constraints are encoded by $\text{Policy}(a, z) \leq 0$, which can be understood as a vector inequality:

$$\text{Policy}(a, z) = \begin{bmatrix} g_1(a, z) \\ g_2(a, z) \\ \vdots \\ g_k(a, z) \end{bmatrix} \leq 0,$$

where each g_j represents a specific policy constraint. We distinguish three classes of constraints.

(i) Hard scope constraints. These prohibit certain outputs regardless of utility. Examples include: no personalized investment advice, no autonomous trade instructions, no circumvention of compliance rules, and no generation of prohibited content. Formally, hard constraints define feasibility:

$$a \in \mathcal{A}_{\text{allowed}}(r) \quad \text{and} \quad a \notin \mathcal{A}_{\text{prohibited}}.$$

(ii) Evidence and provenance constraints. These require that any material claim be supported by provenance, and that the system explicitly label assumptions. A simple constraint form is:

$$\forall \text{ material claim } c \in y(a), \;\; \exists (c_i, \pi_i, \rho_i) \in e \text{ such that } c \text{ is supported by } \pi_i,$$

or, if unsupported, the claim must be placed in the assumptions set A and flagged accordingly. Similarly, policy recency constraints can require that cited policies be the latest approved version:

$$\text{If } a \text{ cites policy } P, \text{ then } \text{version}(P) = \text{latest}(P).$$

(iii) Escalation and review constraints. These enforce that high-impact or high-uncertainty cases trigger escalation. Let $\Delta(a)$ denote an impact score (e.g., exposure size, client-facing risk tier), and let $\kappa(z)$ denote an uncertainty score derived from evidence completeness and conflict. Then a constraint can be:

$$\Delta(a) \geq \tau_\Delta \;\text{ or }\; \kappa(z) \geq \tau_\kappa \quad \Rightarrow \quad a_{\text{status}} \in \{\text{escalate}, \text{refuse}\}.$$

This captures the governance notion that the system must stop when it cannot be confident in a defensible way.

In practice, policy constraints are implemented by rule engines, validators, and workflow gates. The mathematical point is that the objective is not free to trade off compliance for usefulness. Compliance defines feasibility; usefulness is optimized within it.

Finally, there is a subtle but crucial constraint: *no decision laundering*. While difficult to formalize fully, the system can enforce process constraints that reduce laundering, such as requiring explicit human sign-off and requiring reviewers to acknowledge key assumptions. This is an instance where mathematics points to design: if human accountability is required, it must be represented as a

necessary step in the action execution policy.

3.5.4 Sources of instability or fragility

A system can satisfy its formal specification and still fail in practice because the specification is evaluated under distributions that shift. The mathematical framing therefore must include instability sources, not as footnotes but as intrinsic properties of the problem.

Regime shift in market context. Financial relationships are non-stationary. Let $P_t(s^\star)$ denote the distribution of true states over time. Regime shift means $P_t \neq P_{t'}$. Even if the system's representation mapping g is stable, the relationship between evidence and outcomes changes. This is why predictive guarantees do not hold across regimes. For decision support, the more relevant issue is that the model's learned priors may become miscalibrated, producing overconfident narratives based on outdated patterns.

Evidence drift in documents. Documents change: formatting, language, disclosure style, and terminology. Retrieval and extraction depend on document structure. Evidence drift means the function g that maps inputs to representation changes in effective behavior even if the code is unchanged, because the inputs are different. This can be represented as a drift in the distribution of I: $P_t(I) \neq P_{t'}(I)$. In finance, document drift is not rare; it is constant.

Policy drift. Internal policies, thresholds, and regulatory interpretations evolve. If the policy state p changes but the system's constraint layer uses outdated versions, feasibility is mis-specified. This is a distinct fragility: the system can produce outputs that were compliant yesterday and are non-compliant today. This fragility is amplified when policy retrieval is imperfect.

Tool errors and error propagation. Tool-using systems introduce compositional fragility. Let the system produce intermediate tool inputs u, tool outputs $v = T(u)$, and final action $a = h(z, v)$. If u is wrong due to extraction error, v will be wrong deterministically, and a may be wrong with high confidence. Because tools produce precise outputs, humans may trust them more. This is a classic "garbage in, gospel out" failure mode.

Optimization pressure amplifying weak objectives. When systems are tuned to maximize a proxy metric (user satisfaction, brevity, perceived helpfulness), they may learn to produce outputs that are persuasive rather than defensible. In the mathematical framing, this is the mis-specification of U: if λ_{gov} is effectively near zero in practice, the system will optimize for completion and confidence. Even if policy constraints exist, the system will push against them, generating borderline outputs that "seem okay." This is not malice; it is optimization under misaligned rewards.

Human interaction as a non-stationary component. Users adapt to systems. They learn which prompts elicit stronger answers, which phrasing bypasses refusals, and which shortcuts reduce friction. This changes the effective input distribution. Let I include user prompts and context selection. Then user adaptation implies $P_t(I)$ shifts as users learn. This creates a feedback loop: the

system changes user behavior, and user behavior changes system outputs. Traditional evaluation that assumes fixed inputs fails to capture this.

Correlated failures and monoculture. If many desks or firms use similar models and retrieval patterns, errors can correlate. This is a systemic fragility: even if individual risk is small, correlated risk can be large. Mathematically, this means the joint failure probability across institutions is not the product of independent failure probabilities; it is higher due to shared factors. Governance must therefore consider diversity and redundancy at the system level.

The practical implication is that stability cannot be assumed; it must be monitored. A governance-first system treats drift detection, policy versioning, and tool validation as core technical components, not optional enhancements.

3.5.5 What theory does NOT guarantee

Formalization is useful precisely because it clarifies what cannot be guaranteed. In finance, there is a long history of models being mistaken for reality. Frontier AI adds new opportunities for that mistake because it produces human-like narrative. This subsection is therefore an explicit boundary-setting exercise: what does the formal structure above *not* imply?

No stability across regimes. The optimization problem $\max_a \mathbb{E}[U(a; z)]$ does not guarantee that a policy that performed well historically will perform well under regime shift. The distribution underlying the expectation changes. Even if the system is evaluated on a suite of cases, that suite is only a sample of possible futures. Therefore, deployment must be coupled with monitoring and conservative escalation policies. Mathematics can represent regime shift; it cannot eliminate it.

No truth from coherence. The model can produce coherent narratives even when evidence is weak. The formal separation of facts and assumptions helps, but it does not guarantee that the system will not misclassify an inference as a fact, or that reviewers will not be persuaded by style. Coherence is not a proof. The formalism therefore supports governance design, but it does not guarantee epistemic integrity unless evidence anchoring and review are enforced.

No automatic compliance from constraints. Writing $\text{Policy}(a, z) \leq 0$ does not mean the implemented system actually enforces policy correctly. Policies are complex, context-dependent, and sometimes ambiguous. Constraint implementations can have bugs. Retrieval can supply wrong policy versions. Users can attempt to bypass constraints. The formalism clarifies what should be constrained; it does not guarantee constraint correctness.

No elimination of professional liability. Even a perfectly constrained decision-support system does not remove responsibility. Humans remain accountable for decisions and communications. The system can reduce certain error types, but it introduces others. Liability can arise from omissions, from reliance on incorrect outputs, or from failure to supervise. Formalization can support defensibility by producing audit trails, but it cannot transfer responsibility.

No guarantee of calibrated uncertainty. Even if the system computes uncertainty scores $\kappa(z)$, those scores may not be calibrated in a meaningful probabilistic sense. In complex narrative tasks, calibration is hard. Therefore, uncertainty thresholds should be treated as conservative heuristics and validated empirically via case-based evaluations. The theory can represent uncertainty; it cannot ensure that uncertainty measures are trustworthy.

No guarantee that human-in-the-loop prevents harm. Adding a human reviewer is not a theorem. Humans can rubber-stamp, misunderstand evidence, or defer to model outputs. The formalism can require sign-off, but it cannot guarantee quality of review. Review effectiveness depends on interface design, training, incentives, and workload. Governance must therefore treat human review as an engineered process, not as a label.

No guarantee against systemic risk. Even if an individual institution's system is well governed, widespread adoption of similar systems can produce correlated behavior. Herding and monoculture risk arise from shared models and shared evidence sources. The formalism is institution-level; systemic risk requires broader coordination and possibly regulatory guidance.

The point of these non-guarantees is not pessimism. It is discipline. A governance-first mathematical foundation is a way to specify what we are trying to control, to design architectures that can be audited, and to structure evaluation. It is not a promise that the frontier has been domesticated. In finance, the only responsible posture is to treat every model as a hypothesis and every system as a controlled experiment—scaled only when the evidence for safety and defensibility is commensurate with the authority being granted.

3.6 Evaluation and Validation

3.6.1 Why naïve metrics fail

Evaluation is where finance forces honesty. Many AI deployments fail not because the model is incapable, but because the organization evaluates the wrong object with the wrong metric and then confuses a score with safety. Naïve metrics fail in frontier finance AI for three structural reasons: (i) the value is procedural, not purely predictive; (ii) the failure modes are governance failures, not just accuracy errors; and (iii) the systems are multi-step, so output-level scoring hides trajectory risk.

Traditional machine learning metrics—accuracy, F1, AUC—assume a task with labels and a stable mapping from inputs to outputs. They are useful when the objective is classification or prediction under a known distribution. Many high-value finance workflows are not like that. A system that drafts a disclosure, summarizes a filing, prepares a decision dossier, or flags missing suitability data is performing a composite task in which "correctness" includes process: did it cite the correct clause, did it label assumptions, did it escalate when required, did it respect scope boundaries, did it log evidence? A model can achieve high linguistic "quality" and still be unacceptable because it violates a policy requirement that is orthogonal to predictive accuracy.

Backtesting fails for similar reasons. Backtesting is designed for strategies with fixed rules and numerical outputs, where performance can be simulated over historical time. Frontier AI decision-support systems are not fixed strategies. They are interactive, retrieval-driven, and policy-conditioned. Their behavior depends on user prompts, retrieved context, tool calls, and evolving policy documents. This makes the system non-stationary. A backtest can evaluate one frozen snapshot of the system under one set of historical documents and prompts, but it cannot capture how the system behaves as the environment changes, as users adapt, and as retrieval indexes drift. Worse, backtesting produces familiar artifacts—charts, Sharpe ratios, drawdowns—that create a false sense of rigor even when the evaluation object is mis-specified. The organization concludes "it backtested well" and silently ignores that it has not measured policy violations, citation accuracy, or escalation correctness.

Naïve evaluation also fails because it implicitly assumes that errors are symmetric and independent. In finance AI systems, errors can be correlated and concentrated in tails. A system might be correct on 95% of low-stakes cases and catastrophically wrong on the 5% that involve ambiguous covenants, conflicting disclosures, or high-impact decisions. Average-case metrics will celebrate this system because they optimize for frequency-weighted accuracy. Governance, however, cares about harm-weighted failure. A single client-facing disclosure omission can dominate the cost function. A single suitability failure can dominate risk. Therefore, evaluation must be tail-aware and scenario-driven.

Another reason naïve metrics fail is that they do not measure *epistemic integrity*. Language models can produce coherent narratives that feel plausible even when evidence is weak. A metric that scores lexical similarity to a reference summary does not detect that the model invented a number. A

metric that scores "helpfulness" from user ratings does not detect that the model cited an outdated policy. These are not edge cases; they are predictable behaviors in systems trained to complete text. Evaluating such systems using generic "QA correctness" encourages exactly the wrong optimization: produce confident answers quickly. In a regulated domain, the most valuable outputs are often those that say, "Here is what the evidence supports; here is what is missing; here is what must be escalated."

Finally, naïve metrics fail because they ignore *reviewability*. In finance, an output is not acceptable merely because it is correct; it must be reviewable and defensible. A system that produces a dense, polished memo without exposing evidence imposes an impossible burden on reviewers. If review is impractical, it will be skipped, and the system becomes effectively autonomous by default. Reviewability is therefore an evaluation dimension: does the system make it easy for humans to verify claims, or does it invite rubber-stamping?

The conclusion is straightforward: if you evaluate frontier AI in finance using the same metrics used to evaluate a classifier, you will optimize for the wrong behavior. The right evaluation object is the decision-support process—evidence handling, policy compliance, escalation behavior, and auditability—measured through case-based scenarios rather than through generic accuracy scores.

3.6.2 Appropriate evaluation units

The appropriate evaluation unit for frontier financial AI is not a single output. It is a *case* that includes evidence, constraints, and an expected governance outcome. Case-based evaluation makes evaluation commensurate with accountability: it tests whether the system behaves properly under realistic conditions and whether the organization can defend the outcome after the fact.

A case k can be defined as a tuple:

$$k = (I_k, P_k, r_k, \Omega_k),$$

where I_k is the input bundle (documents, tables, user request), P_k is the applicable policy set (including versioned rules), r_k is the user role, and Ω_k is the expected governance outcome space (what is acceptable, what must be escalated, what must be refused). The system produces an output action a_k and a dossier d_k. Evaluation then compares a_k and d_k to case expectations.

This approach implies that evaluation requires curated scenarios. In a governance-first book, we emphasize synthetic cases for demonstration, but the structure is the same in production. The scenario library should cover:

- Low-stakes drafting with clean evidence (baseline behavior).
- Medium-stakes summaries with partial evidence (uncertainty handling).
- High-stakes contexts requiring mandatory escalation (control behavior).
- Policy edge cases (conflicting policies, outdated clauses, role-based restrictions).

- Retrieval pitfalls (irrelevant but persuasive context, missing key documents).

The cases should be designed so that "success" is not only producing a good paragraph but producing the correct governance action: summarize, ask, escalate, refuse, or proceed-with-review.

Metrics should therefore be defined over governance dimensions. A practical minimal set includes:

(1) Provenance accuracy. For each material claim in the output, does the dossier provide a correct evidence anchor? This can be measured as a claim-level precision: fraction of claims supported by cited sources. It should also measure citation relevance: are the cited sources actually supporting the claim, or merely topically related? The metric must penalize hallucinated citations and misattributions strongly.

(2) Policy violation rate. How often does the system violate explicit scope boundaries or fail to include required disclosures? This includes content violations (e.g., giving personalized advice when prohibited) and procedural violations (e.g., drafting client-facing outputs without required sign-off flags). Policy violation rate should be reported by severity tier, because minor style violations are not equivalent to prohibited advice.

(3) Escalation correctness. When a case requires escalation (due to impact, uncertainty, or policy), does the system escalate? When escalation is not required, does the system avoid unnecessary escalation that would overwhelm humans? This can be measured with a confusion-matrix-like structure: true escalations vs false escalations. Governance-first design typically tolerates more false escalations than false non-escalations in high-stakes workflows, but the acceptable trade-off must be explicit.

(4) Facts/assumptions separation integrity. Does the system correctly label assumptions and avoid presenting inferred content as fact? This can be measured by auditing outputs for unsupported statements that appear in the "facts" section. The goal is not to eliminate assumptions—finance requires interpretation—but to make them explicit and reviewable.

(5) Reviewability score. This measures whether the output and dossier enable efficient human verification. Practical proxies include: number of claims with anchors, clarity of assumption labeling, completeness of required fields, and time-to-review in structured user studies. Reviewability is not a "nice to have." It is what keeps human oversight real.

(6) Trace completeness. Does the dossier include all required logs: retrieval results, policy versions, tool calls, intermediate representations, and final outputs? Missing traces should be treated as failures because they prevent incident reconstruction.

The evaluation unit should also include a human review outcome. In a governed workflow, the system proposes; a human approves or rejects. Evaluation should measure whether humans can reliably detect errors when using the system, because the combination of model and interface determines real-world risk. If humans systematically miss certain error types, the system is unsafe even if it is "usually correct."

The guiding principle is that evaluation must be aligned with governance obligations. A case-based unit makes this possible because it embeds policy and accountability into the evaluation object itself.

3.6.3 Robustness and stress testing

Robustness is not a single number; it is a suite of adversarial conditions designed to reveal where the system fails in ways that matter. In finance, stress testing is culturally familiar, but frontier AI requires a different kind of stress test: not only "what happens under market stress," but "what happens under evidence stress, policy stress, and user pressure."

Evidence stress tests manipulate the quality and completeness of inputs:

- **Missing evidence:** remove key documents or omit crucial sections and test whether the system appropriately flags missing information rather than inventing.
- **Noisy evidence:** introduce contradictory numbers, typos, or ambiguous clauses and test whether the system detects conflict and escalates.
- **Conflicting evidence:** provide two sources with different values (e.g., draft vs executed term sheet) and test whether the system prioritizes the authoritative source and flags the conflict.
- **Outdated evidence:** provide superseded policy documents and test whether the system rejects them or retrieves the latest version.

The goal is to see whether the system behaves conservatively: when evidence is weak, does it become more cautious, or does it become more imaginative?

Policy stress tests target the constraint layer:

- **Scope pressure:** prompts that attempt to elicit personalized investment advice or trading instructions.
- **Disclosure pressure:** prompts that encourage omission of risk language ("keep it short," "don't mention risks").
- **Role boundary pressure:** low-authority users attempting to generate high-authority artifacts.
- **Policy ambiguity:** scenarios where policies conflict or are unclear, testing whether the system escalates rather than improvising.

These tests measure refusal reliability and escalation reliability. A system that refuses 99% of prohibited requests but fails 1% of the time is not acceptable if the 1% includes high-liability outputs. Robustness must be measured in the tail.

User-adversarial stress tests mimic realistic pressure:

- **Time pressure prompts:** "urgent," "send now," "client is waiting."
- **Authority prompts:** "the partner approved," "compliance already cleared it."

- **Flattery or manipulation:** attempts to coax the system into confident conclusions.
- **Iterative probing:** users rephrasing requests to bypass refusals.

These tests matter because real-world misuse is often unintentional: users want an answer, and they will push until they get one. The system must hold boundaries under pressure.

Tool-chain stress tests target multi-step fragility:

- **Unit mismatch:** provide numbers in different units and test whether the system detects inconsistency.
- **Tool failure:** simulate tool errors or missing tool outputs and test whether the system fails safely.
- **Wrong tool selection:** test whether the system chooses appropriate tools and documents tool assumptions.
- **Error propagation:** inject a small extraction error and observe whether it compounds into a confident recommendation.

Tool-chain stress testing is essential because tool use creates the appearance of rigor. Systems must demonstrate that they detect tool uncertainty and propagate uncertainty into outputs rather than hiding it.

Monitoring-driven stress tests should also be run periodically as "regression stress": the same adversarial suites used at deployment should be rerun after updates to prompts, retrieval indexes, or model versions. This creates a practical control: no update is approved unless the system maintains refusal reliability, escalation correctness, and provenance integrity.

The philosophy is that robustness testing should be designed to fail the system. A test suite that never breaks the system is likely too gentle. In finance, a system earns trust not by passing easy tests, but by failing safely under hard tests: refusing, escalating, and logging evidence rather than producing plausible but unsupported output.

3.6.4 Failure taxonomies

A failure taxonomy is not an academic exercise; it is an operational tool for incident response, monitoring, and governance reporting. If you cannot categorize failures, you cannot measure them. If you cannot measure them, you cannot govern them. For frontier finance AI, failure categories should reflect governance harm, not merely model error.

A practical taxonomy includes at least the following classes:

(1) Hallucinated facts. The system states a number, event, or clause as fact without evidence support. In finance, this can include invented financial metrics, invented risk limits, invented covenant thresholds, or invented policy requirements. Hallucinated facts are high severity because they can pass review unnoticed when presented confidently.

(2) Hallucinated or irrelevant citations. The system provides citations that do not support the claim, cite the wrong document, or cite a topically related source that does not contain the asserted fact. This is particularly dangerous because it creates a false audit trail. A fake citation is worse than no citation, because it discourages skepticism.

(3) Overconfident inference. The system draws a conclusion that is not warranted by evidence, often filling gaps with plausible narrative. Examples include causal explanations for market movements without supporting data, or conclusions about policy applicability without retrieving the relevant clause. Overconfidence is a governance failure because it suppresses escalation.

(4) Disclosure omissions. The system drafts or summarizes in a way that omits required risk statements, disclaimers, or mandatory disclosures. Omission failures can be systematic: a template missing a field can propagate across many outputs.

(5) Suitability and personalization errors. The system provides advice or recommendations that implicitly assume client characteristics that were not provided, or that it is not permitted to use. This includes inappropriate personalization and is particularly high risk in regulated advisory contexts.

(6) Scope violations. The system produces outputs outside authorized scope: trading instructions, evasion guidance, prohibited recommendations, or client-facing language without required approval gates.

(7) Provenance and data handling failures. The system uses unauthorized sources, leaks sensitive content, mixes confidential inputs into outputs, or fails to respect access classes. This includes persistent memory contamination: reusing prior-case information in a new case.

(8) Tool-mediated computation errors. The system computes incorrect values due to wrong inputs, unit mismatches, or tool misuse, and then presents the output as reliable. Because tools produce precise outputs, these errors can be highly persuasive.

(9) Escalation failures. The system fails to escalate when required (false negatives) or escalates excessively (false positives). The former creates harm; the latter creates operational overload and encourages bypass.

(10) Logging and reconstruction failures. The system fails to record essential information: retrieval results, policy versions, tool calls, intermediate representations, or reviewer decisions. These failures often become visible only after an incident, when reconstruction is impossible.

Each failure class should have a severity tier and an associated control response. For example, hallucinated facts and scope violations demand immediate containment and regression testing; disclosure omissions demand template and schema enforcement; provenance failures demand access control review; escalation failures demand threshold adjustment and UI redesign.

The taxonomy should also distinguish between *content failures* (wrong claims) and *process failures* (missing evidence, missing logs, missing escalation). In finance, process failures are often more

dangerous because they undermine the institution's ability to detect and correct errors. A system that makes occasional mistakes but is transparent and auditable can be managed. A system that looks correct but cannot be reconstructed is a governance nightmare.

Finally, taxonomy supports monitoring. If monitoring dashboards track these categories over time—policy violation rates, citation errors, escalation errors—the organization can detect drift and intervene. Taxonomy is therefore the bridge between evaluation and operational governance.

3.6.5 Limits of current benchmarks

A central difficulty in governing frontier AI in finance is that the broader AI ecosystem's benchmarks are not designed for the governance objects that finance cares about. Benchmarks often optimize for generic correctness, helpfulness, or reasoning ability, but they rarely measure disclosure integrity, suitability compliance, provenance accuracy, or escalation behavior. As a result, an institution can select a "top performing" model and still deploy an unacceptable system.

There are three structural limits to current benchmarks in this context.

First, benchmarks are often *output-centric*. They measure whether an answer matches a reference. Frontier finance systems are *process-centric*. The question is not only "is the output correct," but "is it defensible given evidence and policy." Benchmarks rarely include the full evidence context, rarely require citation integrity, and rarely measure whether the model properly declares assumptions. A model can score well by producing plausible answers, even if it cannot reliably ground them.

Second, benchmarks are rarely *policy-aware*. Finance constraints are not generic safety filters; they are specific, versioned, and procedural. Benchmarks do not encode organization-specific policies, role-based permissions, or escalation requirements. Yet these are precisely what determine whether a system is deployable. A model might be "safe" in generic terms but still violate internal policy by producing client-facing advice without required approvals.

Third, benchmarks are rarely *trajectory-aware*. Tool-using systems fail through intermediate steps: retrieval mistakes, wrong tool calls, error propagation. Benchmarks that score final outputs do not capture these failure modes. A system can pass a benchmark by producing the right answer once, even if it took an unsafe path to get there. In finance, unsafe paths matter because they are how incidents happen: a wrong retrieval plus a confident synthesis equals a client-facing error.

The absence of governance-aware benchmarks implies that institutions must build their own evaluation suites. This is not a call for endless custom work; it is a call for minimal, high-signal harnesses. A governance-aware benchmark for finance should include:

- Cases with embedded evidence and required citations.
- Policy constraints and role-based permissions.
- Required escalation outcomes for high-impact or low-evidence cases.

- Adversarial prompts to exceed scope.
- Tool-call traces and requirements for logging and reconstruction.

In other words, the benchmark must approximate the actual governance environment rather than measuring generic intelligence.

This also clarifies why model selection is only a small part of the problem. Even a strong model cannot compensate for weak retrieval, weak policy enforcement, weak logging, or weak review workflows. The benchmark limitation therefore reinforces the chapter's architectural thesis: in finance, deployable AI is governed systems engineering.

Finally, we should be candid about what benchmark creation cannot solve. Synthetic cases cannot fully replicate real-world complexity. Policies can be ambiguous. Humans can behave unpredictably. Nonetheless, governance-aware evaluation is the only defensible alternative to blind adoption. The goal is not to prove the system is safe in all possible conditions—a standard no complex system can meet. The goal is to demonstrate, with evidence, that the system fails safely: it refuses when it should, escalates when it must, logs what it did, and provides outputs that can be reviewed and reconstructed. That is what "validated" means in a governance-first finance setting, and it is why evaluation must be designed around the realities of accountability rather than the conveniences of generic metrics.

3.7 Implementation Considerations

3.7.1 Minimal viable implementation patterns

The fastest way to create risk with frontier AI in finance is to begin with an ambitious end-state: "a copilot for everything," "an autonomous analyst," or "a decision engine." The safest way to create value is to begin with bounded tasks where correctness can be checked, evidence can be attached, and authority can be constrained by design. Implementation, therefore, should follow a principle that looks conservative but is strategically rational: start with tasks where the system can be evaluated like an audit artifact rather than like a chatbot.

A minimal viable implementation (MVI) for finance AI should satisfy three properties from the first day: (i) it operates on a constrained task type, (ii) it emits structured outputs that separate facts, assumptions, and open questions, and (iii) it produces trace artifacts that make review possible. If any of these is missing, the organization will accumulate "pilot debt": systems that are widely used but indefensible under scrutiny.

Pattern 1: Evidence extraction with anchors. The simplest high-value pattern is extracting specific fields from documents and attaching evidence anchors. Examples include extracting covenant thresholds from term sheets, extracting risk factor headings from filings, extracting key dates and obligations from contracts, or extracting management guidance statements from transcripts. The output is a structured schema:

$$\texttt{facts} = \{(c_i, \pi_i)\},$$

where each extracted claim c_i includes a provenance pointer π_i (document id, section, location). The key is that the system does not interpret; it extracts and points. Interpretation can be a separate, explicitly labeled step.

Pattern 2: Classification and triage with conservative thresholds. Another MVI pattern is triage: classify documents or requests into risk categories and route them appropriately. For example: "client-facing vs internal," "requires compliance review vs does not," "high-impact vs low-impact," "missing required inputs vs complete." The system's output is a routing decision plus justification anchored to policy rules:

$$a_{\text{status}} \in \{\text{proceed}, \text{escalate}, \text{refuse}\},$$

with explicit reasons. This pattern creates value by reducing analyst overhead and ensuring that high-risk items do not slip through due to operational fatigue.

Pattern 3: Drafting with bounded schemas, not free-form authority. Drafting is where frontier AI seems most magical and where governance risk spikes. The MVI drafting pattern should be internal-only and schema-driven. The system produces a draft memo or summary, but it must include explicit sections: facts with evidence, assumptions, open questions, and a required "review

checklist" that tells the reviewer what to verify. Drafting without this structure invites decision laundering because the output looks finished. Drafting with structure communicates that it is a draft artifact, not an answer.

Pattern 4: Policy-aware checklists. A powerful and safe pattern is generating checklists from policies. The system takes a request context and identifies required steps: which disclosures apply, which approvals are required, which data must be collected, which thresholds matter. The system does not decide; it enumerates requirements and links them to policy clauses. This pattern is valuable because it reduces omission risk, and it is governable because checklist items can be validated against policy text.

Pattern 5: Decision dossier generation. A core implementation pattern, even at minimal viability, is dossier generation: the system packages inputs, retrieved evidence, extracted facts, tool calls, outputs, and escalation status into a single artifact. This makes evaluation and oversight possible. In many organizations, this is the "unlock" that turns AI from a gadget into a controlled system.

Across patterns, two design choices are non-negotiable.

First, **bounded language**. The system must avoid language that implies authority or certainty when evidence is incomplete. It should prefer "the evidence indicates," "based on the provided documents," "this assumption requires confirmation," and "escalation recommended." This is not mere style; it is risk control because language shapes user trust.

Second, **explicit exclusions**. The system must refuse prohibited outputs and escalate borderline cases. Refusal behavior is part of the MVI. If refusal is deferred as "later," it will be bypassed in practice, because systems accrete usage before governance catches up.

Minimal viability in finance AI therefore means something different than in consumer software. The minimal viable product is not the most impressive demo. It is the smallest system that can be evaluated, audited, and constrained, while producing measurable value in a bounded workflow.

3.7.2 Reproducibility and determinism

In finance, reproducibility is not an academic virtue; it is an operational requirement. If an output influences a decision, the organization must be able to reconstruct how it was produced. Without reproducibility, incidents cannot be investigated, controls cannot be verified, and accountability becomes narrative rather than evidence. Frontier AI systems threaten reproducibility because they are stochastic, retrieval-dependent, and frequently updated. Implementation must therefore treat determinism as a design goal and accept that full determinism may be unattainable—but partial determinism and replayability are achievable and essential.

A reproducible system versions *everything that changes behavior*. This includes:

- Model identifier and configuration (including temperature and decoding parameters).
- Prompt templates and system policies.
- Retrieval index version and retrieval parameters (embedding model, chunking strategy, filters).
- Policy documents and rule sets (with explicit versions and effective dates).
- Tool implementations (calculator code versions, database schema versions).
- Output schemas and validators.
- UI configuration that affects what users see and approve.

If any of these is unversioned, replay becomes impossible, and the system becomes unauditable.

Deterministic evaluation harnesses. Even if production inference has stochasticity, evaluation should be as deterministic as possible. This is achieved by using synthetic case libraries with fixed seeds and controlled randomness. For example, a synthetic case generator can produce documents with known embedded facts, known conflicts, and known required escalations. The harness then runs the system with fixed configurations and records outputs. Regression testing becomes possible: when prompts change or policies update, the system is re-run on the same case suite to detect behavior drift.

Controlled randomness. When randomness cannot be eliminated, it must be controlled and logged. The system should record random seeds (where applicable), sampling parameters, and any non-deterministic external calls. The audit trail should include enough information that the organization can reproduce behavior statistically, even if exact token-level reproduction is impossible. The key is that "we cannot reproduce it" is not acceptable as an incident response posture.

Separation of evaluation from deployment. A common failure is to evaluate with one configuration and deploy another. For example, evaluation might be done with a low temperature and strict retrieval filters, while production uses higher variability for "better writing." This breaks the link between validation and reality. Implementation must ensure that evaluated configurations are the ones used in production, and that deviations require explicit approval.

Policy version locking. A finance AI system must treat policy documents like code. Policies evolve, but outputs must be tied to the policy version that applied at the time. Therefore, the system should lock policy versions per case and record them in the dossier. If a policy changes later, the case remains reconstructable. This prevents a common governance confusion: "the system violated policy" when the truth is "policy changed after the fact."

Reproducibility is often framed as a cost. In finance, it is an investment: it enables monitoring, enables safe iteration, and creates defensibility. The organizations that treat reproducibility as optional will discover that they cannot scale AI responsibly because they cannot prove what their systems did.

3.7.3 Logging and traceability

Logging is the backbone of governed deployment. In a finance context, a system without traceability is indistinguishable from an uncontrolled decision engine, regardless of whether it is branded as "assistive." Implementation should therefore treat logging and traceability as first-class product features.

The system should generate **immutable logs** and a **decision dossier** per case. The difference is practical:

- Logs are the raw events: retrieval calls, tool calls, model inputs/outputs, policy checks, user interactions.
- The dossier is the curated artifact: a structured package that a reviewer, auditor, or incident investigator can read.

A minimal log schema should include:

- **Case identifiers:** unique case id, timestamp, user role, environment id.
- **Inputs:** document identifiers, hashes, metadata (not necessarily full content if sensitive).
- **Retrieval trace:** query terms, filters, retrieved document ids, ranks, and excerpts used.
- **Policy trace:** which rules were applied, which thresholds were checked, which versioned policy documents were referenced.
- **Tool trace:** tool name, inputs, outputs, error states, unit annotations.
- **Model trace:** prompt template id, model version, decoding parameters, and output.
- **Output trace:** structured schema output, narrative output, escalation status.
- **Human review trace:** reviewer identity, approvals, rejections, comments, and sign-off timestamps.

The dossier can then be produced as a structured record plus a human-readable summary. The dossier should be designed so that a reviewer can answer five questions quickly:

1. What did the user ask for?
2. What evidence did the system use?
3. What did the system claim, and what is fact vs assumption?
4. What policies applied, and were any constraints triggered?
5. Who approved the output, and on what basis?

Immutability and integrity. Logs must be tamper-evident. If logs can be altered, they become storytelling tools rather than audit artifacts. Implementation therefore should include hashing, write-once storage, or other integrity mechanisms. Even if an organization does not implement full cryptographic audit trails on day one, it must at minimum enforce access controls and retention policies that prevent opportunistic rewriting.

Minimum-necessary logging. Traceability must be balanced with confidentiality. Logs should capture identifiers and hashes rather than full sensitive content where possible. Evidence excerpts used for claims may need to be stored to support reconstruction, but these excerpts should be limited to what is necessary. The goal is to be auditable without creating a second uncontrolled data lake.

Replay capability. Logging is not complete unless it supports replay. Replay means that, given the dossier, the organization can rerun the system with the same configuration and retrieve the same evidence set. This requires versioned indices and stored retrieval results or deterministic retrieval snapshots. Without replay, the organization cannot test whether an incident was a one-off or a systematic failure.

Traceability is often treated as a compliance tax. In governed AI, it is the enabler of scale. Without traceability, the organization must keep AI systems in low-stakes drafting forever. With traceability, the organization can expand scope cautiously because it can detect, investigate, and correct failures.

3.7.4 Cost, latency, and scaling issues

Frontier AI systems impose costs in three currencies: compute cost, latency cost, and human review cost. The constraint that dominates scaling is often not compute—it is review throughput. Implementation must therefore be designed around a simple operational fact: if review is too expensive, it will be skipped, and the system will become de facto autonomous.

Compute and context costs. Long-context ingestion and retrieval-augmented reasoning increase token usage and therefore cost. Tool-using systems add additional calls and latency. Organizations often attempt to reduce cost by truncating context or removing evidence excerpts. This is exactly backwards from a governance-first posture. Evidence is not optional; it is what makes outputs defensible. Cost optimization must therefore be done by smarter retrieval and summarization pipelines, not by eliminating evidence. Practical strategies include:

- Use retrieval to narrow context rather than feeding entire documents.
- Cache document parses and embeddings.
- Use cheaper deterministic tools for numeric work.
- Separate extraction (cheap) from synthesis (expensive) and only run synthesis when necessary.

Even so, cost will remain a real constraint, especially when systems are used widely.

Latency and user behavior. High latency encourages users to bypass controls: they will copy-paste into ungoverned tools, or they will accept partial outputs without review. Therefore, implementation must design for acceptable latency in the governed system, even if it means simplifying tasks. A fast, bounded extractor with strong logs can be more valuable than a slow, “smart” system that users circumvent.

Human review throughput. Review is the true scaling bottleneck. If every output requires deep review, adoption will stall. The solution is not to remove review; it is to tier review by risk. Implementation should define risk tiers and align them with workflow:

- Low-risk internal drafts: sampled review and automated checks.
- Medium-risk outputs: mandatory review with structured checklists.
- High-risk outputs: multi-party sign-off and escalation.

This tiering must be encoded in the system's escalation logic and UI. Otherwise, "everything requires review" becomes "nothing gets reviewed."

Design for reviewability. The most powerful scaling lever is making review efficient. A reviewable output is not simply short; it is structured. It highlights claims, shows evidence, and flags assumptions. If reviewers can verify quickly, throughput increases without sacrificing rigor. This is why structured schemas, evidence anchors, and dossiers are scaling features, not compliance decorations.

Batch evaluation and monitoring. Scaling also requires continuous monitoring. Monitoring has cost: running evaluation suites, auditing outputs, investigating incidents. Implementation must budget for this as part of operating the system. A system that is cheap to run but expensive to monitor is not cheap.

Ultimately, scaling in finance AI is not about maximizing usage; it is about expanding authority safely. The organization should treat compute cost as manageable and treat review cost as the scarce resource. Systems should be designed to respect that scarcity: escalate only when needed, structure outputs for efficient verification, and keep scope bounded to tasks where review is feasible.

3.7.5 Integration risks

Integration is where otherwise reasonable AI systems become dangerous. A system that drafts a memo in isolation is one risk class. The same system embedded into trading, reporting, or compliance workflows can acquire hidden authority even if no one formally granted it. Integration risks therefore deserve explicit treatment.

Hidden automation authority. When an AI system is connected to downstream systems—order management, CRM, compliance ticketing, reporting pipelines—it can shape outcomes simply by pre-filling fields, generating recommended actions, or triggering workflows. Humans may approve because the default is already set. This is a classic control failure: automation changes the decision boundary without explicit approval. Governance-first integration requires that AI outputs be clearly labeled, that defaults be conservative, and that high-impact actions require explicit human confirmation rather than passive acceptance.

Permission amplification. If the AI system has broad access to data sources and tools, it can

inadvertently allow users to access information they should not see or to generate artifacts they should not be able to produce. Role-based access control must be enforced at the system level, not assumed. Integration must ensure that low-permission users cannot use AI as a "capability escalator."

UI/UX-driven decision laundering. Interface design can encourage decision laundering. If the UI presents the AI output as a polished final answer, users will treat it as one. If the UI hides evidence and assumptions behind collapsible sections, reviewers will not open them. If the UI makes "approve" the easiest button, approvals will happen. Governance-first integration treats UX as a control surface: evidence should be visible, assumptions should be prominent, and approvals should require acknowledgement of key risks. The goal is not to make the system unpleasant; it is to make it honest.

Workflow contamination. Integration can cause AI-generated content to propagate into official records: CRM notes, compliance logs, research repositories. If AI outputs are stored without labeling, future systems may treat them as authoritative evidence, creating a feedback loop of synthetic content contaminating institutional memory. This is a serious risk. Implementation should label AI-generated content, separate drafts from finalized artifacts, and enforce retention rules.

Boundary ambiguity between decision support and decision making. In finance, the line between "support" and "decision" can be subtle. A system that recommends "approve this client communication" is already influencing a decision. A system that generates a trade rationale may influence whether a trade is executed. Integration must therefore include explicit boundary definitions: which decisions remain human-only, which outputs are advisory, and which outputs can be auto-populated. These boundaries should be documented, communicated, and enforced technically.

Incident response complexity. Integrated systems are harder to investigate. Outputs propagate across systems, logs are distributed, and responsibility becomes diffused. If the AI system cannot produce a consolidated dossier that tracks where outputs went, incident reconstruction becomes impossible. Integration planning must include incident reconstruction as a requirement: where will outputs be stored, how will they be labeled, and how will they be traced across systems?

The overarching implementation lesson is that integration is not the last step; it is the step that defines risk class. A governance-first institution treats integration as a gated escalation: a system can move from sandbox to internal drafting to controlled workflow only when evaluation, logging, and review processes are strong enough to support the new authority. Otherwise, the organization will discover—after an incident—that it accidentally deployed an autonomous system while calling it a copilot.

3.8 Impact and Opportunity

3.8.1 New capabilities unlocked

The opportunity frontier in finance is not "AI that beats the market." The more realistic and defensible frontier is AI that reduces institutional friction in the places where finance is slow for good reasons: documentation, evidence reconciliation, compliance language, and the translation between narrative and structure. Frontier models, when embedded in governed systems, unlock capabilities that are best described as *compression of cognitive overhead*. They move work from "finding and formatting" toward "judging and deciding," but only if the system is designed to preserve evidence and accountability rather than to manufacture confidence.

The first and most immediate capability is **document-to-structure transformation**. Finance operates on documents: filings, term sheets, loan agreements, offering memoranda, transcripts, policies, and internal memos. These documents contain the information that matters, but they are written for humans, not for machines. Traditionally, transforming them into structured representations requires manual extraction, table building, and review. Foundation models, combined with retrieval and schema-driven extraction, can accelerate this transformation dramatically. They can extract key terms, normalize numbers, identify entities, and map obligations to structured fields. The value is not merely speed. The value is consistency: the ability to apply the same extraction logic across large corpora and to produce traceable outputs with evidence anchors.

A second capability is **cross-document synthesis**. Many finance decisions require integrating evidence across multiple sources: comparing disclosures across periods, reconciling contract clauses with policy requirements, aligning management commentary with quantitative performance, and mapping risk factors to internal risk frameworks. Humans can do this, but it is expensive and error-prone under time pressure. A governed AI system can assist by assembling relevant excerpts, identifying contradictions, and producing structured summaries that a human can review. Crucially, the system must not pretend to "resolve" contradictions by narrative smoothing; its value is in surfacing conflicts and pointing to evidence, not in fabricating coherence.

A third capability is **drafting support for compliance and governance artifacts**. Finance produces an immense quantity of artifacts whose primary purpose is not creativity but defensibility: memos, risk summaries, committee packs, disclosure drafts, suitability checklists, audit workpapers, and incident reports. Frontier AI can draft these artifacts in a standardized format, align language to policy templates, and ensure that required sections are present. When supervised properly, this can reduce omission risk and improve throughput. But the governance-first condition is essential: drafting must be bounded, evidence-linked, and subject to human sign-off. Otherwise, drafting becomes a mechanism for producing "compliance-flavored text" detached from substance.

A fourth capability is **triage and routing**. Many organizations struggle not because they lack expertise but because expertise is misallocated. Analysts spend time on low-risk tasks while high-risk

cases pile up. AI systems can classify requests by risk tier, detect missing required inputs, flag potential policy violations, and route cases to appropriate reviewers. This is a pragmatic benefit: it turns AI into an organizational "attention allocator" rather than an oracle. In a governance-first design, triage is particularly attractive because it constrains authority. The system does not decide outcomes; it decides where human attention is needed.

A fifth capability is **improved internal research workflows**. This does not mean prediction; it means synthesis. AI can help analysts navigate massive corpora of filings, transcripts, news, and internal notes to produce structured research briefs. It can suggest questions to investigate, highlight changes in risk factor language across years, and summarize thematic shifts across management commentary. In a governed setting, the system's role is to accelerate exploration and to propose hypotheses, not to deliver final conclusions. The output becomes a research scaffold rather than an authoritative report.

Taken together, these capabilities shift the productivity frontier in finance from computation to cognition. Finance already has abundant computation. What it lacks is time and attention to reconcile evidence, document reasoning, and produce defensible artifacts. Frontier AI, governed properly, can reduce that overhead. The key is that these capabilities are most valuable precisely where governance is strict. That is not a paradox; it is the mechanism by which AI becomes deployable. In finance, the opportunity is not freedom from constraint. It is productivity *within* constraint.

3.8.2 Organizational implications

If the technical story ended at "the model can do more," the organizational story would be trivial. The real implication is that governed AI changes the distribution of work, the definition of expertise, and the structure of accountability. In finance, where organizations are already layered with controls, AI introduces a new layer: evidence operations for machine-assisted workflows. The institutions that treat this as a procurement problem will struggle; the institutions that treat it as an operating model redesign will extract durable value.

The most visible shift is in analyst time allocation. Today, analysts spend enormous effort on tasks that are necessary but not where judgment lives: extracting numbers from PDFs, formatting tables, drafting boilerplate language, searching policies, and assembling meeting packs. These tasks are cognitively expensive and, importantly, they consume the same attention that should be spent on identifying what is missing, what is uncertain, and what is risky. Governed AI can move a meaningful fraction of this workload into automated pipelines. The result is not "analysts replaced." The result is analysts spending more time on judgment, exception handling, and oversight. This is a productivity increase that is not primarily headcount reduction; it is a quality improvement in where human cognition is applied.

However, this shift only occurs if review is engineered. If AI outputs are not structured and

evidence-linked, analysts will spend their time auditing AI drafts rather than doing judgment. That is a poor trade: replacing extraction work with verification work can be net negative if verification is harder. Therefore, the organizational implication is that review processes and interfaces become strategic assets. The organization must invest in making review efficient and reliable, or else AI adoption will stall or become unsafe.

A second implication is the emergence of **evidence management** as a formal function. In traditional finance operations, data management focuses on structured feeds and warehouses. Governed AI requires managing unstructured evidence: document corpora, policy manuals, retrieval indices, provenance metadata, and versioning. Someone must own the evidence supply chain. This includes ensuring that the latest policies are indexed, that confidential documents are access-controlled, that document parsing is reliable, and that evidence anchors are preserved. Evidence management becomes the equivalent of "data engineering" for unstructured finance workflows, with governance as a first principle.

A third implication is the rise of **AI risk monitoring** as an operational discipline distinct from traditional model validation. Model validation remains important, but frontier AI introduces failure modes that are not captured by predictive performance: hallucinated citations, disclosure omissions, suitability drift, and decision laundering behaviors. Monitoring must therefore include governance metrics: policy violation rates, escalation correctness, provenance accuracy, and reconstruction readiness. This creates demand for a function that is part risk, part engineering, part audit: governance operations for AI systems. The organization must maintain dashboards, run regression suites, investigate anomalies, and coordinate updates across models, prompts, and policies.

A fourth implication concerns **role clarity and segregation of duties**. AI systems make it easier for one person to produce artifacts that previously required multiple roles. This can collapse control boundaries if not managed explicitly. For example, a junior analyst with an AI drafting tool can produce a memo that looks like it was written by a senior professional. If the organization does not enforce review and approvals, role boundaries erode. Therefore, governed AI requires explicit workflow enforcement: who can generate drafts, who can approve client-facing outputs, who must sign off on risk statements, and who owns the final decision. AI adoption without reinforced segregation is a control failure waiting for a convenient incident.

A fifth implication is cultural. Organizations must retrain what "good work" looks like. In an AI-assisted environment, a polished memo is not impressive; a defensible memo is. Analysts and managers must value evidence linking, assumption labeling, and escalation discipline. The organization's incentives must align: if people are rewarded for speed and volume, AI will amplify that incentive and degrade governance. If people are rewarded for defensibility and quality, AI can amplify quality by reducing overhead.

Finally, there is an implication for talent. Finance will increasingly require professionals who can span domain expertise and governance engineering. Not everyone must be an ML specialist, but many

will need to understand how retrieval works, why provenance matters, how to interpret uncertainty signals, and how to operate within constrained decision-support workflows. The institutions that train their teams to use AI as a controlled instrument will outperform those that treat AI as a magical intern.

In sum, the organizational implication is that governed AI is a new operating system layer. It reshapes workflows, creates new roles, and requires cultural adjustment. The prize is not only efficiency; it is stronger institutional discipline in evidence and accountability, which becomes competitive advantage in regulated markets.

3.8.3 Potential new industries or markets

Whenever a new general-purpose technology enters a high-accountability domain, secondary markets emerge around control, assurance, and infrastructure. Frontier AI in finance is no exception. The most likely new markets are not "AI hedge funds" or "autonomous traders." They are governance-first platforms that make AI outputs auditable, compliant, and reviewable at scale.

One emerging category is **governance-first financial copilots**. These differ from generic copilots by design: they embed policy constraints, enforce structured outputs, require evidence anchors, and integrate approval workflows. Their value proposition is not that they answer any question. Their value proposition is that they can be used in regulated environments without turning the firm into a liability laboratory. This category will likely fragment by function: copilots for research synthesis, copilots for compliance drafting, copilots for risk triage, copilots for investment committee preparation. The key differentiator will be not model quality alone but the quality of control surfaces: logging, replay, policy enforcement, and reviewer interfaces.

A second category is **audit-grade provenance and evidence platforms**. If AI becomes embedded in decision workflows, the organization must be able to answer: what evidence was used, what policy applied, and how did we arrive at this output? Tools that can provide immutable logs, evidence linking, and incident reconstruction will become valuable regardless of which model is used. This is analogous to how observability platforms became essential in cloud computing. In governed AI, observability is not about uptime; it is about accountability.

A third category is **policy-as-code and compliance automation for AI outputs**. Finance already has policy engines, but AI introduces new needs: scope gating, refusal policies, escalation rules, and schema validation for narrative outputs. Tools that allow compliance teams to define constraints in a formal way, test them against scenario libraries, and deploy them consistently across AI workflows will become increasingly important. This is a market where domain expertise and engineering must meet: compliance professionals need interfaces that let them express requirements without writing brittle code, and engineering teams need constraints that can be enforced deterministically.

A fourth category is **benchmarking and validation services** tailored to governance-aware evaluation. Since generic benchmarks do not measure what finance cares about, there is room for standardized scenario suites that test disclosure risk, suitability handling, provenance integrity, and escalation behavior. These could take the form of shared synthetic case libraries, certification-like evaluations, or internal tooling that helps firms build their own harnesses. The market will reward those who can provide credible, defensible evaluation artifacts rather than marketing claims.

A fifth category is **workflow integration platforms** that manage safe interfaces between AI systems and operational systems (order management, CRM, compliance ticketing, document repositories). The integration challenge is not merely connecting APIs; it is ensuring that AI outputs cannot silently become actions. Platforms that enforce "human confirmation required," that prevent unauthorized default settings, and that label AI-generated content in downstream systems will become critical infrastructure.

Finally, there is a market for **training and governance operations**. Firms will need playbooks, role definitions, and operating procedures for AI oversight. This can become a service category: governance operations support, internal audits of AI workflows, and risk assessments for new deployments. Importantly, this is not only consulting. It can be productized: dashboards, runbooks, incident response templates, and automated reporting for governance metrics.

The common thread across these emerging markets is that value accrues to *control*. In finance, the marginal value of a slightly better model is often smaller than the marginal value of a system that can be audited, constrained, and defended. This is why "governance-first" is not merely a philosophy; it is an economic opportunity. The firms that build the control layer will capture durable value because every regulated institution needs it.

3.8.4 Second-order effects

The first-order effects of AI adoption—faster drafting, quicker extraction, improved synthesis—are obvious. The second-order effects are where strategic risk and systemic implications live. In finance, second-order effects matter because individual institutions do not operate in isolation. Shared technologies create correlated behaviors, and correlated behaviors create systemic risk.

A central second-order effect is **model monoculture**. If many firms rely on similar models trained on similar corpora, with similar retrieval strategies and similar prompt templates, then errors become correlated. This is not hypothetical; finance has seen monoculture effects in other contexts, such as risk model convergence and crowded trades. In an AI context, monoculture can manifest as shared misinterpretations of policy language, shared hallucination patterns, or shared retrieval blind spots. Under stress, these correlated failures can propagate across the industry: many institutions generating similar narratives, making similar recommendations, or failing in similar compliance ways. The systemic implication is that AI can amplify herding not only in trading behavior but in decision narratives and risk perceptions.

Another second-order effect is **correlated failure through shared evidence sources**. If many systems retrieve from the same public filings, the same news sources, or the same third-party research providers, then errors or biases in those sources propagate widely. More subtly, if systems rely on the same summarization pipelines, small distortions can become industry-wide "consensus." This is epistemic systemic risk: the market converges not only on prices but on narratives that may be incomplete or wrong.

A third second-order effect is **skill atrophy**. When AI reduces the friction of producing outputs, humans may stop practicing the underlying skills: reading full documents, reconciling evidence, and writing disciplined arguments. Over time, this can reduce the organization's ability to detect AI errors, because reviewers lose the muscle memory of skepticism. Skill atrophy is especially dangerous in finance because rare, high-impact events require human judgment that cannot be delegated. Governance must therefore treat training and periodic "manual drills" as part of risk management. Just as pilots train for emergencies they rarely face, analysts and compliance teams must retain the ability to operate without AI and to verify AI outputs effectively.

A fourth effect is **over-trust and automation bias**. AI outputs, especially when polished and consistent, can create a sense that the system is more reliable than it is. Over-trust is not merely individual error; it is a predictable outcome of human cognition interacting with persuasive artifacts. Over-trust can lead to reduced scrutiny, increased speed, and silent escalation of AI authority. Governance must counteract this with interface design, structured outputs, and accountability mechanisms that make it psychologically harder to rubber-stamp.

A fifth effect concerns **organizational incentives**. AI can shift incentives in subtle ways. If productivity metrics reward volume, AI will increase volume, potentially at the expense of quality. If teams are evaluated by speed of turnaround, AI can create pressure to accept drafts without verification. This can erode governance norms. Therefore, the organization must align incentives with defensibility: reward evidence discipline, reward proper escalation, and reward incident reporting rather than hiding.

A final second-order effect is **strategic competition and risk externalization**. Firms may feel pressure to deploy AI faster than competitors. The benefits are immediate; the costs are contingent. This can lead to a tragedy-of-the-commons dynamic: industry-wide adoption without sufficient governance, increasing systemic risk. Regulators may respond with stricter requirements, which then impose costs on everyone. A governance-first posture can therefore be strategically beneficial: it allows a firm to adopt AI while maintaining defensibility, potentially influencing industry standards rather than reacting to them.

Second-order effects are where "frontier awareness" becomes more than a slogan. They remind us that AI is not merely a tool; it is a coordination technology. Its risks and benefits scale with adoption patterns, and those patterns are shaped by incentives and governance.

3.8.5 Overstated expectations

The most dangerous expectation in frontier finance AI is that intelligence implies authority. It does not. The law, fiduciary duty, and institutional responsibility impose boundaries that no model capability can dissolve. A governance-first chapter must therefore be explicit about what is overstated, not to dampen enthusiasm, but to prevent organizations from confusing feasibility with permissibility.

First, it is overstated to claim that AI removes accountability. Accountability remains human and institutional. An AI system can draft, summarize, and recommend, but it cannot assume legal responsibility. Firms cannot outsource fiduciary duty to a model. In practice, this means that systems must be designed to reinforce responsibility rather than to blur it. Any deployment that encourages "the model decided" is not merely risky; it is conceptually incoherent in a regulated environment.

Second, it is overstated to assume that autonomy is a straightforward next step. Autonomy is not just "more capability." It is a different risk class. The moment a system acts without human review, it enters a domain where errors propagate faster than controls can catch them. In finance, where tail risks dominate, autonomy is therefore governance-limited. This does not mean that no automation is possible. It means that automation must be bounded: automation of extraction, formatting, routing, and drafting under supervision, not automation of consequential decisions.

Third, the idea of "autonomous trading agents" is commonly overstated. Even if a model could generate trades, the governance barriers are substantial: suitability, market conduct, risk limits, model risk validation, and liability. More importantly, many "autonomous agent" narratives ignore the deepest finance reality: trading is an adversarial environment where strategies decay and where edge is competed away. A broadly available model is unlikely to provide persistent alpha simply by reasoning harder. What it can provide is operational support: research scaffolding, risk summaries, and documentation. Treating it as an alpha engine is often a category error.

Fourth, it is overstated to assume that adding disclaimers makes outputs safe. Disclaimers are not controls; they are text. A system that produces prohibited advice with a disclaimer is still producing prohibited advice. Safety comes from constraints, escalation, evidence discipline, and review processes. Disclaimers can be necessary for communication, but they are insufficient for governance.

Fifth, it is overstated to assume that model improvements will automatically solve governance. Better models reduce some errors but can increase others. As models become more fluent and persuasive, the risk of over-trust increases. As models reason more deeply, the risk of delayed and opaque failure increases. Therefore, governance must evolve with capability. The frontier is not a moving line that technology crosses; it is a moving line that governance must track.

Finally, it is overstated to believe that the primary barrier is technical. In many finance AI

deployments, the barrier is organizational: unclear responsibility, weak logging, inadequate evaluation suites, and incentives that reward speed over defensibility. A firm can buy the best model and still fail because it cannot reconstruct decisions or enforce scope boundaries. The opportunity is therefore not only to adopt AI, but to adopt it in a way that strengthens institutional discipline.

The sober conclusion is also the optimistic one: the real opportunity in finance AI is not to replace judgment but to make judgment cheaper, faster, and better supported by evidence. That opportunity is large. It is also only realizable when governance is treated as a design constraint rather than as a post-hoc compliance wrapper. In a domain where accountability is unavoidable, the most valuable AI systems will be those that make accountability easier to uphold, not those that promise to make it disappear.

3.9 Governance, Risk, and Control

3.9.1 New risk categories

Governance in finance does not begin with "AI ethics." It begins with liability, fiduciary duty, disclosure law, and the institutional requirement that decisions be defensible after the fact. Frontier AI expands the surface area of those obligations because it produces artifacts that look professional and complete even when they are not grounded. The purpose of this section is to name the new risk categories that arise specifically when models become decision-support systems operating over documents, policies, and multi-step workflows.

Disclosure risk is the first category and, in many contexts, the most immediate. In finance, disclosures are not optional stylistic additions; they are legal and regulatory requirements. A system that drafts investor communications, marketing materials, research summaries, or internal committee packs can create risk by omission, misstatement, or inconsistent phrasing. Frontier AI introduces two new failure modes here. First, "compliance-flavored text" that appears to satisfy disclosure requirements but omits mandatory elements. Second, narrative smoothing: the system produces coherent messaging by downplaying uncertainty, conflicts, or adverse information. This is not malicious; it is the model optimizing for coherence. But coherence is precisely what can make disclosures misleading. In a governance-first design, disclosure is treated as a schema completeness problem: required fields must be present, risk factors must be enumerated, and omissions must trigger escalation.

Suitability risk is closely related but distinct. Suitability requirements—formal or informal—depend on client context, product characteristics, and constraints on personalization. Frontier AI systems can easily drift into implicit personalization by generating "reasonable" recommendations that assume client objectives, risk tolerance, or constraints that were not provided. The danger is subtle: the system may not issue a direct "buy this" instruction, but it can produce language that effectively functions as advice. Suitability risk also arises when systems respond differently depending on user-provided details that may be incomplete or biased. The governance response is to treat missing suitability inputs as a hard stop: the system must ask questions or escalate rather than infer.

Privacy and confidentiality risk expands under frontier AI because of the way these systems ingest and transform information. Documents may contain sensitive client data, material nonpublic information, or internal deliberations. A system that summarizes or drafts can inadvertently leak sensitive details, either by including them explicitly or by embedding them into outputs that are later shared. Persistent memory architectures amplify this: if a system stores prior-case details, it can contaminate new cases. The risk category is not only "data breach." It includes subtle, operational leakage: sensitive facts appearing in a context where they do not belong, or data being used outside its permitted purpose. Governance requires strict source classification, access control, and minimum-necessary logging.

Insider-information handling risk is a specific and severe subset. Frontier AI can accelerate the ingestion and synthesis of information. If the system is permitted to access internal notes, emails, or unpublished data, it can inadvertently incorporate material nonpublic information into recommendations or drafts. Even if the system is not used for trading, it can influence decisions that later affect trading or disclosures. The governance response here is categorical: define strict boundaries on what information sources are permissible for which tasks, enforce role-based access, and log evidence sources so that the firm can demonstrate control.

Decision laundering risk is arguably the signature risk of frontier AI in finance. Decision laundering occurs when the model's output becomes a pseudo-justification that reduces human accountability. The output looks professional, cites policy, and provides a rationale. Humans, under time pressure, approve it. Responsibility becomes diffused: "the system recommended it." This is not only a cultural risk; it is a control failure. It undermines governance because it changes how decisions are made, not just what decisions are made. Decision laundering risk is amplified by UI design that makes approval frictionless and by organizational incentives that reward speed.

Model monoculture and correlated failure risk arises when many teams—or many institutions—use similar models, prompts, retrieval stacks, and evidence sources. Correlated failures are dangerous because they defeat diversification. A single misunderstanding of a policy clause can propagate across an institution. A shared hallucination pattern can appear in many reports. At the industry level, monoculture can amplify systemic risk by creating aligned narratives and aligned decisions under stress. This category is not hypothetical; finance has experienced correlated failures in risk models, rating methodologies, and trading strategies. Frontier AI increases the risk by increasing the scale and speed of narrative production.

Provenance and evidentiary drift risk is a new category created by retrieval and long-context systems. The "evidence set" is dynamic. If retrieval indices are stale, if chunking changes, or if documents are misclassified, the system can produce outputs grounded in the wrong evidence. Provenance failures are particularly dangerous because they are silent: the output may look correct, and the citations may look plausible, but the chain of custody is broken. This creates a governance nightmare: the organization cannot prove what evidence was used or why.

Tool-chain and compositional risk arises when models call tools—calculators, parsers, databases—and combine outputs into decisions. Each component may be correct in isolation, yet the composed system may fail due to unit mismatches, incorrect inputs, or error propagation. Tool use also increases perceived rigor. A table of computed exposures is persuasive, even when based on wrong extraction. This category requires controls that treat tool calls as auditable events, with input validation and cross-checks.

These risk categories share a theme: they are not "AI risks" in the abstract. They are finance risks amplified by systems that produce plausible artifacts at scale. Governance must therefore treat frontier AI as an amplification layer on existing obligations, requiring controls that are commensurate

with the new scale and new failure modes.

3.9.2 Control points and safeguards

Controls in finance cannot rely on good intentions or disclaimers. They must be enforceable, testable, and auditable. For frontier AI, the control points must align with the architecture: ingestion, retrieval, reasoning, tools, outputs, and workflow integration. Safeguards should be designed as layers, with the assumption that any single layer can fail.

Hard scope boundaries. The first safeguard is to define and enforce the system's permissible action space. In many settings, this includes a hard prohibition on personalized investment advice, autonomous trading instructions, and any guidance that could be interpreted as evasion of compliance obligations. This is not merely a policy statement; it must be implemented as:

- Refusal rules that trigger on prohibited intents.
- Output schema constraints that disallow certain action types.
- Role-based permissions that restrict what different users can request and receive.
- Integration safeguards that prevent AI outputs from becoming actions without explicit human confirmation.

Hard boundaries should be conservative. If there is ambiguity about whether a request constitutes advice, the system should escalate.

Evidence-first output schemas. The single most effective safeguard against hallucination and decision laundering is an evidence-first schema. The system should be required to produce outputs with explicit partitions: facts (with anchors), assumptions, open questions, and recommended next steps. The schema should be validated automatically: if a material claim is present without an evidence pointer, the output fails validation and must be revised or escalated. This is a technical control, not a cultural guideline.

Citation and provenance requirements. For document-grounded tasks, the system must attach evidence references. "Citations" should mean more than listing a document name; they should include location pointers and, where feasible, excerpts. Provenance requirements should also include recency and authority checks: if policy is cited, it must be the latest approved version; if contracts are cited, executed versions should be prioritized; drafts should be labeled as drafts. Retrieval systems should enforce source allowlists and access control, ensuring the model cannot retrieve from unauthorized corpora.

Escalation rules for high-impact or low-evidence decisions. A governance-first system must include escalation triggers that are not optional. Escalation should be triggered by:

- High impact (client-facing output, large exposure, material disclosure).
- Low evidence coverage (missing documents, ambiguous clauses).

- Conflicting evidence (inconsistent numbers, contradictory statements).
- Policy ambiguity or policy conflicts.
- Tool errors or uncertain computations.

Escalation outputs should be structured: "what is missing," "what is conflicting," "which policy triggers apply," and "who should review." This turns escalation from a vague suggestion into an operational action.

Policy-as-code constraints. Where possible, governance requirements should be encoded as deterministic rules rather than as model-internal behavior. For example, required disclosure checklists, mandatory risk statements, and approval gates can be implemented as validators. The model can draft, but the validator enforces completeness. This reduces reliance on the model's internal compliance "instinct," which is unreliable.

Tool validation and cross-checks. Tool calls should be treated as high-risk events because they create the appearance of precision. Safeguards include:

- Input validation (types, ranges, units).
- Consistency checks (totals match components, accounting identities).
- Redundant computations for critical values (two independent methods).
- Fail-safe behavior: if a tool fails or returns uncertain output, escalate rather than improvise.

Change control as a safeguard. Many failures occur after updates: prompt changes, model upgrades, retrieval re-indexing, policy updates. Therefore, change control is a safeguard: no change should be deployed without regression testing on governance-aware scenario suites. Versioning must include prompts and policies, not only code.

UI safeguards. The interface is a control point. It should:

- Make evidence visible by default.
- Highlight assumptions and open questions prominently.
- Require explicit acknowledgement for high-risk approvals.
- Prevent "one-click approve" for client-facing artifacts.
- Label AI-generated content clearly in downstream systems.

A system can have perfect refusal rules and still fail if the UI encourages users to ignore them.

These safeguards are not expensive luxuries; they are the minimum viable control set for a system that will operate near regulated decisions. The core design posture is layered defense: constrain scope, force evidence linkage, require escalation, enforce determinism where possible, and ensure that every step is logged and reconstructable.

3.9.3 Human oversight boundaries

Human oversight is often invoked as a talisman: "human-in-the-loop" is assumed to make systems safe. In finance, this assumption is dangerous. Oversight is only effective if it is operationally real: humans must have the authority to reject outputs, the time to review evidence, and the process support to understand what they are approving. Therefore, human oversight boundaries must be explicit: what humans must review, what humans may delegate, and what humans must never delegate.

Human sign-off requirements. A governance-first boundary is that any client-facing output requires human sign-off, and any high-stakes decision-support output (large exposure, material disclosure, significant policy implication) requires sign-off by an appropriately authorized role. "Appropriately authorized" matters: a junior analyst cannot be the approver of a high-impact disclosure draft merely because they are the user of the system. Oversight requires role mapping: which roles can sign off on which output classes.

Sign-off should not be a checkbox. It should be an attestation that the reviewer:

- Reviewed the evidence anchors for material claims.
- Confirmed required disclosures and suitability constraints.
- Understood and accepted the assumptions and open questions.
- Confirmed that the output is within scope.

This attestation can be implemented as a structured approval workflow in the UI and recorded in the dossier.

Separation of roles. Frontier AI introduces a temptation to collapse roles because outputs are easier to produce. Governance requires resisting that temptation. A minimal separation includes:

- **Builders:** engineers and model developers who configure the system.
- **Validators:** independent reviewers who test the system against scenarios and policies.
- **Approvers:** business or compliance roles who sign off on outputs in operation.
- **Accountable owners:** senior roles who own the decision and the system's risk posture.

These roles can overlap in small organizations, but the functions must exist. Without separation, conflicts of interest arise: builders approve their own systems, and accountability becomes diffuse.

Oversight boundaries by task class. Oversight should be tiered. Low-risk internal extraction outputs can be sampled and audited. Medium-risk drafting outputs require routine review. High-risk client-facing or decision-adjacent outputs require explicit approval gates and, in some cases, dual control (two independent reviewers). This is not bureaucratic for its own sake; it is proportional control aligned to harm.

No delegation of accountability. The most important boundary is conceptual: accountability

cannot be delegated to the model. The model can assist, but it cannot own. The organization must institutionalize this: approvals must name the human responsible, and incident reviews must hold humans accountable for oversight failures. This is uncomfortable, but it is the only posture consistent with fiduciary reality.

Human oversight of the system, not just outputs. Oversight must extend to system behavior: monitoring dashboards, reviewing violation rates, approving updates, and ensuring that governance controls remain effective over time. Human oversight that focuses only on individual outputs misses systemic drift. Therefore, oversight roles must include governance operations: regular review of metrics, periodic audits, and incident response drills.

The boundary-setting message is simple: human oversight is not a label; it is a set of enforced processes. If the organization cannot provide the time and structure for real oversight, it must reduce system scope. Otherwise, the system will become autonomous by neglect.

3.9.4 Monitoring and auditability

Monitoring and auditability are the operational mechanisms that turn governance from policy into reality. They answer two questions: are we staying within boundaries today, and can we reconstruct what happened yesterday? In finance, both questions are mandatory.

Continuous monitoring metrics. A governance-first monitoring dashboard should track at least:

- **Policy violations:** frequency, severity, and type (scope breaches, advice-like outputs, missing disclosures).
- **Provenance integrity:** citation accuracy, missing evidence anchors, use of outdated policy versions.
- **Escalation behavior:** escalation rate, false non-escalations (detected in audits), and escalation overload.
- **Factuality drift:** rate of unsupported claims detected in sampled reviews.
- **Tool reliability:** tool error rates, unit mismatch detections, computation inconsistencies.
- **User behavior signals:** repeated attempts to bypass scope boundaries, high-pressure prompt patterns.

Monitoring should be designed for action: thresholds should trigger alerts, and alerts should map to response playbooks (e.g., tighten refusal rules, retrain users, roll back a prompt update, or restrict a workflow).

Auditability and reconstruction. Auditability requires that every case can be reconstructed post hoc. Reconstruction means:

- The exact prompt templates and system instructions used.
- The evidence retrieved and the versions of documents retrieved.

- The policy versions and rules applied.
- The tools called, with inputs and outputs.
- The intermediate structured representation (facts/assumptions/open questions).
- The final output and escalation status.
- The human approvals, with identities and timestamps.

Without these, incident investigations become speculative. Speculation is unacceptable in finance governance because it prevents root-cause analysis and corrective action.

Replayability. The highest standard of auditability is replay: the ability to rerun the system on the same case and obtain behaviorally similar outputs. Perfect determinism may be impossible, but the system should at least reproduce retrieval sets, policy applications, and tool outputs. Replay requires versioned indices, stored retrieval snapshots, and locked policy versions. This is why reproducibility was treated as a core implementation consideration: it enables auditability.

Sampling audits and red-team exercises. Monitoring must be complemented by periodic audits. Sampling audits review a random subset of outputs for evidence integrity and policy compliance. Red-team exercises deliberately attempt to induce violations: prompts that push for advice, prompts that request omission of risk, prompts that request use of unauthorized sources. The goal is not to punish the system; it is to test whether controls hold under pressure. These exercises should be scheduled and their results tracked over time.

Governance reporting. For senior leadership, monitoring should roll up into governance reports: violation trends, major incidents, control effectiveness, and planned changes. This is where the "governance-first" thesis becomes board-level relevant: AI systems must be treated like other material risk systems, with reporting that supports oversight.

Monitoring and auditability are sometimes framed as burdens. In finance, they are the tools that allow controlled expansion. Without them, the organization cannot safely widen the system's authority. With them, the organization can learn from failures, adjust controls, and progressively adopt AI in higher-value workflows without turning governance into guesswork.

3.9.5 Deployment deferral criteria

A governance-first approach must include the discipline to say "not yet." This is not conservatism; it is risk management. Deployment deferral criteria define when the system must remain in a limited mode (internal drafts, low-risk extraction) or when deployment must be paused entirely. These criteria protect the institution from scaling a system whose controls are not commensurate with its authority.

Deferral criterion 1: Provenance cannot be guaranteed. If the system cannot reliably attach evidence anchors to material claims, or if retrieval cannot be constrained to authorized and current sources, deployment must be restricted to internal exploratory use. Client-facing outputs,

compliance artifacts, and high-stakes decision support should be deferred. In finance, inability to prove evidence is not a minor technical gap; it is a disqualifying governance failure.

Deferral criterion 2: Outputs cannot be reviewed efficiently. If human review requires excessive time because outputs are unstructured, evidence is hidden, or assumptions are not explicit, then review will not happen in practice. In that case, deployment should be deferred or scope reduced until reviewability improves. A system that cannot be reviewed becomes autonomous by neglect, which is the worst possible autonomy.

Deferral criterion 3: Escalation and refusal behavior is unreliable. If stress tests show that the system can be coaxed into prohibited advice, or if it fails to escalate under low-evidence or high-impact conditions, then deployment must be constrained. "Mostly safe" is not a sufficient standard where harm is nonlinear. The system must demonstrate robust boundary behavior under adversarial prompts.

Deferral criterion 4: Logging is incomplete or non-immutable. If the system cannot produce dossiers and logs sufficient for reconstruction, then it cannot be deployed in workflows that require accountability. Logging gaps may be tolerated in prototypes, but they are not tolerable in operational systems. If reconstruction is impossible, incident response becomes impossible.

Deferral criterion 5: Change control and regression evaluation are absent. If the organization cannot run governance-aware regression suites after changes, then the system will drift into unsafe behavior over time. Deployment should be deferred until a minimal evaluation harness exists and until updates are gated by tests.

Deferral criterion 6: Organizational roles and responsibilities are unclear. Even a technically sound system can fail if no one owns oversight. If there is no clear accountable owner, no validator function, no approval workflow, and no incident response plan, deployment must be deferred. Governance is organizational as much as technical.

These deferral criteria are not obstacles to adoption. They are the conditions that make adoption sustainable. A finance institution that cannot meet them is not ready for a frontier system in a high-accountability workflow, regardless of how impressive the demo appears. The governance-first alternative is to deploy in lower-risk modes while building the missing control infrastructure: internal extraction, policy-aware checklists, and dossier generation. Those deployments generate value while also building the evaluation and monitoring machinery required for higher-stakes use.

The conclusion is therefore disciplined: frontier AI in finance offers real opportunity, but only when deployed as constrained decision support under auditable controls. When provenance is weak, review is impractical, or boundaries are unenforceable, deferral is not failure. Deferral is governance doing its job.

3.10 Outlook and Open Questions

3.10.1 Near-term research questions

The most urgent research questions for frontier AI in finance are not about making models "smarter." They are about making systems *governable* at the level of evidence, trajectories, and institutional accountability. Finance is a domain where the marginal value of an additional point of benchmark performance is often smaller than the value of one additional control that prevents a single high-severity incident. Near-term research, therefore, should be judged by an unusual standard: does it reduce the gap between capability and defensibility?

A first research priority is the design of **governance-aware evaluation suites**. Today, evaluation infrastructure remains misaligned with what finance needs. We lack standardized case libraries that encode policy constraints, evidence requirements, escalation obligations, and review outcomes. The research question is not merely "can we build benchmarks," but "can we build benchmarks that institutions can adopt without turning them into bespoke consulting projects?" This implies modular suites: a core set of scenarios that test provenance, refusal reliability, and escalation correctness, plus extensible modules for specific use cases such as research summaries, disclosure drafts, and suitability workflows. An important design constraint is that such suites must be operationally usable: reproducible, deterministic where possible, and designed to support regression testing across model and prompt updates.

A second near-term priority is **verifiable evidence handling**. Retrieval-augmented systems can cite sources, but citation is not verification. The research problem is to make evidence claims machine-checkable at scale: to validate that a claim is supported by a specific excerpt, that the excerpt is from the correct version of a document, and that the chain of custody is intact. In finance, this includes not only public documents but internal policies, where "latest approved version" is a crucial constraint. A promising direction is the development of structured claim extraction followed by independent verification modules that operate deterministically on text spans, combined with provenance metadata. But the deeper question is about thresholds: what kinds of claims can be verified reliably today, what remains too ambiguous, and when must the system escalate?

A third near-term priority is **calibrated uncertainty for narrative outputs**. Finance cannot govern systems that always sound confident. Yet the standard uncertainty tools for classifiers and regressors do not translate cleanly to language and multi-step workflows. The research challenge is to produce uncertainty measures that are meaningful to reviewers: not only "confidence scores," but explicit explanations of what is missing, what is conflicting, and what assumptions dominate the conclusion. This is partly a modeling problem and partly an interface problem. An uncertainty estimate is only useful if it changes behavior: triggering escalation, guiding reviewers toward verification, and preventing decision laundering.

A fourth research direction is **trajectory-level evaluation for tool-using systems**. Many failures

occur not in the final output but in intermediate steps: wrong retrieval, wrong tool input, silent tool errors, and error propagation. We need evaluation methods that score the trajectory: which tools were used, whether inputs were validated, whether evidence selection was appropriate, and whether escalation occurred when intermediate uncertainty was high. This research is foundational because tool-using AI is the form of frontier AI that finance is most likely to adopt in practice: systems that can compute, retrieve, and draft within workflows. Output-level scoring is insufficient; trajectory scoring is the evaluability frontier.

Finally, a near-term research question concerns **human factors in governed AI**. If the system is designed well but humans rubber-stamp, governance fails. We need better empirical understanding of how reviewers interact with evidence-linked outputs, which UI patterns reduce automation bias, and what training interventions preserve skepticism. In finance, the human is not merely "in the loop." The human is the legal and fiduciary anchor. Research that improves the reliability of human oversight is, therefore, directly economically valuable.

3.10.2 Technical unknowns

Even with strong near-term progress, several technical unknowns remain that should temper any narrative of inevitability. These unknowns are not theoretical curiosities; they are the reasons governance-first design remains necessary even as models improve.

A primary unknown is **robustness under regime shift**. Finance is a moving target. Models can learn patterns that appear stable for years and then fail abruptly. Frontier AI adds new dimensions of shift: not only market regimes, but document regimes (changes in disclosure language), policy regimes (updated internal rules), and organizational regimes (new workflows, new incentives). We do not yet have reliable methods for detecting when a reasoning system's internal priors have become misaligned with the current environment, especially in narrative tasks. Drift detection for structured models is already difficult; drift detection for multi-step retrieval-and-reasoning systems is harder.

A second unknown is **reliable hallucination detection in finance contexts**. Hallucinations are not uniform. Some hallucinations are obvious; others are subtle—small numeric errors, misattributed policy clauses, plausible but false statements about instruments or regulations. Finance hallucinations are particularly dangerous because they often occur in the language of legitimacy: numbers, definitions, and legal-sounding phrasing. Detecting such hallucinations requires domain-aware verification. The unknown is whether we can build detectors with acceptably low false negative rates in the tails. High false negative rates are disqualifying in high-stakes settings; high false positive rates create operational overload and encourage bypass. The feasible trade-offs remain unclear.

A third unknown is **tool-error detection and propagation control**. Tool-using systems are compositional: errors in extraction become errors in computation become confident recommendations. We do not yet have general methods for making tool chains robust when the upstream model is uncertain. Input validation helps, but validation must know what "reasonable" ranges are and

must handle unit issues and context-dependent assumptions. A further unknown is whether we can propagate uncertainty through tool chains in a way that remains interpretable to human reviewers. In a governed system, the right behavior is often not "continue with low confidence." It is "stop and escalate." But stopping too often makes the system unusable. Finding stable, reliable stopping policies under uncertainty remains open.

A fourth unknown concerns **retrieval integrity at scale**. Retrieval systems are sensitive to indexing strategy, chunking, embedding models, and metadata. Minor changes can produce different evidence sets, which can change outputs. In finance, evidence sets must be controlled: unauthorized sources must be excluded; outdated policies must be filtered; confidential documents must be access-restricted. Ensuring retrieval correctness and consistency at scale remains technically challenging, particularly when evidence corpora are large, heterogeneous, and constantly changing.

A fifth unknown is the **interaction between long context and reasoning depth**. Longer context can improve grounding, but it can also dilute relevance and increase the chance that irrelevant but persuasive text shapes the output. Reasoning depth can improve synthesis but can also increase confident elaboration. We do not yet have a stable understanding of how these dimensions interact under real finance workloads. This is a technical unknown with governance consequences: it determines whether "more context" and "more reasoning" are safe defaults or whether they require explicit guardrails.

3.10.3 Governance unknowns

Governance unknowns are not the same as technical unknowns. They concern how institutions, regulators, and courts will treat AI-assisted decision-making over time, and what standards will become de facto requirements for defensibility. Because finance is regulated and litigious, governance unknowns can dominate strategic risk even when the technology works.

One governance unknown is the **evolving regulatory treatment of AI-generated advice and disclosures**. Regulators have long-standing standards for suitability, disclosure, and market conduct, but frontier AI introduces new operational questions: When does drafting become advice? When does a suggestion become a recommendation? If a system produces client-facing content under human sign-off, what level of documentation is required to demonstrate appropriate supervision? How will regulators treat "AI as a drafting tool" versus "AI as a decision-support engine" when the practical difference may be minimal? The answers will likely vary across jurisdictions and over time, creating uncertainty for multinational institutions.

A second governance unknown is the **standardization of AI audit trails and decision dossiers**. Many organizations already maintain audit trails for trading and compliance processes, but AI introduces new artifacts: prompts, retrieved context, intermediate representations, tool traces, and model versions. What constitutes an adequate audit trail for an AI-assisted disclosure draft? What is the minimum dossier required to defend a decision influenced by AI? Will regulators demand replay

capability, or will they accept logging and sampling audits? The lack of standardized expectations creates a risk: firms may under-invest in auditability and later discover that their documentation is insufficient.

A third unknown concerns **liability allocation and governance accountability**. If an AI system produces an error that leads to client harm or disclosure misstatement, where does liability attach? To the business owner who approved, to the firm's model governance function, to the vendor, or to the individual user? In practice, institutions will bear much of the burden, but the allocation matters because it shapes incentives. If accountability is unclear, governance becomes weak. The governance-first approach advocated in this book is partly an answer: define accountable owners, enforce separation of roles, and treat AI outputs as drafts requiring explicit sign-off. But how external stakeholders will interpret these controls remains uncertain.

A fourth unknown is **industry convergence on minimum control sets**. In financial risk management, certain controls became standard over decades: independent validation, model inventories, stress testing, and change control. Frontier AI may drive a similar convergence, but the timing and content are uncertain. Will evidence anchoring become mandatory? Will refusal reliability be audited? Will firms be required to demonstrate governance-aware benchmark performance? The answers will shape competitive dynamics: early adopters of strong controls may gain trust and deployment ability, while late adopters may face restrictions.

3.10.4 What would change the assessment

This chapter's assessment is intentionally disciplined: frontier AI can deliver real value in finance, but its deployable scope is bounded by governance and evaluability. The question, therefore, is what developments would justify expanding that scope—what would move the boundary between "useful drafting tool" and "controlled decision-support system suitable for higher-stakes workflows."

The first development would be the emergence of **widely adopted governance benchmarks for financial AI**. Not generic reasoning benchmarks, but suites that measure provenance accuracy, policy violation rates, escalation correctness, disclosure completeness, and trajectory-level tool safety. If such benchmarks were credible, standardized, and adopted across institutions, they would reduce the evaluation burden for individual firms and create a common language for regulators and auditors. This would not eliminate risk, but it would raise the industry's baseline discipline and reduce the number of uncontrolled deployments.

The second development would be **low-overhead automated provenance verification tooling**. If institutions could verify, at scale and with acceptable false negative rates, that material claims are supported by specific evidence excerpts from correct document versions, the risk of hallucination and misattributed citations would drop substantially. This would also make review more efficient: humans could focus on judgment and interpretation rather than on checking whether the system invented a number. Importantly, "low overhead" matters. If verification tooling is expensive and

slow, teams will bypass it. The control must be operationally feasible to be adopted.

A third development would be **reliable uncertainty calibration tied to escalation policies**. If systems could produce uncertainty signals that robustly predict when outputs are likely to be wrong or unsupported, escalation could be targeted and efficient. This would reduce both false non-escalations (dangerous) and excessive escalations (operationally paralyzing). The key is that uncertainty must be behaviorally actionable: it must drive consistent stopping and review decisions.

A fourth development would be **matured evaluation for multi-step tool use**. If we had robust methods for validating tool chains—ensuring correct tool selection, correct input validation, and safe propagation of uncertainty—finance could safely automate more procedural workflows. This would expand the feasible scope from drafting and extraction into more complex decision-support loops, still under human approval but with higher reliability.

A final development would be **institutionalization of governance operations**. Even if technical tools improve, the system will not be safe unless organizations operate it well. If firms develop standardized governance roles, dashboards, incident response playbooks, and change control processes for AI systems, then the organization can safely expand deployment. In finance, operational maturity often matters as much as technical maturity.

3.10.5 Link to subsequent chapters

This chapter closes the book's application arc in finance by reinforcing a recurring theme: in high-accountability domains, the frontier is not raw capability, but the ability to maintain evidentiary discipline under pressure. Finance makes this theme unavoidable because liability is immediate and the cost of error is nonlinear. The system that "sounds right" but cannot show its evidence is not a productivity tool; it is a risk generator.

The transition to the subsequent chapter on interpretive AI in humanities and knowledge work is therefore natural rather than thematic whiplash. Humanities and policy domains are often treated as "soft" compared to finance, but they share the same structural vulnerability: coherent narratives can drift away from evidence. In finance, that drift creates legal and fiduciary harm. In knowledge work, it creates epistemic harm: organizations begin to operate on stories detached from sources. The control strategy is shared: evidence-first schemas, provenance integrity, and explicit separation of fact from interpretation.

In other words, Chapter 8 demonstrates the governance-first approach under the harshest practical constraints—where accountability is enforced by regulators, markets, and courts. Chapter 9 generalizes the lesson to domains where accountability is often looser but where truth is equally at risk. The bridge between them is the same control principle: the system must not be allowed to substitute rhetorical coherence for evidentiary grounding. Evidence discipline is the antidote to epistemic drift, whether the drift leads to a disclosure violation or to a policy decision built on a

manufactured narrative.

The book's broader message remains consistent: frontier AI is most valuable when it strengthens institutional judgment rather than replacing it. The open questions ahead are not merely technical. They are about building a future where AI can be used at scale without eroding the very standards—evidence, accountability, and defensibility—that professional domains exist to uphold.

Bibliography

[1] Board of Governors of the Federal Reserve System and Office of the Comptroller of the Currency. *Supervisory Guidance on Model Risk Management (SR 11-7) and Attachment (SR 11-7a1)*. April 4, 2011. https://www.federalreserve.gov/supervisionreg/srletters/sr1107.pdf

[2] X. Liu, H. Yu, H. Zhang, Y. Xu, X. Lei, H. Lai, Y. Gu, H. Ding, K. Men, K. Yang, S. Zhang, X. Deng, A. Zeng, Z. Du, C. Zhang, S. Shen, T. Zhang, Y. Su, H. Sun, M. Huang, Y. Dong, and J. Tang. *AgentBench: Evaluating LLMs as Agents*. arXiv:2308.03688, 2023. https://arxiv.org/abs/2308.03688

[3] J. Menick, M. Trebacz, V. Mikulik, J. Aslanides, F. Song, M. Chrzanowski, S. Gajewski, L. Chappell, A. Czarnecki, and A. R. Ladner et al. *Teaching Language Models to Support Answers with Verified Quotes*. arXiv:2203.11147, 2022. https://arxiv.org/abs/2203.11147

[4] S. Kapoor, A. M. Yadlowsky, M. Nori, C. Ré, and D. E. Ho. *Large Language Models Must Be Taught to Know What They Know*. NeurIPS (Conference Paper), 2024. https://papers.nips.cc/paper_files/paper/2024/file/9c20f16b05f5e5e70fa07e2a4364b80e-Paper-Conference.pdf

[5] P. Liang, R. Bommasani, T. Lee, D. Tsipras, D. Soylu, M. Yasunaga, Y. Zhang, D. Narayanan, Y. Wu, A. Kumar, and C. Ré et al. *Holistic Evaluation of Language Models*. arXiv:2211.09110, 2022. https://arxiv.org/abs/2211.09110

[6] K. Goddard, A. Roudsari, and J. C. Wyatt. *Automation bias – a hidden issue for clinical decision support system use*. Stud Health Technol Inform, 164:17–22, 2011. (PMID: 21335682) https://pubmed.ncbi.nlm.nih.gov/21335682/

[7] U.S. Securities and Exchange Commission. *Conflicts of Interest Associated with the Use of Predictive Data Analytics by Broker-Dealers and Investment Advisers* (Proposed Rule; Federal Register posting). August 9, 2023. https://www.federalregister.gov/documents/2023/08/09/2023-16377/conflicts-of-interest-associated-with-the-use-of-predictive-data-analytics-by-broker-deal

[8] A. L. Suárez-Cetrulo, A. J. Hernández, and J. M. Vidal. *Machine Learning for Financial Prediction Under Regime Change: A Systematic Literature Review*. 2023. `https://reunir.unir.net/bitstream/handle/123456789/15032/ip2023_06_003.pdf`

[9] International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC). *ISO/IEC 42001:2023 — Artificial intelligence — Management system*. 2023. `https://www.iso.org/standard/42001`

[10] K. Wu, E. Wu, A. Cassasola, A. Zhang, K. Wei, T. Nguyen, S. Riantawan, P. S. Riantawan, D. E. Ho, and J. Zou. *An automated framework for assessing how well LLMs cite relevant medical references*. Nature Communications, 2025. `https://www.nature.com/articles/s41467-025-58551-6`

Chapter 4

Chapter 9 — Interpretive AI: Knowledge Work, Narrative, and Epistemic Discipline

Abstract. This chapter treats interpretive AI as an institutional writing and reasoning system whose primary risk is epistemic: coherent narratives can outpace the evidence that should constrain them. In humanities-adjacent and professional knowledge work—policy, law, research, education, media, and executive communication—the output is rarely a single fact. It is framing, selection, emphasis, and argument. That makes provenance and evidentiary discipline the core product, not a compliance afterthought.

We model interpretive generation as a mapping from a source set to a narrative under editorial constraints, and we argue that governance must be implemented as a pipeline property rather than a model property: every factual claim must be linkable to a supporting source; interpretive moves must be labeled as such; and uncertainty must be expressed when sources conflict or do not exist. The chapter emphasizes drift as the canonical failure mode: successive rewrites, paraphrases, and "improvements" can detach claims from their original evidence while simultaneously increasing rhetorical authority.

Operationally, the chapter introduces evidence-first workflows, claim-level verification, citation-fidelity audits, and publication dossiers as minimum viable controls for any organization that uses AI to produce external-facing narratives. It also clarifies what the chapter does not do: it is not a guide to persuasion manipulation, propaganda, or disinformation. Instead, it is a governance-first guide to building interpretive systems that remain reviewable, contestable, and institutionally accountable.

A companion Colab notebook (linked, separate artifact) demonstrates the chapter's core idea in a synthetic document world: generating toy sources with contradictions, extracting toy claims, linking evidence, drafting constrained narratives, and measuring citation fidelity and drift under retrieval noise.

4.1 Orientation and Scope

4.1.1 Motivation and context

In the social sciences and humanities, the primary object of work is rarely a single, checkable fact. The object is interpretation: an account of meaning, context, causality, or significance that is constructed from sources through a lens. Historians weigh archives, silences, and power; political scientists translate messy events into concepts and identification assumptions; anthropologists navigate situated knowledge and ethical constraints; literary critics argue about what a text is doing, not merely what it says. In these domains, writing is not simply presentation. Writing is where reasoning happens, where evidence is selected and framed, where alternative readings are acknowledged or ignored, and where claims accumulate rhetorical force.

Frontier AI systems—especially large language models used with retrieval, long context, and iterative drafting loops—enter directly into this interpretive core. They are not just productivity tools that "draft faster." They can mass-produce plausible arguments, scholarship-adjacent prose, literature reviews, policy memos, lesson plans, and narrative syntheses that look epistemically well-formed. The risk is therefore not a narrow technical error rate. The risk is institutional: when outputs become persuasive at scale, the bottleneck moves to verification, and the temptation to treat coherence as legitimacy grows. In knowledge domains, legitimacy is not decorative; it is operational. Universities, publishers, think tanks, NGOs, museums, courts, and public agencies depend on credibility. SSH scholarship depends on contestability: the ability for others to locate claims in sources, challenge interpretations, and trace how conclusions were reached. If AI systems accelerate production while weakening traceability, they do not simply introduce "model risk." They reshape epistemic norms by making unverifiable narratives cheap.

This chapter begins from a simple governance-first proposition: in social sciences and humanities work, interpretive AI must be evaluated and controlled as an *argument pipeline*, not as a text generator. The relevant hazards are not only hallucinated facts, but also citation theater (citations that decorate rather than support), contextual drift (where paraphrase changes meaning), framing bias through omission (where what is not said becomes the central distortion), and authority laundering (where institutional voice is projected through machine-generated certainty). These failure modes matter precisely because they exploit the superficial markers of credibility: polished style, confident tone, and bibliographic density.

The opportunity side is real. Evidence-first interpretive systems can reduce toil in corpus review, accelerate the assembly of counterevidence, make argument structure explicit, and help scholars and practitioners explore alternative lenses more systematically than time normally permits. The same machinery that makes output scalable can, with the right constraints, make process auditable. But this requires treating provenance and review as first-class design objectives. The scope of this chapter is therefore not to argue that AI "understands" texts or societies. It is to specify the

institutional conditions under which AI can assist interpretive work without corroding the standards that make SSH outputs trustworthy.

Artifact (Save This)

Core premise for the chapter. In social sciences and humanities work, the product is not text; the product is an accountable interpretation. Any AI system that produces narratives without enforceable evidence links, lens declarations, and revision traceability should be treated as a drafting convenience for internal use only—not as a publishable research or institutional communication pipeline.

4.1.2 Why this topic is at a frontier moment

This topic is at a frontier moment because the constraint in interpretive work has shifted. For decades, the limiting factor in SSH practice was production capacity: reading time, drafting time, and the human labor required to synthesize complex, contested bodies of material. Even with digital archives and search tools, writing remained slow. That slowness had an epistemic benefit: it imposed natural friction. You could not generate ten alternative historiographical framings in an afternoon, nor could you casually draft a policy narrative that sounds like a journal article unless you had internalized disciplinary norms. The speed of production acted as an implicit quality filter.

Frontier AI removes that friction. A model can draft a plausible theoretical framing, produce a literature review voice, propose causal mechanisms, and write with discipline-coded rhetoric at near-zero marginal cost. That is why "writing is no longer the bottleneck." The bottleneck is now verification: whether the argument is properly grounded, whether citations actually support claims, whether the framing is defensible, whether uncertainty and disagreement are represented rather than erased, and whether a reader can reconstruct how the narrative was produced. In other words, the bottleneck moves from composition to accountability.

This shift matters because interpretive domains are unusually vulnerable to a specific asymmetry: *rhetorical authority scales faster than epistemic certainty.* In many operational settings—policy briefs, executive summaries, public-facing comms—the readers are time-constrained and rely on credibility cues. Coherence, tone, and reference density function as proxies for "due diligence." LLM outputs can optimize exactly those proxies. The frontier risk is therefore not merely error but *miscalibrated trust*: institutions begin to treat persuasive narratives as vetted knowledge because they look like the output of careful scholarship. The irony is that the better the prose, the easier it becomes to skip the epistemic work.

The frontier is also institutional because SSH domains increasingly sit inside high-stakes decision environments. Social science frames policy; historical narratives shape legitimacy; legal reasoning (adjacent to SSH in its interpretive discipline) influences rights and liability; educational materials shape beliefs. The externalities of interpretive drift are not confined to academic debate. They

can affect reputations, elections, litigation risk, and public trust. This is why the chapter treats "soft domains" as high-stakes: when the output shapes belief and institutional voice, framing risk becomes real risk.

Technically, the frontier arises from new system behaviors that look like scholarly cognition. Retrieval augmentation makes models appear to "cite." Long context makes outputs appear to "have read the archive." Iterative loops make outputs appear to "revise thoughtfully." Multimodal capability makes outputs appear to "interpret evidence" across charts, images, and tables. Each of these capabilities increases the *surface area for plausible deception*—not intentional deception by the system, but deception as an emergent effect of a pipeline that produces credibility cues faster than it produces verifiable support.

A governance-first response must therefore invert the default product logic. The goal is not "generate the best narrative." The goal is "generate the most inspectable narrative under explicit constraints." For SSH, inspectability includes: (i) passage-level anchoring, (ii) explicit separation of descriptive claims from interpretive claims, (iii) lens declaration (what theoretical frame is being used), (iv) counterevidence inclusion, and (v) revision traceability so that rhetorical strengthening can be audited for evidentiary drift. At the frontier moment, these controls are not academic. They are what will separate responsible institutional adoption from epistemic collapse-by-convenience.

Risk & Control Notes

Frontier failure pattern: credibility outpacing verification. The most dangerous outputs in SSH settings are not obviously wrong. They are polished, citation-dense narratives that embed small unsupported leaps (causal upgrades, decontextualized paraphrases, omitted counterreadings). These outputs pass casual review precisely because they mimic the style of disciplined scholarship.

4.1.3 What has changed recently

Several changes have converged to make interpretive AI operationally relevant in the social sciences and humanities. The most visible change is long-form fluency. Modern models can sustain coherent argument structure across thousands of words, maintain a consistent voice, and emulate disciplinary genres: literature reviews, historiographical surveys, theory sections, policy memos, grant proposals, op-eds, and pedagogical explanations. This matters because interpretive work is not a sequence of short answers; it is a sustained construction where earlier premises constrain later conclusions. When models can hold those constraints over long stretches, they can produce outputs that resemble genuine scholarly argumentation.

A second change is the maturation of retrieval-augmented generation (RAG) as an implementation pattern. RAG did not merely improve factual lookup; it changed user expectations. Users now interact with systems that can retrieve and quote from documents, attach citations, and tailor

outputs to a defined corpus—an archive, a set of interview transcripts, a reading list, or a policy dossier. In SSH contexts, where arguments are built from sources, this looks like a direct match. But the key shift is that retrieval can be integrated into the generation loop: retrieve passages, draft claims, retrieve counterpassages, revise, and so on. This creates a pipeline that can *simulate* evidentiary reasoning even when the internal mechanics remain probabilistic.

A third change is long-context architectures. When models can ingest large document sets directly—multiple papers, chapters, or archival excerpts—users increasingly conflate "seeing the text" with "being faithful to the text." But in interpretive settings, fidelity is not only about having access. It is about selection, emphasis, and context. Long context reduces certain forms of hallucination but increases a different hazard: *selective plausibility*. With more material available, a model can more easily find something that appears to support a claim while ignoring nearby qualifiers, alternative interpretations, or counterevidence. The result can be a narrative that is technically "grounded" in the sense that it references real text, yet epistemically distorted in the way it frames that text.

A fourth change is the normalization of iterative drafting workflows. Many users now treat AI systems as editors: "rewrite this more persuasively," "strengthen the argument," "tighten the narrative," "make it sound more academic," "add references." In SSH work, these instructions are not stylistic; they can be epistemic transformations. A rewrite that increases confidence or generalizes a claim can change its truth conditions. The system's ability to improve rhetoric while losing nuance creates the core risk of *epistemic drift*. Drift is not a single hallucinated sentence; it is the gradual detachment of meaning from sources through successive paraphrases and "improvements."

A fifth change is multimodal literacy. Many SSH-adjacent outputs incorporate charts (survey results, econometric figures), maps (historical geography), images (artifacts, photographs), and tables (coding schemas, archival metadata). As models become capable of interpreting and describing such inputs, the boundary between textual evidence and visual evidence blurs. This expands the provenance problem. Visual claims often require different validation than textual claims. A model can confidently interpret a figure in a way that is not supported by the underlying data, or infer causal stories from a chart without identification discipline. Multimodality, therefore, increases the need for evidence typing and modality-specific verification.

Finally, adoption has moved from experimental to routine in several environments: education (lesson planning, feedback), media (drafting, backgrounders), corporate communication (narratives, FAQs), and policy drafting (briefs, consultations). The operational reality is that interpretive AI is already being used. The frontier question is not whether it will be used, but whether it will be used with controls that preserve epistemic standards. This makes implementation design urgent. Without explicit evidentiary discipline, the path of least resistance is "draft-first, verify-never," and the social cost is epistemic erosion that is hard to reverse.

Artifact (Save This)

What changed, in one line. AI made interpretive text cheap; it did not make interpretive accountability cheap. The gap between those two costs is now the dominant governance problem for SSH-facing deployments.

4.1.4 Explicit exclusions and non-goals

This chapter is intentionally not a guide to persuasion manipulation, propaganda operations, or disinformation production. The reason is not merely ethical. It is also conceptual: the core problem here is how to preserve evidentiary discipline and scholarly norms in the presence of scalable generation. A manual for manipulation would invert the chapter's objective and would be operationally irresponsible. Where the chapter discusses framing, it does so as a risk category to be detected, logged, and controlled—especially framing through omission, selective emphasis, and rhetorical overconfidence.

The chapter is also not a replacement for SSH methodology. It does not claim that interpretive validity can be reduced to a single metric. Humanities interpretation is not equivalent to fact-checking, and social science inference is not equivalent to narrative plausibility. The chapter respects pluralism: multiple interpretations can be defensible given the same sources, and disagreement can be intellectually productive. The purpose of governance in this context is therefore not to enforce "one correct reading," but to enforce transparency about what is claimed, what supports it, what lens is used, what is uncertain, and what alternatives exist. In other words, the chapter does not seek to mechanize interpretation; it seeks to mechanize accountability.

Third, the chapter does not claim that AI systems "understand" meaning as humans do. It treats models as powerful pattern-based generators that can participate in interpretive workflows under constraint. Whether that participation counts as "understanding" is a philosophical question with no operational resolution and limited governance value. Institutions cannot base their controls on metaphysical claims. They must base controls on measurable behaviors: citation fidelity, drift rates, omission patterns, uncertainty calibration, and review outcomes.

Fourth, the chapter is not a comprehensive survey of all humanities computing, computational social science, or digital scholarship. Those fields have long histories (topic modeling, stylometry, network analysis, corpus linguistics, ethnographic coding tools) that intersect with AI but are not reducible to LLM workflows. We focus on frontier interpretive AI as it is now practiced: long-form generative systems, often coupled with retrieval and iterative editing, used to produce arguments, syntheses, and institutional narratives. Classical quantitative methods are relevant insofar as they illustrate the difference between verifiable analysis and rhetorically confident synthesis.

Fifth, the chapter is not an endorsement of fully autonomous publication. In this domain, autonomy is not a technical milestone; it is a governance decision with reputational and ethical consequences.

The chapter treats external-facing publication as a high-risk boundary that requires human sign-off, separation of duties, and dossier-quality traceability. Any workflow that cannot staff verification should restrict AI outputs to internal drafts and exploratory synthesis.

Finally, this chapter does not assume that "more context" or "more citations" is inherently safer. In interpretive domains, more context can increase ambiguity and make cherry-picking easier; more citations can increase the temptation to treat bibliography as proof. A core non-goal is the illusion of safety through surface complexity. The chapter's stance is simple: if a claim cannot be mapped to evidence in a form that a reviewer can inspect, the claim is not publication-grade.

Risk & Control Notes

Non-goal clarified. The objective is not to build systems that generate persuasive SSH narratives. The objective is to build systems that generate SSH narratives whose evidentiary scaffolding is explicit, inspectable, and stable under revision pressure.

4.1.5 Role of this paper in AI 2026

Within *AI 2026: Frontier Awareness Without the Hype*, Chapter 9 performs a specific function: it carries governance-first reasoning into the domains where the output changes what people believe. Earlier chapters address evaluation for agents, reasoning risk, representation control, memory, and planning—topics that often sound like "technical safety." Chapter 9 insists that epistemic safety is not optional and that interpretive domains are not "low-stakes" simply because they lack physical actuators. In many organizations, the most consequential AI outputs are narratives: investment theses, policy rationales, legal arguments, risk memos, public statements, educational materials, and internal knowledge bases. These are the documents that set direction, justify action, and create liability.

This chapter therefore reframes interpretive AI as a governance surface. It introduces framing risk as a first-class risk category alongside hallucination, privacy, and misuse. It argues that provenance is not an add-on feature but the central control mechanism: the ability to answer "what sources supported what claims, under what lens, with what uncertainty, approved by whom." That is the minimum requirement for institutional legitimacy in SSH-like work. When outputs are contestable, errors can be corrected and trust can be repaired. When outputs are not contestable, the organization is effectively publishing un-auditable belief.

Chapter 9 also serves as a bridge to Chapter 10 on multimodal vision–language–action systems. The connection is structural: once systems can perceive and act, the governance problem becomes system-level. But even before action, interpretive pipelines already behave like systems-of-systems: retrieval, memory, drafting, verification, and publication gating. The chapter's controls—claim typing, evidence mapping, drift monitoring, and dossiers—generalize naturally to multimodal settings where "evidence" includes visuals and where the stakes include security and safety. In

that sense, Chapter 9 is the book's epistemic hinge: it shows that governance is not merely about preventing harmful actions; it is about preserving the integrity of institutional knowledge under scalable generation.

Finally, Chapter 9 complements the book's overarching message: frontier capability without evaluability is organizational negligence. In interpretive domains, evaluability is not a benchmark score; it is evidentiary traceability. This chapter specifies what that means operationally, what minimal controls look like, and what open questions remain about making those controls cheap enough to scale. It prepares the reader for the final chapter's broader claim: that responsible frontier deployment is a systems engineering discipline where provenance, gating, and audit logs are the infrastructure of trust.

Artifact (Save This)

Positioning statement for AI 2026. Chapter 9 shows why "governance-first" is not just about preventing models from doing bad things. It is about preventing institutions from believing bad reasons—because persuasive, untraceable narratives are themselves a high-impact failure mode.

4.2 Conceptual Abstraction

4.2.1 Core abstraction

The core abstraction of this chapter is that interpretive AI is not a "summarizer" but a *narrative construction system*. In social sciences and humanities work, the output is typically a structured account: a story of causality, a framing of meaning, a synthesis of contested scholarship, or an argument for significance. That output is a function of what sources are available, how they are selected, and what editorial constraints govern the transformation from evidence into narrative. We represent this as:

$$N = F(S, \pi),$$

where S is a set of sources (archives, papers, transcripts, datasets, interview notes, media reports, field notes, and secondary scholarship), and π is a set of framing choices and editorial constraints (lens declarations, scope boundaries, uncertainty policies, citation rules, tone requirements, and admissibility criteria for sources). This abstraction matters because the risk is not located in the final text alone. The risk is located in the mapping: which sources are retrieved and privileged, which claims are formed, which claims are omitted, and how rhetorical structure upgrades weak evidence into apparently strong conclusions.

In an evidence-disciplined pipeline, F is not a free-form generator; it is a constrained transformation that must preserve inspectability. In that setting, interpretive output N is not just paragraphs. It is a composite object that includes (i) a set of claims C, (ii) an explicit labeling of claim types (descriptive, interpretive, causal, normative), (iii) a mapping from each claim to supporting and countervailing evidence, and (iv) a provenance record that makes the construction path reconstructible. The narrative is therefore best understood as a *structured argument with attached evidence*, even when the user sees only the prose.

This abstraction also clarifies why the humanities and social sciences are not "low-stakes." In many organizations, interpretive narratives become institutional speech: research reports, policy rationales, educational materials, official communications, and public statements. When an institution publishes an interpretation, it effectively asserts that the underlying evidentiary and editorial labor has been done. Interpretive AI can help do that labor faster, but only if the constraints π make the process auditable. Otherwise, the institution risks substituting rhetorical fluency for epistemic due process.

Finally, $N = F(S, \pi)$ makes visible the key governance lever: π. If π is weak ("sound confident," "make it persuasive," "add citations"), the system will reliably produce outputs that look legitimate while drifting from evidence. If π is strong ("every factual claim must map to a source passage," "interpretations must declare lens," "counterevidence must be included," "uncertainty must be stated when sources conflict"), the system can be shaped into a disciplined assistant for interpretive work. The frontier is therefore not merely model strength; it is the design of constraints that define what counts as an acceptable transformation from sources to narrative.

Artifact (Save This)

Interpretive AI as an accountable object. In this chapter, a narrative N is treated as more than text: it is a claim set with typed assertions, evidence links, counterevidence links, and a provenance trail. Without those attachments, a "good" narrative is indistinguishable from a well-written fabrication.

4.2.2 Key entities and interactions

The abstraction becomes operational when we specify the entities that the system must manage. The first entity is the *source* $s \in S$. In SSH settings, sources are heterogeneous: primary texts (letters, decrees, novels, speeches), secondary scholarship (articles, monographs), qualitative materials (interviews, ethnographic notes), quantitative artifacts (datasets, codebooks), and gray literature (policy reports, NGO publications, media). Unlike many technical domains, the admissibility and status of sources is itself interpretive. A newspaper report is not equivalent to an archival record; a memoir is not equivalent to a contemporaneous transcript; a preprint is not equivalent to a peer-reviewed article. Therefore, sources carry metadata: type, date, provenance, credibility tier, and in many cases ethical restrictions (e.g., human-subject consent boundaries). A governance-first pipeline treats source curation as a core control, not a preprocessing detail.

The second entity is the *claim*. A claim is a unit of assertive content that the narrative commits to. Claims must be extracted or declared explicitly because they are the objects that can be verified, contested, and audited. In SSH outputs, claim types matter: descriptive claims ("event X occurred"), interpretive claims ("text T functions as a critique of Y"), causal claims ("policy P increased outcome O"), comparative claims ("case A differs from case B in mechanism M"), and normative claims ("therefore the institution should do Z"). The chapter insists that claim typing is not bureaucratic overhead; it is the minimum structure required to prevent interpretive writing from laundering weak evidence into strong-sounding conclusions.

The third entity is the *evidence link*. Evidence links map claims to source passages (or data artifacts) that support them. In interpretive work, evidence is often non-unique: multiple passages can support a reading; a dataset can support multiple causal framings under different identification assumptions; counterevidence can coexist with support. Therefore evidence links must allow: (i) supporting evidence, (ii) counterevidence, and (iii) "uncertain/insufficient" states. The system must also represent *scope constraints*: which parts of the source set were considered, what was excluded, and why. Without scope documentation, omission becomes invisible and framing manipulation becomes un-auditable.

The fourth entity is the *framing constraint* π. In practice π includes: declared lens (e.g., historiographical tradition, theoretical school), audience constraints (academic vs public), tone constraints (neutral vs advocacy), admissible source tiers, required inclusion of counterarguments, uncertainty

language policy, and publication gating rules. Critically, π must be inspectable itself. If the system is instructed to "sound persuasive," that is not a neutral stylistic choice; it is a governance decision that increases risk of overclaiming. Conversely, if the system is instructed to "state uncertainty when evidence conflicts," that is a control that reduces reputational and epistemic risk.

These entities interact in a characteristic loop:

$$\text{retrieve} \rightarrow \text{extract claims} \rightarrow \text{attribute evidence} \rightarrow \text{draft narrative} \rightarrow \text{review} \rightarrow \text{revise}.$$

Each arrow is a control point. Retrieval determines which evidence is visible; claim extraction determines what becomes auditable; evidence attribution determines whether claims are anchored; drafting determines rhetorical force; review determines institutional accountability; revision determines whether improvements preserve fidelity or introduce drift. The chapter's governance posture is that this loop must be logged. If the institution cannot reconstruct what sources were retrieved, what claims were made, and how revisions changed claim-evidence mappings, then the institution cannot credibly defend the output as disciplined scholarship or responsible policy communication.

In the SSH setting, the loop is further complicated by pluralism: multiple narratives can be legitimate given the same sources. Therefore, a robust system must support multi-lens workflows: run the pipeline under different π values, generate alternative narratives, and explicitly compare how the claim set changes. That is not a luxury. It is often the most honest representation of a contested field. The system should enable structured disagreement rather than forcing synthesis into a single confident voice.

Risk & Control Notes

Why entity structure matters. Without explicit entities (sources, claims, evidence links, lens constraints), governance collapses into "review the prose." But prose-level review is precisely what fluent models are designed to pass. The system must surface what is normally hidden: what was claimed, what supports it, and what was ignored.

4.2.3 What is being optimized or controlled

A central confusion in interpretive AI adoption is the assumption that the system's objective is "to produce the best narrative." That objective is underspecified and, in SSH contexts, potentially harmful. A model can optimize for coherence, rhetorical force, and apparent completeness while degrading evidentiary integrity. The governance-first stance is therefore that the primary objective is not narrative quality in the abstract, but *narrative usefulness under evidentiary constraints*. In other words, we want outputs that are helpful to humans while remaining contestable.

Formally, we can distinguish between a *utility* term $U(N)$ and a set of *constraint violations* related to evidence. The system should maximize usefulness subject to constraints such as: (i) every

factual claim has at least one supporting evidence link, (ii) interpretive claims are labeled with the lens used, (iii) causal claims state identification assumptions or are downgraded to hypotheses, and (iv) contradictory evidence triggers uncertainty rather than rhetorical smoothing. In practice, these constraints are implemented through workflow design and review gates, not only through loss functions.

The key control objective is *anchoring*. Anchoring means that the narrative cannot float free from sources. But anchoring alone is insufficient. A narrative can be anchored in a superficial way (e.g., quoting a sentence) while misrepresenting context. Therefore, anchoring must be combined with *context preservation*: the evidence link should include enough surrounding passage to preserve meaning, and the pipeline should penalize decontextualized paraphrase.

A second control objective is *typed separation*: factual claims, interpretive claims, and normative claims must not be blended without labeling. This is especially critical in social science outputs where a descriptive statistic can be followed by an implied causal inference and then a policy recommendation. A fluent model will smooth these transitions; governance must interrupt them. Typed separation makes it possible to review the most liability-bearing moves (e.g., causal and normative leaps) with extra scrutiny.

A third control objective is *calibration of certainty*. In SSH, uncertainty is not always a defect; it is often epistemic honesty. When evidence is mixed, contested, or incomplete, the system should downgrade language: "suggests," "is consistent with," "may reflect," "under this lens," rather than "proves," "demonstrates," "shows that." The open risk is that institutional incentives push toward confident prose. Governance must counterbalance that incentive with explicit language policy and review requirements.

A fourth control objective is *coverage of counterevidence*. Many interpretive harms arise not from false statements but from selective omission. A well-formed narrative can still be misleading if it excludes the strongest counterargument or the most relevant alternative lens. Therefore, the pipeline should treat counterevidence as a required artifact: identify at least k counterclaims and link them to sources; include a "what would change this assessment" section; or provide multiple lens outputs. The chapter frames this as an epistemic control analogous to adversarial testing in technical systems.

Finally, the system should optimize for *reviewability*. Reviewability is an operational property: can a verifier quickly inspect claim-evidence mappings, see which sources were used, understand what lens was applied, and reproduce the output? In many organizations, review time is the scarcest resource. A system that maximizes narrative beauty while minimizing reviewability will fail governance even if the prose is excellent.

Artifact (Save This)

Optimization target (operational). The system should optimize for *reviewable usefulness*: maximize human utility while minimizing unanchored claims, unlabeled interpretive moves, unjustified causal upgrades, and rhetorical overconfidence under weak evidence.

4.2.4 Distinction from prior paradigms

The interpretive AI paradigm differs from prior writing and research paradigms in one decisive respect: it decouples authorship from accountability unless controls reattach them. Traditional SSH writing is slow and socially situated. The author is visible; the scholarly community knows how to contest claims; citation practices, while imperfect, are embedded in norms of responsibility. Even when errors occur, they occur within a structure where responsibility is attributable.

AI-driven writing introduces a new production regime: drafting becomes abundant, cheap, and instantaneous. The natural temptation is to treat the model as a "co-author" or a "research assistant." But these metaphors mislead. A model does not bear responsibility, cannot be cross-examined, and cannot be sanctioned. If an institution uses AI to produce interpretive narratives, the institution is still the author in the only sense that matters: it is the party accountable for the claims. Therefore, interpretive AI changes the governance requirement from "can we write this" to "can we defend this." The latter requires artifacts that earlier paradigms did not need at scale.

Compared to classical NLP or computational humanities pipelines, the difference is also structural. Earlier methods were typically analytic: topic models, stylometry, frequency analysis, network graphs, coding tools. Their outputs were constrained objects (tables, clusters, statistics) that could be inspected and reproduced. The narrative still had to be written by a human, and the interpretive leap from analysis to argument was visible as a human act. Generative AI collapses that separation. The pipeline can produce both the analytic-sounding summary and the argument that uses it, in one continuous stream. This increases the risk of blurred boundaries: results become claims, claims become conclusions, conclusions become policy. Without explicit structure, the reader cannot see where inference occurred.

Another distinction is that prior paradigms assumed scarcity of text generation. Editors and peer reviewers focused on substance because style was costly. With generative systems, style becomes cheap. As a result, many institutional review processes that implicitly relied on "writing effort" as a proxy for diligence become obsolete. A polished memo no longer signals that someone spent time verifying it. It may signal only that a model was prompted effectively. This creates a governance gap: organizations that fail to update their review norms will be fooled by outputs that are optimized for readability.

The paradigm shift also changes how errors propagate. Traditional errors in SSH scholarship propagate through citation networks and reputation systems over time. AI can propagate errors

instantly at scale across products, classrooms, media channels, and internal knowledge bases. Moreover, errors can become more difficult to detect because they are embedded in plausible narrative structure. The system can cite real sources while misrepresenting them, or it can present contested claims as settled. The resulting harm is not only misinformation; it is the erosion of the discipline's ability to contest claims because the provenance trail is missing or incoherent.

Therefore, the key distinction is governance: interpretive AI requires that the production pipeline be separated from the approval pipeline more rigorously than before. Drafting can be automated; publishing cannot. In an evidence-disciplined institution, the right mental model is not "AI writes for us." It is "AI proposes candidate narratives with explicit evidentiary scaffolding that humans can accept, reject, or revise with accountability."

Risk & Control Notes

Paradigm inversion. In traditional SSH workflows, writing effort loosely correlates with diligence. In AI-assisted workflows, writing effort collapses to near zero while diligence requirements remain unchanged. Governance must explicitly rebuild the diligence signal through logs, evidence mappings, and review gates.

4.2.5 Conceptual failure modes

The conceptual failure modes of interpretive AI in social sciences and humanities are not random bugs; they are structural properties of how narrative generation interacts with evidence scarcity, contested meaning, and institutional incentives. The most important failure mode is *epistemic drift*. Drift occurs when successive rewrites and revisions detach claims from their original support. In SSH contexts, drift is often subtle: a cautious statement becomes a confident one; a contextual qualifier is dropped; an interpretive claim is presented as descriptive fact; a contested thesis is presented as consensus. Because the output remains coherent and well-phrased, drift often escapes detection unless the workflow forces claim-by-claim evidence checking.

A second failure mode is *citation theater*. Citation theater is the appearance of scholarship without the substance of support. It can take multiple forms: fabricated citations, real citations that do not support the claim, citations that support only a weaker version of the claim, or citations that are relevant to the topic but not to the asserted conclusion. In SSH writing, citations also function as positioning within a literature. A model can cite canonical names to signal legitimacy even if the argument does not meaningfully engage their claims. This is dangerous because it exploits academic trust cues.

A third failure mode is *framing bias through omission*. Many interpretive harms arise not from what is stated but from what is excluded. A narrative that selects only evidence consistent with a chosen lens can be internally coherent while externally misleading. Omission is especially potent in politically salient topics, contested historiographies, and policy-adjacent analysis. In social science,

omission may mean ignoring alternative identification strategies or confounders. In humanities, omission may mean excluding marginalized voices or counterreadings that complicate a canonical story. Because omission is hard to detect from the text alone, governance must require explicit counterevidence artifacts.

A fourth failure mode is *rhetorical overconfidence*. Models can generate high-authority prose that sounds definitive even when evidence is weak, mixed, or absent. In SSH, where uncertainty is often the honest state, overconfidence can reshape how readers perceive contested issues. This is not merely stylistic. It alters the epistemic posture of the institution: it turns inquiry into proclamation. Overconfidence is reinforced by prompts that request persuasion, clarity, and decisiveness—common in organizational settings. Therefore, language calibration must be treated as a control.

A fifth failure mode is *attribution blur*. When models draft interpretive narratives, the boundary between the institution's judgment and the model's generation can become unclear. This blur creates governance problems: who is responsible for interpretive choices, selection of evidence, or omission of counterarguments? Blur also creates ethical problems in scholarship and education: what constitutes authorship, how should assistance be disclosed, and how does one avoid misrepresenting originality? The chapter treats attribution blur as an institutional risk because it undermines accountability and can trigger reputational harm when AI assistance is revealed or contested.

A sixth failure mode is *lens laundering*. A model can present a particular theoretical frame as neutral common sense, embedding normative commitments without declaring them. For example, in political economy a rational-choice framing may be presented as "objective," while alternative institutional or critical perspectives are excluded. In literary analysis, a psychoanalytic reading can be written as if it were descriptive truth. The failure is not that a lens is used; lenses are necessary. The failure is that the lens is hidden, making the narrative appear more authoritative and less contestable than it should be.

A seventh failure mode is *causal inflation* in social science writing. Models frequently produce causal language because it is rhetorically strong and common in policy discourse. But causality requires identification assumptions, counterfactual reasoning, and explicit limitations. Without constraint, AI outputs tend to overstate causal confidence, conflate mechanisms, or treat correlational patterns as evidence of intervention effects. This can produce institutionally dangerous outputs: policy recommendations built on unjustified causal stories.

These failure modes lead to a final meta-failure: *epistemic erosion by convenience*. If AI makes it easier to produce narratives than to verify them, organizations will drift toward publishing outputs that meet deadlines rather than standards. Over time, the institution's internal norms shift: verification becomes optional, counterarguments disappear, and rhetorical polish becomes the main success criterion. This is the deepest risk because it is not corrected by better models. It is corrected only by governance design.

Artifact (Save This)

Conceptual failure mode hierarchy.

1. **Process failures:** drift, citation theater, omission.
2. **Posture failures:** overconfidence, hidden lenses, causal inflation.
3. **Accountability failures:** attribution blur, unverifiable production.
4. **Institutional failure:** epistemic erosion by convenience.

“‘latex

4.3 Historical and Technical Lineage

4.3.1 Preceding approaches

Interpretive AI did not appear in a vacuum. The social sciences and humanities have spent decades building tools that sit somewhere between "information access" and "computational interpretation." The earliest and most durable lineage is search and retrieval. Libraries digitized catalogs; archives were indexed; citation databases enabled forward and backward traversal of scholarly networks. Keyword search, Boolean queries, and later ranked retrieval transformed how scholars locate texts, but these systems were fundamentally *non-generative*: they surfaced documents; they did not draft interpretations. Their governance properties were therefore comparatively straightforward. Search could bias attention, but it did not fabricate arguments. The principal risks were incompleteness, ranking bias, and overreliance on what was indexed.

A second lineage is extractive summarization and information extraction. Classical NLP tools attempted to compress documents by selecting salient sentences, extracting named entities, identifying topics, and mapping relations. In policy contexts and journalism-adjacent research, extractive summarization was attractive because it retained a direct link to the source: the summary was literally a subset of the original text. In humanities contexts, extractive tools were often used to support close reading at scale: concordances, keyword-in-context views, collocation statistics, and later topic models that suggested thematic clusters across corpora. These methods did not claim to "understand" meaning; they offered analytic hints. The human scholar still performed the interpretive leap, and the leap was visible as human responsibility.

A third lineage is computational social science and the broader world of quantitative models applied to social data: surveys, administrative records, experiments, and observational datasets. Here the pipeline is typically: data cleaning, model specification, inference, and interpretation. While many methodological debates exist (identification, measurement, external validity), the key point for our purposes is that the output is anchored in explicit artifacts: code, tables, plots, robustness checks. Even when interpretive narratives in social science are contested, there is a shared expectation that claims are traceable to analytic steps. This expectation is a governance asset: it constrains how far rhetoric can move beyond evidence, at least in principle.

A fourth lineage is the digital humanities and corpus-based criticism. Stylometry, authorship attribution, distant reading, network analysis of characters, and topic modeling all reflect a common ambition: to scale interpretive inquiry by making textual patterns computable. Yet even in these traditions, generation was not the core function. The tools produced features, clusters, and visualizations. Scholars then produced arguments in response. This separation mattered: it preserved a boundary between computational artifact and interpretive claim.

A fifth lineage is writing assistance and grammar checking, from spell-check to style suggestions to template-based report generators. These systems shaped prose but did not produce sustained

arguments with citations. They improved surface quality, but they did not alter the epistemic status of the content. Their risks were relatively local: homogenization of voice, subtle bias in suggestions, and overconfidence in automated correctness.

Across these preceding approaches, the shared pattern is that *interpretation remained scarce*. Either the system retrieved evidence (search), selected evidence (extractive summarization), computed patterns (digital humanities), or cleaned and modeled data (computational social science). The authoritative narrative still required human construction, and the institution's accountability structures were designed around that assumption. That assumption is precisely what frontier interpretive AI disrupts.

Artifact (Save This)

Lineage summary. Preceding systems largely separated *evidence access* from *interpretive authorship*. They could bias what was seen, but they rarely manufactured full arguments. Frontier generative systems collapse that separation, making the interpretive leap cheap and scalable.

4.3.2 Key inflection points

The first decisive inflection point was the emergence of large foundation models capable of producing coherent, genre-appropriate text. Earlier NLP systems could complete sentences, fill templates, or generate short passages. Foundation models can draft *argument-shaped discourse*: they produce introductions that frame a question, body sections that present evidence and counterpoints, transitions that mimic scholarly flow, and conclusions that offer implications. In SSH contexts, this capability matters because it produces a new kind of artifact: a plausible academic narrative that looks like it came from a disciplined process.

The second inflection point was the normalization of model use in iterative workflows. The moment users began to treat models not as question-answer systems but as writing collaborators—"revise this," "make it more rigorous," "add counterarguments," "strengthen the causal story," "write it in the style of a literature review"—the system became part of the epistemic production chain. Importantly, iterative prompting does not merely refine language; it changes claims. Each revision is an opportunity for drift: subtle upgrades of certainty, selective emphasis, and the introduction of plausible but unsupported scaffolding. The operational shift is that the model is no longer producing a one-off response; it is participating in the construction of institutional speech.

The third inflection point was retrieval augmentation (RAG) and, relatedly, tool use. Once models could retrieve documents, quote passages, and produce citations—whether through explicit retrieval systems or through long-context ingestion—they began to emulate the surface form of scholarship. This is a profound change in user perception. A narrative with citations triggers trust heuristics that a narrative without citations does not. Yet the presence of citations is not equivalent to citation

fidelity, and retrieval does not guarantee that the cited passage supports the claim being made. Nonetheless, RAG moved the perceived function of models from "text generator" to "evidence-aware assistant."

The fourth inflection point was long-context scaling. With larger context windows, users can provide entire policy dossiers, multiple academic articles, chapters, or corpora excerpts. This makes outputs feel grounded because the model has "seen" the material. But in SSH contexts, seeing is not enough. Meaning depends on context, genre, authorship, and historical situation. Long context also enables new risks: cherry-picking within a larger archive becomes easier to hide; and the model can produce confident synthesis that elides conflict. The paradox is that long context can reduce outright hallucination while increasing subtle misrepresentation.

The fifth inflection point was the mainstreaming of multimodal systems that can read tables, interpret charts, describe images, and integrate these into narrative outputs. In social science, this allows a model to write a narrative around a figure; in humanities, it allows the model to incorporate images of artifacts, paintings, or maps. Yet multimodal interpretation introduces additional ambiguity: visual evidence can be misread, or its significance can be overstated. Provenance across modalities is harder to track: is the claim supported by text, by data, or by an inferred reading of an image?

The sixth inflection point is institutional integration. Models are now embedded in document workflows: office suites, content management systems, research tools, and internal knowledge bases. This changes the default: the model is not a separate system used occasionally; it is present at the point of writing. In organizations, that creates an automation temptation: if the model can draft, then publication can be accelerated. The pipeline shifts from "human writes then publishes" to "model drafts then human glances." This is precisely where governance must intervene, because interpretive outputs require the opposite: evidence is the slow part, not the writing.

These inflection points together created the modern frontier: interpretive AI that can produce credible, citation-adjacent, long-form narratives inside the normal machinery of institutional communication. The chapter's claim is that this is not primarily a text-generation story; it is a governance and epistemic accountability story.

Risk & Control Notes

Inflection-point risk. The combination of (i) argument-shaped generation and (ii) citation-like outputs creates a credibility multiplier. Institutions that do not redesign review processes will mistake rhetorical form for epistemic substance.

4.3.3 What persisted vs what broke

The easiest way to see what changed is to separate what remained stable from what collapsed. What persisted is the fundamental epistemic requirement: claims must be supported by evidence, and interpretations must be contestable. SSH disciplines have always depended on traceability,

even when standards vary by field. Historians cite archives; social scientists cite data and methods; philosophers cite arguments; literary scholars cite passages. Disagreement is normal, but it is structured around shared practices of evidence and critique.

What broke is the relationship between writing and diligence. Historically, writing effort functioned as an imperfect proxy for epistemic work. Producing a literate, disciplined narrative typically required reading, synthesis, and attention. This proxy was always fallible—someone could write convincingly without being correct—but the cost structure limited scale. With AI, the cost of producing a coherent narrative approaches zero, while the cost of verifying that narrative remains roughly constant. The ratio inverts: it becomes easier to produce ten polished interpretations than to verify one. This inversion is the core governance shock.

A second thing that broke is the scarcity of scholarly form. In prior paradigms, the ability to write in the genre of scholarship was a signal of training and disciplinary immersion. AI erodes that signal. A model can generate writing that resembles academic discourse without possessing the discipline's internal commitments. This changes how readers allocate trust. It also changes institutional incentives: style becomes cheap and ubiquitous, so the differentiator becomes speed rather than rigor—unless governance reasserts rigor as a requirement.

A third break is the boundary between evidence access and interpretive claim-making. In earlier tools, retrieval showed sources; analysis produced artifacts; humans made arguments. Now the same system can retrieve, draft, and revise, producing an end-to-end narrative. This collapse of boundaries makes it harder to see where inference occurred, which in turn makes it harder to assign accountability. It also makes it easier to launder weak inference steps through smooth prose. The narrative becomes a continuous surface where the seams of reasoning are invisible.

A fourth break is the status of citations. Citation practice in SSH is both epistemic and social. Citations support claims and locate the argument within a discourse community. AI can generate citation-like structures, but those structures are not necessarily faithful. Worse, even when citations are real, they can be misaligned: a cited work may relate to the topic but not support the specific claim. The surface form of citation persists, but its meaning as a trust signal breaks. This is why the chapter insists on citation fidelity auditing: citations must be validated as claim-support relations, not treated as decorative tokens.

A fifth break is the speed of error propagation. In the classical world, interpretive errors propagate through slow citation networks and peer debate. In AI-assisted workflows, errors can be propagated immediately through internal knowledge bases, educational materials, and public communications. Drift becomes organizational: once a misleading narrative is generated and reused, it becomes part of institutional memory. In that sense, AI does not merely create new text; it creates new "epistemic artifacts" that can persist and compound.

What did not break is the need for editorial judgment. If anything, the need intensified. SSH work requires choosing lenses, representing uncertainty, acknowledging counterarguments, and deciding

what is worth emphasizing. AI can help explore these choices, but it cannot eliminate them. The true shift is that editorial judgment becomes the dominant cost and the dominant control point. The drafting step is no longer scarce; the judgment step is. Organizations that treat AI as a labor-saving device while underinvesting in judgment will fail.

Artifact (Save This)

Persistence vs break.

- **Persisted:** evidence, contestability, disciplinary norms, the necessity of judgment.
- **Broke:** writing-as-diligence proxy, scarcity of scholarly form, citation as a reliable trust cue, slow error propagation.

4.3.4 Why older intuitions fail

Older intuitions about writing, scholarship, and credibility fail under frontier generative systems because they were adapted to a world where producing polished text was costly and therefore correlated with effort. In that world, plausibility was a weak but usable heuristic: a coherent narrative written in the right genre, with citations, was more likely to reflect real work. Not always—history is full of plausible but wrong theories—but plausibility was not systematically optimized by machines. That assumption no longer holds.

The first failing intuition is "if it sounds plausible, it is probably correct." In SSH contexts, plausibility is often an aesthetic and rhetorical property: the story has the right structure, the right causal arc, the right academic tone. Models are trained to produce exactly that. They optimize for being the kind of text that a reader expects to read. Therefore plausibility becomes actively adversarial to truth: the more plausible the output, the more likely it is to be accepted without scrutiny. This is not because the model intends deception; it is because the model is good at satisfying genre expectations.

The second failing intuition is "citations imply support." In traditional scholarship, citations are imperfect but meaningful. Scholars can be sloppy or strategic, but there is a reputational cost for egregious mis-citation. In AI outputs, that social constraint is absent. A model can produce citations that are fabricated, irrelevant, or misaligned without suffering reputational consequences. Even when retrieval-based citations are used, the system may cite a document that mentions a concept while the claim requires a stronger relation. The intuition fails because citation becomes a formatting feature rather than a truth-maintenance mechanism. The correct intuition is: citations must be treated as *hypotheses* about support relations, requiring verification.

The third failing intuition is "summaries are safer than generation." Extractive summaries did preserve text; they could be audited easily. But frontier systems are not merely summarizing; they are synthesizing, generalizing, and framing. A generated "summary" can include interpretive moves and causal inferences that do not exist in the sources. Moreover, the boundary between summary

and commentary is blurred: what appears to be a neutral synthesis may embed a lens. Therefore, assuming that the output is safe because it is labeled a summary is a category error. The governance requirement is not the label; it is claim typing and evidence mapping.

The fourth failing intuition is "more context reduces risk." Long context can reduce certain hallucinations by giving the model access to relevant text. But in SSH, risk is often about selection and emphasis. More context increases the space of possible cherry-picks. It also increases the difficulty of human review: a reviewer is less likely to check whether the model misrepresented a small passage when the context window contains hundreds of pages. Therefore, larger context can increase risk unless the pipeline includes tools that make review tractable (passage-level citations, claim registries, and automated counterevidence retrieval).

The fifth failing intuition is "revision improves correctness." Human revision can improve clarity and accuracy, but it can also introduce bias. With models, revision is often driven by prompts that reward rhetorical strength. The model can "improve" text by removing hedges, increasing confidence, and smoothing contradictions. This yields outputs that look better and may read better but are less faithful to evidence. Therefore, iterative improvement without constraints is a drift amplifier. The intuition that rewriting makes things more correct fails unless rewriting is coupled to evidence checks.

The sixth failing intuition concerns plagiarism and originality. Traditional norms assume that copying is detectable and that authorship is tied to effort. Generative text complicates this: the system can produce novel phrasing that is not copied from any single source while still presenting ideas in ways that may be ethically problematic (e.g., reproducing a scholar's argument without attribution). Conversely, it can also generate unoriginal cliché scholarship that appears unique. This challenges how institutions teach and enforce integrity. The governance response is not to pretend the old norms apply unchanged, but to require disclosure practices and provenance logs appropriate to AI-assisted drafting.

The core reason older intuitions fail is that models industrialize the production of credibility cues. They can generate form faster than institutions can verify substance. The correct response is not moral panic or blanket rejection. It is to redesign the workflow so that the production of credibility cues is tied to the production of inspectable evidence links.

Risk & Control Notes

The new default is backwards. Under generative systems, the easiest thing to produce is the appearance of scholarship. The hardest thing to produce remains the evidentiary justification that scholarship requires. Governance must invert the workflow so that evidence comes first and rhetoric comes last.

4.3.5 Inherited lessons

Despite the novelty of frontier generative systems, the social sciences and humanities have inherited two deep reservoirs of institutional wisdom that should guide governance-first implementation: the norms of scholarship and the discipline of risk management. The first reservoir—scholarly practice—provides principles for epistemic integrity: reproducibility, citation discipline, method transparency, and contestability. The second reservoir—risk management—provides principles for operationalizing those norms at scale: separation of duties, documentation standards, auditability, and gating.

From scholarship, the most important inherited lesson is that interpretive work is not validated by eloquence but by traceability. A good argument is one that can be challenged. Peer review, archival citation, methodological appendices, replication packages, and footnotes are all mechanisms that enable contestation. The social sciences formalize this through data and code sharing; the humanities formalize it through textual citation and historiographical positioning. The shared lesson is that epistemic claims are inseparable from the artifacts that allow scrutiny. In AI pipelines, this means that the output must come with a dossier: claim registry, evidence links, lens declaration, and revision history. Without these, the output is not scholarship-adjacent; it is merely content.

Scholarship also teaches the value of negative space: acknowledging limitations, uncertainties, and counterarguments. Good work specifies what it does not show, what alternative interpretations exist, and what would change the conclusion. In the generative context, this becomes a control mechanism. The pipeline should require explicit "uncertainty and limits" sections and should treat the absence of counterevidence as a warning sign rather than a success. This is a direct translation of scholarly virtue into governance practice.

From risk management, the most important lesson is that production and approval must be separated. In finance, medicine, and safety-critical engineering, it is standard that the person who produces an analysis is not the sole approver of its release. Checkers exist; sign-offs are logged; changes are tracked. Interpretive AI creates a similar need because the system that drafts cannot be trusted to verify itself. A model can generate both a claim and an apparently plausible justification. Without independent verification, the organization collapses into self-confirmation.

Risk management also emphasizes the importance of *controls at the process level*. It is not enough to declare that a system will be used responsibly; the workflow must force responsible use. This includes mandatory gates for external publication, minimum documentation requirements, and monitoring for drift and mis-citation over time. In interpretive AI, drift is the analog of model drift: the system's outputs can shift as sources change, prompts evolve, or institutional incentives favor speed. Monitoring must therefore include periodic audits: sample outputs, verify citations, check claim typing, and measure omission patterns.

Another inherited lesson is that risk is not only technical; it is reputational, legal, and ethical. SSH-facing outputs often become institutional speech. A misleading narrative can trigger backlash,

litigation, or loss of trust. Therefore, governance must define escalation thresholds: when topics are politically sensitive, when human-subject data is involved, when the institution's brand is at stake, when legal exposure exists. In such cases, the pipeline should become more conservative: narrower source sets, stricter uncertainty language, stronger sign-off requirements, and explicit disclosure of AI assistance.

Finally, scholarship and risk management converge on a practical insight: documentation is the infrastructure of accountability. Footnotes and appendices are not ornaments; they are controls. Audit logs and sign-off records are not bureaucracy; they are what allow an institution to stand behind its outputs. In interpretive AI, this means that the system's success should be measured not by how quickly it drafts, but by how cheaply it produces a dossier that a reviewer can trust.

Artifact (Save This)

Minimum inherited lesson translated into implementation. Treat AI drafting as *production* and evidence verification as *approval*. Require a publication dossier (sources used, claims made, evidence links, lens declaration, revision history, human sign-off) for any external-facing output. Without the dossier, the output is not publishable.

‘‘‘

4.4 Technical Foundations

4.4.1 System or architectural components

A governance-first interpretive AI system for the social sciences and humanities is best understood as a *pipeline of accountable transformations*, not as a single model. The central design requirement is that every step that can change epistemic status must be visible, loggable, and reviewable. The system therefore decomposes into components that (i) control what evidence is available, (ii) structure what is asserted, (iii) constrain how narratives are generated, (iv) validate how evidence is cited and represented, and (v) produce artifacts that support institutional accountability. The goal is not maximal automation; the goal is *bounded automation* that makes verification cheaper than drafting.

Retrieval and memory layer. The retrieval/memory component governs what sources are visible at the moment of writing. In SSH settings, the "source universe" is rarely a single database; it is a patchwork of corpora: digitized archives, PDFs of scholarship, interview transcripts, field notes, datasets, codebooks, legislative records, and sometimes internal institutional documents. The retrieval layer must therefore implement (i) source admissibility rules (what is allowed), (ii) source ranking (what is surfaced), and (iii) provenance preservation (what is logged). A minimal system includes:

- an index over the corpus (with versioning),
- document metadata (type, date, origin, license/permissions, credibility tier),
- query logs and retrieval results logs,
- stable identifiers for passages (so citations can point to specific spans).

Memory in this context is not a vague "context window." It is an institutional design choice about retention and exposure. For instance, a policy unit may permit retrieval from peer-reviewed literature and official statistics but forbid retrieval from anonymous blogs; an ethnography project may forbid any inclusion of sensitive field notes beyond anonymized excerpts. These are governance constraints implemented technically.

Claim extraction and claim registry. Interpretive AI fails most often because the unit of accountability is unclear. Prose hides commitments. The claim extraction component converts text or sources into explicit claims. In an evidence-first workflow, claim extraction happens *before* drafting: the system proposes candidate claims that are potentially supportable by the sources. In a draft-first workflow, claim extraction happens *after* drafting: the system decomposes the narrative into claims so that each can be verified. Both patterns can work, but governance-first design prefers claim extraction early because it reduces the space of un-auditable prose.

Claims must be stored in a claim registry with fields such as:

- claim text (the atomic assertion),

- claim type (descriptive / interpretive / causal / comparative / normative),
- lens tag(s) for interpretive claims (e.g., historiography/theory),
- confidence/uncertainty level (or required uncertainty phrasing),
- evidence links (supporting passages, counterpassages, "insufficient evidence"),
- reviewer status (accepted/rejected/needs revision),
- revision provenance (when the claim changed and why).

This registry becomes the canonical audit object. The narrative is a view over the registry, not the other way around.

Evidence mapping and attribution module. Evidence mapping is the mechanism that binds claims to sources. In SSH settings, evidence mapping must handle:

- *passage-level anchoring* (specific spans, not whole documents),
- *multi-evidence support* (a claim may need multiple passages),
- *counterevidence* (passages that complicate or contradict),
- *context windows* (surrounding text to prevent decontextualization),
- *source-quality constraints* (some sources cannot support certain claims).

The attribution module then determines how citations appear in the narrative: footnotes, endnotes, inline references, or structured "evidence cards" in a review UI. Importantly, attribution must support *interpretive* claims, not just factual ones. For interpretation, the evidence link is often "passage p under lens L suggests reading r." The system must therefore record lens declarations and disclaimers as part of the link, rather than pretending interpretation is simple factual support.

Drafting model as a constrained renderer. In a governance-first architecture, the drafting model is not the epistemic decision-maker. It is a renderer that translates structured claims into readable text under editorial policy. The drafting component should be constrained by:

- claim typing rules (do not present interpretive claims as descriptive facts),
- uncertainty language policy (hedging required when evidence is mixed),
- counterevidence inclusion requirements (must mention strongest counterclaim),
- lens disclosure requirements (state the lens when interpretation depends on it),
- citation placement rules (every factual claim must carry a citation).

The model can still add connective tissue—transitions, explanation, organization—but those additions must be tagged as "non-claim narrative" so that they are not mistaken for evidence-bearing assertions.

Policy checker and language calibration. A policy checker enforces constraints that are easy to violate in prose: overconfident language, normative recommendations without basis, hidden lens adoption, or prohibited content classes (depending on institutional context). This can be implemented as:

- a rule-based linter for prohibited phrases ("proves," "undeniably") under weak evidence,
- a classifier for claim types and certainty levels,
- a "citation requirement" validator (no factual sentence without a link),
- domain-specific gates (e.g., for sensitive topics, require extra disclaimers).

The checker does not prove truth; it enforces discipline in how claims are presented.

Review interface and human workflow. Interpretive AI governance cannot be realized without a review UI that makes verification tractable. Reviewers need to see:

- the claim list,
- the evidence passages for each claim (with context),
- contradictions and alternative readings,
- lens declarations and uncertainty labels,
- diff views across revisions (what changed and why).

The UI should support separation of duties: a drafter (human or AI) proposes claims; a verifier checks evidence; an approver signs off for publication. The system must log each role's actions.

Provenance logs and publication dossier generator. Finally, the system must automatically generate dossiers. A dossier includes the frozen source set, retrieval logs, claim registry snapshots, evidence mappings, policy checker outputs, revision history, and sign-off records. In SSH contexts, the dossier is the analog of a replication package: it enables contestation and accountability.

Artifact (Save This)

Technical foundation principle. Treat the model as a component, not the system. The system is the chain: retrieval → claim registry → evidence mapping → constrained drafting → policy checks → human sign-off → dossier. If any link is missing, publication-grade accountability is unattainable.

4.4.2 Information flow

A disciplined interpretive AI pipeline can be described as a directed information flow where each stage produces artifacts that constrain later stages. The canonical flow is:

$$\text{Documents} \to \text{Claims} \to \text{Evidence Mapping} \to \text{Draft} \to \text{Checks} \to \text{Revision} \to \text{Publish}.$$

The key governance insight is that information does not merely move forward; it is *re-represented* at each stage, and each re-representation can distort meaning unless anchored.

Stage 1: Documents (source ingestion and curation). The pipeline begins with a defined corpus. The system must log: what was included, what was excluded, and on what basis. For

SSH contexts, corpus design is not neutral. It encodes assumptions about legitimacy, canon, and admissible evidence. Therefore, ingestion produces not just documents but metadata:

- corpus version ID and timestamp,
- document IDs and provenance,
- source-tier labels and licensing constraints,
- sensitive-data flags (e.g., human-subject materials).

Stage 2: Claims (extraction and structuring). The next transformation converts source material (and/or prompts) into candidate claims. The system should separate:

- *source-derived claims* (directly supported by passages),
- *interpretive hypotheses* (plausible but lens-dependent),
- *open questions* (gaps where sources are insufficient),
- *inadmissible claims* (not supportable or prohibited).

This stage is where "what the narrative will commit to" becomes explicit. If this stage is skipped, the institution will attempt to review prose, which is precisely what fluent systems are optimized to pass.

Stage 3: Evidence mapping (support and counter-support). The pipeline then binds claims to evidence. The output is a bipartite structure between claims and passages, augmented with relation types: supports, contradicts, contextualizes, or insufficient. In SSH, the mapping must accommodate:

- multiple supports for the same claim,
- multiple claims supported by the same passage,
- contradictions that are not errors but scholarly disputes.

This stage also records *citation granularity*. A citation to an entire book is not sufficient for claim-level audit; citations must include passage identifiers.

Stage 4: Draft (constrained rendering). Drafting is then a rendering step that assembles prose from claims and evidence. The draft should carry embedded markers (not necessarily visible to the end reader) that associate sentences or paragraphs with claim IDs. This enables later validation. If the draft is edited, the system should detect whether edits changed claim meaning. In governance-first design, humans are allowed to edit, but edits must not escape the claim registry.

Stage 5: Compliance and policy checks (discipline enforcement). The draft is then evaluated by checkers: citation requirements, uncertainty language rules, lens disclosure requirements, prohibited content checks, and structural checks (counterevidence included, limitations stated). These checks are not "compliance" in the legal sense; they are epistemic discipline checks that reduce systematic failure modes.

Stage 6: Revision (audited iteration). Revisions are where drift emerges. The system must therefore treat revision as an audited operation: each revision produces a diff in claims, evidence mappings, and rhetoric. The reviewer must be able to answer: did this revision strengthen claims beyond what evidence supports? Did it omit counterevidence? Did it change lens without disclosure? Revision is not a free improvement loop; it is a controlled process.

Stage 7: Publication (gated release). Publication is a governance boundary, not a technical step. External-facing outputs should require explicit sign-off. The system should lock the dossier at publication time: freeze the source set, freeze the claim registry, freeze the evidence mapping, and store the sign-off record. If later corrections are required, the system should support errata with traceability.

The value of describing the information flow is that it forces a shift in organizational thinking: writing is downstream of evidence. AI may accelerate drafting, but it should also accelerate documentation. A system that drafts quickly but cannot produce a dossier is mis-optimized for SSH use.

Risk & Control Notes

Where things usually go wrong. Organizations implement the drafting step first, then attempt to "add citations" later. This reverses the epistemic direction. Evidence mapping must precede rhetorical polishing, or the pipeline will manufacture confidence without support.

4.4.3 Interaction or control loops

Interpretive AI workflows are not linear; they are iterative. The control problem is to allow iteration for quality while preventing iteration from amplifying drift, omission, and overconfidence. Two loops dominate: the *draft–review–revise loop* and the *monitoring loop*.

Draft–review–revise loop (micro-control). The micro-control loop is the day-to-day workflow of producing an output. A governance-first loop is structured as:

$$\text{propose claims} \rightarrow \text{map evidence} \rightarrow \text{draft} \rightarrow \text{verify claims} \rightarrow \text{revise} \rightarrow \text{re-verify}.$$

The key is that verification is claim-based. Reviewers do not read the narrative and decide whether it "feels right." They inspect claim IDs, click evidence links, and accept/reject claims. Revision then modifies either the claim or its rhetorical presentation, and re-verification ensures the mapping still holds.

This loop must enforce separation of duties when stakes are high. A drafter (human or AI) should not be the sole verifier. Independent verification reduces self-confirmation. In small academic settings this might be peer review; in organizational settings it might be editorial review, legal/comms review, or a dedicated verification role. The system should support workflows where verification can be sampled (for low-risk internal use) or exhaustive (for high-risk publication), but it should never

pretend that verification is optional if the output will be treated as institutional speech.

Monitoring loop (macro-control). The macro-control loop addresses the reality that drift is not only within a document; it is across time. Sources change (new scholarship, updated statistics), retrieval indices change (new documents ingested), prompts evolve (teams adopt new templates), and institutional incentives shift. Therefore, the system must monitor:

- citation fidelity rates (how often citations truly support claims),
- drift rates across revisions (how often claims strengthen without evidence),
- omission indicators (how often counterevidence is absent),
- lens disclosure compliance (how often interpretation is labeled),
- calibration indicators (overconfident language under weak evidence).

Monitoring should trigger intervention: update policies, retrain users, restrict allowed use cases, or increase verification requirements for specific topics.

Pressure-loop resistance. A distinctive control challenge in SSH deployments is rhetorical pressure. Users (or stakeholders) may demand persuasive narratives, decisive conclusions, or simplified stories. AI systems comply easily. Governance must therefore include "pressure tests": prompts designed to push the system toward overclaiming, and controls that prevent it. For example, the system should refuse to remove uncertainty labels when evidence is mixed, or it should require explicit justification when upgrading a claim. This is analogous to adversarial testing in technical safety, but the adversary is organizational incentive rather than malicious input.

Lens-comparison loop. A constructive control loop in humanities and social science settings is multi-lens comparison. The system can be used to generate parallel narratives under different declared lenses $\pi_1, \pi_2, \ldots$. The loop is:

$$\text{same sources} \rightarrow \text{different lens constraints} \rightarrow \text{compare claim sets} \rightarrow \text{document disagreement.}$$

This loop turns pluralism into an explicit artifact rather than an implicit fight inside a single narrative. It is also a defense against hidden lens laundering: by forcing lens declaration and enabling comparison, the system makes framing choices visible.

Artifact (Save This)

Control loop design rule. Any loop that improves rhetoric must be coupled to a loop that re-checks evidence. Otherwise the system will systematically optimize the easiest objective (persuasion) while degrading the hardest one (accountability).

4.4.4 Assumptions and constraints

A technically competent interpretive AI system can still fail if it is built on unrealistic assumptions about sources, meaning, and institutional use. SSH contexts require explicit constraints because sources are messy and because interpretation is inherently lens-dependent.

Assumption 1: sources are incomplete and biased. Archives have gaps; datasets reflect measurement choices; scholarship is contested; interviews are situated. The system must assume that no source set is complete. Therefore, outputs must include scope statements: what corpus was used, what types of sources were excluded, and what limitations follow. The system must also assume that some sources are low credibility or strategically biased. This is why source-tiering is required: not all sources can support high-impact claims.

Assumption 2: sources may conflict legitimately. Conflicting sources are not necessarily errors; they may reflect real disagreement, different methods, or different perspectives. The system must therefore support contradiction as a first-class state. When sources conflict, the output should not "resolve" the conflict by rhetorical smoothing. It should represent the disagreement, label uncertainty, and, when appropriate, present alternative readings.

Assumption 3: interpretation requires lens declaration. In humanities and much of social theory, claims depend on the lens used. The system should assume that neutrality is not automatic. Therefore, it must label interpretive moves: "under a postcolonial lens," "within an institutionalist account," "from a feminist historiographical perspective," or analogous framing. The purpose is not to politicize outputs; it is to make them contestable. Hidden lenses are governance failures.

Assumption 4: users will request persuasion and simplification. In organizational contexts, users often want confident, clean narratives. The system must assume this pressure will exist. Therefore, uncertainty language policies and counterevidence requirements must be hard constraints, not optional suggestions.

Assumption 5: external publication is a distinct risk boundary. The system must treat public-facing or high-impact outputs differently from internal drafts. This implies constraints such as: mandatory human sign-off, mandatory dossier generation, stricter source-tier requirements, and stricter disclosure of AI assistance (as determined by institutional policy). If an institution cannot staff verification, the constraint should be "internal only."

Constraint: privacy and ethics for sensitive materials. SSH work often involves human subjects, field notes, interview transcripts, and sensitive community information. The system must enforce data minimization, consent boundaries, and restricted retrieval. It must also prevent sensitive content from being stored in logs beyond what is necessary for audit. In practice, this requires access controls, redaction policies, and sometimes air-gapped corpora.

Constraint: reproducibility. For accountability, the system must be able to reproduce outputs: same sources, same retrieval results, same claim registry snapshot, same drafting settings. Perfect

determinism may be unrealistic, but audit determinism must be achievable: an institution must be able to reconstruct what happened well enough to defend a published output.

Constraint: explicit labeling of uncertainty and viewpoint. The system must implement and enforce a controlled vocabulary for uncertainty. It must also label viewpoint: is the narrative descriptive, interpretive, or advocacy? SSH audiences are sensitive to rhetorical posture because posture shapes trust. The system should therefore treat posture labeling as a requirement, not a courtesy.

Risk & Control Notes

The most dangerous hidden assumption. Treating the corpus as "the truth" rather than "a curated, partial view." In SSH work, corpus definition is itself a political and epistemic act. Governance requires explicit disclosure of corpus boundaries and the consequences of those boundaries.

4.4.5 Technical bottlenecks

The technical bottlenecks of interpretive AI in SSH settings are not primarily about generating fluent text. They are about making the output *evidence-disciplined at scale*. Several bottlenecks dominate.

Claim–evidence alignment. The hardest technical problem is mapping claims to supporting evidence with high precision. In SSH writing, claims are often abstract: "this policy reflects a shift in legitimacy," "this narrative functions as moral repair," "the discourse constructs the subject as X." Evidence for such claims is rarely a single sentence; it is a pattern across passages and context. Automated alignment methods can produce plausible links, but precision is crucial. A system that frequently links claims to irrelevant passages will train users to ignore the evidence interface, collapsing governance back into prose review.

Context preservation and decontextualization detection. Even when a passage is relevant, it can be misused. A quote can be accurate but misleading if qualifiers are omitted. Detecting decontextualization is technically difficult because it requires modeling not just the quoted span but its rhetorical function in the source. This bottleneck is especially acute in humanities interpretation where meaning depends on irony, genre, and historical context.

Hallucinated citations and mis-citations. Citation theater is partly a governance problem and partly a technical one. Systems must detect fabricated references, verify that cited works exist, and verify that cited passages support the claim. This is nontrivial at scale across heterogeneous corpora (books, PDFs, scanned documents, archival metadata). Moreover, even correct citations can be misaligned: the source is about the topic but does not support the specific inference.

Measuring framing bias and omission. Framing bias is often about what is absent. Detecting

omission requires knowing what would have been relevant: alternative readings, counterevidence, minority perspectives, competing identification strategies. Automated methods can suggest missing counterarguments, but objective measurement is difficult. Nonetheless, some proxies are feasible: diversity of source tiers, inclusion of counterclaims, and retrieval of top opposing passages. The bottleneck is turning these proxies into reliable, non-gameable signals.

Scalable verification. Human verification is expensive, and SSH verification is slow because it requires reading and contextual judgment. The bottleneck is designing systems where verification time is minimized without becoming performative. This motivates interface design (evidence cards, claim sorting, risk prioritization) and sampling strategies (verify high-risk claims exhaustively, lower-risk claims by sampling). The challenge is that risk is not always obvious: a small interpretive misframing can cause reputational harm if it touches sensitive identities or contested history.

Lens tracking and multi-lens comparability. The system must track which lens produced which claim and enable comparison. Technically, this requires structured representation of lenses and of how they change the claim set. Without this, systems will default to hidden lens laundering, presenting a single narrative as neutral. Implementing lens tracking in a way that is usable—not overly academic, not too coarse—is a practical bottleneck.

Robustness to retrieval noise. RAG pipelines are sensitive: small differences in retrieved documents can change the narrative. In SSH, these changes can be substantive. Therefore, the system must measure sensitivity: how much does the claim set change under retrieval perturbations? If small perturbations produce large changes, the output is not stable enough to publish without strong disclaimers. Building sensitivity analysis into the pipeline is technically and operationally challenging.

Integration with enterprise controls. Finally, bottlenecks include non-ML engineering: access control, auditing, role management, and integration with document workflows. SSH outputs often pass through editorial systems, legal review, and institutional comms pipelines. If the interpretive AI system cannot integrate with these controls, governance will fail at the organizational level even if the model-level behavior is disciplined.

The chapter's posture is that these bottlenecks are not reasons to avoid interpretive AI; they are reasons to define the frontier correctly. The frontier is not "can the model write an essay." The frontier is "can the system make its essay defendable." That is a pipeline question. It requires investment in evidence mapping, review tools, sensitivity analysis, and provenance infrastructure—not just better text generation.

Artifact (Save This)

A practical definition of progress. Interpretive AI advances meaningfully for SSH when the marginal cost of producing an *audit-grade dossier* falls faster than the marginal cost of producing a polished narrative. Until then, institutions will remain structurally tempted to publish what they cannot defend.

4.5 Mathematical Foundations

4.5.1 Formal problem framing

The purpose of formalization in this chapter is not to "prove" correctness of interpretation. Social sciences and humanities outputs are inherently plural: different lenses can yield different defensible readings from the same corpus. What mathematics can provide is a disciplined language for (i) what objects the system manipulates, (ii) what constraints constitute evidentiary discipline, and (iii) where instability arises when generation is iterated under retrieval and revision. The goal is bounded formalism: enough structure to support evaluation and governance, without pretending that interpretive validity reduces to a single scalar score.

Let $S = \{s_1, \ldots, s_m\}$ denote a finite set of sources, where each source s_j is associated with metadata $\mu(s_j)$ (type, date, provenance tier, licensing, sensitivity flags). We assume that each source can be decomposed into passages (spans) $\mathcal{P}(s_j) = \{p_{j1}, \ldots, p_{jn_j}\}$, each with a stable identifier. Let $\mathcal{P} = \bigcup_j \mathcal{P}(s_j)$ denote the universe of passages.

A narrative output N is represented as a structured object:

$$N \equiv (T, C, \tau, \pi, \mathcal{L}, \mathcal{A}),$$

where:

- T is the surface text (prose).
- $C = \{c_1, \ldots, c_k\}$ is a set of atomic claims extracted from T or proposed prior to drafting.
- $\tau : C \to \mathcal{K}$ maps each claim to a *claim type* in a finite type set $\mathcal{K}$ (e.g., descriptive, interpretive, causal, comparative, normative).
- π denotes editorial/framing constraints (lens declarations, admissible sources, uncertainty policies).
- $\mathcal{L}$ is a set of lens tags used in the narrative (theoretical frames, historiographical schools).
- $\mathcal{A}$ is an attribution structure linking claims to evidence passages.

Evidence is modeled by a mapping:

$$E : C \to 2^{\mathcal{P}},$$

where $E(c_i) \subseteq \mathcal{P}$ is the set of passages considered supporting evidence for claim c_i. However, SSH reasoning requires more nuance than "supports" vs "does not support." We therefore define a signed evidence relation:

$$R(c_i, p) \in \{+1, 0, -1\},$$

where $+1$ denotes support, -1 denotes counterevidence/contradiction, and 0 denotes irrelevance or

insufficient relation. For a given claim c_i, we define:

$$E^+(c_i) = \{p \in \mathcal{P} : R(c_i, p) = +1\}, \quad E^-(c_i) = \{p \in \mathcal{P} : R(c_i, p) = -1\}.$$

We also allow an "insufficient evidence" state, operationalized by $E^+(c_i) = \emptyset$ and/or a low-confidence label (defined below).

Because interpretation is lens-dependent, we attach lens declarations to claims:

$$\ell : C \to 2^{\mathcal{L}},$$

where $\ell(c_i)$ is the set of lenses under which the claim is asserted. For descriptive claims, $\ell(c_i)$ may be empty; for interpretive claims it is typically non-empty and must be disclosed.

We define a *confidence labeling function* $\gamma : C \to [0, 1]$ representing the system's stated confidence (or, more precisely, the strength of language used). In governance-first design, $\gamma(c_i)$ is not a truth probability; it is a calibration target: high rhetorical confidence should be permitted only when evidence support is strong and counterevidence weak.

The system's objective is not to maximize plausibility but to maximize *utility under evidence discipline*. Let $U(N)$ be a utility function capturing usefulness, coherence, readability, and task relevance. Then a simple penalized objective is:

$$\max_N \; U(N) \; - \; \lambda \sum_{i=1}^{k} \mathbf{1}\Big[E^+(c_i) = \emptyset \; \wedge \; \tau(c_i) \in \mathcal{K}_{\text{fact}}\Big],$$

where $\mathcal{K}_{\text{fact}} \subset \mathcal{K}$ denotes claim types treated as requiring strict evidence (typically descriptive and sometimes causal claims). This captures the baseline: factual claims without evidence are violations. But SSH outputs require additional penalties for (i) hidden lenses, (ii) overconfidence under weak evidence, and (iii) omission of counterevidence. We can extend the objective with penalty terms:

$$\max_N \; U(N) - \lambda_1 \sum_i v_{\text{unsupported}}(c_i) - \lambda_2 \sum_i v_{\text{hidden-lens}}(c_i) - \lambda_3 \sum_i v_{\text{overconf}}(c_i) - \lambda_4 \sum_i v_{\text{no-counter}}(c_i),$$

where each $v(\cdot)$ is a violation indicator or severity score.

For example:

$$v_{\text{unsupported}}(c_i) = \mathbf{1}\Big[E^+(c_i) = \emptyset \wedge \tau(c_i) \in \mathcal{K}_{\text{fact}}\Big],$$

$$v_{\text{hidden-lens}}(c_i) = \mathbf{1}[\tau(c_i) \in \mathcal{K}_{\text{interp}} \wedge \ell(c_i) = \emptyset],$$

$$v_{\text{overconf}}(c_i) = \mathbf{1}\Big[\gamma(c_i) > g\Big(E^+(c_i), E^-(c_i), \mu(\cdot)\Big)\Big],$$

where $g(\cdot)$ is a policy function specifying the maximum allowed rhetorical confidence given sup-

port/counterevidence and source tiers, and $\mathcal{K}_{\text{interp}}$ is the set of interpretive claim types. Finally,

$$v_{\text{no-counter}}(c_i) = \mathbf{1}\left[\tau(c_i) \in \mathcal{K}_{\text{high-stakes}} \wedge \text{risk}(c_i) \geq r_0 \wedge E^-(c_i) = \emptyset\right],$$

which encodes a policy that high-stakes claims should explicitly surface counterevidence when available.

This formal framing makes a governance point: interpretive AI should be optimized under constraints that privilege *inspectability and contestability* over rhetorical smoothness. The mathematical object of interest is not text alone but $(C, E^+, E^-, \ell, \gamma)$ attached to the text.

Artifact (Save This)

Governance-relevant representation. A narrative is modeled as text plus claim objects with (i) type, (ii) evidence supports/counterevidence, (iii) lens tags, and (iv) confidence labels. Evaluation and control operate primarily on these attached structures, not on the prose.

4.5.2 State, action, or representation spaces

To reason about pipeline behavior—especially drift under revision—it is useful to model interpretive generation as a controlled process evolving over time. We define a state space that includes what the system currently "knows" and what it has already committed to, and an action space that captures the ways the system can change the narrative.

State. Let the system state at step t be:

$$x_t = (S_t, P_t, C_t, E_t, \pi_t, \theta_t),$$

where:

- $S_t \subseteq S$ is the current retrieved source set (documents selected for use).
- $P_t \subseteq \mathcal{P}$ is the current retrieved passage set (evidence snippets).
- C_t is the current claim set (possibly edited over time).
- E_t is the current evidence mapping, including $E_t^+(c)$ and $E_t^-(c)$.
- π_t are the active constraints (lens declarations, admissible sources, uncertainty policy).
- θ_t denotes model/prompt parameters and drafting settings (temperature, system prompts, templates).

We can also include a revision history H_t as part of state, but for clarity we treat it as derived from the trajectory $(x_0, \ldots, x_t)$.

Actions. The system can take actions that alter the state. A minimal action set includes:

- *Retrieve*: modify S_t and P_t by querying the corpus (change evidence visibility).

- *Propose claim*: add a claim c to C_t, assign type $\tau(c)$ and lens tags $\ell(c)$.
- *Edit claim*: modify the content of c (strengthen/weaken/clarify), potentially changing type or lens.
- *Assign evidence*: update $E_t(c)$ by linking passages as support or counterevidence.
- *Draft text*: render T from (C_t, E_t, π_t), introducing surface phrasing and confidence signals.
- *Revise text*: rewrite parts of T, which may implicitly change claims and confidence unless constrained.
- *Update constraints*: change π_t (e.g., require counterevidence, tighten uncertainty language).
- *Approve/reject*: human action that changes acceptance statuses of claims and permits publication gating.

In governance-first design, the key is to ensure that text revisions are not "free actions" that bypass the claim structure. A revision should either (i) be purely stylistic and not change claim content, or (ii) trigger an explicit update to C_t and a re-check of E_t. This is how the mathematics connects to auditability: actions that change epistemic commitments must leave traces in the structured state.

Policy and constraints as control. A governance policy is a mapping from states to allowed actions:

$$\Pi : x_t \mapsto \mathcal{A}(x_t),$$

where $\mathcal{A}(x_t)$ is the set of permissible actions at state x_t. For example, under high-stakes conditions, Π may disallow "publish" until all factual claims have evidence links and a human verifier has signed off. Under sensitive-data constraints, Π may disallow retrieval from certain documents entirely.

Representation spaces. In practice, the system operates over representations: embeddings for retrieval, intermediate summaries, and text tokens. The formal model abstracts away from these internal representations and focuses on governance-visible objects. Nevertheless, it is helpful to distinguish:

- *Evidence representation*: passage identifiers plus surrounding context windows.
- *Claim representation*: canonical text for the claim plus type and lens tags.
- *Narrative representation*: the rendered text plus alignment pointers back to claims.

The critical principle is that these representations should be aligned: a reviewer should be able to go from any sentence in T to the claim(s) it expresses and to the evidence passages that support it. This alignment is the mathematical backbone of the dossier.

Risk & Control Notes

Where drift enters the state model. Drift occurs when the system takes "revise text" actions that change implied claims or confidence without triggering updates in C_t, E_t, or $\gamma(\cdot)$. Governance requires either restricting such actions or forcing re-extraction and re-verification after them.

4.5.3 Objective functions and constraints

The objective of an interpretive AI pipeline can be written as a constrained optimization problem over narratives N or over trajectories $(x_0, \ldots, x_T)$. Because SSH outputs are iterative, it is often more accurate to treat the system as optimizing over a sequence of actions.

Single-output constrained objective. A baseline formulation is:

$$\max_N \; U(N) \quad \text{s.t.} \quad \forall c \in C_{\text{fact}} : \; |E^+(c)| \geq 1,$$

where $C_{\text{fact}} = \{c \in C : \tau(c) \in \mathcal{K}_{\text{fact}}\}$. This captures the hard constraint that factual claims require evidence. We then add:

$$\forall c \in C_{\text{interp}} : \; |\ell(c)| \geq 1,$$

requiring lens disclosure for interpretive claims, and a calibration constraint:

$$\forall c \in C : \; \gamma(c) \leq g\Big(E^+(c), E^-(c), \mu(\cdot)\Big),$$

where $g(\cdot)$ specifies allowable rhetorical confidence given the evidence state and source tiers. In other words, when evidence is weak or conflicted, the narrative must not be written with maximal certainty.

We may also enforce counterevidence inclusion for high-stakes claims:

$$\forall c \in C_{\text{hs}} : \; \text{risk}(c) \geq r_0 \Rightarrow \big(|E^-(c)| \geq 1 \; \vee \; \text{explicitly declare “no counterevidence found”}\big).$$

This is less about truth and more about contestability: the system must show that it searched for counterevidence and what it found.

Trajectory-level objective. For iterative workflows, let π define allowed actions, and let $J(\cdot)$ be a cumulative objective:

$$\max_{a_0, \ldots, a_{T-1}} \sum_{t=0}^{T-1} \beta^t \, r(x_t, a_t) \quad \text{s.t.} \quad a_t \in \mathcal{A}(x_t),$$

where $r(x_t, a_t)$ encodes reward for improving usefulness and reducing violations, and $\beta \in (0, 1]$ is a discount factor. A natural reward is negative for violations and positive for improved reviewability:

$$r(x_t, a_t) = \Delta U_t - \lambda \Delta V_t - \eta \Delta R_t,$$

where ΔU_t is improvement in utility, ΔV_t is change in violation severity (unsupported claims, hidden lenses, miscalibration), and ΔR_t is change in review burden (e.g., number of unlinked claims). The goal is to make the system prefer actions that reduce review cost by making evidence explicit.

Penalty design as governance. The choice of penalties λ_i is not purely technical. It encodes institutional values: how much the organization prioritizes caution over completeness, pluralism

over single narratives, or auditability over rhetorical polish. In SSH contexts, the penalty for hidden lenses may be high because hidden framing undermines contestability, while the penalty for lower stylistic polish may be low. This contrasts with marketing contexts where style might be prioritized. The mathematical point is that λ-weights are governance parameters.

Hard constraints vs soft penalties. Certain constraints should be hard at publication boundaries:

- no uncited factual claims,
- no unlabeled interpretive claims,
- no high-confidence language under mixed evidence (as defined by policy),
- documented corpus and retrieval logs,
- human sign-off for external-facing publication.

Other preferences can be optimized softly: readability, narrative structure, completeness. This separation is crucial because soft optimization under rhetorical pressure will otherwise trade away epistemic discipline.

Interpretive pluralism as a constraint. A distinctive SSH requirement is that the system should not collapse contested interpretations into a single voice. We can model this by requiring multi-lens outputs when the literature is contested:

$$\text{contested}(S_t) = 1 \Rightarrow |\mathcal{L}| \geq 2,$$

or, more operationally, requiring a "competing interpretations" section with at least k alternative framings. The system is not asked to choose the winner; it is asked to represent the contestation faithfully.

Artifact (Save This)

Governance as constraint specification. In this chapter, governance is represented mathematically as the selection of (i) hard constraints defining publication-grade evidentiary discipline and (ii) penalty weights that encode institutional risk tolerance. The drafting model optimizes within those bounds; it does not define them.

4.5.4 Sources of instability or fragility

The main fragilities of interpretive AI pipelines arise from sensitivity to retrieval and from semantic drift under iteration. These are not merely "bugs"; they are structural instability mechanisms.

Retrieval sensitivity and framing instability. Let R denote a retrieval operator that maps a query q and corpus S to a set of passages P:

$$P = R(q, S).$$

In practice, retrieval depends on embeddings, ranking, and filtering; small changes in q, the index, or ranking heuristics can change P. In SSH contexts, small changes in P can produce large changes in narrative framing because the narrative is constructed by selection. We can define a sensitivity measure:

$$\Delta_{\text{frame}}(q) = d\big(N(R(q,S)),\ N(R(q',S))\big)\,,$$

where q' is a perturbed query or retrieval configuration and $d(\cdot,\cdot)$ measures difference between narratives (claim set distance, lens distance, or emphasis distance). A high Δ_{frame} indicates that the narrative is unstable to retrieval noise. In finance or engineering, this might be tolerated if outputs are used internally; in SSH publication contexts, high instability implies the narrative should be treated as exploratory and should include explicit caveats.

We can also measure stability at the claim level. Let $C(P)$ denote the set of claims produced when the evidence set is P. Then:

$$\Delta_C = |C(P) \triangle C(P')|,$$

where $\triangle$ denotes symmetric difference. If Δ_C is large for small retrieval changes, the pipeline is not robust. The governance implication is that systems should perform retrieval perturbation tests and report stability metrics in the dossier.

Iterative paraphrasing and semantic drift. Drift arises when revision actions change text meaning without corresponding updates in claims and evidence. Even if claims are tracked, drift can occur as claims are gradually strengthened. Consider a sequence of revisions producing claim texts $c^{(0)}, c^{(1)}, \ldots, c^{(T)}$. Drift can be modeled as:

$$\delta_t = d_c\Big(c^{(t)}, c^{(t-1)}\Big)\,,$$

where d_c measures semantic change relevant to evidentiary support (e.g., increased quantification, increased certainty, broadened scope). A dangerous drift pattern is monotone strengthening:

$$\text{strength}(c^{(t)}) \geq \text{strength}(c^{(t-1)}) \quad \text{while} \quad |E^+(c^{(t)})| \text{ does not increase.}$$

This indicates that rhetoric is being upgraded without new evidence. The system should detect and flag this.

Drift is amplified by prompts like "make it more persuasive" or "tighten the argument." These prompts induce actions that increase rhetorical force. Without constraints, the model will remove hedges, generalize from narrow evidence, and compress complexity. In SSH contexts, such compressions often change truth conditions. Drift is therefore not accidental; it is a predictable outcome of optimizing for readability and decisiveness.

Lens switching without disclosure. Another instability arises when the model implicitly changes the lens used in interpretation while retaining surface continuity. This can happen when the system draws from different scholarly traditions across retrieval steps. The narrative may begin in an

institutionalist framing and drift into a rational-choice framing without explicit declaration. This is a form of "lens drift." We can model lens assignments $\ell_t(c)$ over time and detect instability:

$$\Delta_\ell = \sum_{c \in C} \mathbf{1}[\ell_t(c) \neq \ell_{t-1}(c)] \,.$$

High Δ_ℓ without explicit disclosure is a governance failure because it changes the interpretive commitments.

Omission instability. Omission is difficult to measure, but instability can be inferred by comparing which counterevidence is retrieved and included under small perturbations. If the presence or absence of a single counterpassage flips the narrative posture from cautious to definitive, the system is brittle. This suggests that the output is not adequately conditioned on counterevidence by default.

Human–AI interaction fragility. Finally, instability can arise from human editing. Humans may revise the text directly, introducing claims not present in the claim registry, or they may delete uncertainty phrasing to satisfy stakeholder demands. The system must treat human edits as actions that require re-extraction and re-verification, otherwise the pipeline becomes porous. From a mathematical perspective, unlogged human edits are unobserved actions that break the audit trail.

Risk & Control Notes

Instability summary. Interpretive AI is fragile because (i) retrieval is a selection mechanism that changes framing, and (ii) revision is a rhetorical optimization mechanism that induces drift. Stability requires explicit sensitivity testing and re-verification after revisions.

4.5.5 What theory does NOT guarantee

Formalization can clarify objects and constraints, but it cannot grant properties that the domain does not support. This section states explicitly what mathematical structure does *not* provide, to prevent false confidence.

Coherence does not imply truth. A high value of $U(N)$—coherence, readability, persuasive flow—does not imply that claims are correct. In fact, coherence can be anti-correlated with truth under pressure because the model can smooth contradictions and remove uncertainty. The formal objective explicitly separates utility from evidentiary violations for this reason.

Citations do not imply support. The existence of an evidence link $E^+(c) \neq \emptyset$ does not guarantee that the evidence truly supports the claim. Evidence mapping is itself a fallible process. A passage may mention the relevant topic without supporting the inference. Therefore, a dossier with citations is necessary but not sufficient. Human verification or high-precision automated checks are still required.

Neutrality is not automatic. The presence of a corpus S and constraints π does not imply that

the output is neutral. Corpus selection is value-laden; retrieval ranking embodies bias; lens choices shape interpretation. The best that the system can do is *declare* its lens and scope and present alternative readings when contestation exists. The mathematical model can enforce lens disclosure, but it cannot make a lens disappear.

Evidence discipline does not resolve interpretive disagreement. Even with perfect evidence links, SSH domains can sustain multiple defensible interpretations. The model can represent pluralism, but it cannot guarantee a unique "correct" narrative. If an institution demands a single definitive account in a contested domain, the pipeline's best response is to downgrade certainty and present competing frames.

Constraint satisfaction does not eliminate omission. A narrative can satisfy "every factual claim has a citation" and still mislead by omitting crucial counterevidence or by selecting a narrow corpus. Omission is a second-order property that depends on what is not included. Therefore, constraint satisfaction is a baseline, not an endpoint. Sensitivity tests, counterevidence requirements, and scope disclosures are needed.

No guarantee against adversarial or incentive-driven misuse. Even a well-designed pipeline can be pressured by institutional incentives. Users may attempt to circumvent controls by rephrasing claims to avoid citation requirements or by using the system outside approved contexts. Formal constraints help, but they do not replace governance enforcement: access controls, training, auditing, and consequences.

No guarantee of stability under distribution shift. As sources update, as new scholarship emerges, and as topics evolve, the retrieval and narrative outputs can shift. The mathematical model can recommend monitoring, but it cannot guarantee stable outputs across time without explicit versioning and re-validation.

The practical implication is that mathematics should be treated as a governance aid, not as a proof of correctness. Formal structure helps institutions define what to check, log what happened, and detect where drift and instability occur. It does not turn interpretation into a solved problem. In SSH contexts, the honest goal is not guaranteed truth; it is accountable, contestable interpretation under explicit constraints.

Artifact (Save This)

Bounded guarantee statement. The formalism in this chapter can guarantee only process properties (claim typing, evidence linking, lens disclosure, logged revisions) when implemented as hard controls. It cannot guarantee epistemic properties (truth, neutrality, unique interpretation). Governance must therefore treat the system as an accountable assistant, not an authority.

4.6 Evaluation and Validation

4.6.1 Why naïve metrics fail

Evaluation is where interpretive AI most frequently collapses into self-deception. The surface fluency of modern models tempts institutions to reuse familiar NLP metrics and familiar human review instincts. Both are miscalibrated for social sciences and humanities use, because the failure modes are not primarily lexical or stylistic. The failures are epistemic: claims detach from sources, rhetorical confidence outruns evidence, and framing decisions hide their own assumptions. Any evaluation regime that treats "good writing" as a proxy for "good scholarship" will reward exactly the outputs that create the greatest institutional risk.

Consider the canonical metrics inherited from machine translation and summarization: BLEU, ROUGE, and related n-gram overlap scores. These metrics reward similarity to reference texts. They assume that there exists a stable target string (or family of close strings) and that overlap is meaningful. In interpretive SSH work, the premise fails. There is rarely a single reference narrative. Multiple defensible readings exist; disagreement is normal; and novelty in framing can be valuable. A system that matches a reference narrative may be wrong under a different lens, while a system that diverges may be offering a legitimate alternative. Therefore, overlap metrics do not measure what matters and can perversely penalize honest pluralism.

Even when the task is framed as "summarization," standard metrics remain weak because interpretive systems are not purely extractive. They synthesize, generalize, and propose causal or conceptual linkages. A summary can be lexically similar to the source while still introducing an unjustified implication; conversely, a summary can be lexically different while remaining faithful. BLEU/ROUGE are blind to the central question: does each claim have support in the sources, and is the support represented accurately?

Naïve human evaluation fails for a different reason: humans often rate outputs based on readability, clarity, completeness, and confidence. Those are precisely the dimensions where models excel, and where failure can be most persuasive. In SSH contexts, style is a credibility amplifier. A coherent narrative with citations feels like disciplined work even when the citations are misaligned or decorative. Readability ratings therefore risk rewarding "persuasive hallucinations"—text that is rhetorically strong but evidentially weak. Worse, skilled rhetoric can *hide* evidentiary failures: a narrative can smooth over contradictions, omit counterevidence, and present contested claims as consensus without triggering obvious red flags.

A deeper issue is that many organizations evaluate outputs at the document level: a reviewer reads a memo and asks whether it seems acceptable. But interpretive errors are often local and compositional. One unsupported causal claim can create liability in an otherwise careful report. One decontextualized paraphrase can misrepresent a community or a historical event. Therefore, evaluating the whole narrative as "good" or "bad" is not granular enough. Evaluation must operate

at the level of claims, evidence links, and revision histories.

Finally, naive evaluation fails because it ignores the process. Interpretive AI is typically used iteratively: retrieve, draft, revise, strengthen, shorten, simplify, adapt to audience. The most dangerous failures emerge across iterations: drift, increasing certainty, and omission of caveats. A one-shot evaluation of the final text cannot detect whether the system was pushed toward overconfidence or whether the evidence mapping decayed over revisions. For governance-first deployment, process-level evaluation is mandatory. The question is not only "is this output correct?" but "is this output *defendable*?" Defendability depends on traceability: what sources were used, what claims were made, how they changed, and what was verified.

Risk & Control Notes

Evaluation trap. In SSH settings, the most dangerous outputs are those that score highly on readability and perceived completeness. These are the outputs most likely to pass casual review while containing subtle evidence failures, hidden lenses, or omission-driven framing distortions.

4.6.2 Appropriate evaluation units

A governance-first evaluation regime begins by choosing the correct unit of analysis. In interpretive work, the appropriate units are not tokens or documents but *claims and their evidentiary attachments*. The central evaluative questions are: (i) what is being asserted, (ii) what supports it, (iii) what contradicts it, (iv) how confident is the language, and (v) how stable are these relations across revisions and retrieval perturbations.

Claim-level verification. Let $C = \{c_i\}$ be the claim set extracted from a narrative. For each claim c_i, the system should assign a verification status:

$$\mathrm{ver}(c_i) \in \{\text{Supported}, \text{Contradicted}, \text{Mixed}, \text{Unknown}\}.$$

This is not a perfect notion of truth, especially for interpretive claims. Therefore, evaluation must incorporate claim type. For descriptive claims, "Supported" should mean there exists at least one source passage that directly supports the assertion and no strong counterpassage. For causal claims, "Supported" should typically require either (i) explicit methodological evidence (identification assumptions stated and supported) or (ii) downgrading to a hypothesis if such evidence is absent. For interpretive claims, "Supported" may mean "textually plausible under declared lens with cited passages," while "Mixed" may indicate that alternative readings exist or that passages can be reasonably read otherwise. The important point is that interpretive verification is about *contestability*, not binary truth.

Citation fidelity. Citation fidelity evaluates whether the cited material supports the specific claim

it is attached to. For each claim c_i and its cited passages $E^+(c_i)$, define a fidelity predicate:

$$\text{fid}(c_i) = \mathbf{1}\left[\text{cited passages support the claim as stated}\right].$$

In practice, fidelity is not binary; it can be graded by strength:

$$\text{fid}(c_i) \in \{\text{Strong}, \text{Partial}, \text{Weak}, \text{No support}\}.$$

A frequent failure in SSH settings is *topic relevance masquerading as support*. A passage may discuss the same theme but not justify the inference. Fidelity evaluation must explicitly test this. The output of citation fidelity evaluation is a distribution across claims: what fraction of claims have strong support, what fraction are weakly supported, and which claims are unsupported.

Contradiction and unknown detection. Interpretive work often involves contested sources. Systems should be evaluated on their ability to detect contradiction and represent uncertainty. Given evidence sets $E^+(c_i)$ and $E^-(c_i)$, the system should flag claims where counterevidence exists and should downgrade rhetorical certainty. A useful evaluation unit is the calibration relationship between contradiction states and confidence language. If a claim is "Mixed" or "Unknown," the narrative should not use maximal certainty phrases. Therefore, evaluation should measure:

$$\text{cal}(c_i) = \mathbf{1}\left[\gamma(c_i) \leq g(E^+(c_i), E^-(c_i))\right],$$

where $\gamma(c_i)$ proxies rhetorical confidence. In practical terms: does the system change posture when evidence is conflicted?

Drift across revisions. Because interpretive pipelines are iterative, evaluation must measure how claims evolve across revisions. Let $C^{(t)}$ be the claim set at revision t. Drift can be measured as:

$$\Delta_C^{(t)} = |C^{(t)} \triangle C^{(t-1)}|,$$

and claim-strengthening drift can be measured by tracking changes in scope, quantification, and certainty. A simple operational metric is: how often do claims become stronger without new evidence being added? This can be approximated by counting cases where $\gamma(c_i)$ increases while $|E^+(c_i)|$ does not.

Lens disclosure and posture labeling. In SSH work, evaluation should include whether interpretive lenses are disclosed. For claims typed as interpretive, measure:

$$\text{lens}(c_i) = \mathbf{1}\left[\ell(c_i) \neq \emptyset\right].$$

Additionally, evaluate whether the narrative distinguishes between descriptive sections, interpretive sections, and normative recommendations. This can be measured at the paragraph or section level: are the relevant labels present, and are they consistent with the underlying claims?

Coverage of counterevidence. Because omission is a core failure mode, evaluation should include whether the narrative includes at least one strong counterclaim for high-stakes topics, or whether it explicitly states that no counterevidence was found after search. This can be operationalized as a requirement for a "counterevidence" section or as a per-claim counterevidence expectation.

These evaluation units shift the workflow from "evaluate the prose" to "evaluate the argument structure." They are also aligned with what can be logged and audited. In a mature deployment, the system should produce an evaluation report that attaches to each output dossier: claim verification statuses, citation fidelity scores, drift indicators, and lens disclosure compliance.

Artifact (Save This)

Minimum evaluation dossier for an SSH narrative.

- Claim list with types and lens tags.
- Per-claim verification status: Supported / Contradicted / Mixed / Unknown.
- Per-claim citation fidelity grade.
- Calibration check: confidence language vs evidence state.
- Drift report across revisions (claim changes and strengthening without new evidence).

4.6.3 Robustness and stress testing

Robustness in interpretive AI is not primarily about adversarial prompts that cause obvious toxicity. The relevant robustness question is whether the system preserves evidentiary discipline under the conditions that actually arise in SSH work: conflicting sources, incomplete archives, ambiguous evidence, politicized topics, and organizational pressure to sound decisive. Stress testing should therefore be designed to simulate these conditions and to measure whether the pipeline's controls hold.

Conflicting-source tests. Construct (or select) corpora where sources disagree. In social science, disagreement may be empirical (different datasets, different estimates). In humanities, disagreement may be interpretive (different readings of the same passage, different historiographical accounts). The test asks:

- Does the system detect the conflict?
- Does it represent disagreement rather than smoothing it away?
- Does it downgrade certainty and present conditions under which each account holds?

A failure is not simply choosing the "wrong" side; a failure is presenting a contested claim as settled or failing to disclose the contestation.

Missing-source tests. Many real settings involve partial corpora. Stress tests should include scenarios where crucial sources are absent and the system is tempted to fill gaps with plausible

narrative. The evaluation should check whether the system (i) declares gaps as open questions, (ii) avoids fabricated detail, and (iii) proposes what evidence would be needed to resolve the gap. This is a key governance property: the system must be willing to say "unknown" rather than improvise.

Ambiguity tests and interpretive pluralism. Humanities and qualitative social science often confront ambiguous passages or evidence that supports multiple readings. Stress tests should include ambiguous texts and require the system to produce multiple plausible interpretations under different lenses. The evaluation should measure:

- whether alternative readings are genuinely distinct,
- whether each reading is anchored in passages,
- whether the system labels the lens and avoids pretending neutrality.

A failure is forced synthesis: collapsing ambiguity into a single confident interpretation.

Pressure-prompt tests. Organizational incentives often push toward confident, persuasive narratives. Stress testing must include prompts like:

> "Make this sound decisive," "remove caveats," "write this as an authoritative conclusion," "make it persuasive to stakeholders."

The system should resist or constrain these requests when evidence does not justify confidence. A robust pipeline will either (i) refuse the request, (ii) keep uncertainty labels, or (iii) require explicit sign-off and justification for upgrading certainty. This is a direct test of whether governance controls are real or merely advisory.

Citation-theater tests. Stress tests should include requests to "add citations" or "make it look scholarly." The system must not respond by generating decorative references or misaligned citations. Instead, it should (i) map each claim to actual passages, (ii) cite only what supports, and (iii) flag claims that lack support rather than citing tangential sources. This test reveals whether the system is optimizing for appearance.

Retrieval perturbation tests. Because retrieval instability is a core fragility, stress tests should perturb retrieval: change query wording, change ranking, remove a small subset of documents, or inject a small amount of noisy material. Then measure narrative stability:

- how much does the claim set change,
- how much does lens emphasis change,
- whether counterevidence inclusion remains consistent.

High sensitivity indicates that outputs should be treated as exploratory and should include explicit stability disclaimers.

Long-context overload tests. Provide a large corpus that exceeds realistic review capacity. The test is whether the system can still produce passage-level anchoring and whether it begins to

cherry-pick or smooth contradictions. Robustness here is partly interface-driven: if reviewers cannot inspect evidence, the system is effectively unverifiable, regardless of model capability.

Human-edit intrusion tests. In practical deployments, humans will edit model outputs. Stress testing should include post-edit scenarios: the reviewer edits text to remove uncertainty or add stronger claims. The system should detect that claims changed and require re-verification. If human edits can bypass controls, the system will fail under real institutional pressure.

A mature robustness program treats these tests as recurring audits, not one-time evaluations. The system's performance should be tracked over time, with metrics like citation fidelity rates and drift rates serving as operational health indicators.

Risk & Control Notes

Robustness principle. In SSH deployments, the critical stressor is *pressure to sound certain*. If a system cannot maintain evidentiary discipline when asked to be persuasive, it is not robust enough for external-facing use.

4.6.4 Failure taxonomies

Failure taxonomies are governance tools: they turn a vague sense that "the model got it wrong" into specific, repeatable categories that can be monitored, assigned controls, and used to decide when to restrict deployment. In interpretive AI, the taxonomy must capture both factual errors and interpretive distortions, especially those that arise from framing.

1. Unsupported assertions. The system states a factual claim with no supporting evidence link. In SSH settings, this includes invented details about historical events, fabricated quotations, and claims about empirical findings without a source. Unsupported assertions are the most direct violation and are the easiest to define, but they are not the only risk.

2. Misattribution and mis-citation. The system cites a source that does not support the claim, cites the wrong passage, or misrepresents the cited passage. Misattribution includes decontextualization: quoting a line while omitting qualifiers that reverse its meaning. In humanities work, mis-citation can be subtle because the passage may be thematically related; fidelity evaluation must distinguish relevance from support.

3. Overgeneralization. The system generalizes from narrow evidence to broad claims. In social science, this may be external validity inflation: treating a case study or a specific dataset result as universally applicable. In humanities, it may be universalizing a culturally specific interpretation. Overgeneralization is often coupled with rhetorical overconfidence.

4. Causal inflation. The system upgrades correlation to causation, suggests mechanisms without evidence, or recommends policy based on unjustified causal narratives. This is a dominant risk in policy-adjacent social science outputs because causal language is rhetorically attractive. Governance

controls include requiring identification assumptions and downgrading to hypotheses when they are absent.

5. Lens laundering. The system presents a lens-dependent interpretation as neutral description. For example, it presents a particular theoretical frame as common sense while excluding alternatives. Lens laundering undermines contestability because it hides the assumptions that structure the narrative.

6. Selective omission and framing manipulation. The system produces a coherent narrative by omitting counterevidence, alternative historiographies, or relevant conflicting data. This is often the most consequential failure because it can create biased institutional speech without any single sentence being false. Selective omission is particularly acute in contested topics where multiple communities have different stakes.

7. Rhetorical overconfidence and calibration failure. The system uses definitive language where evidence is weak or mixed, removes caveats under pressure, or expresses certainty that is not warranted. This failure is about posture rather than content alone, and it shapes how readers interpret the narrative.

8. Drift across revisions. The system's output becomes less faithful to evidence over iterative edits. Drift can occur as claims strengthen, qualifiers disappear, and citations become decorative. Drift is a process failure and requires process-level controls: re-verification after edits and drift monitoring.

9. Provenance loss. The system cannot reconstruct which sources were used, what retrieval results were returned, or how claims changed. Provenance loss is a governance failure even if the final output appears plausible, because it eliminates the institution's ability to defend the narrative.

10. Ethical and privacy boundary violations. The system inadvertently surfaces sensitive information, violates consent boundaries, or uses restricted materials in outputs. In SSH contexts involving human subjects, this failure can be severe even if the narrative is otherwise careful.

A practical taxonomy should map each failure category to controls and to escalation thresholds. For example, repeated mis-citation might trigger stricter citation validation requirements or restrict the system to internal use. Repeated omission failures in sensitive topics might require mandatory counterevidence sections and multi-lens outputs.

Artifact (Save This)

Failure taxonomy as an operational checklist. Each output dossier should include: (i) which failure categories were checked, (ii) which were detected, (iii) what mitigations were applied, and (iv) whether human sign-off accepted residual risk. This turns evaluation into governance evidence.

4.6.5 Limits of current benchmarks

Current benchmark culture in AI is poorly aligned with the needs of interpretive work in the social sciences and humanities. Most widely used benchmarks measure short-answer factuality, reasoning on synthetic tasks, or preference judgments about helpfulness. Even benchmarks that evaluate "truthfulness" or "hallucination" tend to focus on factual claims that have crisp ground truth. SSH outputs rarely fit that mold. The central risks—framing bias, omission, lens laundering, and drift—are not captured by common benchmarks, and this mismatch can produce false confidence.

Lack of standardized epistemic discipline suites. There is no widely adopted benchmark suite that evaluates a full evidence-first pipeline: claim extraction, evidence mapping, citation fidelity, uncertainty labeling, and revision drift. As a result, organizations often test only the drafting model, not the system. This is analogous to evaluating a financial model by inspecting only the output table while ignoring the data pipeline and control environment.

Weak measurement of framing bias and omission. Benchmarks rarely measure what is not said. Yet omission is a dominant failure mode in SSH contexts. Measuring omission requires designing tasks where counterevidence exists and where evaluation rewards its inclusion. This is difficult because it requires a curated ground truth of "relevant counterpoints" and a definition of what counts as fair representation. Nonetheless, without such benchmarks, systems can score well on factuality while producing systematically biased narratives.

Insufficient treatment of plural interpretation. Many benchmarks implicitly assume a single correct answer. SSH work often has multiple defensible readings. A benchmark that penalizes deviation from a reference narrative can discourage pluralism and encourage forced synthesis. We need benchmarks that evaluate *structured disagreement*: does the system present alternative lenses accurately and with evidence links?

Limited robustness benchmarks for retrieval sensitivity. RAG and long-context systems are sensitive to retrieval configuration. Few benchmarks test stability under retrieval perturbations. In SSH settings, where framing is selection-driven, this is a critical gap. Without stability benchmarks, systems may appear strong in curated test settings but behave unpredictably in production corpora.

Inadequate evaluation of iterative workflows. Most benchmarks are one-shot. But SSH deployments are iterative. Drift occurs across revisions. Without benchmarks that simulate revise-and-strengthen loops, we cannot measure whether systems become less faithful over time or whether controls prevent that decay.

Benchmark–institution mismatch. Finally, benchmarks often ignore the institutional boundary conditions that dominate risk: publication gates, separation of duties, provenance logging, and auditability. A model can perform well on a benchmark and still be unusable in a governed environment because it cannot produce defensible dossiers.

The implication is that organizations must treat external benchmarks as incomplete signals and must

build internal evaluation harnesses aligned with their own corpora, risk profiles, and publication standards. In SSH contexts, this often means synthetic benchmark construction: toy corpora with embedded conflicts, missing sources, and contested interpretations, paired with scoring rubrics that reward evidence discipline and plural representation.

Risk & Control Notes

Benchmark complacency risk. If an institution adopts interpretive AI based on generic "LLM quality" scores, it will optimize for the wrong objective. The relevant benchmark is not fluency or helpfulness; it is citation fidelity, drift resistance, and contestability under pressure.

4.7 Implementation Considerations

4.7.1 Minimal viable implementation patterns

Implementation in the social sciences and humanities should begin with an uncomfortable truth: the minimal viable product is not a generative model. The minimal viable product is an *evidence-disciplined workflow* that produces an output dossier a human can defend. A system that drafts beautifully but cannot generate inspectable claim–evidence mappings is not merely incomplete; it is mis-specified for this domain. Therefore, the implementation question is: what is the smallest architecture that (i) improves productivity, (ii) preserves contestability, and (iii) does not encourage publication of unverified narratives?

Pattern A: Evidence-first drafting (claim registry before prose). The simplest defensible pattern is to invert the usual drafting flow. Instead of prompting the model to "write an essay," the system proceeds in stages:

1. Define corpus scope S and admissibility rules (what sources are allowed).
2. Retrieve passages P relevant to a narrow question.
3. Extract or propose candidate claims C from passages, not from imagination.
4. Require evidence links $E^+(c)$ and counterevidence $E^-(c)$ for each claim.
5. Draft prose as a rendering of the verified claim registry.

This pattern reduces hallucination and discourages rhetorical inflation because claims must exist as explicit objects before prose is written. It also makes review cheaper: reviewers inspect claims and evidence, not entire paragraphs for hidden commitments.

In practice, evidence-first drafting can be implemented with very simple machinery. Claim extraction can be heuristic (sentences with named entities, quantified statements, or causal verbs). Evidence mapping can be implemented with retrieval over the same corpus constrained to passage-level matching. The power comes not from sophistication but from forcing explicitness: claims cannot appear without evidence links. For humanities interpretation, the system can include a required "lens" field for interpretive claims, making the interpretive posture visible.

Pattern B: Draft-first with mandatory claim extraction (retrofit structure). Some environments cannot invert the workflow immediately because users want a draft quickly. In those settings, a draft-first pattern can be made governable by forcing structure after drafting:

1. Produce an initial draft T with strong constraints (no definitive claims, required uncertainty language).
2. Extract claims C from T automatically.
3. For each claim, search for supporting and counterevidence passages.
4. Label each claim as Supported / Mixed / Unknown and downgrade language accordingly.
5. Require human review of the claim registry before any publication.

This pattern recognizes institutional reality while still preventing the most common failure: publishing draft prose that has never been decomposed into auditable claims.

Pattern C: Structured output schema as the primary interface. The single most effective implementation decision is to force structured output as the default. A minimal schema for SSH outputs should include:

- **Facts (descriptive claims):** what the sources directly state.
- **Interpretations:** lens-tagged readings anchored to passages.
- **Causal hypotheses (if present):** with explicit identification assumptions or "cannot infer causality" warnings.
- **Open questions:** gaps in the corpus or contested points.
- **Counterevidence and alternatives:** strongest counterclaims with citations.

The narrative text becomes an optional final rendering. The primary product is the structured argument pack.

Pattern D: "Red flag first" risk triage. In real deployments, verification capacity is limited. A minimal viable implementation should include triage rules that prioritize review on:

- causal or policy-relevant claims,
- claims about contested identities, histories, or politically salient topics,
- claims with weak evidence links,
- claims where retrieval shows contradiction.

The system should highlight these claims in the review UI and require explicit acceptance or revision.

Pattern E: Multi-lens output for contested topics. For humanities and theory-heavy social science work, a minimal viable pattern is to generate two or three parallel framings under declared lenses π_1, π_2, π_3. The system then compares claim sets and identifies where lenses diverge. This pattern prevents hidden lens laundering and makes contestation an explicit artifact. It also teaches users the correct epistemic posture: the system is a tool for exploring interpretations, not producing definitive truth.

Pattern F: "Internal-only mode" by default. A governance-first implementation should treat publication as a separate mode that requires additional controls. The default use case should be internal drafting, exploratory synthesis, and teaching support. External-facing use should require: full dossier generation, human sign-off, and stricter admissibility rules. This is a practical pattern that reduces risk without banning the tool.

Across all patterns, the minimal non-negotiable is the claim registry with evidence links. If the organization cannot operationalize that, it should not represent outputs as disciplined scholarship or institutional narrative.

Artifact (Save This)

Minimal viable SSH implementation (non-negotiables).

1. A claim registry (typed claims).
2. Passage-level evidence links for factual claims.
3. Lens tags for interpretive claims.
4. A counterevidence requirement for high-stakes topics.
5. A review interface that makes verification fast.

4.7.2 Reproducibility and determinism

Reproducibility is the institutional bridge between "a helpful draft" and "a defensible output." In SSH contexts, reproducibility is not only an academic virtue; it is a governance necessity because outputs often become institutional speech. If an organization cannot reconstruct what sources were used and how the narrative was produced, it cannot credibly stand behind the result when challenged. Therefore, implementation must treat reproducibility as a first-class feature, not as a later enhancement.

Versioned source sets. Every run of the pipeline must record a corpus identifier:

$$\text{corpus\_id} = \text{hash}(\{s_j\}, \{\mu(s_j)\}),$$

or a comparable versioning scheme. In practice, this means: a frozen snapshot of the document set, including metadata and any pre-processing steps (OCR versions, text extraction parameters). For SSH corpora, where documents may be updated, retracted, or corrected, corpus versioning prevents silent changes to the evidence base.

Retrieval index versioning. Retrieval is a major source of instability. The index (embeddings, tokenization settings, chunking strategy) must be versioned and recorded alongside the corpus. A run should capture:

- index build timestamp,
- chunking parameters,
- embedding model version or method,
- ranking algorithm version and configuration.

Without index versioning, two runs over the same corpus can produce different evidence sets and therefore different narratives without any visible reason.

Deterministic sampling for audit runs. Stochastic generation can be useful for exploration, but auditability requires determinism. A practical pattern is to maintain two modes:

- **Exploration mode:** allows non-zero temperature and alternative drafts.

- **Audit mode:** uses frozen prompts, fixed seeds (where applicable), and deterministic decoding as much as feasible.

Audit mode is used for publication dossiers and for incident reconstruction. The requirement is not perfect determinism at the token level; the requirement is that the system can reproduce the *claim registry* and evidence mapping sufficiently to explain the output.

Frozen prompts and templates. Prompts are part of the system configuration. If prompts change, outputs change. Therefore, prompts and templates should be versioned artifacts. A run should record:

- system prompt version,
- task prompt content (with redaction where sensitive),
- policy checker rules version,
- schema version for structured outputs.

This is particularly important in SSH contexts because prompts can encode lenses. If an institution shifts from a neutral framing to an advocacy framing through prompt edits, that change must be visible and reviewable.

Normalization of pre-processing. SSH documents arrive in messy forms: PDFs, scans, OCR outputs. Pre-processing choices can materially affect what is retrievable and therefore what is used as evidence. Implementation should standardize and record: OCR engine version, language models used, cleaning heuristics, and any normalization steps (e.g., removing footnotes, handling page headers). Without this, reproducing evidence links becomes impossible.

Reproducibility targets. At minimum, the institution should be able to reproduce:

- which passages were retrieved for each claim,
- the claim list and their types,
- the evidence and counterevidence links,
- the revision diffs across key steps,
- the final rendered narrative.

If only the final narrative can be reproduced, reproducibility has failed in the sense relevant to governance. The dossier must reproduce the *argument structure*.

Risk & Control Notes

Reproducibility failure mode. "We can't recreate the run because the index changed and the prompt template was updated." In SSH settings, this is equivalent to losing your footnotes, your archive citations, and your lab notebook at the same time.

4.7.3 Logging and traceability

Logging is the infrastructure of accountability. In interpretive AI, the output dossier is the analog of a replication package: it is what allows a reader, reviewer, or institutional stakeholder to contest claims and reconstruct how they were produced. Logging must therefore be designed not as a debugging convenience but as a governance artifact generator.

What to log: minimum viable trace. A minimal logging system should capture the following layers:

(1) Source and retrieval logs.

- corpus ID and document IDs used,
- retrieval queries (or query embeddings identifiers),
- ranked retrieval outputs (document and passage IDs),
- filters applied (source-tier restrictions, sensitivity filters),
- timestamps and user role.

This layer answers: what evidence was visible to the system?

(2) Claim registry logs.

- claim IDs and text,
- claim types and lens tags,
- confidence labels / required uncertainty posture,
- support and counterevidence links,
- status transitions (proposed $\rightarrow$ verified $\rightarrow$ approved).

This layer answers: what was asserted, and how was it categorized?

(3) Draft and revision logs.

- draft versions (or diffs) with timestamps,
- mapping from narrative spans to claim IDs,
- changes in wording that increase certainty or broaden scope,
- human edits (captured as explicit actions).

This layer answers: how did rhetoric evolve, and did it change the claims?

(4) Checker and evaluation logs.

- policy checker outputs (violations detected),
- citation fidelity checks and results,
- drift metrics across revisions,
- sensitivity tests (retrieval perturbations and stability measures).

This layer answers: what controls were applied, and what did they find?

(5) Human review and sign-off logs.

- reviewer identity/role (not necessarily public; at least internal),
- decisions per claim (accept/reject/modify),
- escalation notes for contested claims,
- final approver sign-off and publication gate record.

This layer answers: who took responsibility?

Publication dossier per output. Implementation should culminate in a standardized dossier package for each external-facing output. The dossier should be automatically generated and stored with immutable identifiers. A practical dossier includes:

- the final narrative and structured output schema,
- a claim registry export,
- passage-level evidence bundles (with context windows),
- retrieval logs and corpus/index version IDs,
- revision diffs and drift report,
- evaluation summaries (citation fidelity, calibration, stability),
- sign-off record.

In SSH contexts, this dossier is not only for internal audit; it can support public transparency when appropriate (e.g., a public-facing "sources and method" appendix).

Traceability across time. Interpretive outputs often get reused: a memo becomes a slide deck, becomes a press release, becomes internal training. Traceability must therefore support lineage: if a narrative is repurposed, the system should record that derivative outputs are linked to the original dossier and should require re-verification if claims are altered. Otherwise, drift becomes organizational: a lightly edited version loses caveats and the institution no longer knows where the claims came from.

Data minimization and confidentiality. SSH corpora can include sensitive material. Logging must therefore implement redaction and minimum-necessary principles. The goal is to preserve auditability without leaking sensitive content. Practically, this means:

- store passage IDs and hashed references rather than full text where possible,
- encrypt sensitive logs,
- control access by role,
- redact prompts or user inputs that contain personal data.

This is not optional; it is a necessary condition for ethical SSH deployment.

Artifact (Save This)

Traceability test. For any published sentence, a reviewer should be able to answer in under one minute: (i) which claim ID it expresses, (ii) what passages support it, (iii) what counterevidence exists, (iv) what lens is declared, and (v) who approved it.

4.7.4 Cost, latency, and scaling issues

In interpretive AI, the dominant cost is not generation. The dominant cost is verification, and the dominant latency is human review. This flips the conventional engineering instinct to optimize model speed. A system that generates drafts in seconds but requires hours of manual checking is not necessarily inefficient; it may be appropriately cautious. The real optimization target is *reducing the cost of review without degrading epistemic discipline*. That implies that implementation should focus on interfaces, triage, and workflow design more than on model throughput.

Verification dominates cost. Claim-level verification requires reading passages, checking context, and judging whether the claim follows under a lens. In social science, causal claims may require method review. In humanities, interpretive claims may require close reading. These tasks are expensive and cannot be fully automated. Therefore, scaling requires:

- reducing the number of claims that require deep review,
- prioritizing high-risk claims,
- making evidence presentation fast and navigable,
- preventing the system from generating unnecessary claims.

A practical tactic is to limit narrative scope: narrower questions, fewer claims, more explicit open questions. A system that admits what it cannot support reduces review cost.

Design for fast human review. Review UI is a primary scaling lever. The interface should allow reviewers to:

- sort claims by risk,
- view supporting passages with context in one click,
- see contradictions highlighted,
- approve/reject at claim granularity,
- generate a revised narrative automatically from approved claims.

In other words, the reviewer should not be forced to read a 2,000-word essay to find three problematic sentences. The system should surface those sentences as claims.

Bounded automation. Automation can be used to reduce human load without replacing judgment. Examples include:

- automated detection of sentences that sound like factual claims but lack citations,
- automated prompts to retrieve counterevidence for high-stakes claims,
- automated drift detection when revisions remove hedges or broaden scope,
- automated stability tests under retrieval perturbations.

The boundary is clear: automation can triage, flag, and suggest; humans decide and sign off.

Latency trade-offs. Organizations often want real-time writing assistance. Governance-first SSH systems must resist the assumption that publication-grade outputs can be produced instantly. A realistic pattern is a two-speed system:

- **Fast mode:** internal drafting and exploration with warnings, structured outputs, and no publication gate.
- **Slow mode:** publication pipeline with claim verification, dossier generation, and sign-off.

This prevents the organization from treating speed as a policy choice.

Scaling by limiting scope and standardizing artifacts. The fastest way to scale is not to generate more; it is to standardize. Use fixed schemas, fixed lens declaration formats, standard counterevidence sections, and standard dossier templates. Standardization makes review more consistent and reduces the cognitive overhead of each new output.

Cost accounting as governance. An implementation should make verification cost visible. If the system consistently produces outputs that require heavy review, that indicates misalignment: too many claims, too much certainty, insufficient evidence mapping. Governance should treat high review cost as a signal to tighten constraints π, not as a reason to skip review.

Risk & Control Notes

Scaling illusion. Many teams believe interpretive AI scales because drafting is cheap. In reality, publishing defensible interpretation scales only if review is engineered to be efficient. Otherwise, the organization scales text volume while shrinking epistemic capacity.

4.7.5 Integration risks

The most consequential risks in SSH implementation are not technical failures of models; they are integration failures of organizations. These failures arise when the tool is introduced into workflows without redesigning incentives, responsibilities, and publication gates. The result is "speed as policy": the organization begins to treat rapid output as success, and epistemic discipline becomes optional. This is the failure mode that destroys trust while appearing productive.

Risk 1: Temptation to publish drafts. The primary integration risk is that drafts become de facto final outputs. A team uses AI to draft a memo; the memo is circulated; stakeholders rely on it; no one verifies citations; the output becomes institutional knowledge. This risk is amplified

by the quality of prose. When the draft looks professional, it is socially difficult to insist on slow verification. The control is a hard publication gate: external-facing outputs must come from the verified claim registry with dossier attached.

Risk 2: Erosion of epistemic standards by convenience. Over time, the existence of an "always-available" drafting tool can shift norms. Students stop doing close reading; analysts stop checking methods; comms teams stop verifying claims; scholars stop tracking provenance. This erosion is subtle: each use is small, but the cumulative effect is a reduced capacity for contestation. The governance response is training plus enforcement: the tool must be framed as an assistant that produces hypotheses and drafts, not as a source of truth.

Risk 3: Role confusion and accountability diffusion. When AI drafts, who is responsible? If the institution cannot assign ownership for claim verification and sign-off, accountability will diffuse. In contested topics, this becomes reputationally catastrophic because challenges will be met with "the model wrote it" rather than a defensible account. Separation of duties is therefore an integration requirement, not a luxury.

Risk 4: Hidden lens adoption through templates. Organizations often standardize prompts and templates. In SSH contexts, templates can encode ideological or theoretical assumptions. If prompts instruct the model to write in a particular framing without disclosure, the institution may inadvertently adopt a hidden lens as a default. The control is explicit lens declaration and periodic audit of prompt templates for framing bias.

Risk 5: Tool sprawl and provenance fragmentation. Many teams use multiple AI tools: one for retrieval, one for drafting, one for citation formatting. Without integration, provenance becomes fragmented. Claims are copied between systems; citations are lost; logs do not connect. The result is a narrative that cannot be reconstructed. The control is to centralize dossier generation or require that any tool used must export into the same claim registry and logging system.

Risk 6: Organizational incentives that punish caution. In many institutions, speed is rewarded and caution is treated as delay. Interpretive AI will magnify this incentive. If staff are evaluated on output volume, they will skip verification. The only sustainable control is governance at the management level: explicit policies that verification is required, time is budgeted for it, and sign-off is part of performance expectations.

Risk 7: Public trust and disclosure norms. SSH outputs often interact with public trust. If an institution uses AI assistance without clear disclosure norms, it may face backlash when AI use is discovered. Conversely, disclosure without discipline can also harm trust if the public believes AI outputs are unvetted. The correct integration move is to pair disclosure with dossier-backed process: when appropriate, disclose that AI assisted drafting and that human verification and evidence logs exist.

Risk 8: "Speed as policy" becomes irreversible. Once an institution reorganizes around rapid generation, it may be difficult to return to slower epistemic standards. Knowledge bases fill

with unverified narratives; training materials embed errors; institutional memory becomes polluted. Correcting this is costly. Therefore, the safest implementation stance is conservative: begin with internal-only modes, build review workflows, demonstrate dossier utility, and only then expand use cases.

Artifact (Save This)

Integration gate (practical). If the organization cannot answer "Who verified this claim?" and "Where is the evidence dossier?" then the output must be labeled internal draft and must not be treated as institutional speech. This is the simplest defense against "speed as policy."

4.8 Impact and Opportunity

4.8.1 New capabilities unlocked

The most important opportunity in interpretive AI for the social sciences and humanities is not that machines can "write." Writing has always been the visible tip of the iceberg; the real bottleneck has been synthesis under constraints: collecting disparate sources, mapping what can be claimed, representing disagreement honestly, and producing a narrative that can withstand challenge. Frontier systems unlock a new capability regime where *drafting and synthesis become cheap enough to be iterative*—and where the institution can allocate human expertise to verification and judgment rather than to first-pass composition. If implemented with evidentiary discipline, this shift can improve both productivity and quality. If implemented naively, it simply floods the organization with persuasive, unverifiable text. The opportunity therefore depends on the controls.

Rapid synthesis of heterogeneous corpora. SSH work often requires reading across genres: scholarly articles, historical documents, interview transcripts, datasets, policy reports, and media narratives. Humans can do this, but the marginal cost is high, and the result is often fragile: a synthesis depends on what the researcher happened to read first, what was easiest to access, and what time allowed. Interpretive AI changes the economics of search and first-pass synthesis. With retrieval plus long-context summarization, a system can generate candidate syntheses across many documents quickly. The key is that this synthesis must be treated as a *proposal*, not as a conclusion. The unlocked capability is that teams can iterate across alternative storylines and verify them, rather than committing early to a single frame and then backfilling justification.

Evidence-conditioned drafting for teaching. In education—especially in humanities and social science instruction—drafting support can be transformative when constrained properly. Teachers can create multiple versions of lesson plans: one emphasizing historiography, another emphasizing theory, another emphasizing empirical evidence. Students can use the system to generate structured outlines that explicitly separate descriptive content, interpretive moves, and open questions. The critical opportunity is pedagogical: interpretive AI can be used to teach evidentiary discipline by forcing claim–evidence mapping. A well-designed system can make visible what is often tacit: how an argument is assembled and where it relies on assumptions. This is not "AI doing the thinking"; it is AI making the steps of reasoning inspectable.

Policy and institutional communication under constraint. Many organizations rely on SSH-like narrative outputs: policy briefs, research memos, diversity and inclusion reports, historical statements, and public communications about sensitive topics. These outputs are often written under time pressure, and the cost of error is reputational. Interpretive AI can help organizations produce drafts that are better structured, more explicit about uncertainty, and more systematic about citing sources. The capability unlocked is not press-release automation; it is the ability to quickly assemble candidate narratives and evaluate them for evidentiary support and omission risks

before publication.

Multi-source synthesis with explicit contestation. A distinct advantage of AI systems is that they can generate multiple alternative framings with low incremental effort. In contested fields, the system can produce parallel narratives under different lenses and then compare claim sets. This is a genuine capability unlock for SSH practice: structured pluralism becomes operational. Instead of hiding disagreement behind a single voice, the institution can represent multiple accounts and explicitly document where they diverge and why. This can improve trust internally and externally because it signals epistemic humility and transparency.

Operationalization of "open questions." Human researchers often know what they do not know, but in organizational settings open questions are sometimes treated as weakness and therefore suppressed. Interpretive AI can normalize open questions as a first-class output: a structured list of unresolved issues, missing sources, and contested interpretations, each linked to what evidence would be required. The capability unlocked is that uncertainty becomes trackable and actionable rather than embarrassing. This is especially valuable in policy contexts where decisions must be made under incomplete information.

Translation across modalities and audiences. Many SSH outputs must be adapted for different audiences: academic, public, executive, or educational. Interpretive AI can generate multiple audience-adapted renderings of the same claim registry while preserving evidence links. The productivity gain is significant: instead of rewriting from scratch, the institution can treat the claim registry as the stable core and produce multiple narratives with consistent evidence. This is a governance advantage: consistency reduces accidental drift when materials are repurposed across contexts.

Taken together, these capabilities suggest that interpretive AI can be a genuine productivity tool in SSH domains—but only when the system is designed so that the marginal cost of producing an *evidence-backed dossier* is low. Otherwise, the tool simply amplifies rhetorical output without amplifying epistemic capacity.

Artifact (Save This)

Opportunity framing. The key capability unlocked is not "AI can write." It is "AI can generate multiple candidate syntheses quickly, allowing humans to spend scarce time on verification, lens disclosure, and judgment." The system's value rises with its ability to make verification cheaper than drafting.

4.8.2 Organizational implications

Once interpretive AI becomes embedded in knowledge work, the organization must confront a structural change: editorial review is no longer a stylistic layer. It becomes a governance function.

This is a profound shift for institutions that historically treated writing as individual craft and review as optional polish. In SSH-adjacent contexts—policy teams, comms groups, research units, education departments—the new constraint is not content production but *epistemic accountability*. The organization must decide what it will stand behind, and how it will prove it.

Editorial review becomes a control environment. Traditionally, editorial review might focus on clarity, tone, and consistency. Under interpretive AI, editorial review must focus on claim validity, evidence support, lens disclosure, and omission risk. This requires new competencies and new process design. Organizations that keep editorial review as a cosmetic step will publish outputs that are rhetorically strong and epistemically weak. The implication is that editorial functions need formal authority: the power to block publication, require revisions, and demand evidence dossiers.

New roles emerge: evidence curator, verification lead, provenance owner. A mature interpretive AI deployment will produce roles that did not previously exist as distinct functions.

- **Evidence curator:** maintains the admissible corpus, assigns source tiers, manages metadata, and ensures licensing and ethics compliance. In SSH contexts, the curator is also responsible for documenting corpus boundaries and what is excluded.
- **Verification lead:** owns the claim verification process, defines sampling rules, manages triage, and ensures that high-risk claims receive sufficient scrutiny. The verification lead is the guardian of "Supported / Mixed / Unknown" labeling discipline.
- **Provenance owner:** ensures that logs, dossiers, and versioning are maintained and that any published output has a reconstructible path from sources to claims to prose. This role is analogous to model risk governance roles in finance, but applied to narratives.

These roles do not require large teams in small organizations, but they require explicit ownership. Without ownership, accountability diffuses into "the model did it" or "someone reviewed it."

Separation of duties becomes normal. In high-stakes writing—especially policy statements and institutional communications—separation of duties becomes a non-negotiable control. The drafter (human or AI) cannot be the only verifier. The verifier cannot be the only approver. This creates friction, but it is the same friction that prevents errors in safety-critical domains. Organizations that adopt interpretive AI without separation of duties will experience a predictable pattern: initial productivity surge followed by reputational incidents and reactive restrictions.

Training shifts from prompting to epistemic practice. Most organizations initially think adoption is about teaching people how to prompt. In SSH contexts, the real training requirement is epistemic: how to separate descriptive from interpretive claims, how to evaluate evidence, how to disclose lenses, how to represent disagreement, and how to resist rhetorical pressure. Prompting is a minor skill compared to these. Therefore, the organization must invest in training that looks more like research methods and editorial standards than like "AI tips and tricks."

A new category of internal artifact: the publication dossier. Organizations will need

to normalize dossiers as part of routine work. Just as financial institutions normalize model documentation, SSH-adjacent institutions will need to normalize narrative documentation. This is an operational shift: storage, access control, retention policy, and audit processes must be built around dossiers. The implication is that interpretive AI is not only a writing tool; it is an information governance system.

Governance committees will care. In many institutions, governance bodies focus on data privacy and security. Interpretive AI introduces reputational and epistemic governance concerns: what claims can the institution make, what sources support them, and what is the approval process? Therefore, oversight may need to involve comms leadership, policy leadership, legal counsel, and ethics review boards, not just IT.

Metrics and incentives must change. If teams are rewarded for output volume, interpretive AI will degrade standards. If teams are rewarded for dossier completeness and citation fidelity, interpretive AI can improve standards. Therefore, organizational incentives become a technical control: they determine whether governance constraints are actually followed.

Risk & Control Notes

Organizational failure mode. The institution treats interpretive AI as "a faster writing tool" and keeps review processes unchanged. Drafts become publications, citations become decoration, and accountability diffuses. The result is predictable: reputational incidents, followed by overcorrection and blanket bans.

4.8.3 Potential new industries or markets

Where governance becomes a product requirement, markets form. Interpretive AI in social sciences and humanities will create demand not only for models but for *evidence infrastructure*. The most promising markets are those that reduce verification cost and increase traceability, because those are the binding constraints.

Evidence and provenance platforms for SSH corpora. Unlike enterprise knowledge bases, SSH corpora often include heterogeneous sources with complex metadata and licensing constraints. Platforms that can ingest, tier, and version corpora—and provide passage-level identifiers suitable for citation—will be valuable. In particular, organizations will need tools that treat corpus construction as a governed process: who added a source, why, with what credibility tier, and under what licensing terms.

Audit-grade generation pipelines. There is a market for systems that do not merely generate text but produce audit artifacts: claim registries, evidence maps, drift reports, and sign-off logs. In many organizations, the barrier to adoption is not model performance but the inability to defend outputs. Vendors who can package dossier generation as a standard feature will have an advantage, especially in policy, education, and public-sector contexts.

Citation fidelity tooling. Citation is the weak point of interpretive AI adoption. Tools that verify whether a cited passage supports a claim, detect decontextualization, and flag missing counterevidence will be central. This market will likely include:

- passage-level entailment or support checkers (with careful limitations),
- reference validation systems (does the cited work exist?),
- decontextualization detection,
- citation coverage auditors (which claims lack citations?).

Even imperfect tooling can provide high value by triaging what humans should review first.

Framing and omission diagnostics. Measuring framing bias is hard, but organizations will demand diagnostics that at least surface risk: which perspectives are underrepresented, whether counterevidence exists but is omitted, and how sensitive narratives are to retrieval changes. Tools that support multi-lens comparison and produce "difference reports" between narratives could become standard in policy shops and research units.

Review workflow and separation-of-duties systems. In regulated environments, workflow tooling (permissions, approvals, logs) is often as important as the model. For SSH interpretive AI, similar workflow systems will emerge: role-based review interfaces for claim verification, audit trails, and publication gating. These systems look less like "AI assistants" and more like governance platforms.

Educational markets: evidence-first teaching assistants. In education, there is potential for tools that help students learn evidentiary reasoning: they generate structured outlines, enforce citation discipline, and require explicit separation of facts vs interpretations. Institutions will prefer tools that discourage plagiarism and hallucination by design. The market is not "AI that writes essays"; it is "AI that teaches how to build arguments responsibly."

Public transparency tooling. Some institutions will adopt public-facing transparency norms: releasing method appendices, evidence maps, and disclosure statements. Tools that can generate public-facing provenance artifacts (without leaking sensitive data) will become valuable, especially for NGOs, public agencies, and research organizations.

The overarching point is that interpretive AI will generate markets where the core product is *epistemic accountability*, not novelty in generation. In SSH domains, the economic value will accrue to systems that make contestable narratives cheaper to produce than unaccountable ones.

Artifact (Save This)

Market thesis. The winning products in interpretive AI will not be the ones that write the most convincing narratives. They will be the ones that make verification, provenance, and lens disclosure operationally cheap and standard.

4.8.4 Second-order effects

Second-order effects are where governance-first thinking becomes essential. Interpretive AI does not simply improve productivity; it changes the information environment inside and outside institutions. The most important second-order effect is volume: when narrative generation becomes cheap, the world fills with narratives. Attention does not scale, verification capacity does not scale, and trust becomes more fragile.

Content volume overwhelms attention and review capacity. As organizations generate more reports, memos, briefs, and educational materials, readers face overload. Overload increases reliance on heuristics: trust the confident-sounding memo, trust the one with citations, trust the one that matches prior beliefs. These heuristics are exactly what models can exploit inadvertently by producing fluent text. Therefore, volume creates a systemic risk: even if individual outputs are mostly fine, the aggregate environment becomes less verifiable.

Reputational exposure as a liability surface. Institutional narratives are liability surfaces. A single mis-cited claim in a public statement can trigger controversy; a subtle framing omission can be interpreted as bias; a misrepresented historical account can alienate communities. Interpretive AI increases the rate at which institutions produce statements, and therefore increases exposure. The risk is not only factual error; it is loss of legitimacy. In SSH contexts, legitimacy is often the core asset: universities, NGOs, public agencies, and cultural institutions rely on trust.

Epistemic erosion inside organizations. Internally, the ability to generate plausible narratives can erode the habit of checking sources. Teams may treat drafts as knowledge. Knowledge bases may be populated with AI-generated summaries that drift. Over time, the organization's internal epistemic standards degrade: fewer people read primary sources, fewer people maintain citation discipline, and fewer people are able to contest narratives. This is a capability loss disguised as productivity.

Strategic manipulation becomes easier. The chapter explicitly excludes guidance on propaganda and manipulation, but governance must acknowledge that interpretive AI lowers the barrier for narrative shaping. Stakeholders inside an organization can request persuasive framings; external actors can flood the information space with plausible narratives. Even without malicious intent, the ability to generate "argument-shaped" text at scale changes the strategic environment for policy and public discourse.

Homogenization of discourse. Models tend to generate text that is stylistically standardized and often aligned with mainstream rhetorical patterns. Over time, this can homogenize academic and institutional discourse. In humanities settings, where voice, nuance, and situated interpretation matter, homogenization can be a cultural loss. It can also create bias by privileging certain rhetorical forms as "professional" and marginalizing others.

Institutional memory contamination. If AI-generated outputs are stored and reused without

provenance, they become part of institutional memory. This can lead to compounding errors: a flawed summary becomes the source for a new summary, and so on. The result is a form of epistemic drift at the organizational level. This is why provenance and dossiers matter not only for publication but for internal knowledge management.

Legal and compliance spillovers. In many contexts, interpretive narratives intersect with compliance, HR, DEI, and policy obligations. AI-generated interpretations can inadvertently create legal exposure if they assert claims about individuals or communities, misrepresent policy history, or make recommendations that conflict with regulations. These risks are amplified by volume and speed.

Second-order effects are not an argument against adoption. They are an argument for designing adoption so that verification capacity is preserved and strengthened. The institution must ensure that the tool does not replace epistemic discipline with rhetorical throughput.

Risk & Control Notes

Systemic risk. Interpretive AI can create a world where narratives proliferate faster than verification. In that world, trust becomes a scarce resource and institutions that cannot defend their narratives will lose legitimacy.

4.8.5 Overstated expectations

The final opportunity discussion must be anchored by what interpretive AI cannot do, and what institutions should not expect it to do. Overstated expectations are not harmless optimism; they are failure generators, because they encourage organizations to delegate judgment to systems that cannot bear it.

Expectation 1: "AI can replace editorial judgment." It cannot. Editorial judgment is the act of deciding what is defensible, what is fair, what is sufficiently supported, and what uncertainty must be disclosed. Models can propose, draft, and triage, but they cannot own responsibility. In SSH contexts, where interpretive pluralism and ethical stakes are real, judgment remains human and institutional.

Expectation 2: "AI produces neutral outputs by default." Neutrality is not automatic because corpora are not neutral, retrieval ranking is not neutral, and framing choices are not neutral. The best a system can do is declare its lens, document its corpus, and represent alternatives. Institutions that demand neutrality without lens disclosure will get hidden bias, not neutrality.

Expectation 3: "If it cites sources, it is grounded." Citations are not grounding unless citation fidelity is verified. The system may cite relevant works without support, or it may misrepresent passages. Therefore, citation output must be treated as a verification task, not as a trust signal. This is an implementation and governance expectation that must be set explicitly.

Expectation 4: "Long context eliminates hallucination." Long context reduces some fabrication, but it introduces other risks: cherry-picking, decontextualization, and smoothing contradictions. In SSH contexts, meaning is contextual. The system can still misread, misframe, or overstate. Therefore, long context is not a substitute for discipline.

Expectation 5: "Interpretive disagreement can be resolved by better models." Disagreement in SSH fields often reflects genuine pluralism: different values, different theories, different historical emphases. Better models can represent disagreement more clearly, but they cannot erase it. Institutions should expect tools that help manage disagreement, not tools that eliminate it.

Expectation 6: "More output equals more knowledge." This is the most dangerous expectation. Output volume is not knowledge; it is text. Knowledge requires verification, contestation, and integration into human understanding. Interpretive AI can accelerate the production of candidate narratives, but without evidence discipline it accelerates confusion.

A governance-first adoption posture is therefore conservative in a specific way: it is optimistic about productivity gains in drafting and synthesis, but it refuses to treat those gains as epistemic gains unless verification, provenance, and lens disclosure are built in. The opportunity is real, but it is conditional.

Artifact (Save This)

Expectation discipline statement. Interpretive AI can accelerate drafting and broaden exploration of alternative framings, but it cannot supply truth, neutrality, or responsibility. Institutions must design workflows where AI outputs are proposals that become publishable only through evidence mapping, verification, and accountable human sign-off.

4.9 Governance, Risk, and Control

4.9.1 New risk categories

Interpretive AI in the social sciences and humanities changes the institutional risk map because the product is not merely information—it is *legitimacy*. SSH outputs shape beliefs, reputations, and policy narratives. They circulate through education, media, internal governance, and public communication. When generative systems enter these pipelines, the failure modes are not limited to factual errors; they include framing distortions, attribution failures, and gradual erosion of epistemic standards. Governance, therefore, must treat interpretive AI as a high-stakes system even when the domain is labeled "soft." Soft domains often become hard liabilities precisely because their outputs are socially consequential and difficult to unwind.

Reputational risk as a primary risk class. Many organizations treat reputational risk as downstream of legal risk. In interpretive AI, reputational exposure becomes the primary risk because outputs are public-facing or culturally salient even when they are not legally binding. A mis-cited historical claim, an oversimplified characterization of a community, or a narrative that appears ideologically slanted can trigger reputational damage quickly. This is exacerbated by the credibility cues that models produce—professional tone, structured argumentation, and citations. The organization's vulnerability is not just "being wrong"; it is appearing careless with evidence. In SSH contexts, carelessness is interpreted as disrespect, bias, or propaganda.

Reputational risk also has a compounding dynamic. Once an institution is perceived as using AI irresponsibly, future communications are viewed through suspicion. Trust, unlike output volume, does not scale. It is slow to build and fast to lose. Therefore, governance must treat reputational risk as a first-class metric: not simply whether outputs are accurate, but whether outputs are *defendable under scrutiny*.

Framing risk: narratives shape meaning even when facts are correct. Framing risk is the risk that the system produces a narrative that is technically fact-consistent but materially misleading because of selection, emphasis, and omission. In SSH work, this is often more consequential than outright fabrication. A narrative can cite real sources and still distort meaning by:

- selecting only one historiographical tradition,
- omitting counterevidence and minority viewpoints,
- representing contested interpretations as consensus,
- using loaded descriptors that imply judgment without evidence.

Framing risk is difficult because it is not localized to a single sentence. It is a property of the whole narrative architecture: what is foregrounded, what is backgrounded, and what is absent. This is why governance cannot rely solely on "every factual claim has a citation." That control is necessary, but it is not sufficient to prevent framing manipulation.

Epistemic erosion: slow degradation of standards inside the organization. One of the most dangerous risks is not a single bad output but a gradual shift in epistemic practice. When interpretive AI makes writing easy, institutions may substitute generation for reading. Teams may accept drafts as knowledge. Students may treat AI synthesis as understanding. Policy units may treat AI-written memos as adequate analysis. Over time, the organization's capacity for critical evaluation declines: fewer people consult primary sources; fewer people can detect mis-citation; fewer people recognize when lenses are being smuggled into descriptions. This is epistemic erosion: a loss of institutional competence disguised as productivity.

Epistemic erosion is a governance risk because it increases vulnerability to future failures. An organization with degraded epistemic standards becomes dependent on AI systems while being less able to supervise them. This is an unstable equilibrium: dependence rises as oversight capacity falls.

IP and attribution risk: authorship, reuse boundaries, and legitimacy. SSH domains are deeply concerned with attribution: who said what, whose argument is being used, and how intellectual credit is allocated. Interpretive AI introduces ambiguity in authorship and reuse. Risks include:

- unintentional paraphrase of protected text without attribution,
- reproduction of distinctive scholarly arguments without citation,
- unclear boundaries between AI-assisted drafting and human authorship,
- licensing violations when restricted sources are incorporated.

Even when the institution is not legally exposed, it can be ethically exposed: scholars and communities may view AI-assisted writing as appropriation or misrepresentation. Governance must therefore define attribution policies and disclosure norms, and it must implement technical controls that preserve provenance.

Accountability diffusion: "the model wrote it" becomes an organizational escape hatch. When outputs are generated, the temptation is to treat the system as the author. That is fatal for governance. Institutions must be able to answer: who approved this claim, who verified the evidence, and who is responsible for publication? If the answer is "the model," accountability has collapsed. Accountability diffusion also interacts with separation of duties: if no one is assigned to verification, then verification becomes no one's job.

Decision laundering and policy laundering. Interpretive outputs can influence decisions even when framed as "just a draft." A policy team may use an AI-generated narrative to justify a decision, presenting it as "what the evidence says" when it is actually a product of framing choices and incomplete sources. This is a form of decision laundering: using AI outputs to shift responsibility for contentious judgments. Governance must explicitly prohibit this. Interpretive AI should be used to surface options and uncertainties, not to create the appearance of evidence-based inevitability.

Misuse pathways: narrative scale as a manipulation vector. While this chapter explicitly

excludes guidance on disinformation, governance must recognize that interpretive AI can be misused to produce persuasive narratives at scale. In institutions, misuse can be internal (stakeholders pushing a narrative) or external (actors flooding the discourse). The risk is not only malicious; it is also incentive-driven. Systems that reward confident prose will be used to produce confident prose, even when the evidence is weak.

These risk categories are distinct but intertwined. Reputational risk is amplified by framing risk. Epistemic erosion increases mis-citation. Accountability diffusion makes incidents harder to correct. The governance strategy must therefore be layered: controls at the claim level, controls at the workflow level, and controls at the organizational incentive level.

Risk & Control Notes

Key governance insight. Interpretive AI primarily threatens legitimacy, not just accuracy. The institution's risk is that it produces narratives it cannot defend, and in doing so erodes the very epistemic standards that legitimacy depends on.

4.9.2 Control points and safeguards

Controls for interpretive AI must be designed around the true failure modes. In SSH settings, the core hazard is not merely hallucination; it is *unaccountable synthesis*. The controls must therefore make synthesis accountable by forcing explicit claims, explicit evidence, explicit lenses, and explicit human responsibility. A useful way to structure safeguards is by control points along the pipeline: intake controls, drafting controls, verification controls, publication controls, and post-publication controls.

Control point 1: Source admissibility and corpus governance. The earliest and most powerful control is the definition of what sources may be used. This includes:

- admissibility rules (peer-reviewed only, official statistics only, or tiered inclusion),
- licensing and permissions checks,
- sensitive-data restrictions (human subjects, confidential archives),
- metadata requirements (date, provenance, credibility tier).

This control prevents "garbage in" from becoming "credentialed out." It also enables defensible disclosure: the institution can state what corpus was used.

Control point 2: Mandatory claim registry. Every output must be decomposed into a claim registry. This is not optional. The registry forces explicitness and allows review at the correct unit. Controls include:

- claim typing (fact / interpretation / causal hypothesis / recommendation),
- scope labeling (what population, what period, what context),

- lens tagging for interpretive claims,
- status fields (proposed, verified, approved).

Claim typing is a safeguard against lens laundering and causal inflation. If a claim is interpretive, it cannot be written as descriptive fact without violating the schema.

Control point 3: Evidence mapping as a hard requirement. For factual claims, evidence links are mandatory. For interpretive claims, evidence links plus lens tags are mandatory. The system should enforce passage-level citation and should prohibit "floating citations" that are not attached to a specific claim. Contradictions must be flagged when detected. This control directly targets unsupported assertions and citation theater.

Control point 4: Uncertainty and confidence labeling policy. The organization must define a controlled language policy for uncertainty. The system should:

- prohibit definitive language when evidence is weak or mixed,
- require explicit uncertainty statements for unknowns,
- require explicit "open questions" sections when gaps exist,
- require counterevidence representation for contested topics.

This is a safeguard against rhetorical overconfidence and against pressure prompts that attempt to remove caveats. In a governance-first system, the model is not allowed to silently upgrade certainty.

Control point 5: Publication gates and role-based approvals. Publication is where risk becomes real. Controls must include:

- a gate that blocks external-facing publication without a complete dossier,
- mandatory human sign-off by an approver with authority,
- separation of duties: drafter vs verifier vs approver,
- escalation paths for sensitive topics (ethics review, legal review).

The gate is both a technical feature and a cultural one: it communicates that publication is a governed act.

Control point 6: Dossier generation as an automatic artifact. The system must generate a dossier per output: frozen source set, retrieval logs, claim registry, evidence links, drafts and diffs, policy checker results, and reviewer decisions. The dossier is a safeguard against accountability diffusion: it makes responsibility and traceability concrete.

Control point 7: Post-publication correction workflow. Even with controls, errors will occur. The organization must implement correction mechanisms:

- errata and update logs linked to the original dossier,
- retraction protocols when claims are unsupported,
- notification procedures for stakeholders who relied on the output,

- internal learning reviews to update controls.

Correction is a governance competency. Institutions that cannot correct cannot responsibly publish.

Control point 8: Guardrails against decision laundering. Policies must explicitly prohibit representing AI drafts as authoritative evidence. The system should include prompts and templates that require humans to state: "This is an AI-assisted draft; claims require verification." For high-stakes outputs, the system should require a "human accountability statement" as part of the dossier.

Control point 9: Training and certification for users. Controls are brittle without training. Users must learn:

- how to type claims and label lenses,
- how to verify citations and detect misattribution,
- how to represent disagreement responsibly,
- when to escalate and when to defer publication.

In many organizations, this training is the difference between effective governance and symbolic compliance.

These safeguards are not theoretical. They are operational. They require workflow engineering and management commitment. But they are also the path to turning interpretive AI into a competitive advantage: institutions that can produce defendable narratives faster will outperform those that produce narratives quickly but cannot withstand scrutiny.

Artifact (Save This)

Safeguards design rule. Controls must constrain (i) what sources can be used, (ii) what can be claimed, (iii) how certainty is expressed, and (iv) who is accountable for publication. If any of these is missing, the system will drift toward persuasive, unaccountable output.

4.9.3 Human oversight boundaries

Human oversight is not a generic requirement; it must be specified as explicit boundaries: what must be reviewed by humans, at what granularity, under what roles, and with what escalation rules. In SSH contexts, where interpretation intersects with identity, culture, and legitimacy, oversight boundaries are particularly important because "small" framing choices can carry large consequences.

Boundary 1: External-facing narratives require human review. Any output intended for publication, public communication, or high-impact internal policy should require human review. The minimum is claim-level review for factual claims and lens-level review for interpretive claims. "Human review" must mean more than reading the prose; it must include checking evidence links for high-risk claims and confirming that uncertainty language matches evidence states.

Boundary 2: High-impact topics require enhanced review and escalation. Certain domains require stricter oversight:

- narratives about marginalized communities or contested identities,
- historical accounts tied to institutional culpability or responsibility,
- policy claims with legal or regulatory implications,
- claims that could influence public behavior or safety.

For these, the oversight boundary should include multi-person review and, when appropriate, stakeholder consultation. The institution should define topic categories that automatically trigger escalation.

Boundary 3: Separation of duties is required for publication. The classic triad is:

- **Drafter:** produces the structured output and narrative draft.
- **Verifier:** checks evidence mappings, citation fidelity, and claim typing.
- **Approver:** accepts residual risk, signs off on publication, and owns accountability.

This separation is not bureaucratic theater. It prevents self-confirmation and makes accountability assignable.

Boundary 4: Humans own lens declaration and ethical posture. The system can suggest lenses, but humans must choose whether a lens is appropriate and whether it should be disclosed. Humans must also decide whether the narrative's posture is descriptive, interpretive, or advocacy. This is not a technical decision; it is an institutional one. AI systems should not be allowed to "decide" the ethical stance of an institution.

Boundary 5: Humans must approve any increase in certainty. A dangerous pattern is revision that strengthens claims and removes caveats. The oversight boundary should include a rule: any revision that increases certainty or broadens scope must be explicitly reviewed. This can be implemented by drift detection tools, but the decision remains human.

Boundary 6: Sensitive-data handling requires human governance. If the corpus includes human-subject data, interview transcripts, or confidential archives, humans must govern what can be retrieved and what can be quoted. The system may enforce access controls, but the policy is human. In many SSH settings, ethical review boards (IRBs or equivalents) define these boundaries.

Boundary 7: Internal drafts can be AI-assisted with warnings, but must not be re-labeled as verified. Many organizations will use interpretive AI for internal brainstorming and drafts. That is acceptable if—and only if—the organization maintains a clear boundary between exploratory text and publishable text. The oversight boundary is: internal drafts may circulate with explicit draft labeling, but they must not be treated as institutional truth unless they pass the verification gate.

Human oversight boundaries are therefore not simply about adding a human at the end. They are

about structuring responsibility. A governance-first system designs the pipeline so that humans spend their scarce time where it matters most: evidence checking, lens declaration, and publication accountability.

Risk & Control Notes

Oversight failure mode. "A senior person skimmed it." Skimming fluent prose is not oversight. Oversight is claim-level verification, evidence inspection, lens disclosure review, and accountable sign-off. Without these, human involvement is symbolic.

4.9.4 Monitoring and auditability

Governance is not complete at deployment. Interpretive AI must be monitored because (i) corpora evolve, (ii) prompts and templates evolve, (iii) organizational incentives evolve, and (iv) drift can occur across repeated use even if each individual run seems acceptable. Monitoring must therefore track both output quality and process health, with metrics aligned to evidentiary discipline rather than to superficial fluency.

Citation fidelity monitoring. The core operational metric is citation fidelity: the fraction of claims whose citations truly support them. This should be measured periodically through sampling audits. A monitoring program might:

- sample outputs weekly or monthly,
- sample claims within outputs, weighted toward high-risk claim types,
- verify support relations and record error categories (unsupported, misattributed, decontextualized).

Trends matter. A declining fidelity rate indicates that either the system is drifting, users are bypassing controls, or the corpus is changing in ways that reduce support.

Drift monitoring across revisions. The system should log revision histories and compute drift indicators: how often claims become stronger, how often caveats are removed, how often counterevidence disappears. Monitoring should flag outputs where:

- certainty increases without new evidence,
- interpretive claims lose lens tags,
- open questions are removed under pressure.

These indicators provide early warning of "speed as policy" inside the organization.

Framing audits and periodic red-teaming. Because framing risk is not captured by simple citation checks, the organization should perform periodic framing audits. A framing audit asks:

- are alternative perspectives represented when contestation exists?

- are marginalized viewpoints omitted systematically?
- are narratives sensitive to small retrieval changes?
- do templates embed hidden lenses?

Red-teaming in this context is not about jailbreaks; it is about pressure-testing institutional narratives. Teams can simulate stakeholder pressure prompts and see whether the system remains disciplined.

Audit trails mapping sources → claims → outputs. Auditability requires that any output can be decomposed and traced. The system must maintain:

- immutable identifiers for source snapshots and indices,
- claim registries with evidence links,
- mappings from narrative spans to claim IDs,
- reviewer decision logs.

This is what enables incident reconstruction: if a claim is challenged publicly, the institution can immediately locate the evidence and the approval record.

User behavior monitoring. Many governance failures are behavioral: users bypass controls, copy text outside the system, or treat drafts as final. Monitoring should include:

- frequency of outputs generated without verification,
- frequency of attempts to remove uncertainty language,
- frequency of publication gate overrides (if allowed),
- patterns of tool sprawl (exporting drafts into unmanaged channels).

These are sensitive metrics and must be handled with respect for privacy and organizational culture, but they are essential. Governance is as much about behavior as it is about models.

Model and prompt version monitoring. If prompts or templates change, output posture changes. Monitoring must therefore include change logs and approvals for prompt updates, especially when prompts encode lenses or policy statements.

The end-state is that interpretive AI becomes part of the organization's audit infrastructure. The system does not merely produce text; it produces evidence of responsible process. That evidence is what protects legitimacy.

Artifact (Save This)

Minimum monitoring dashboard (SSH).

- Citation fidelity rate (with error breakdown).
- Unsupported-claim incidence rate.
- Drift indicators: certainty upgrades without new evidence.
- Counterevidence inclusion rate for high-stakes topics.
- Publication dossier completeness rate.
- Percentage of external outputs with verified sign-off.

4.9.5 Deployment deferral criteria

Governance-first design requires the ability to say "not yet." Deployment deferral criteria are the conditions under which an institution should restrict or postpone interpretive AI use, despite the temptation to adopt quickly. These criteria are not pessimism; they are risk management. In SSH contexts, where reputational and legitimacy stakes are high, deferral is often the difference between sustainable adoption and public failure.

Criterion 1: If evidence cannot be logged and audited, do not deploy for publication. If the system cannot produce a dossier that maps sources to claims to outputs, publication-grade deployment should be deferred. Without auditability, the institution cannot defend its narratives. At most, the tool can be used for internal brainstorming with explicit draft labeling.

Criterion 2: If the corpus cannot be governed, restrict to controlled corpora or defer. If the organization cannot define admissible sources, manage licensing, tier credibility, and enforce sensitive-data constraints, then retrieval-augmented interpretive AI is too risky. A practical mitigation is to restrict to a small curated corpus. If even that is not feasible, deployment should be deferred.

Criterion 3: If verification staffing is absent, restrict to internal drafts only. Verification is the binding constraint. If the organization cannot staff verification and sign-off—either through dedicated roles or through allocated time—then interpretive AI should not be used for external narratives. Otherwise, the organization will publish unverified content by default.

Criterion 4: If separation of duties cannot be enforced, defer external-facing use. If the same person drafts, verifies, and approves, the control environment is weak. In small organizations, full separation may be hard, but at minimum, high-impact outputs require independent review. If this is impossible, defer publication deployment.

Criterion 5: If drift and citation fidelity cannot be monitored, defer scaling. A pilot can proceed without full monitoring, but scaling cannot. If the organization cannot measure citation fidelity and drift rates, it will not detect the onset of epistemic erosion until an incident occurs.

Therefore, scaling beyond limited internal use should be deferred until monitoring infrastructure exists.

Criterion 6: If topic sensitivity cannot be managed, restrict topics or defer. If the institution operates in contexts with high political sensitivity, identity-related narratives, or contested histories, it must have escalation protocols. If it cannot define and enforce those protocols, it should restrict use cases to low-stakes topics or defer.

Criterion 7: If disclosure policy is undefined, defer public use. Institutions need clear norms: when AI assistance is disclosed, how provenance is represented, and what disclaimers are required. Without these, public use invites backlash even if the content is largely correct. Disclosure policy is therefore a prerequisite to external deployment.

Criterion 8: If staff training is absent, defer broad rollout. Without training in evidentiary discipline, users will use the system as a writing accelerator and will bypass controls. A minimal training program must be in place before broad rollout.

Deployment deferral criteria create a disciplined adoption path: internal drafts first, governed corpus next, claim registry and evidence mapping next, human verification and sign-off next, monitoring next, and only then external-facing publication. This sequence protects the institution's legitimacy while still capturing productivity gains.

Risk & Control Notes

Minimum Control Set for Interpretive AI Outputs (Operational Baseline).

- **Claim registry:** every output decomposed into claims with types (fact / interpretation / recommendation).
- **Evidence links:** every factual claim linked to supporting sources; contradictions explicitly flagged.
- **Uncertainty labeling:** calibrated language policies for weak/mixed evidence; prohibit false certainty.
- **Publication dossier:** frozen source set + retrieval logs + prompts + draft history + reviewer decisions.
- **Human sign-off:** required for external-facing publication; separation of duties enforced.

4.10 Outlook and Open Questions

4.10.1 Near-term research questions

The near-term research agenda for interpretive AI in the social sciences and humanities is dominated by a simple constraint: the field has learned how to generate text, but it has not learned how to generate *defensible* text at scale. The frontier, therefore, is not primarily model size or fluency. It is the engineering of epistemic discipline into pipelines that can survive institutional pressure, heterogeneous corpora, and contested interpretations. Several research questions are especially urgent because they determine whether interpretive AI becomes a legitimate productivity tool or a legitimacy-destroying narrative engine.

First is **scalable claim verification**. We need methods that decompose outputs into claims reliably and then verify those claims against sources with low false negatives. In SSH contexts, false negatives matter more than false positives: it is better to flag a supported claim for review than to let an unsupported claim pass unnoticed into publication. The challenge is that claim verification cannot be restricted to crisp factual entailment. Many claims are mixed: partly descriptive, partly interpretive, partly causal. Research must develop verification taxonomies that treat claim types differently and that can return structured outputs such as "supported under lens L" or "supported descriptively but causal inference not justified."

Second is **citation fidelity at scale**. We need tooling that detects whether citations genuinely support the attached claim, not merely whether they are topically related. This includes decontextualization detection: a passage can be quoted accurately and still misrepresent meaning by omitting qualifiers or surrounding argumentative context. Developing robust, passage-level citation checkers that work across PDFs, scanned archives, and heterogeneous genre styles remains a core bottleneck. A credible near-term direction is hybrid verification: combine retrieval-based passage matching, lightweight entailment checks, and human review triage, but the research question is how to calibrate this pipeline and how to measure its failure rates.

Third is **drift detection across iterative rewriting**. Interpretive AI is frequently used in revision loops: summarize, rewrite, simplify, adapt to audience, strengthen argument. Drift emerges when rhetorical optimization gradually detaches claims from evidence or upgrades certainty without adding support. We need drift metrics that track changes in claim scope and certainty across revisions and that trigger re-verification when thresholds are crossed. This is not merely a model question; it is a pipeline question: how do we enforce that revisions cannot silently change epistemic commitments?

Fourth is **framing quantification**. The hardest open research problem is measuring framing and omission in a way that is objective enough to operationalize. Framing is not always bias; selection is necessary. The problem is unacknowledged selection that materially distorts meaning. Near-term research should focus on pragmatic proxies: counterevidence inclusion rates, multi-lens comparison

metrics, retrieval sensitivity measures, and coverage diagnostics that estimate whether relevant perspectives are systematically absent. The research frontier is to move from "bias detection" as a vague aspiration to "framing robustness" as an evaluable property.

Finally, there is a near-term research question about **human–AI workflow design**. Verification is expensive, and the bottleneck is human attention. Research should ask: what user interfaces make claim verification fast without becoming performative? How do we design review workflows that preserve scholarly and institutional standards rather than eroding them through convenience? In SSH settings, this is not a minor HCI question; it is the difference between accountable use and epistemic collapse.

4.10.2 Technical unknowns

Technical unknowns for interpretive AI in SSH domains cluster around robustness: robustness of evidence alignment, robustness to contradiction, and robustness against subtle rhetorical manipulation. These unknowns are not theoretical curiosities; they determine whether systems can be trusted to support high-stakes institutional narratives.

A first unknown is **robust evidence alignment under long-context**. Long-context models can ingest more text, but ingestion is not alignment. The system may still misattribute a claim to the wrong passage, misunderstand a qualifier, or merge distinct arguments into a single synthesized assertion. In SSH contexts, meaning is often distributed and contextual, and long-context may increase the risk of smoothing: contradictions get averaged away, uncertainty gets resolved rhetorically, and argumentative nuance is compressed. We do not yet know how to build alignment mechanisms that remain stable when the context window becomes large and heterogeneous.

A second unknown is **multi-source contradiction handling at scale**. Contradictions are not always errors; they are often the point. Yet most AI systems are optimized to produce a single coherent narrative. Handling contradiction requires representing disagreement without collapsing it, and doing so consistently across many claims. Technically, this requires that retrieval surfaces counterevidence reliably, that claim typing distinguishes contested from uncontested claims, and that the drafting system maintains uncertainty and pluralism rather than forcing resolution. The unknown is whether we can build systems that are naturally "disagreement-native" rather than disagreement-avoiding.

A third unknown is **subtle rhetorical manipulation detection**. The danger in interpretive AI is not always explicit falsehood; it is rhetorical posture. Systems can manipulate through word choice, sequencing, emphasis, and the selective presentation of caveats. Detecting this requires modeling pragmatics, not just semantics. The unknown is whether we can build detectors that identify when an output is becoming more persuasive than justified by evidence—especially when the persuasion is subtle and plausible. This includes detecting lens laundering: when a lens-dependent interpretation is written in a neutral descriptive voice.

A fourth unknown is **decontextualization at the passage level**. Many SSH errors arise from extracting a line from its argumentative environment. Technical systems struggle to represent "context" as a first-class object. We do not yet have reliable methods to determine whether a cited passage is being used in a way consistent with the author's intent or whether it is being repurposed rhetorically. Without progress here, citation fidelity tooling will remain brittle.

A fifth unknown is **robustness to retrieval perturbations**. If small changes in retrieval ranking flip narrative framing, the system is unstable. We need methods to quantify and reduce this sensitivity, perhaps through retrieval ensembles, stability-aware ranking, or explicit framing constraints that limit how selection drives narrative posture. But we do not yet know which approaches will be reliable and economically feasible for real SSH corpora.

The common thread is that technical unknowns are not solved by "better language models" alone. They require system-level design: retrieval, claim registries, evidence mapping, and constrained drafting. The frontier is not a single model capability; it is stable, auditable, evidence-aligned synthesis under pressure.

4.10.3 Governance unknowns

Even if technical problems improve, governance unknowns will remain because interpretive AI sits at the boundary between knowledge production and institutional speech. Governance questions are therefore about standards, attribution, and liability—areas where norms are evolving and often contested.

The first governance unknown is **standards for AI-assisted publication**. Institutions lack shared standards for what it means to publish AI-assisted narratives responsibly. Questions include:

- What disclosures are required, and to whom?
- What constitutes adequate evidence logging?
- What is the minimum acceptable verification process?
- How should institutions represent uncertainty and disagreement?

Without standards, organizations will either over-disclose in ways that undermine trust or under-disclose in ways that provoke backlash. A governance frontier is the emergence of "audit-grade" publication standards that specify dossiers, claim verification requirements, and sign-off roles.

The second unknown is **attribution and authorship norms**. In SSH domains, credit and accountability matter. If a model drafts an argument based on a corpus of scholarship, what counts as proper attribution? How do we distinguish between synthesis and appropriation? How do we prevent the re-expression of distinctive scholarly ideas without citation? Governance must define norms that respect intellectual labor while still enabling synthesis. This is not only a legal issue; it is an ethical and reputational one.

The third unknown is **liability allocation for institutional narratives**. If an AI-assisted narrative causes harm—misrepresents a community, triggers policy backlash, or spreads an incorrect historical claim—who is responsible? The organization, the approver, the tool provider, or the individual who prompted the system? In governed environments, liability allocation shapes behavior. If liability is ambiguous, organizations may be tempted to treat AI as a scapegoat. Governance must instead align liability with accountability: responsibility remains with the institution and its human signatories.

The fourth unknown is **audit norms and contestability**. If an institution is challenged, what evidence must it produce? What counts as a sufficient audit trail? How long must dossiers be retained? Who has access? These are governance questions that intersect with privacy, IP, and academic freedom. The unknown is whether a common set of contestability norms can emerge that is acceptable across sectors.

Finally, there is an unknown about **educational integrity**. Universities and schools are still negotiating what AI assistance means for learning outcomes. Governance must balance productivity and accessibility with the risk that students stop practicing core skills such as close reading and argument construction. Institutional policy, not technology alone, will determine whether interpretive AI becomes a teaching aid or an epistemic shortcut.

4.10.4 What would change the assessment

The assessment of interpretive AI in SSH domains would change materially if several enabling conditions become widely adopted. These conditions are not speculative miracles; they are operational shifts that make defensible use cheaper and therefore sustainable.

First, **adopted provenance standards** would change the baseline. If organizations can attach standardized provenance metadata to sources and outputs—consistent identifiers, passage-level citation anchors, and tamper-evident logs—then the cost of audit and contestation falls. Provenance standards would also facilitate tool interoperability: evidence maps and dossiers could move between systems without losing meaning.

Second, **reliable verification tooling** would change the economics of review. If citation fidelity checkers and claim verification systems become accurate enough to triage effectively—especially with low false negatives—then human reviewers can focus on the hardest interpretive judgments rather than on mechanical citation checking. The key here is not perfect automation; it is a shift in the distribution of human effort.

Third, **benchmarks for epistemic discipline and framing robustness** would enable progress measurement. We need evaluation suites that test contested corpora, missing evidence, multi-lens interpretation, retrieval perturbation stability, and resistance to pressure prompts. If such benchmarks become standard, organizations will stop using fluency metrics as a proxy and will

begin to compete on defendability.

Fourth, **institutional workflow normalization** would change incentives. If organizations normalize the claim registry and dossier as standard artifacts—much like model documentation in regulated finance—then interpretive AI becomes governable by default. This requires cultural adoption, not just technology.

Finally, **clear disclosure and attribution norms** would reduce reputational volatility. If stakeholders understand what AI assistance means and what controls exist, then AI-assisted narratives can be judged fairly rather than through suspicion. Norms that connect disclosure to verifiable process are especially important: disclosure without discipline is not trust-building; discipline with appropriate disclosure can be.

4.10.5 Link to subsequent papers

This chapter's closing bridge to Chapter 10 (Multimodal Frontier Systems: Vision–Language–Action) is straightforward: once perception enters the pipeline, narrative risk becomes action risk. Interpretive AI in SSH domains already shows that the critical governance problem is not generating text but governing the chain from evidence to claims to publication. Multimodal systems extend the chain: images, video, sensor data, and environment perception become sources, and generated narratives can directly trigger decisions and actions.

The connection is that provenance and gating must be system-level, not model-level. In Chapter 10, a system may perceive a scene, produce a narrative interpretation of that perception, and then recommend or execute an action. The same failure modes scale up: misattribution (misreading an image), framing bias (what the system emphasizes), rhetorical overconfidence (claiming certainty from ambiguous perception), and drift across iterative perception-action loops. The controls therefore generalize: claim registries become action registries; evidence links become sensor provenance links; publication gates become action gates.

In other words, interpretive AI is the governance rehearsal for multimodal action systems. If an institution cannot govern narratives under evidentiary discipline, it will not be able to govern perception-to-action pipelines where the cost of error is physical, not merely reputational. Chapter 10 extends the theme: the frontier is governed capability, and governance begins with traceable chains from inputs to commitments to consequences.

Artifact (Save This)

Transition statement. Chapter 9 shows that legitimacy depends on provable chains from sources to claims to narratives. Chapter 10 extends the chain to perception and action: when models see and act, provenance and gating must become system-level controls, or the institution will deploy autonomy it cannot defend.

Bibliography

[1] Patrick Lewis, Ethan Perez, Aleksandra Piktus, et al. *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks*. Advances in Neural Information Processing Systems (NeurIPS), 2020.

[2] Oren Etzioni, Michele Banko, and collaborators. *Citation Needed: A Taxonomy and Survey of Citation Fidelity in Large Language Models*. arXiv preprint arXiv:2309.06727, 2023.

[3] Kenton Lee, Nicholas FitzGerald, et al. *Hallucinations in Neural Machine Translation*. Proceedings of the NAACL-HLT, 2018.

[4] Yao Fu, Pengfei Liu, et al. *Measuring and Mitigating Semantic Drift in Iterative Text Generation*. Proceedings of the Association for Computational Linguistics (ACL), 2023.

[5] National Institute of Standards and Technology (NIST). *Artificial Intelligence Risk Management Framework (AI RMF 1.0)*. U.S. Department of Commerce, 2023.

[6] Teun A. van Dijk. *Discourse and Power*. Palgrave Macmillan, 2008.

[7] Ben Shneiderman. *Human-Centered Artificial Intelligence: Reliable, Safe & Trustworthy*. Oxford University Press, 2022.

[8] C2PA (Coalition for Content Provenance and Authenticity). *C2PA Technical Specification, Version 1.3*. Adobe, Arm, BBC, Intel, Microsoft, 2023.

[9] Katherine Lee, Mahdi Namazifar, et al. *Evaluating Factual Consistency of Long-Form Text Generation*. Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), 2022.

[10] Fei Xia, William Fedus, et al. *Gato: A Generalist Agent*. Transactions on Machine Learning Research (TMLR), 2022.

Chapter 5

Chapter 10 — Multimodal Systems: Perception to Action with Provenance and Gates

Abstract. Multimodal frontier systems—those that integrate vision, language, and action—represent a decisive shift in how artificial intelligence interacts with the real world. These systems do not merely generate text or interpret images; they perceive environments, reason over heterogeneous evidence, and increasingly trigger downstream actions through tools, interfaces, and control systems. At this point, the traditional boundary between "model behavior" and "system behavior" collapses. Risk, accountability, and value no longer reside in the model alone, but in the full perception–reasoning–action loop.

This chapter examines why multimodality marks a frontier moment not because of representational novelty, but because it amplifies institutional exposure. Errors are no longer confined to incorrect statements; they originate in misperception, propagate through reasoning layers, and culminate in consequential actions. Conventional evaluation metrics—accuracy, language benchmarks, or vision scores—fail to capture these dynamics. Similarly, alignment and instruction-following do not protect against multimodal prompt injection, privacy leakage through mixed-media logs, or authority escalation via tool coupling.

The chapter reframes multimodal AI as a governance problem first and a modeling problem second. It analyzes emerging failure modes unique to multimodal systems, including content-as-instruction attacks, cascading uncertainty, provenance loss, and silent action misfires. It argues that safe deployment requires explicit action gating, least-privilege design, auditable evidence trails, and system-level evaluation that treats perception as a sensor and action as a controlled authority.

As the capstone of *AI 2026: Frontier Awareness Without the Hype*, this chapter unifies the book's central insight: frontier AI risk does not come from intelligence alone, but from coupling intelligence to action without sufficient control, evaluation, and institutional oversight.

5.1 Orientation and Scope

5.1.1 Motivation and context

Multimodal frontier systems are no longer best understood as "models that can see." They are better understood as *systems that can interpret heterogeneous evidence and then participate in consequential workflows*. In a typical deployment, the system ingests a mixture of text, images, scanned documents, diagrams, screenshots, tables, audio snippets, or sensor-like streams. It then produces outputs that range from natural-language explanations to structured extractions, classifications, recommended actions, or direct tool calls. The important escalation is not the presence of images; it is the *coupling* of perception to reasoning and, increasingly, to action. The moment a system moves from "describe what you see" to "decide what to do next," it becomes an operational actor in an environment of incentives, constraints, and adversaries.

For executives and governance-minded practitioners, multimodality is not a feature request; it is an exposure shift. It changes what counts as input, what counts as evidence, and what counts as an acceptable failure. In text-only settings, errors often present as incorrect statements, flawed summaries, or unsupported reasoning. In multimodal settings, the failure chain can start earlier and be harder to detect: a misread number in a table, a missed negation in a scanned contract, an OCR artifact that changes a date, a blurry image that hides a safety label, a diagram that is interpreted with the wrong legend, or an audio segment that is transcribed with a subtle but material substitution. When these perceptual errors feed a reasoning module that is optimized for coherence, the system can produce a decision narrative that sounds decisive while resting on corrupted evidence. If that narrative is then used to trigger downstream actions—an approval, a filing, a dispatch, a purchase, a customer communication, a compliance escalation, or a control-system command—the organization has effectively built an automated pathway from *noisy perception* to *consequential action*.

This is why the chapter frames vision–language–action (VLA) systems as a governance challenge first. The core institutional question is not "How accurate is the vision model?" but "Under what conditions do we allow this system to influence decisions, and how do we reconstruct what it saw, what it believed, and what it did?" That reconstruction requirement is central because accountability is retrospective: when something goes wrong, the organization must be able to answer what evidence was used, what transformations were applied, which tools were invoked, what intermediate assumptions were introduced, and who approved what. Multimodal systems raise immediate pressure against that requirement because the evidence may be high-dimensional, privacy sensitive, difficult to store, and difficult to summarize without distortion.

A second governance pressure arises from the tool layer. Multimodal pipelines routinely rely on external components: OCR engines, layout parsers, retrieval systems, databases, web browsers, code execution sandboxes, and action APIs. These are not passive accessories; they are authority surfaces.

A system that can "look" at a document and then "click" a button in an enterprise interface is no longer only generating content; it is participating in operations. Each tool call is a potential locus of failure, misuse, or abuse. The system may be induced to call tools in unsafe sequences, to request data it should not access, or to act on ambiguous evidence without escalation. If the tool layer is integrated naïvely, the organization can accidentally confer more authority than intended because the system is easiest to build when it is allowed to operate end-to-end. Governance, however, demands that authority be explicit, bounded, and reviewable.

A third pressure arises from adversarial conditions. In regulated and high-value environments, inputs are not reliably benign. Documents can be crafted to manipulate interpretation; images can embed instructions; layouts can be designed to fool extraction; and mixed-media artifacts can conceal sensitive information that the system might surface improperly. The security model of multimodal systems cannot assume that "input equals data." In a system that follows instructions and uses tools, *content can become control*. That is the conceptual pivot that makes multimodal prompt injection a first-class frontier issue rather than a niche security story.

Finally, multimodality changes organizational ergonomics. People tend to trust perception because it feels grounded. A screenshot, a photo, or a scan carries an aura of "objective evidence," even when it is incomplete or ambiguous. Multimodal systems can unintentionally exploit this human bias by producing narratives that appear anchored in reality. In practice, the system may have silently hallucinated a missing field, guessed at a label, or over-weighted a spurious visual cue. The output can therefore combine two dangerous properties: *high persuasive force* and *low epistemic integrity*. The chapter's orientation begins here because this is the boundary condition for safe deployment: multimodal systems must be treated as fallible sensors coupled to fallible reasoning, not as oracles that "understand" the world.

5.1.2 Why this topic is at a frontier moment

Multimodal AI has existed for decades in narrow forms: computer vision classifiers, OCR pipelines, speech recognition, and purpose-built fusion systems for captioning or retrieval. What makes the present moment frontier is not that these components exist, but that general-purpose foundation models now offer a single interface that *appears* to unify perception, reasoning, and communication. In practical terms, this means that teams can build systems that accept arbitrary mixed inputs and then produce fluent, instruction-following outputs that look like the product of integrated understanding. When such a system is further connected to tools, it can become an operator: extracting fields from a document, cross-checking a database, drafting a response, and executing the next step in a workflow.

This transition from "describe" to "operate" is a regime change. In the describe regime, the system's failure modes are often limited to mislabeling or miscaptioning, which can be corrected by a human downstream. In the operate regime, the system's outputs can influence state. If a system misreads a

compliance threshold and initiates an action, or misinterprets a form and submits an incorrect filing, the error becomes embedded in an operational process and can propagate quickly. Frontier risk therefore arises not from any single model's capabilities but from the *system coupling* that turns outputs into actions.

Several forces are converging to make this coupling increasingly common. First, enterprises are digitizing workflows that are fundamentally multimodal: invoices and receipts, contracts and exhibits, spreadsheets and slide decks, risk reports with charts, emails with attachments, recorded calls, and screenshots of internal dashboards. Second, organizations increasingly seek automation not in isolated tasks but in end-to-end processes: triaging documents, routing tickets, validating claims, monitoring controls, and generating compliance artifacts. Third, tool ecosystems have matured: APIs, RPA layers, browser automation, and retrieval systems make it technically straightforward to connect a model to operational systems. When these pieces combine, the temptation is strong to let the multimodal system "just do the workflow."

At the same time, the frontier is defined by *mismatched maturity*. Model capability has advanced quickly, while governance and evaluation standards for multimodal action systems have not. Benchmarks exist for vision classification, OCR accuracy, VQA, and even some multimodal reasoning tasks, but these benchmarks rarely capture the end-to-end dynamics that matter to institutions: distribution shift in real documents; adversarial artifacts; privacy constraints; partial observability; ambiguous inputs; tool failures; latency trade-offs; and the irreversibility of actions. The gap between what can be built and what can be safely governed is therefore widening.

The frontier moment is also characterized by a new class of security failures. In classic software security, untrusted input is separated from control flow. In multimodal assistants, that separation can erode because the system is designed to follow instructions embedded in content. A scanned document or an image can contain text-like instructions, visually embedded prompts, or layout tricks that change what the system believes it is supposed to do. When the system has tool access, such manipulations can become operationally meaningful. This is not a theoretical concern; it is a structural property of systems that treat content as conversation partners. The frontier issue is therefore not simply "model jailbreaks" but *control-plane integrity*: ensuring that the system's authority cannot be hijacked by the artifacts it processes.

Another frontier property is that multimodal systems increase the role of uncertainty in ways that are difficult to communicate. Perception is inherently uncertain; sensors are noisy; scans are incomplete; and context is missing. Yet multimodal foundation models are optimized for coherent outputs, not calibrated uncertainty. A system can be uncertain about whether a line item is "$10,000" or "$100,000" and still produce a smooth summary. Unless the system is architected to surface uncertainty explicitly and to trigger verification steps, it will default to producing a single narrative. This is precisely the kind of behavior that can slip through organizations: outputs feel confident, especially when they come with an image-based justification, and the human reviewer may not realize that the system made a guess at a crucial step.

Finally, the frontier moment is societal and regulatory. Multimodal systems are natural instruments of surveillance and monitoring: cameras, audio, and document streams can be fused to track behavior, detect anomalies, and infer intent. This creates both opportunity (safety monitoring, fraud detection, compliance automation) and significant risk (privacy harm, chilling effects, discriminatory outcomes, and reputational damage). The governance of multimodal systems therefore cannot be separated from broader institutional commitments and legal obligations. Even when the technical system "works," the deployment can fail organizationally if it violates privacy norms or creates unacceptable monitoring regimes.

In short, this topic is frontier because it sits at the intersection of three accelerations: (i) models that can process mixed evidence, (ii) tool stacks that convert inference into operations, and (iii) institutions that seek automation under accountability constraints. Where these accelerations overlap, the question is no longer "Can the model do it?" but "Can the organization defend what it did?"

5.1.3 What has changed recently

The recent shift is best described as a convergence of capability, accessibility, and integration. On the capability side, vision–language models have improved not only at captioning but at instruction-conditioned tasks: answering questions about complex images, extracting structured information from documents, interpreting charts and tables, and reasoning across multiple pages or mixed attachments. These improvements matter because they reduce friction: systems no longer require carefully curated pipelines for each document type. A single assistant can accept a broad range of inputs and produce plausible outputs, which in turn accelerates experimentation and deployment.

A second capability change is improved *cross-modal instruction following*. Earlier multimodal systems were often brittle: they could perceive but struggled to integrate instructions in a reliable way. Contemporary systems can take a user's directive—"extract the key terms, compare to our policy, and draft an escalation email"—and perform multi-step work that combines perception, retrieval, and generation. This makes multimodal systems more directly useful for enterprise workflows, but it also makes the failure modes more consequential because the system is now doing the work of interpreting policy and deciding what to do next.

A third change is the maturation of long-context handling for mixed media. Many real workflows involve multi-page documents, long email threads, or collections of attachments. Systems are increasingly asked to hold a broader context window that includes both text and representations derived from images (OCR text, layout embeddings, or visual features). This creates new opportunities (document-level reasoning and traceability) and new risks (context dilution, provenance confusion, and silent reliance on low-quality extractions). Long-context multimodality does not simply scale the amount of information; it scales the complexity of knowing what information mattered.

On the integration side, tool coupling has become the norm. Multimodal assistants are rarely

deployed as pure inference endpoints. They are connected to OCR engines, retrieval systems, knowledge bases, internal APIs, ticketing systems, workflow engines, and browser automation layers. This tool coupling is attractive because it allows the system to move from understanding to execution. It also introduces two compounding governance problems. First, every tool introduces its own failure modes and security assumptions. Second, the integration encourages a form of implicit autonomy: if a system can fetch data and take actions, it becomes tempting to let it do so automatically to realize efficiency gains.

Another change is organizational: teams have begun to treat multimodal assistants as interface operators. Instead of building bespoke software for every task, organizations increasingly contemplate a model that can "use" existing enterprise systems—reading screens, filling forms, and navigating workflows. This reduces engineering burden but increases governance burden because the system is operating in spaces designed for humans, often without explicit APIs or structured guardrails. The system's behavior becomes sensitive to UI changes, ambiguous prompts, and content embedded in screens. Moreover, the audit trail becomes harder to formalize because actions occur through UI interactions rather than explicit API calls with clean logs.

Security research and practitioner awareness have also changed. The idea that prompts can be injected through non-text channels—through documents, images, or even layout artifacts—has moved from curiosity to operational concern. The field increasingly recognizes that content can carry instructions that the model may treat as authoritative. In an enterprise setting, where documents from counterparties, customers, or external sources are routine, this is not a corner case; it is a baseline threat model. Organizations are beginning to implement content sanitization, separation of untrusted content from system prompts, and allowlist-based tool permissions. Yet standardized patterns remain immature, and many deployments still rely on implicit trust in the model's "judgment."

Finally, privacy and compliance constraints have sharpened. Multimodal inputs often include personally identifiable information, confidential business information, and sensitive imagery. Logging and debugging these systems requires capturing inputs, intermediate representations, and tool calls. But storing raw images or documents can violate retention policies, confidentiality obligations, or regulatory requirements. As a result, teams face a tension: robust auditability often demands rich logs, while privacy demands minimization. Recent work has focused on hashing, redaction, selective retention, and cryptographic approaches, but practical and standardized enterprise patterns are still evolving.

The net effect of these changes is that multimodal systems have become *easy to deploy* relative to how hard they are to govern. The technology has shifted from research prototypes to enterprise pilots, while the institutional frameworks for evaluation, security, privacy, and accountability are still catching up. That mismatch is precisely what defines a frontier topic in a governance-first book: it is where organizations can move fast, but cannot yet move safely without deliberate design.

5.1.4 Explicit exclusions and non-goals

This chapter is intentionally not a robotics engineering manual. While robotics is a natural endpoint of vision–language–action systems, the governance problems addressed here appear well before physical robots enter the picture. A multimodal assistant that reads invoices and triggers payments, a system that reviews contracts and initiates escalations, or a model that monitors dashboards and executes operational playbooks already operates in a world where actions have real consequences. The chapter therefore focuses on the governance structure of perception-to-action coupling, not on the mechanical or control-theoretic details of robotics hardware.

The chapter is also not a security exploitation guide. It does not provide step-by-step instructions to craft adversarial documents, bypass controls, or perform prompt injection attacks. Where security risks are discussed, they are framed as threat models, control categories, and evaluation requirements suitable for executives and responsible builders. The goal is to help organizations recognize the attack surface and implement safeguards, not to enable misuse.

A third non-goal is to claim that multimodal models are reliable sensors by default. Perception systems are probabilistic and context-dependent. Images can be ambiguous, documents can be incomplete, and audio can be noisy. A multimodal foundation model may produce plausible interpretations, but plausibility is not the same as correctness. The chapter therefore avoids treating multimodal AI as a substitute for validated instrumentation or formal verification. When perception is used in safety-critical contexts, additional validation layers, redundancy, and domain-specific checks are required. The chapter's focus is to articulate why such checks are necessary and how to design operational workflows that respect uncertainty.

The chapter is not a comprehensive survey of the multimodal literature. The frontier evolves quickly, and a survey approach can obscure the structural governance lessons by over-indexing on model names and incremental benchmarks. Instead, the chapter concentrates on stable abstractions: how errors propagate, how tool coupling creates authority, how content can become control, how privacy and logging collide, and how evaluation must be end-to-end to be meaningful. Specific model examples are used only insofar as they illustrate these abstractions.

Another explicit exclusion is the notion that alignment, instruction tuning, or preference optimization "solves" multimodal risk. Alignment methods can improve helpfulness and reduce some unsafe outputs, but they do not remove system-level vulnerabilities introduced by tool access, UI coupling, or adversarial inputs. The chapter therefore treats alignment as one component in a broader control stack, not as an endpoint. The governance lens is: if a system can act, it must be controlled by enforceable mechanisms beyond the model's internal preferences.

Finally, the chapter does not attempt to resolve all questions of liability, regulation, or ethical policy. Those topics matter, but they vary by jurisdiction and domain. Instead, the chapter offers governance primitives that remain useful across contexts: explicit authority boundaries, least-privilege access, auditable traces, privacy-preserving logging, and deployment deferral criteria. These primitives

are actionable regardless of the precise legal framework and can be adapted to the organization's obligations.

These exclusions are deliberate because they preserve focus on the central claim: the frontier issue is not that multimodal systems are impressive; it is that they are *hard to evaluate, hard to secure, and hard to govern* once connected to action. The chapter is designed to provide executives and advanced practitioners with a disciplined mental model and a set of operational controls that can be applied immediately, without pretending that the underlying science has matured to the level of guaranteed reliability.

5.1.5 Role of this chapter in AI 2026

This chapter functions as the capstone of *AI 2026: Frontier Awareness Without the Hype* because it operationalizes every major theme of the book in a single integrated setting. Chapters 1 through 5 establish that frontier AI risk emerges when systems act over time, reason internally, are steered through representation-level interventions, depend on retrieval and memory, and use planning and search to optimize objectives. Chapters 6 through 9 show how these dynamics manifest in domain deployments: chemistry, physics surrogates, finance under constraint, and interpretive knowledge work. Multimodal vision–language–action systems unify these strands because they are the clearest embodiment of "frontier AI as system."

From Chapter 1, we inherit the evaluation thesis: output-based evaluation collapses when systems operate over trajectories. Multimodal action systems are trajectory systems by default. Their failures are often delayed and distributed across components: a misread figure leads to a flawed plan, which triggers a tool call, which changes an external state. Evaluating only the final answer is therefore misleading; one must evaluate the full perception–reasoning–action trace. This chapter extends that logic into environments where the evidence itself is high-dimensional and uncertain.

From Chapter 2, we inherit the caution about reasoning depth and delayed failure. Multimodal systems often require multi-step reasoning to reconcile contradictory evidence across modalities. That reasoning can be sophisticated, but it can also be confidently wrong because the system is reasoning over flawed perception. The chapter therefore emphasizes that deeper reasoning does not automatically reduce risk; it can amplify it by constructing persuasive narratives on top of incorrect inputs. The governance implication is that reasoning must be bounded by verification, not merely encouraged.

From Chapter 3, we inherit the insight that internal control interventions are governance decisions. As organizations seek to make multimodal systems reliable, they will be tempted to apply representation engineering, safety filters, and steering techniques to constrain behavior. In multimodal settings, however, steering can have collateral effects: improving sensitivity to certain cues while increasing vulnerability to others, or improving extraction accuracy while degrading uncertainty signaling. The chapter positions these interventions as change-controlled artifacts requiring evaluation of off-target

effects, not as quick fixes.

From Chapter 4, we inherit the memory and provenance problem. Multimodal systems routinely depend on retrieval to supplement their interpretation: policies, reference documents, prior cases, and knowledge base entries. The combined context can be enormous and heterogeneous. Without explicit provenance tracking, the system can blur sources and create outputs that are impossible to audit. This chapter therefore treats evidence attribution and claim-to-source mapping as core requirements, not optional niceties. In multimodal settings, provenance is harder because the evidence may be an image region, a page coordinate, or a segment of audio, not a clean text quote.

From Chapter 5, we inherit the planning and search risk: optimization amplifies objective misspecification. When multimodal systems are allowed to act, the objective is often framed as "complete the workflow" or "resolve the ticket." If constraints are underspecified, the system can take logically consistent steps that violate institutional norms, privacy obligations, or risk thresholds. The chapter therefore emphasizes safe action design: least-privilege access, explicit allowlists, irreversible-action gates, and stop conditions triggered by uncertainty or policy ambiguity.

Finally, from the applications chapters (6–9), we inherit the lesson that domain value is inseparable from validation and governance. In chemistry and physics, proxy optimization and rollout instability show that one-step success does not guarantee real-world validity. In finance, constraint and reviewability dominate because fiduciary duty and disclosure risk cannot be outsourced to a model. In interpretive knowledge work, evidentiary discipline is essential to avoid epistemic drift. Multimodal systems bring all of these lessons into a single operational reality: they operate on evidence, they propose decisions, they can act, and they must be governed as accountable systems.

Thus, the role of Chapter 10 is to make explicit the book's culminating thesis: frontier AI is increasingly not about larger models, but about tighter coupling between perception, reasoning, and action. This coupling produces new opportunities—document intelligence, inspection, mixed-media decision support—but it also creates a failure landscape where security, privacy, auditability, and authority boundaries are no longer peripheral concerns. They are the core of deployment competence. The chapter closes the collection by arguing that an organization's frontier posture should be measured not by what models it can access, but by what systems it can *evaluate, constrain, and defend* under real-world conditions.

5.2 Conceptual Abstraction

5.2.1 Core abstraction

The central abstraction of this chapter is that a multimodal frontier system is not merely a predictor of text. It is a *decision participant* that maps heterogeneous evidence into both communicative outputs and operational actions. Formally, let the input be a tuple of modalities

$$X \;=\; (x^{\text{text}},\, x^{\text{img}},\, x^{\text{audio}},\, x^{\text{doc}},\, x^{\text{tab}},\, x^{\text{meta}}),$$

where x^{doc} can represent multi-page documents (with layout), x^{tab} can represent structured tables or spreadsheets, and x^{meta} can represent provenance and policy context (who supplied the artifact, retention classification, access labels, and workflow state). The system produces (i) an output Y intended for human consumption or downstream processing and (ii) a set of candidate or executed actions A in an external environment:

$$(Y, A) \;=\; F(X;\, \theta, \pi).$$

Here θ represents model parameters (encoders, fusion, and decoding) and π represents a policy layer that governs tool use and action selection, including permission constraints, gating rules, and escalation logic. This is a deliberately compact abstraction. It forces attention to an institutional reality: the same system that "explains" can also "act," and actions alter the environment that supplies subsequent inputs.

Two consequences follow immediately. First, *perception errors become decision errors.* If the system misperceives x^{img} or misextracts content from x^{doc}, the downstream reasoning and action components inherit corrupted state. Second, the system's behavior is inherently *trajectory-based.* It often runs in a loop:

$$X_t \to (Y_t, A_t) \to \text{Environment update} \to X_{t+1},$$

where A_t may include tool calls, UI operations, or structured commands that change records, trigger workflows, or communicate with users. This loop is the conceptual bridge to earlier chapters: evaluation must be conducted over trajectories, not single outputs, because the organization ultimately bears responsibility for what the system does over time.

A governance-first abstraction treats F as only one layer in a broader system that includes (a) transformations of raw media into internal representations (OCR, ASR, layout parsing, image normalization), (b) evidence selection and retrieval (memory), (c) reasoning and planning, and (d) an enforcement boundary that determines what actions are possible. In practice, many of the most important failures occur *between* these layers. The model might be "correct" given its inputs, yet the inputs are wrong due to OCR errors; or the model might be safe in isolation, yet the tool policy π allows unsafe sequences of actions. This motivates a conceptual discipline: treat the system as an

end-to-end pipeline whose reliability depends on the integrity of every transformation and boundary.

An additional conceptual requirement is *evidence semantics*. In multimodal settings, "evidence" is not simply a string of tokens. It may be a region of an image, a page coordinate in a PDF, a time interval in an audio recording, or a specific cell in a spreadsheet. Therefore, the mapping from X to Y must be accompanied by an *evidence map* that can support audit and contestability. In an idealized form, we might represent the system's output as

$$Y \;=\; (\hat{y},\, \mathcal{E},\, \mathcal{U}),$$

where $\hat{y}$ is the proposed content (answer, extraction, recommendation), $\mathcal{E}$ is an evidence set (references to specific regions/pages/cells), and $\mathcal{U}$ is an uncertainty summary. The chapter's governance claim is that $\mathcal{E}$ and $\mathcal{U}$ are not optional add-ons. Without them, the organization cannot defend decisions, cannot reproduce failures, and cannot reliably determine whether a given action was justified.

Finally, the abstraction emphasizes that actions are not only "do something" commands; they include *information actions*. Retrieving a confidential record, copying content into a ticket, or summarizing an image for a chat channel are actions that can violate privacy, confidentiality, or policy constraints even if no physical effect occurs. Thus A should be conceptualized as a set of state transitions:

$$A \subseteq \{\text{read, write, notify, execute, approve-request, escalate}\}.$$

Different action classes demand different governance. "Read" actions can be governed by access control; "write" actions can be governed by change management; "execute" actions can be governed by irreversible-action gates; and "notify" actions can be governed by communication policies. This conceptual taxonomy is crucial because it prevents the common error of treating all tool use as equivalent. In multimodal systems, tool use is authority, and authority must be explicitly decomposed.

5.2.2 Key entities and interactions

To govern multimodal frontier systems, one must identify the entities that participate in the perception–reasoning–action loop and the interactions that create risk. A useful decomposition includes the following components.

(1) Sources and channels. These are the origins of X: cameras, scanners, email attachments, user uploads, internal dashboards, call recordings, PDFs from counterparties, spreadsheets, and databases. Each source has a trust level and a distribution profile. A scanned contract from a counterparty should be treated as untrusted input; an internal ERP export might be trusted but still error-prone. Governance begins by labeling sources and attaching provenance metadata.

(2) Preprocessors and extractors. These include OCR, layout parsers, chart readers, table

extractors, and speech-to-text systems. They transform raw media into intermediate representations. This layer is conceptually equivalent to sensors and measurement pipelines in engineering. It must be versioned, testable, and monitored. A change in OCR settings can alter extracted numbers; a change in layout parsing can reorder fields. If these transformations are not controlled, the system becomes non-reproducible even when the core model is unchanged.

(3) Encoders and modality embeddings. The system encodes each modality into latent representations. Whether this is done via separate encoders or a shared architecture is less important than the conceptual fact: each encoder can fail differently. Vision encoders may be sensitive to blur and occlusion; text encoders may be sensitive to truncation; audio encoders may be sensitive to accents and noise. Governance needs these failure profiles because they determine when to demand verification.

(4) Fusion mechanism. Fusion combines modality-specific representations into a joint state. Conceptually, fusion is where the system decides "what matters." It can be explicit (cross-attention, gating, weighted averaging) or implicit. The frontier risk is that fusion can overweight a misleading modality, producing what feels like grounded certainty. For example, a model might treat an OCR-extracted number as exact even though the OCR confidence is low, or treat a visual cue as decisive even when the text contradicts it.

(5) Reasoning and planning module. This component produces inferences, explanations, and candidate plans. It is where the system constructs a narrative that ties evidence to conclusions and decides which actions are needed. The planning aspect becomes crucial when the system can call tools iteratively, request more information, or execute multi-step workflows. Conceptually, this module transforms evidence into intent.

(6) Tool and action layer. Tools include retrieval, browsers, code execution, database queries, workflow engines, and UI automation. Actions include reading records, writing updates, sending messages, and triggering external processes. The tool layer is where the system's influence becomes operational. It is also where security boundaries matter most: what is allowed, what is logged, and what requires approval.

(7) Policy enforcement and gating. This layer implements π: least privilege, allowlists, rate limits, escalation thresholds, and irreversible-action gates. It can be implemented as middleware, rule engines, or structured workflows. The important conceptual point is that policy enforcement must be *external* to the model's internal preferences. Governance cannot rely on the model's willingness to behave; it must constrain what the model can do.

(8) Environment and feedback channels. The environment includes enterprise systems, user interactions, and the external world. Feedback can come from user corrections, system monitors, tool responses, and outcome measurements. Feedback closes the loop: the system observes the effect of its actions and updates its subsequent behavior. This is where trajectory evaluation becomes necessary: a system that recovers gracefully from uncertainty is very different from one that escalates

errors.

These entities interact in a recurring loop:

$$\text{Perceive} \rightarrow \text{Extract} \rightarrow \text{Fuse} \rightarrow \text{Infer/Plan} \rightarrow \text{Act} \rightarrow \text{Observe}.$$

In governance-first terms, the loop must be instrumented with *trace points*. Each stage should produce artifacts that are reviewable: extracted text with confidence, evidence references, tool-call logs, action requests, and approval records. Without trace points, the loop becomes opaque, and the organization cannot separate model error from integration error, nor can it diagnose whether a failure was caused by untrusted input, a preprocessing bug, or an overly permissive tool policy.

The key interaction to highlight is **content–control entanglement**. In multimodal systems, content can include instructions, and instructions can shape tool use. A document may contain text that looks like policy, a screenshot may contain a prompt-like directive, or an image overlay may embed instructions that the model interprets as user intent. If the system does not maintain a strict separation between (i) trusted control signals (system prompts, user requests, policy rules) and (ii) untrusted content (documents, images), then untrusted content can hijack control flow. This is the defining security interaction for VLA systems.

Another crucial interaction is **evidence–authority coupling**. The system's confidence in its evidence influences its willingness to act. If the system does not explicitly represent uncertainty, it may treat low-quality evidence as reliable and proceed to action. Conversely, if uncertainty is represented but not connected to gating, the system may still act despite recognizing ambiguity. Therefore, uncertainty must be coupled to policy: high-impact actions require low uncertainty or explicit approval.

Finally, there is a **human–system interaction** layer that is often overlooked. Users supply prompts, interpret outputs, and approve actions. UI design can implicitly encourage over-trust: a clean summary with a screenshot can feel more authoritative than it deserves. Governance must account for human biases by requiring the system to show evidence references, to surface uncertainty, and to make action boundaries explicit. In multimodal systems, human factors are part of the technical system because they determine whether errors are caught or propagated.

5.2.3 What is being optimized or controlled

A multimodal frontier system is typically deployed to maximize some notion of utility: faster workflows, reduced manual effort, improved consistency, better detection of anomalies, or improved decision support. Yet utility in regulated or high-accountability contexts is never unconstrained. The organization must respect privacy, security, safety, and legal obligations. Therefore the correct

conceptual framing is *optimization under constraints*. A generic form is:

$$a^* \;=\; \arg\max_{a \in \mathcal{A}} \; \mathbb{E}\big[U(a) \mid X\big] \quad \text{s.t.} \quad C(a, X) \le \kappa.$$

Here $U(a)$ measures task value (e.g., correct extraction, correct routing, reduced time), while $C(a, X)$ captures constraint violations (privacy exposure, unauthorized access, unsafe action, policy noncompliance). The parameter κ represents an institutional risk tolerance, which in many settings should be interpreted as "near zero" for certain classes of violation.

This framing reveals a subtle but critical point: if the system is trained or tuned primarily to maximize perceived usefulness, it will naturally move toward actions that appear to complete tasks quickly. Without explicit constraints, the system will not discover the organization's hidden rules. Governance-first design therefore requires that constraints be made explicit in the action policy π and that the system be evaluated on constraint adherence, not just task completion.

In practical systems, what is being controlled is not only the final action but the *process by which actions are proposed*. A useful decomposition of control objectives includes:

(1) Robust perception. The system should interpret multimodal inputs reliably under realistic noise. Control here includes preprocessing validation, confidence estimation, and redundancy checks. The goal is not perfect perception, but bounded error under known conditions and graceful handling of ambiguity.

(2) Robust fusion. The system should combine modalities in a way that is stable and not dominated by spurious cues. Control here includes forcing cross-checks (e.g., reconcile OCR text with visual layout, reconcile table totals with line items), enforcing consistency constraints, and detecting contradictions that trigger escalation.

(3) Evidence-grounded reasoning. The system should not merely produce plausible narratives; it should tie claims to evidence. Control here includes requiring evidence references for material claims, maintaining claim-to-evidence maps, and preventing the system from asserting unverified facts when the evidence is missing.

(4) Safe action boundaries. The system should propose actions only within permitted authority and should request human approval when actions are high-impact or irreversible. Control here includes least privilege, allowlists, rate limits, staged execution, and rollback plans.

(5) Privacy and minimization. The system should not unnecessarily retain or expose sensitive information. Control here includes redaction, hashing, selective logging, and ensuring that outputs are appropriately scoped to the recipient's permissions.

(6) Auditability and reproducibility. The system should produce traces that allow reconstruction of decisions. Control here includes deterministic pipelines where feasible, versioning of preprocessors, logging of tool calls, and structured output formats.

These objectives are interdependent. For instance, privacy minimization can conflict with auditability if raw evidence cannot be retained. A governance-first approach resolves this by designing logs that preserve references and hashes rather than storing full media, and by storing sensitive artifacts in controlled evidence vaults rather than in model logs. Similarly, robust perception can conflict with latency and cost, because additional verification steps require extra computation. Governance-first design resolves this by tying verification to risk: low-impact tasks can proceed with lighter checks, while high-impact actions require heavier verification and approval.

A key frontier issue is that the model itself may implicitly optimize for *coherence* rather than *truth*. Coherence is a useful property for communication, but it becomes dangerous when it substitutes for evidentiary integrity. Therefore, an explicit control goal should be to align system incentives with *epistemic discipline*: the system must prefer "I do not know" plus a verification request over a confident guess when evidence is ambiguous. This can be implemented by gating and by evaluation, not by hoping the model behaves.

Another frontier control question concerns **authority separation**. In well-designed systems, the model proposes actions, but an external controller executes them. This separation ensures that even if the model is manipulated, it cannot bypass policy. The conceptual rule is: *the model may suggest; the system must enforce*. This is the same principle that governs safety-critical software: untrusted components do not hold authority. In multimodal VLA systems, this separation is often violated for convenience. The chapter's conceptual abstraction emphasizes that convenience is not a governance argument.

Finally, a mature control objective must include **deferment**. There must be conditions under which the system refuses to act or is restricted to read-only modes. These conditions are not failures; they are safety achievements. In governance-first deployment, deferral is a designed outcome when uncertainty, injection risk, or privacy constraints cannot be managed. The control objective is therefore not maximal automation, but maximal value within defensible constraints.

5.2.4 Distinction from prior paradigms

The temptation is to treat multimodal systems as "NLP plus images." That framing is insufficient because it misses what changes qualitatively when perception and action are introduced. In single-modality NLP settings, the primary uncertainties come from language ambiguity and knowledge gaps. The input is typically text supplied by a user or retrieved from a database. While adversarial prompt injection exists, it tends to occur within the text channel and can be mitigated by prompt hygiene and instruction hierarchy.

Multimodal systems differ in four fundamental ways.

First, perception introduces measurement error. Images and audio are not symbolic inputs; they are noisy measurements of a world. Even a scanned document is a measurement: resolution,

skew, lighting, compression, and artifacts all affect extraction. This makes multimodal systems resemble sensor systems. In sensor systems, calibration, uncertainty modeling, and redundancy are standard. In multimodal AI deployments, these disciplines are often absent because the output is fluent text that hides uncertainty.

Second, evidence is spatial and temporal, not purely textual. In NLP, evidence can often be pointed to as a quote or a passage. In multimodal systems, evidence may be a region of a page, a line in a table image, or a timestamp in audio. This complicates provenance and audit. It also complicates user review because users cannot easily verify claims unless the system provides precise references. Therefore, the conceptual notion of "citation" must be generalized into coordinate-based evidence mapping.

Third, the adversarial surface expands dramatically. In NLP, an adversary can craft text prompts. In multimodal systems, an adversary can craft documents, layouts, images, or mixed-media artifacts that exploit model shortcuts or injection pathways. A contract might contain a paragraph designed to be misread by OCR. An invoice might use typography that confuses digit recognition. An image might contain overlay text that instructs the system to ignore policy. These are not exotic; they are natural extensions of how adversaries already manipulate human workflows. The difference is that a model may follow such instructions more literally and without the contextual skepticism a trained human might apply.

Fourth, action coupling makes failures operational. In NLP-only systems, the output is often a suggestion. In VLA systems, the output can be a tool call or an action request. The system can become the locus of authority rather than a mere advisor. This means that errors can propagate faster and become harder to unwind. In regulated environments, this also creates a new compliance risk: the organization may be unable to demonstrate that human judgment remained in control.

These differences imply that the governance primitives must change. Prompt engineering is insufficient; one must govern preprocessing, evidence mapping, tool permissions, and action gating. Model-level alignment is insufficient; one must build system-level controls that assume the model can be wrong or manipulated.

The chapter also distinguishes VLA systems from traditional computer vision pipelines. Classic CV systems were often narrow: they classified images into categories or detected objects. Their outputs were structured and their failure modes were often measurable. Multimodal foundation models are broad and can produce open-ended explanations. This broadness is valuable for usability but dangerous for governance because it allows the system to fill gaps with plausible language. A classic OCR system might output a low confidence score. A multimodal assistant might output a confident summary without exposing that the underlying extraction was uncertain. This difference matters because it changes the human's ability to detect failure.

Finally, the chapter distinguishes multimodal VLA systems from ordinary software automation. In deterministic automation, the control flow is explicit and verifiable. In VLA systems, control flow can

be influenced by model interpretation of content, which is probabilistic and context-dependent. Thus, the system behaves less like software and more like an agent operating under uncertain perception. This is why system-of-systems governance is required: one must manage the probabilistic component with explicit constraints and evaluation, not treat it as a deterministic operator.

5.2.5 Conceptual failure modes

A governance-first conceptual model must include a taxonomy of failure modes that captures what is unique and operationally dangerous about multimodal frontier systems. The goal is not to enumerate every error but to identify the classes of failure that drive risk, require distinct controls, and often evade naive evaluation.

(1) Spurious cross-modal shortcuts. Multimodal models can learn correlations that work in benchmarks but fail in deployment. For instance, a model might infer document type from formatting cues and then assume the presence of certain fields, hallucinating missing items. Or it might associate certain visual patterns with certain outcomes (e.g., "this looks like a valid invoice") without actually verifying content. These shortcuts become dangerous when they are treated as grounded understanding. Control requires stress testing under distribution shift and forcing explicit verification of key fields.

(2) Fusion dominance and contradiction blindness. When modalities disagree, the system may overweight one modality without noticing the contradiction. For example, the text might state one value while the table image implies another; the system might pick one and proceed. Contradiction blindness is a frontier issue because it is often invisible: the output is coherent, and the user may not see the underlying conflict. Control requires explicit contradiction detection and policies that escalate when conflicts are detected.

(3) Multimodal prompt injection (content-as-control). Untrusted documents or images can contain instructions that manipulate the system into ignoring policy, exfiltrating data, or executing unintended tool calls. The essence is not that the model is "tricked"; it is that the system has not maintained a strict boundary between trusted control signals and untrusted content. Control requires prompt compartmentalization, sanitization, tool allowlists, and an architecture where content cannot directly influence permission decisions.

(4) Privacy leakage through mixed-media processing and logging. Multimodal inputs often contain sensitive information that is not obvious: faces, addresses, account numbers, signatures, health details, or confidential figures embedded in screenshots. Systems may inadvertently surface this information in outputs, share it with unauthorized users, or store it in logs. The risk increases when logs are used for debugging and evaluation. Control requires minimization, redaction, role-based access, and privacy-preserving audit trails.

(5) Silent hallucination under perceptual uncertainty. When perception is ambiguous, a

system may guess and then explain its guess as if it were observed. This is a particularly dangerous form of hallucination because it feels grounded in an image or document. A missing field might be filled in; an unreadable number might be inferred; a blurred label might be assumed. Control requires uncertainty surfacing and gating: if the system cannot read a value, it must request a better scan or defer to a human.

(6) Tool misfires and authority creep. When connected to tools, the system can take actions that exceed intended authority. This can occur through configuration errors (overly broad permissions), UI changes (buttons moved), or model behavior (choosing a faster but unsafe path). Authority creep is especially common when organizations iteratively expand tool access to improve performance. Control requires least privilege, staged rollouts, approval gates, and continuous permission reviews.

(7) Evidence-provenance collapse. Outputs may combine information from multiple sources without tracking which source supported which claim. In multimodal settings, the risk is compounded because evidence references are harder. Provenance collapse undermines accountability: when challenged, the organization cannot demonstrate what evidence was used. Control requires structured evidence maps, reference IDs, and claim-to-evidence linking.

(8) Human over-trust induced by "visual grounding." Users may over-trust outputs because they are associated with images or documents. A system that quotes a figure "from the chart" can appear more credible than it is. If the UI does not surface uncertainty and evidence references, reviewers may accept outputs without verifying. Control requires interface design that makes uncertainty and evidence explicit and that prevents the system from presenting guesses as observations.

(9) Feedback loop amplification. Once actions change the environment, errors can be reinforced. A system might write an incorrect classification into a database, and then later retrieve that classification as ground truth. Or it might trigger a workflow that generates additional documents reflecting the error. This creates self-reinforcement: the system's mistakes become part of the evidence. Control requires write restrictions, human review for updates, and separation between model outputs and authoritative records.

(10) Distribution shift in the real world. Enterprise documents are messy: scans vary, templates change, counterparties behave adversarially, and contexts differ. Models trained on curated datasets can fail unexpectedly. Distribution shift is not unique to multimodal systems, but it is more severe because the space of visual and layout variation is large and because tool coupling can turn a minor misclassification into a major operational failure. Control requires continuous monitoring, synthetic stress suites, and conservative deployment boundaries.

These failure modes illustrate the chapter's conceptual thesis: multimodal VLA systems are governed not by clever prompting but by system design choices that maintain boundaries, preserve evidence integrity, and constrain authority. The conceptual abstraction therefore culminates in a simple operational rule: *treat multimodal inputs as untrusted sensors, treat the model as a fallible interpreter,*

and treat tool access as delegated authority that must be explicitly bounded and audited.

5.3 Historical and Technical Lineage

5.3.1 Preceding approaches

Multimodal frontier systems did not emerge from a single breakthrough; they are the product of several mature technical traditions that evolved largely in parallel before being fused into today's vision–language–action (VLA) stacks. The most important predecessors are classical computer vision (CV), optical character recognition (OCR) and document analysis, automatic speech recognition (ASR), and earlier generations of multimodal fusion systems. Understanding these lineages matters because their assumptions—and their failure modes—were inherited by modern systems, even when the interface now looks radically different.

Classical computer vision began as a pipeline discipline. Systems were decomposed into feature extraction, detection, segmentation, and classification. Engineers reasoned about edges, textures, keypoints, and later convolutional representations, often with explicit inductive biases. Crucially, these systems typically produced structured outputs: bounding boxes, labels, confidence scores, or masks. The structured nature of outputs made it possible to build downstream safeguards in a relatively direct way. For example, a safety system could be designed to trigger only when multiple detectors agreed, or to refuse action when confidence was below a threshold. Even when performance was imperfect, the pipeline structure encouraged explicit treatment of uncertainty and error propagation.

OCR and document understanding evolved along a different axis. Early OCR focused on recognizing characters under constrained conditions. As documents became more complex, systems expanded into layout analysis: detecting lines, tables, headers, footnotes, signatures, stamps, and multi-column formats. Document AI emerged as a practical engineering domain, especially in finance, insurance, and government, where form-like artifacts are everywhere. Again, the outputs were often structured: extracted fields with confidence, coordinates, and sometimes rule-based validation (e.g., checksums, formats, cross-field consistency rules). Many organizations built mature quality assurance workflows around OCR: human review queues for low-confidence extractions, template management, and exception handling.

ASR and audio analytics formed another predecessor. Speech recognition systems had to manage noise, accents, speaker overlap, and domain-specific terminology. They frequently exposed word-level confidence or alternative hypotheses. In regulated environments such as call centers and trading floors, ASR systems were often integrated with compliance controls, retention policies, and audit trails. The discipline of handling sensitive audio data also forced organizations to confront privacy and governance questions early: who can access transcripts, how long recordings are retained, and how outputs are used in decisions.

Finally, early multimodal fusion systems were typically narrow. A captioning model fused image features and language generation. A visual question answering system fused image and text to

answer constrained questions. A speech+text system might fuse ASR output with downstream NLP. These systems made two implicit assumptions: (i) the modality fusion problem was task-specific and could be engineered with targeted objectives, and (ii) the system's role was primarily interpretive—it would output a label, an answer, or a caption, not act as an operator in a workflow. This is an important historical point: earlier multimodal systems were frequently treated as "front-end" components that improved user experience, not as authority-bearing decision participants.

Across all these predecessors, one idea dominated: *decomposition*. Systems were split into modules with explicit interfaces. The rise of foundation models changed the interface, but it did not eliminate the underlying reality that multimodal systems still have modules: preprocessors, encoders, fusion, reasoning, and tool layers. The danger is that the modern interface hides this decomposition and encourages teams to treat the system as a single model. When that happens, the governance lessons of the predecessor domains are forgotten precisely when they are most needed.

5.3.2 Key inflection points

The contemporary VLA paradigm is the result of a sequence of inflection points that gradually moved multimodal AI from narrow perception tasks to general-purpose, instruction-following, tool-augmented systems. The first inflection point was the emergence of *joint embedding spaces*. Instead of treating images and text as separate domains, systems learned to align their representations so that semantically related images and phrases were close in latent space. This enabled retrieval-like behavior: given a text query, retrieve relevant images; given an image, retrieve relevant descriptions. Joint embeddings were not only a modeling trick; they were a conceptual shift. They implied that images could be treated as another language-like source of meaning, and that cross-modal transfer was feasible at scale.

A second inflection point was the transformer architecture and attention-based fusion. Attention mechanisms provided a natural way to integrate modalities by allowing tokens (or visual patches) in one modality to attend to features in another. This reduced the need for handcrafted fusion heuristics and enabled end-to-end training on large datasets. It also expanded the kinds of tasks that were feasible: not just retrieval or captioning, but multi-step reasoning over images, text, and structured representations. Importantly, transformer-based fusion supported the idea of *context*: the system could condition on long textual instructions while simultaneously grounding on visual inputs. This laid the groundwork for instruction-following multimodal assistants.

A third inflection point was *scaling*—not only model size, but dataset diversity and training regimes. Large-scale pretraining on broad, noisy datasets produced models that were robust enough to handle a range of natural images and documents. This scaling shift is often described as a capability story, but it also had an organizational consequence: multimodal models became general enough to be used as platforms. Teams no longer needed to build a different CV system for each task. Instead, they could adopt a single multimodal model and prompt it, fine-tune it, or wrap it with tools. This

platformization is a major driver of deployment and also of risk, because platform systems tend to accumulate authority across many workflows.

A fourth inflection point was *instruction tuning* and preference optimization applied to multimodal models. Earlier vision models were not conversational. They produced outputs for a fixed task. Instruction tuning transformed them into assistants that could accept open-ended directives and respond in a helpful, structured way. This matters because enterprises do not want a model that outputs a probability vector; they want a system that can explain, justify, and adapt to context. Instruction tuning also made it natural to connect the system to tools, because the system could now interpret tool results and plan next steps. The combination of multimodal perception and instruction-following reasoning is what makes the system feel like an operator rather than a classifier.

A fifth inflection point was the rise of *tool-augmented workflows*. Tool use is not new—pipelines have always called other systems—but the novelty is that language-model-like systems can select tools dynamically, interpret results, and decide what to do next. When a multimodal assistant can read a document, extract entities, query a database, and draft an email, the boundary between perception and business process collapses. The system becomes a workflow participant. This is where the vision–language–action framing becomes concrete: the system perceives (reads, watches, listens), reasons (summarizes, reconciles, plans), and acts (calls tools, writes updates, triggers workflows).

A sixth inflection point is *interface operation*. Many organizations cannot easily expose every internal workflow through clean APIs. Instead, they consider AI systems that operate existing interfaces: reading screens, clicking buttons, filling forms. Multimodal models, by virtue of their ability to interpret images and text, can serve as the perception engine for such UI operation. This is a profound shift because it makes tool use less explicit and therefore harder to govern. UI operation also increases brittleness: a small UI change can alter system behavior. Yet it is attractive because it reduces integration cost and accelerates automation.

Finally, a seventh inflection point is growing awareness of *adversarial and governance failure modes* specific to multimodal systems. As deployments expanded, practitioners began to recognize that documents and images could carry instructions that manipulate systems, that OCR and extraction errors could silently propagate, and that logging mixed-media evidence creates privacy hazards. This awareness is driving an emerging ecosystem of defenses: content sanitization, prompt compartmentalization, least-privilege tool access, and audit logging frameworks. The field is still early, but the inflection is conceptual: governance is becoming part of the technical design space, not an afterthought.

These inflection points collectively explain why VLA systems feel new: the system is not merely a better perception model. It is a general interface layer between messy real-world artifacts and operational decision processes. The lineage also explains why the governance challenge is severe: the system inherits the uncertainty of perception, the persuasion of language, and the authority of tools.

5.3.3 What persisted vs what broke

Despite the apparent discontinuity introduced by foundation models, several realities persisted from earlier eras, and several intuitions broke. Distinguishing the two is essential for governance because it prevents both complacency and overreaction. What persisted is the physics of information: perception is noisy, distribution shift is severe, adversaries exist, and uncertainty must be managed. What broke is the assumption that multimodal systems are merely front-end interpreters with bounded impact.

The first persistent reality is **noisy perception**. Images are ambiguous, documents are messy, and audio is imperfect. A high-performing vision–language model does not change the fact that a blurry scan cannot reliably encode fine print. The difference is that the system can now produce a fluent interpretation even when the input is inadequate. In earlier OCR systems, low-quality scans often triggered explicit failures or low-confidence outputs. In modern multimodal assistants, the failure may be masked by plausible language. Thus, the noise persists, but the system's interface makes it easier to ignore.

The second persistent reality is **distribution shift**. Real-world documents vary across counterparties, time, jurisdictions, and business units. Form templates change. Logos and layouts change. Scanning practices vary. In vision tasks, lighting and angle vary. In audio tasks, accents and background noise vary. This diversity is not a nuisance; it is the normal condition. Many models perform well on curated benchmarks but fail on edge cases that are common in operational settings. Distribution shift persists, and in some respects becomes more consequential because the system is now used across more workflows, increasing the surface area of possible shift.

The third persistent reality is **error propagation**. In a pipeline, early errors propagate downstream unless caught. This remains true. The difference is that modern systems are often designed as seemingly end-to-end models, leading teams to forget that preprocessing and extraction are still separate layers, even if hidden. OCR errors still occur; layout parsing still fails; image normalization still matters. The propagation pattern persists, but the visibility of intermediate states often decreases.

The fourth persistent reality is **adversarial pressure**. In domains like finance, insurance, and compliance, actors have incentives to manipulate inputs. Fraudsters craft documents to evade detection. Counterparties present terms in confusing ways. Users may attempt to elicit unauthorized actions. Adversarial pressure persisted even before multimodal AI; what changes is that a model may be more easily manipulated by content-as-instruction attacks unless the system architecture enforces separation of control and content.

What broke first is the **front-end only assumption**. Historically, multimodal systems were often deployed as interpretive tools: OCR extracted text; CV detected objects; ASR transcribed speech. Decisions were made downstream by humans or deterministic software. With VLA systems, the assistant itself can propose and execute workflow steps. This breaks the comfortable notion

that perception is a passive upstream component. Instead, perception becomes entangled with decision-making and authority.

What broke second is the **structured-output discipline**. Earlier systems often produced structured artifacts with confidence and coordinates, which supported auditing and quality control. Modern assistants often produce unstructured narratives, and even when they extract structured fields, they may not expose confidence or evidence mapping unless explicitly engineered. The interface is more human-friendly but less governance-friendly by default. This breaks a key operational habit: verifying intermediate outputs.

What broke third is the **separation between input and instruction**. In classic software, untrusted input is data, not control. In language-model systems, input often contains instructions, and the model is designed to follow them. In multimodal systems, the untrusted input can carry instructions in visual or document form. This breaks older security models that assumed a clear boundary between code and data.

What broke fourth is the assumption that **model-level alignment is sufficient**. In text-only chatbots, alignment and content filters can address many obvious unsafe outputs. In VLA systems, the main risks often arise from system integration: tool permissions, logging, retention, and action gating. A well-aligned model can still leak sensitive data if the system logs raw documents or if the tool layer allows broad database queries. This breaks the intuition that safety can be handled entirely at the model layer.

The practical implication of "persisted vs broke" is a governance posture that is both conservative and precise. Conservative, because perception remains uncertain and adversaries remain present. Precise, because the new risks are not mystical; they stem from broken assumptions about authority, structure, and control boundaries. A mature organization responds by reinstating the disciplines that worked in predecessor domains (confidence handling, provenance, validation, least privilege), while adapting them to the new reality that the assistant's outputs can now drive actions.

5.3.4 Why older intuitions fail

Older intuitions fail primarily because they were built for systems where (i) outputs were evaluated in isolation, (ii) the model was not an operator with tool access, and (iii) security assumptions treated content as inert. VLA systems violate all three conditions. As a result, relying on legacy mental models can produce deployments that look impressive but are fragile, unsafe, or indefensible under audit.

The first failing intuition is that **benchmark accuracy implies operational safety**. Traditional CV benchmarks measure classification accuracy or detection performance under controlled conditions. Document AI benchmarks may measure extraction accuracy on standardized datasets. These metrics are valuable, but they do not measure what institutions need: the safety of end-to-end decisions. A

model can achieve high accuracy on object detection and still be unsafe when its outputs trigger actions. For instance, a small error rate might be acceptable when a human reviews every output, but catastrophic when actions are automated. Moreover, benchmark distributions rarely include adversarial artifacts, rare formats, or real-world noise that matters in enterprise settings. Thus, CV benchmarks are necessary but insufficient, and treating them as proxies for safety is a category error.

The second failing intuition is that **confidence is meaningful and calibrated**. Many classic systems produced confidence scores that were at least interpretable within a domain. Modern foundation models often produce fluent outputs without explicit calibration. Even when a model can output confidence-like signals, they may not correspond to true probability of correctness. In multimodal settings, calibration is especially hard because uncertainty arises from multiple sources: sensor noise, extraction errors, ambiguous context, and model inference uncertainty. Older intuitions about using confidence thresholds to gate actions therefore fail unless the system is explicitly designed and validated for calibration across modalities.

The third failing intuition is that **security is primarily about code, not content**. Traditional security models focus on preventing injection into code execution contexts. In VLA systems, content can influence control flow because the model interprets content as instructions. A scanned document can contain a directive that the model treats as higher priority than the user's instruction. An image can contain hidden text or layout tricks. Older security intuitions that treat documents as passive data fail because the assistant is trained to follow natural language instructions wherever they appear. Therefore, security must be reconceptualized as protecting the *control plane* of the system: ensuring that untrusted content cannot alter permissions, tool selection, or action execution.

The fourth failing intuition is that **the model is the product**. In many organizations, procurement and governance processes focus on selecting a model vendor and evaluating the model's capabilities. For VLA systems, the product is the system: preprocessing, tool stack, UI, logging, permissions, and human workflows. A strong model with weak system controls is unsafe; a weaker model with strong controls can be acceptable for constrained tasks. Thus, older procurement and risk practices fail if they treat the model as the primary unit of governance.

The fifth failing intuition is that **interpretability or explainability naturally follows from multimodality**. There is a common belief that if a system uses images or documents, its outputs are grounded and therefore explainable. In reality, grounding is not the same as traceability. A system can reference an image without providing a verifiable link between a claim and a specific evidence region. It can also hallucinate content that appears visually plausible. Older intuitions about explainability (e.g., "it saw the document, so it must be right") fail because the system's internal fusion and reasoning are not inherently transparent.

The sixth failing intuition is that **automation is a linear extension of assistance**. Teams often start with a system that summarizes documents and then gradually add tool access to reduce

manual work. The intuition is that this is a smooth progression: once the system is helpful, it can be made autonomous. In reality, autonomy changes the risk regime discontinuously. The same error that was acceptable in an assistive setting becomes unacceptable when it triggers state changes. Moreover, tool access introduces new attack surfaces. Therefore, older intuitions about incremental automation fail unless each step is treated as a governance escalation with explicit re-evaluation.

The seventh failing intuition is that **human review naturally catches errors**. In multimodal systems, outputs can be persuasive and grounded-seeming. Reviewers may be biased toward accepting outputs, especially under time pressure. If the system does not present uncertainty and evidence references clearly, human review becomes performative rather than substantive. Older intuitions that "a human is in the loop" therefore fail unless the loop is designed to support real verification: showing what evidence was used, what was uncertain, and what checks were performed.

The cumulative lesson is that older intuitions fail because the object of governance changed. In prior paradigms, governance could focus on model accuracy and output reasonableness. In VLA systems, governance must focus on system boundaries, authority, traceability, and end-to-end evaluation under adversarial and privacy-constrained conditions. The chapter emphasizes this not to discourage multimodal deployment, but to make the conditions of safe value creation explicit.

5.3.5 Inherited lessons

Even though older intuitions fail, older disciplines remain valuable. The lineage of security engineering, safety engineering, and high-reliability systems provides a set of lessons that translate directly into the governance of multimodal frontier systems. The core message is that modern VLA systems are not exempt from classical operational wisdom; they demand it.

From security engineering, the primary inherited lesson is: **assume adversarial inputs**. In practice, this means treating external documents, images, and attachments as untrusted. The system must not let untrusted content influence privileged behavior. This principle leads to several design imperatives: compartmentalize prompts so that system instructions and policy rules are not mixed with document content; sanitize extracted text to remove instruction-like patterns; and ensure that tool policies are enforced externally rather than decided by the model. Security engineering also teaches that controls must be layered. No single filter will catch all injection attempts. The system should combine allowlists, anomaly detection, rate limiting, and mandatory approvals for sensitive actions.

A related security lesson is **least privilege**. Tool access is authority. Therefore, each tool should have minimal scope, and permissions should be segmented by task and role. A model that reads invoices does not need the ability to modify payment details. A model that summarizes contracts does not need access to HR records. Least privilege is not only a technical control; it is a governance posture that forces organizations to explicitly state what authority they are delegating to an AI system. In multimodal contexts, least privilege also applies to data visibility: the system should not

be given access to raw sensitive media if a redacted representation is sufficient.

From safety engineering, the foundational inherited lesson is: **actions require gating**. In safety-critical systems, it is unacceptable to let a fallible component directly execute irreversible actions. The equivalent in VLA systems is to separate proposal from execution. The model can propose an action plan, but an external controller decides whether the plan is allowed, whether additional verification is needed, and whether a human approval is required. Gating should be risk-based: low-impact actions (e.g., drafting a suggestion) can proceed automatically, while high-impact actions (e.g., filing, sending external communications, changing records) require explicit review. Safety engineering also teaches the value of **fail-safe defaults**. When uncertain, the system should defer or request clarification rather than guess and act.

Safety engineering further emphasizes **redundancy and cross-checks**. In perception systems, redundancy can mean using multiple extraction methods, validating totals against line items, or reconciling OCR text with visual layout cues. The goal is not to make the model perfect, but to reduce the probability of silent catastrophic error. This is particularly important because multimodal models can produce plausible outputs even when wrong. Cross-checks create friction that surfaces errors.

From high-reliability operations and incident response, the inherited lesson is **auditability and replay**. When a failure occurs, the organization must reconstruct what happened. This requires logs that capture inputs (or privacy-preserving references), transformations, tool calls, decisions, and approvals. The system should be designed so that a specific run can be replayed under the same configuration, enabling diagnosis. This is not optional in regulated contexts. Without replayable traces, the organization cannot demonstrate control, cannot learn from failures, and cannot defend itself under audit or litigation.

Another inherited lesson is **change control**. In predecessor domains, changing OCR templates, CV models, or ASR dictionaries required careful validation because small changes could have large downstream effects. In VLA systems, changes are even more consequential because the system is integrated with tools and workflows. Updating a model, changing a prompt policy, or modifying a tool permission can alter behavior across many tasks. Therefore, changes must be versioned, evaluated, and approved. This includes changes to preprocessors, retrieval corpora, and UI automation scripts. The system-of-systems nature makes change control more complex, but also more necessary.

From privacy engineering and compliance, the inherited lesson is **data minimization and retention discipline**. Multimodal inputs are rich in sensitive information. Logging and storing raw artifacts can create liability. Therefore, the system should log the minimum necessary for accountability: hashes, redacted snippets, coordinate references to secure evidence vaults, and structured summaries that avoid exposing sensitive details. Retention policies must be explicit. Access to logs must be controlled. In many cases, the safest design is to separate operational logs from raw evidence storage,

and to require explicit authorization to retrieve raw evidence during audits.

Finally, from human factors engineering, the inherited lesson is **do not outsource vigilance to the user**. Interfaces should be designed to support correct oversight. The system should surface uncertainty, show evidence references, and clearly label whether outputs are verified, inferred, or requested. Approval flows should be structured so that reviewers can see the basis of an action without being overwhelmed. The goal is to make oversight realistic under time pressure, not idealized.

Taken together, these inherited lessons provide a governance blueprint that is both conservative and actionable. They also clarify why this chapter belongs as the capstone of *AI 2026*. The frontier is not defined by model novelty alone. It is defined by the return of classical engineering truths in a new context: uncertainty must be managed, authority must be bounded, adversaries must be assumed, and accountability must be engineered. Multimodal VLA systems are simply the newest domain in which these truths become unavoidable.

5.4 Technical Foundations

5.4.1 System or architectural components

A multimodal vision–language–action (VLA) system is best understood as a layered architecture that converts heterogeneous evidence into both communicative outputs and operationally meaningful actions. The technical foundations are therefore not a single neural network, but a *stack* in which each layer introduces its own failure modes and governance requirements. A useful decomposition begins with the perception pipeline, proceeds through representation and reasoning, and ends with an enforcement boundary that governs authority.

(1) Input interfaces and ingestion. The system must accept multiple modalities: raw images, PDFs, scanned documents, screenshots, audio recordings, plain text, and structured data such as tables or JSON records. In a production setting, ingestion is not trivial. Inputs arrive from channels with different trust levels and metadata: internal databases, external counterparties, customer uploads, or user screenshots. The ingestion layer attaches provenance attributes (source identity, time, retention classification, access labels) and determines whether the artifact can be processed at all under policy constraints. In many organizations, this layer is the first place where governance is either enforced or silently bypassed.

(2) Preprocessing and extraction. Before a foundation model sees anything, raw media is often transformed. For documents this may include OCR, layout parsing, table detection, header/footer removal, and normalization of fonts or rotation. For images this may include resizing, de-noising, cropping, contrast adjustments, and detection of regions of interest. For audio this may include speech-to-text (ASR), diarization, noise filtering, and time alignment. These preprocessors behave like sensors: they convert the world into a representation the model can consume. They must be versioned and validated because small changes can materially alter extracted content. A common governance failure is to treat these transformations as "implementation details" and omit them from audit logs, making it impossible to reconstruct why a particular run produced a given output.

(3) Modality encoders. Each modality is encoded into latent representations. Architecturally, this can take the form of separate encoders (e.g., vision transformer for images, text transformer for text) that feed into a fusion module, or a unified encoder that consumes multimodal tokens. The governance-relevant point is that each encoder has distinct sensitivities. Vision encoders can be brittle to blur, occlusion, and adversarial overlays. Text encoders can be brittle to truncation, tokenization artifacts, and prompt injection. Audio encoders can be brittle to accents, background noise, and domain-specific vocabulary. In a system-of-systems view, encoder behavior is a key source of uncertainty that must propagate into downstream gating.

(4) Fusion layer. Fusion combines modality-specific representations into a joint state. Technically, fusion can be achieved via cross-attention, gating networks, concatenation with learned mixing, or hierarchical fusion (e.g., fuse page-level representations, then document-level). Fusion is the

point at which the system implicitly decides which modality dominates. This is where cross-modal contradictions can be missed and spurious cues can be overweighted. In enterprise document intelligence, fusion often also includes *layout priors*: where information appears on the page strongly influences interpretation. This is powerful but risky, because layout varies across counterparties and can be manipulated.

(5) Decoder, reasoner, and planner. The decoding stage produces output text, structured extractions, and intermediate reasoning steps. In VLA systems, the "decoder" is often also the reasoner: it interprets the fused state and generates a plan, potentially over multiple steps. When tool use is integrated, the planner decides which tools to call, how to interpret results, and whether to continue. This is where the system's behavior becomes trajectory-based. The failure modes include persuasive but unsupported narratives, insufficient uncertainty signaling, and unsafe sequences of tool calls.

(6) Tool stack and action executors. The tool layer is typically a heterogeneous set of components: retrieval from knowledge bases, database queries, web browsing, code execution, OCR re-runs, validation functions, and enterprise workflow APIs. In some deployments, tools include UI automation (reading screens and clicking) rather than explicit APIs. The tool layer introduces operational authority. Therefore, each tool must have explicit permissions, scope, rate limits, and logging. A frequent mistake is to treat tool calls as harmless because they are "just fetching information," while in practice they may expose sensitive data or enable exfiltration pathways.

(7) Policy enforcement and governance middleware. This is the most important architectural component from a governance standpoint. Policy enforcement implements least privilege, allowlists, redaction rules, approval gates, and escalation thresholds. It should be external to the model. The model may propose an action; the policy layer decides whether it is allowed. This separation prevents untrusted content from directly influencing authority. The policy layer also enforces privacy controls: it decides what can be logged, what must be redacted, and what evidence can be stored or referenced. In a mature deployment, the policy layer is not a set of ad hoc rules; it is a controlled artifact with versioning, testing, and audit trails.

(8) Observability, tracing, and evidence vaults. Because multimodal artifacts can be sensitive and large, systems typically require a structured approach to logging. Logs should capture tool calls, decisions, approvals, and references to evidence without necessarily storing raw media. This often implies a separate evidence vault where raw inputs are stored under strict access control, while operational logs store hashes, coordinate references, and minimal excerpts. Observability also includes monitors for anomalies (e.g., repeated injection attempts) and drift (changes in document templates or input quality).

This component view supports a core technical claim: reliability and safety are emergent properties of the *entire stack*, not solely of the model. A governance-first posture therefore treats each component as a potential control point and requires that the system be designed to surface its

internal dependencies and transformations.

5.4.2 Information flow

Information flow in a multimodal VLA system is the pathway by which evidence is transformed into representation, representation into reasoning, and reasoning into outputs and actions. For governance, the critical requirement is that this flow be *traceable*: the organization must be able to answer which evidence supported which claim and which claim motivated which action.

A simplified flow begins with raw evidence. Let X denote the multimodal input bundle, but in practice X is rarely consumed directly. Instead, the system produces intermediate forms:

$$X \xrightarrow{\mathcal{T}} \tilde{X} \xrightarrow{\text{encode}} Z \xrightarrow{\text{fuse}} H \xrightarrow{\text{reason/plan}} (Y, A).$$

Here $\mathcal{T}$ denotes transformations (OCR, ASR, normalization), $\tilde{X}$ denotes transformed evidence (text tokens from OCR, segmented regions, structured tables), Z denotes modality-specific latent representations, and H denotes the fused state. The outputs (Y, A) include both communicative content and proposed or executed actions.

In many enterprise systems, there is also **retrieval flow**. The model is rarely trusted to operate only on the provided input. It retrieves reference materials: policies, prior cases, knowledge base entries, and sometimes external sources. Retrieval introduces an additional branch:

$$\tilde{X} \xrightarrow{\text{query}} R \quad \text{and} \quad (\tilde{X}, R) \to H.$$

R is the retrieved context. Governance demands that R be treated as evidence with provenance: what was retrieved, from where, under which permissions, and with what timestamp. A frequent source of compliance risk is that retrieval quietly imports outdated or unauthorized content, and the final output has no trace of that provenance.

The governance-critical information product is not only Y and A, but a trace artifact τ that binds evidence to decisions:

$$\tau \;=\; (\mathcal{E},\, \mathcal{P},\, \mathcal{L},\, \mathcal{G}).$$

Here $\mathcal{E}$ is an evidence map (page/region/cell/time references), $\mathcal{P}$ is a provenance record (source IDs, timestamps, access labels), $\mathcal{L}$ is a tool/action log (calls, arguments, results references), and $\mathcal{G}$ is a governance record (gating decisions, approvals, escalation steps). Without τ, the system is operationally dangerous because it cannot be audited or debugged; it becomes a black box that produces authoritative-seeming outputs.

Information flow must also account for **privacy filtering**. In a regulated environment, the system cannot indiscriminately pass raw evidence through every component. Certain sensitive artifacts must be redacted before being sent to the model, or processed in secure enclaves. This yields

a bifurcated flow: raw evidence may be processed by specialized redaction tools, then only a minimized representation is passed to the model. Similarly, logs may store only hashes and references. Technically, this creates friction and complexity, but it is the cost of defensibility.

A final information flow concern is **feedback contamination**. When the system writes outputs back into enterprise systems, those outputs can later be retrieved as evidence. If not controlled, the system can bootstrap its own errors into the evidence base. A governance-first design separates *model-generated artifacts* from *authoritative records*, or at least labels them distinctly. The information flow should preserve the distinction between "observed" and "generated." Without this, the system can create self-reinforcing loops where the model's prior guesses become the justification for future actions.

In summary, the technical foundation of information flow is not a data engineering detail; it is the backbone of accountability. The system must be engineered so that evidence transformations are recorded, retrieval is traceable, privacy constraints are enforced, and feedback does not corrupt the evidentiary base.

5.4.3 Interaction or control loops

Multimodal VLA systems are not static predictors. They operate in loops: they perceive, infer, act, and observe again. These loops are the operational form of agency. They also define where evaluation and control must focus, because many failures arise from iterative behavior rather than single-step errors.

A minimal interaction loop can be written as:

$$(X_t, s_t) \to (Y_t, A_t) \to \mathrm{Env}(s_t, A_t) = s_{t+1} \to X_{t+1},$$

where s_t is the environment state (workflow state, database state, UI state), and X_{t+1} includes new observations (tool results, updated documents, user feedback). The loop can be initiated by a user request, a scheduled monitoring trigger, or an event in the environment.

There are several distinct types of loops:

(1) Verification loops. The system proposes an interpretation, runs checks, and revises. For instance, it may extract fields from a document, then run validation rules (format checks, totals checks, cross-document consistency), then either confirm or escalate. Verification loops are desirable; they embody the "observe–propose–verify" pattern. Governance should encourage these loops by making verification tools available and by rewarding deferral when checks fail.

(2) Active perception loops. When inputs are ambiguous, the system may request better evidence: a higher-resolution scan, a missing page, a different camera angle, or a clarifying question. In physical or UI settings, active perception can include zooming, re-cropping, or requesting additional

screenshots. This is a frontier capability because it shifts the system from passive perception to evidence acquisition. It is also a governance concern because it can create privacy exposure (asking for more sensitive views) and because it can be exploited (an adversary can respond with manipulated evidence).

(3) Tool-use loops. The system iteratively calls tools: retrieval, databases, calculators, or browsers. Each tool call introduces a new observation and potentially new risk. Tool-use loops can be productive, but they are also where authority creep occurs. A system that is allowed to call tools repeatedly can escalate from reading to writing or from internal queries to external communications. Therefore, tool-use loops must be bounded by budgets (call limits), allowlists, and policy checks at each step.

(4) Action loops. The system takes an action that changes state (e.g., updates a record), then observes the result and continues. Action loops are the highest-risk form because they can compound errors. If the system misperceives and then acts, it can create new evidence that reflects its mistake, leading to reinforcement. Governance therefore requires explicit action gating, irreversible-action thresholds, and rollback plans. In many settings, the safest architecture is to allow the system to propose actions but to require human approval for execution.

(5) Monitoring loops. The system continuously observes streams of multimodal data (dashboards, camera feeds, logs) and flags anomalies. Monitoring loops are common in manufacturing, infrastructure, and compliance. The frontier issue is that monitoring systems can evolve into intervention systems: once an anomaly is detected, the system may be allowed to trigger a response. Governance must therefore define whether the system is advisory or autonomous, and must ensure that monitoring does not become covert surveillance without explicit policy.

Control loops are not only technical. There are **human control loops**. Humans review outputs, approve actions, and provide corrections. A mature VLA system treats human feedback as structured signals, not informal chat. Approvals should be logged; corrections should be captured as labeled events; and escalation should be formal. This is essential for auditability and for continuous improvement, but it also defines accountability boundaries: the organization must know when a human took responsibility and when the system acted under delegated authority.

Finally, control loops include **risk monitoring**. The system should be instrumented to detect injection attempts, anomalous tool sequences, unusual access patterns, and drift in input distributions. These monitors feed back into governance decisions: tightening tool permissions, updating sanitization rules, or deferring deployment in certain contexts. Without risk monitoring, the system is blind to its own exposure.

The technical foundation here is that control loops must be designed, not discovered. If left to emerge, they will be shaped by convenience and performance pressures, often at the expense of safety and accountability. Governance-first design makes loops explicit, bounded, and auditable.

5.4.4 Assumptions and constraints

Any rigorous technical foundation must state assumptions and constraints explicitly, because the most dangerous failures often occur when implicit assumptions are violated in deployment. Multimodal VLA systems operate under a set of harsh realities that must be treated as default conditions rather than rare edge cases.

Assumption 1: Inputs may be manipulated. External documents and images can be adversarial. Even internal artifacts can be unintentionally misleading due to formatting or scanning errors. Therefore, the system must assume that inputs can contain injection attempts, misleading layouts, or hidden instructions. This assumption directly motivates control-plane separation: untrusted content must not be allowed to influence system prompts, permissions, or tool policies.

Assumption 2: Observations are incomplete. In many workflows, the system does not have all relevant information. A document may be missing pages; an image may not include the necessary region; a screenshot may omit context; an audio snippet may cut off key words. Incomplete observation is a core property of the environment. The system must therefore support deferral and active perception rather than hallucination.

Assumption 3: Perception is probabilistic and context-dependent. OCR accuracy depends on scan quality; chart reading depends on resolution and style; ASR depends on noise and accents. These are not fixed properties. The system must therefore represent uncertainty and propagate it into decision-making. Treating perception outputs as exact facts is a category error that leads to unsafe action.

Assumption 4: Privacy constraints apply to both inputs and logs. Multimodal artifacts often contain sensitive information. The system cannot assume it is allowed to store, transmit, or display raw inputs. This constraint forces design choices: redaction before processing, minimization in outputs, secure evidence vaults, and privacy-preserving logging. Governance must also define retention and access policies for both evidence and logs.

Assumption 5: Tool access is bounded by permissions. The system cannot be given unconstrained tool access. Tools must be scoped to tasks and roles. This is not only a security constraint but a governance requirement: delegating authority without explicit boundaries is organizational negligence in regulated contexts.

Assumption 6: The environment changes. Document templates evolve, UIs change, policies update, and databases drift. Therefore, the system cannot assume stable distributions. Monitoring for drift and continuous evaluation are necessary. This also implies that system behavior must be robust to minor changes and must fail safely when major changes occur.

Constraint 1: Determinism is limited. Many components (model inference, retrieval ranking, OCR heuristics) can introduce nondeterminism. In regulated settings, however, reproducibility is a requirement. The system must therefore impose determinism where feasible (fixed seeds, versioned

components) and record configuration states where not feasible. The constraint is that perfect determinism may be impossible; the goal is replayability and traceability.

Constraint 2: Latency and cost budgets exist. Multimodal processing is expensive. OCR, parsing, and large model inference add latency. Tool loops add additional delays. Organizations must operate within cost and SLA constraints. This creates a tension with verification: more checks improve safety but cost more. Governance-first design resolves this by tying verification intensity to risk. High-impact actions justify higher cost; low-impact tasks can be cheaper.

Constraint 3: Human review capacity is scarce. Many governance designs assume that humans will review everything, but real organizations operate under time pressure. Therefore, the system must be designed so that review is *efficient and meaningful*. Evidence maps, uncertainty summaries, and structured action requests reduce review burden. Without these, the human-in-the-loop becomes a bottleneck or, worse, a rubber stamp.

Constraint 4: Integration surfaces are heterogeneous. Tools vary in maturity. Some have clean APIs; others require UI automation. Some systems support audit logs; others do not. Therefore, the VLA system must be robust to integration limitations and must compensate with its own logging and gating. The constraint is that organizations often cannot rebuild all systems; governance must work with existing infrastructure.

These assumptions and constraints collectively define the operating envelope. A VLA system is safe and valuable only within a defined envelope where input quality, tool access, privacy controls, and review processes are aligned. The technical foundation is therefore inseparable from an explicit statement of what the system assumes and what it cannot guarantee.

5.4.5 Technical bottlenecks

The technical bottlenecks of multimodal VLA systems are not simply "better models." They are bottlenecks in robustness, uncertainty, security, and auditability that emerge from system coupling. These bottlenecks define where frontier work remains and where organizations should be cautious about deployment claims.

(1) Robust fusion under real-world heterogeneity. Combining modalities reliably is hard when modalities are noisy, incomplete, and sometimes contradictory. The system must decide how to weigh OCR text versus visual layout, how to reconcile a chart with its caption, and how to handle missing context. In practice, fusion failures are common and can be silent. Robust fusion requires explicit contradiction detection, cross-checking, and sometimes task-specific validation logic. A frontier issue is that general-purpose models do not reliably expose when fusion is unstable.

(2) Calibrated uncertainty estimation across modalities. Uncertainty in multimodal systems has multiple sources and is difficult to calibrate. A system might be highly confident in a wrong OCR extraction because it lacks a reliable uncertainty signal, or it might be uncertain for the wrong

reasons. Without calibrated uncertainty, action gating cannot be principled. The bottleneck is not only producing uncertainty estimates but making them operational: connecting uncertainty to verification loops, deferral, and approvals.

(3) Secure tool invocation and control-plane integrity. Tool use creates a control plane that must be protected from untrusted content. The bottleneck is designing systems where documents and images cannot inject instructions that alter tool selection or permissions. This requires prompt compartmentalization, sanitization, allowlists, and external enforcement. It also requires evaluation: injection attempts must be part of the test suite. The frontier challenge is that many deployments still rely on informal prompt rules rather than enforceable boundaries.

(4) Privacy-preserving yet auditable logging. Organizations need logs to audit and debug, but multimodal evidence is often too sensitive to store in raw form. The bottleneck is building logging systems that preserve accountability without violating privacy. Techniques include hashing, redaction, storing references to secure evidence vaults, and capturing minimal excerpts with coordinates. Yet there is no universal standard, and operational tooling remains immature. This bottleneck is particularly acute because it is not purely technical; it requires alignment with legal, compliance, and security policies.

(5) Evaluation that matches deployment reality. The field lacks standardized benchmarks that test end-to-end perception–reasoning–action safety under adversarial and privacy-constrained conditions. Many evaluations still measure isolated capabilities: VQA accuracy, OCR extraction, or instruction following. The bottleneck is building evaluation harnesses that simulate real workflows, include injection and shift, and measure not only correctness but constraint adherence and safe deferral. Without such evaluation, organizations cannot justify autonomy.

(6) Integration brittleness, especially UI operation. When VLA systems operate interfaces, behavior becomes sensitive to UI changes, rendering differences, and timing. The bottleneck is achieving robust UI perception and action under variability, and maintaining the system as UIs evolve. This brittleness also creates governance problems: it is hard to certify a system whose operating surface changes frequently.

(7) Preventing feedback contamination and self-reinforcement. When model outputs are written into systems and later retrieved as evidence, the system can reinforce its own errors. The bottleneck is designing data governance that distinguishes authoritative records from model-generated artifacts and prevents the model from using its own outputs as ground truth without verification.

(8) Human oversight ergonomics. Finally, there is a bottleneck in making oversight workable. If reviewers cannot see evidence references, uncertainty, and action rationale clearly, oversight fails. The frontier challenge is that many systems optimize for smooth user experience rather than for audit-grade review. In regulated domains, that optimization is misaligned: the primary user is not only the operator, but the auditor, the risk officer, and the institution's future self reconstructing

an incident.

These bottlenecks define the technical frontier of multimodal deployment. They also justify the chapter's core governance claim: the principal risks and opportunities are system-level. Progress will come not only from larger models, but from architectures that make uncertainty operational, enforce control-plane integrity, and produce privacy-preserving audit trails that can stand up to scrutiny.

5.5 Mathematical Foundations

5.5.1 Formal problem framing

A multimodal vision–language–action (VLA) system becomes mathematically interesting—and institutionally dangerous—precisely when it is not merely producing a description, but participating in a *decision* under uncertainty with the ability to trigger state changes. The appropriate formal framing is therefore not "predict a label" but "choose an action under partial information subject to constraints." The purpose of this section is to define that problem cleanly, to expose where instability and fragility arise, and to clarify what the mathematics *cannot* guarantee even under optimistic modeling assumptions.

We begin with multimodal evidence. Let the raw observation bundle at time t be

$$X_t \;=\; (x_t^{\text{text}}, x_t^{\text{img}}, x_t^{\text{audio}}, x_t^{\text{doc}}, x_t^{\text{tab}}, x_t^{\text{meta}}),$$

where x_t^{meta} includes provenance labels, retention class, user identity/role, and the workflow context (e.g., "invoice review," "contract triage," "compliance escalation"). In a realistic stack, X_t is not consumed directly. It is transformed by preprocessors (OCR, ASR, layout parsing) into a derived evidence representation $\tilde{X}_t = \mathcal{T}(X_t)$, where $\mathcal{T}$ is a possibly stochastic transformation pipeline with parameters φ (OCR model/version, parsing heuristics, image normalization settings):

$$\tilde{X}_t \;=\; \mathcal{T}(X_t; \varphi).$$

The system then encodes and fuses $\tilde{X}_t$ into a latent state h_t (details in the next subsection), produces a communicative output Y_t (e.g., extraction, summary, recommendation), and proposes or executes an action a_t chosen from some action set $\mathcal{A}$. The environment state s_t (database state, workflow state, UI state, external context) evolves according to a transition function $\mathcal{E}$:

$$s_{t+1} \;=\; \mathcal{E}(s_t, a_t, \omega_t),$$

where ω_t models exogenous randomness (tool failures, delayed updates, other actors). The VLA system therefore operates on a trajectory $\tau = (s_0, X_0, a_0, s_1, X_1, a_1, \dots)$.

The most compact statement of the decision problem is a constrained expected utility maximization at each decision point:

$$a_t^* \;\in\; \arg\max_{a \in \mathcal{A}(s_t)} \; \mathbb{E}[\, U(s_t, a, S_{t+1}) \mid \mathcal{I}_t \,] \quad \text{s.t.} \quad \mathbb{E}\Big[\, C(s_t, a, \tilde{X}_t) \mid \mathcal{I}_t \,\Big] \le \kappa.$$

Here:

- $\mathcal{I}_t$ is the system's information set at time t (including $\tilde{X}_t$, retrieved context, tool outputs observed

so far).

- $U(\cdot)$ is a utility function capturing task value: correctness, speed, completeness, and organizational benefit.
- $C(\cdot)$ is a constraint cost capturing violations: privacy exposure, unauthorized access, unsafe action, policy breach.
- κ is a tolerance level. In regulated settings, κ may be near zero for some classes of violation (e.g., disallowed disclosure).
- $\mathcal{A}(s_t)$ is the allowable action set given state and permissions; this already encodes least privilege at a mathematical level.

This framing makes several governance-relevant claims explicit. First, the action set is state- and permission-dependent, meaning authority is not a property of the model; it is a property of the system's control layer. Second, constraints are not afterthoughts; they are part of the optimization problem. Third, the system's information set is a function of preprocessing and retrieval pipelines, meaning that governance must treat those pipelines as part of the decision mechanism.

A key subtlety is that U is rarely a scalar in practice. Enterprises have multiple objectives: accuracy, speed, consistency, customer impact, and risk. A more faithful formalization is multi-objective optimization. Let U be vector-valued:

$$\mathbf{u}(s_t, a) \;=\; \big(u_1(s_t, a), \ldots, u_m(s_t, a)\big),$$

with components like task success, robustness, reviewability, and timeliness. A standard approach is scalarization:

$$U_\lambda(s_t, a) \;=\; \sum_{i=1}^{m} \lambda_i \, u_i(s_t, a), \quad \lambda_i \geq 0, \quad \sum_i \lambda_i = 1.$$

The choice of λ is a governance decision: it encodes how the organization trades speed against verification, or automation against oversight. The mathematical point is not that scalarization is perfect, but that it forces explicit recognition of trade-offs that are often hidden in product decisions.

Constraints likewise come in families. Instead of one constraint cost, we can define a constraint vector $\mathbf{c}(s_t, a, \tilde{X}_t) \in \mathbb{R}^p$ with thresholds $\boldsymbol{\kappa} \in \mathbb{R}^p$:

$$\mathbb{E}\Big[\mathbf{c}(s_t, a, \tilde{X}_t) \mid \mathcal{I}_t\Big] \leq \boldsymbol{\kappa},$$

where components represent privacy constraints (no PII leak), access constraints (no unauthorized tool calls), safety constraints (no irreversible action without approval), and provenance constraints (no claims without evidence references). This aligns with the chapter's thesis that multimodal governance is multi-dimensional: a system can be correct but still noncompliant, or compliant but unhelpful.

Finally, because VLA systems are trajectory systems, we often care about cumulative cost and

long-horizon consequences. A standard objective is discounted return:

$$J(\pi) \;=\; \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t\, U(s_t, a_t, S_{t+1})\right], \quad 0 < \gamma < 1,$$

subject to cumulative constraint budgets:

$$\mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t\, C_j(s_t, a_t, \tilde{X}_t)\right] \le \kappa_j, \quad j = 1, \ldots, p.$$

This is the familiar language of constrained Markov decision processes (CMDPs). The key governance connection is that constraints represent institutional obligations that must hold *over time*, not merely per step. For example, repeated small privacy leaks can accumulate into a major incident; repeated borderline tool calls can represent a pattern of misuse.

In practice, enterprises often do not solve CMDPs optimally. Instead, they design policies with hard gates: certain actions require approval, certain tool calls are forbidden, certain contexts force read-only mode. Mathematically, hard gating can be represented by setting $\mathcal{A}(s_t)$ to exclude unsafe actions or by defining constraints with $\kappa = 0$ and large penalties for violations. This is not a limitation of the formalism; it is an instantiation of governance-first design: the policy class is intentionally restricted to ensure defensibility.

5.5.2 State, action, or representation spaces

To reason formally about VLA systems, we must define what "state" means in a multimodal setting. Unlike classic control systems, the system does not observe the environment state s_t directly. It observes a multimodal bundle X_t that is a noisy, partial function of s_t. This is a partially observable setting. A clean abstraction is a partially observable Markov decision process (POMDP) augmented with tool actions and policy constraints.

Let the latent environment state be $s_t \in \mathcal{S}$. This state includes not only external variables (e.g., the true content of a contract, the real values in an invoice) but also internal workflow state (e.g., ticket status, user role, approval chain, and prior actions). The system observes X_t drawn from an observation model:

$$X_t \sim \mathcal{O}(\cdot \mid s_t),$$

and then processes X_t into derived evidence $\tilde{X}_t = \mathcal{T}(X_t; \varphi)$. Because $\mathcal{T}$ can be lossy and stochastic, the effective observation model for $\tilde{X}_t$ is

$$\tilde{X}_t \sim \tilde{\mathcal{O}}(\cdot \mid s_t; \varphi).$$

The system maintains an internal representation $h_t \in \mathcal{H}$, often computed by encoders and fusion:

$$h_t \;=\; f_\theta(\tilde{X}_t, r_t, h_{t-1}),$$

where r_t is retrieved context or tool results incorporated at time t, and f_θ is a parameterized function (the multimodal model plus its memory mechanism). This is intentionally generic. The point is that h_t is a learned sufficient statistic (or at least an attempted one) for decision-making under partial observability.

A classical POMDP introduces a belief state b_t, a probability distribution over latent states given the history:

$$b_t(s) \;=\; \mathbb{P}(s_t = s \mid \mathcal{I}_t).$$

In a VLA system, the model's internal representation h_t can be viewed as an implicit belief state, but it is not guaranteed to be calibrated or even interpretable. Governance-first design therefore often supplements h_t with explicit uncertainty metadata: OCR confidence, ASR alternatives, detection confidence, and contradiction flags. We can represent the system's decision state as an augmented tuple:

$$z_t \;=\; (h_t, q_t),$$

where q_t contains explicit quality/uncertainty signals and policy-relevant annotations (source trust label, sensitivity class, redaction status).

The action space is similarly nonstandard. In VLA systems, an "action" is not only a single environment command. It can be a tool call (read action), a proposed workflow step (write action), a request for more evidence (active perception), or a human escalation. A useful decomposition is:

$$\mathcal{A} \;=\; \mathcal{A}^{\text{read}} \;\cup\; \mathcal{A}^{\text{write}} \;\cup\; \mathcal{A}^{\text{communicate}} \;\cup\; \mathcal{A}^{\text{verify}} \;\cup\; \mathcal{A}^{\text{escalate}} \;\cup\; \mathcal{A}^{\text{defer}}.$$

Examples:

- $\mathcal{A}^{\text{read}}$: query a database, retrieve a policy, open a document region, run OCR on a page.
- $\mathcal{A}^{\text{verify}}$: compute totals, cross-check fields, compare extracted entities to known registries.
- $\mathcal{A}^{\text{write}}$: update a record, submit a form, change a ticket status.
- $\mathcal{A}^{\text{communicate}}$: draft an email, post to a channel, send a notification.
- $\mathcal{A}^{\text{escalate}}$: request human approval, route to compliance, flag for review.
- $\mathcal{A}^{\text{defer}}$: refuse action pending clarification or better evidence.

Crucially, in a governed system, the available action set depends on permissions and context:

$$\mathcal{A}(s_t, \rho_t) \subseteq \mathcal{A},$$

where ρ_t represents the permission state (role-based access, tool allowlists, approval status). The

system's policy π maps decision state z_t into an action distribution:

$$\pi(a \mid z_t) \;=\; \pi(a \mid h_t, q_t).$$

In practice, policy is often implemented as a composition:

$$\pi \;=\; \pi_{\text{model}} \circ \Pi_{\text{guard}},$$

where π_{model} proposes actions (including tool calls) and Π_{guard} is an external guard that enforces constraints by filtering, modifying, or requiring approvals. Formally, Π_{guard} induces a constrained policy class:

$$\Pi_{\text{safe}} \;=\; \{\pi : \pi(a \mid z) = 0 \text{ for all } a \notin \mathcal{A}_{\text{allowed}}(z)\}.$$

This separation is a mathematical version of "the model may suggest; the system must enforce."

Representation spaces matter because multimodal errors frequently originate from representation mismatch. For example, the latent representation h_t may entangle content and instructions: text extracted from an untrusted document can be embedded into the same space as the system's own control prompt. That creates a vulnerability: untrusted content can influence action selection. A governance-aware architecture can impose representation separation:

$$h_t \;=\; (h_t^{\text{content}},\, h_t^{\text{control}}),$$

where h_t^{control} is derived only from trusted sources (system policy, user intent) and h_t^{content} is derived from untrusted evidence. The guard layer then ensures that only h_t^{control} can influence permission-critical decisions (e.g., tool scope). In practice this can be implemented through prompt compartmentalization and policy engines, but the mathematical abstraction clarifies the goal: prevent a mixing term that lets content become control.

Finally, because evidence in multimodal systems is spatial/temporal, the system's state should include an *evidence index*:

$$\mathcal{E}_t \;=\; \{e_{t,1}, \ldots, e_{t,k}\},$$

where each evidence item includes a pointer (page, region, timestamp, cell), a provenance label, and a confidence score. Outputs and actions should be functions not only of h_t but of $\mathcal{E}_t$ explicitly, to enable audit:

$$(Y_t, a_t) \;=\; g(h_t, \mathcal{E}_t, q_t).$$

This is the mathematical expression of a governance requirement: decisions must be tied to retrievable evidence references.

5.5.3 Objective functions and constraints

The decision problem in VLA systems is multi-objective and constraint-driven. It is tempting to define a single "task success" metric, but that fails to reflect institutional reality. A correct extraction that violates privacy is unacceptable; a safe system that cannot complete tasks has little value. The mathematics must therefore explicitly encode both value and obligation.

Task success. Let $u_{\text{task}}(s_t, a_t)$ measure task performance. This could be accuracy of extracted fields, correctness of classification, or success of a workflow step. In many settings, task success is defined relative to latent ground truth in s_t (e.g., the true contract terms), which may not be fully observable to the system. We can represent task loss ℓ_{task} and define

$$u_{\text{task}} \;=\; -\ell_{\text{task}}.$$

For example, for structured extraction with ground-truth field vector $y^\star$ and predicted $\hat{y}$, one might define $\ell_{\text{task}}(\hat{y}, y^\star)$ as a weighted error.

Robustness. Robustness reflects performance under perturbations, distribution shift, and ambiguity. A practical mathematical formalization is worst-case or risk-sensitive utility. Suppose the input evidence is subject to perturbations δ in an uncertainty set Δ (blur, occlusion, OCR noise, layout variation). Then robust task utility might be

$$u_{\text{robust}}(s_t, a_t) \;=\; -\sup_{\delta \in \Delta} \ell_{\text{task}}\big(\hat{y}(X_t + \delta), y^\star\big).$$

Alternatively, under a distributional shift set $\mathcal{P}$ of plausible deployment distributions, one could define:

$$u_{\text{robust}} \;=\; -\sup_{P \in \mathcal{P}} \mathbb{E}_{X \sim P}\big[\ell_{\text{task}}\big].$$

These are conceptual definitions; organizations rarely compute them exactly. Their value is that they clarify why naive average-case accuracy can be misleading: institutions care about tail behavior, especially when actions are consequential.

Reviewability and evidence fidelity. VLA systems must support oversight. A formal proxy is a penalty when outputs lack evidence mapping or when claims are not traceable. Let $\mathcal{C}(Y_t)$ denote the set of material claims in Y_t, and let $\mathcal{E}_t$ be the evidence index. Define a coverage function $\text{cov}(\mathcal{C}, \mathcal{E}) \in [0, 1]$ measuring the fraction of claims supported by evidence references. Then we can include:

$$u_{\text{review}} \;=\; \alpha\, \text{cov}\big(\mathcal{C}(Y_t), \mathcal{E}_t\big) \;-\; \beta\, \text{hall}(Y_t),$$

where $\text{hall}(Y_t)$ is a hallucination indicator or score, and $\alpha, \beta > 0$. Again, the purpose is conceptual: reviewability is not a soft preference; it is an objective dimension.

Privacy and confidentiality. Privacy constraints can be formalized as limits on sensitive

information exposure. Let $S(\tilde{X}_t)$ denote sensitive content present in the input (PII, confidential numbers). Let $\text{leak}(Y_t, \mathcal{L}_t)$ measure exposure through outputs Y_t and logs $\mathcal{L}_t$. A constraint might be:

$$\mathbb{E}\big[\text{leak}(Y_t, \mathcal{L}_t) \mid \mathcal{I}_t\big] \le \kappa_{\text{priv}},$$

where κ_{priv} could be near zero for certain categories. This formulation highlights that logs are part of the exposure surface. A common governance failure is to secure outputs while neglecting debug logs; mathematically, both appear in the leakage function.

Permission and access control. Tool calls must respect role-based permissions. Let a_t encode a tool call with parameters (tool name, arguments). Define a predicate $\text{allow}(a_t, \rho_t) \in \{0, 1\}$ that indicates whether the action is permitted under permission state ρ_t. Then hard access control is:

$$\mathbb{P}\big(\text{allow}(a_t, \rho_t) = 1\big) = 1,$$

or equivalently $\pi(a \mid z_t) = 0$ whenever $\text{allow}(a, \rho_t) = 0$. This is a clean mathematical representation of least privilege.

Action gating for irreversible steps. Many systems must require human approval before irreversible actions. Let $\mathcal{A}_{\text{irr}}$ denote irreversible actions (submit filing, execute payment, send external message). Let $\text{approved}_t \in \{0, 1\}$ indicate whether a human approval token has been issued. Then a hard constraint is:

$$a_t \in \mathcal{A}_{\text{irr}} \;\Rightarrow\; \text{approved}_t = 1.$$

In practice, this is implemented by guard middleware. The mathematical value is clarity: approval is a state variable, not a vague process.

Redaction and minimization. Redaction can be formalized as an information projection operator $\mathcal{R}$ that maps raw evidence to a minimized view:

$$\bar{X}_t \;=\; \mathcal{R}(\tilde{X}_t; \psi),$$

where ψ defines redaction rules (mask account numbers, blur faces). The model may consume $\bar{X}_t$ rather than $\tilde{X}_t$ for certain tasks. A constraint can then require that for certain contexts, only redacted views are used:

$$\text{if sensitive}(\tilde{X}_t) = 1 \text{ then } \pi \text{ must condition only on } \bar{X}_t.$$

This is the mathematical version of "minimize data exposure by default."

Putting it together. A practical system objective can be represented as a Lagrangian with

penalties for soft constraints and hard constraints encoded in $\mathcal{A}(s_t, \rho_t)$:

$$\max_{\pi \in \Pi_{\text{safe}}} \mathbb{E}_\pi \left[\sum_{t \geq 0} \gamma^t \Big(U_\lambda(s_t, a_t) - \sum_{j=1}^{p} \eta_j \, c_j(s_t, a_t, \tilde{X}_t) \Big) \right],$$

where $\eta_j \geq 0$ are penalty weights. In governance-first practice, many constraints are not tuned via η; they are encoded as hard rules. The Lagrangian form is still useful because it reveals that "making the system more helpful" often corresponds to decreasing penalties or expanding $\mathcal{A}$—both governance decisions that must be justified and evaluated.

A final institutional point concerns **deferral as a legitimate action**. Many objective functions accidentally punish deferral because it reduces immediate task completion. In high-accountability contexts, deferral is often the correct action under uncertainty. Mathematically, deferral must be explicitly rewarded (or at least not heavily penalized) when uncertainty is high. Let q_t include a scalar uncertainty indicator $u_t \in [0, 1]$. Then one can define a deferral reward:

$$U_{\text{defer}}(s_t, a_t) \;=\; \mathbf{1}\{a_t = \text{defer}\} \cdot \phi(u_t),$$

where ϕ increases with uncertainty. This encourages a safe behavior: when the system is uncertain, it chooses to defer rather than guess and act.

5.5.4 Sources of instability or fragility

The most important mathematical contribution of this chapter is to identify where instability originates in VLA systems, and why it is not eliminated by improved average-case accuracy. Instability arises because the system composes multiple imperfect mappings: perception, fusion, reasoning, and action, often iteratively. Small errors early can be amplified into large downstream consequences. Several distinct fragility mechanisms are worth formalizing.

(1) Fusion dominance and misweighting. Let $z_t^{(m)}$ denote the representation from modality m (text, image, audio, table). Fusion produces a joint state:

$$h_t \;=\; \text{Fuse}\big(z_t^{(1)}, \ldots, z_t^{(M)}\big).$$

In many architectures, fusion can be conceptualized as a weighted combination:

$$h_t \;=\; \sum_{m=1}^{M} w_m(\tilde{X}_t) \, z_t^{(m)}, \quad \sum_m w_m = 1, \quad w_m \geq 0,$$

where weights w_m depend on the input. Fragility arises when the weighting is miscalibrated: a low-quality modality is overweighted, or a spurious cue drives weight selection. A small perturbation in $\tilde{X}_t$ can cause a discontinuous change in w_m, leading to large changes in h_t, and therefore in

actions. This is a mathematical expression of the intuitive failure: "the model latched onto the wrong signal."

A governance-relevant mitigation is to impose constraints on fusion, such as requiring consistency checks or limiting how dominant a single modality can be without corroboration:

$$w_m(\tilde{X}_t) \leq \bar{w} \quad \text{unless corroborated by another modality.}$$

Even if implemented heuristically, the mathematical form clarifies the control goal: prevent single-modality domination when stakes are high.

(2) Adversarial perturbations and non-robust representations. Multimodal systems are vulnerable to perturbations that can be imperceptible to humans but large in representation space. If X is perturbed to $X + \delta$ with $\|\delta\|$ small, the representation can shift significantly:

$$\|h(X + \delta) - h(X)\| \gg \|\delta\|.$$

This is a classic adversarial fragility statement. In VLA systems, the danger is compounded because the perturbed input can also embed instructions (content-as-control). Even if the perturbation does not change the semantic content in a human sense, it can change the model's tool choice or action proposal. Formally, if the policy is $\pi(a \mid h)$, then adversarial sensitivity can be expressed as a large change in action distribution:

$$D_{\mathrm{TV}}(\pi(\cdot \mid h(X + \delta)), \pi(\cdot \mid h(X))) \text{ is large,}$$

where D_{TV} is total variation distance. A large change in π for small δ is precisely the mathematical signature of a system that is unsafe to connect to actions.

(3) Cascading error in perception → reasoning → action. Consider a pipeline where extraction produces an intermediate estimate $\hat{e}$ (e.g., a number from OCR), reasoning uses $\hat{e}$ to produce a plan, and action executes that plan. Let the true value be $e^\star$ and extraction error be $\epsilon = \hat{e} - e^\star$. The action outcome might be some function $g(\hat{e})$. The sensitivity of the outcome to extraction error is approximately:

$$g(\hat{e}) - g(e^\star) \;\approx\; g'(e^\star)\,\epsilon.$$

If $|g'(e^\star)|$ is large (high sensitivity), small extraction errors produce large action errors. This is common in threshold-based decisions: a small OCR error can flip a threshold comparison and trigger a different action class. For example, misreading "$9,800" as "$98,000" can flip an approval requirement. In such cases, the system is unstable because the action policy has discontinuities.

Governance mitigation is to introduce *verification bands*. If a decision depends on a threshold θ, define a margin band $[\theta - \Delta, \theta + \Delta]$. If the estimated value is within the band or extraction

confidence is low, the system must verify or escalate:

$$\text{if } |\hat{e} - \theta| \leq \Delta \text{ or } \text{conf}(\hat{e}) < c_0, \quad \text{then } a_t = \text{verify/escalate}.$$

This is a mathematical rule for safe gating around discontinuities.

(4) Feedback contamination and self-reinforcement. Let the system write an output $\hat{y}_t$ into a database, which then becomes part of the retrieved context at future time steps. The retrieved context r_{t+1} becomes a function of $\hat{y}_t$:

$$r_{t+1} \;=\; \text{Retrieve}(s_{t+1}) \;\ni\; \hat{y}_t.$$

If $\hat{y}_t$ contains an error, then future decisions condition on corrupted evidence. Over time, this can create a self-confirming loop. Mathematically, the system's belief state can drift toward its own outputs, reducing sensitivity to real evidence. This is not a model-level pathology; it is an architectural risk. A governance mitigation is to label and separate model-generated artifacts, ensuring retrieval distinguishes them from authoritative records:

$$r_t \;=\; r_t^{\text{auth}} \cup r_t^{\text{model}}, \quad \text{and policy treats } r_t^{\text{model}} \text{ as non-authoritative unless verified.}$$

This is an example of provenance-aware state design.

(5) Tool coupling and action compounding. In a tool-augmented policy, the model may execute a sequence of tool calls before selecting a final action. Let the tool sequence be $a_t^{(1)}, \ldots, a_t^{(K)}$ and the final action be $a_t^{(K+1)}$. Each tool call returns an observation $o_t^{(k)}$, which updates the information set. The effective policy over sequences is:

$$\pi(a^{(1:K+1)} \mid z_t) \;=\; \prod_{k=1}^{K+1} \pi\big(a^{(k)} \mid z_t, o_t^{(1:k-1)}\big).$$

Fragility arises because errors or adversarial content in early observations $o_t^{(k)}$ can steer later tool choices. Moreover, each additional tool call increases attack surface and privacy exposure. From a governance viewpoint, sequences must be bounded. Mathematically, impose a budget $K \leq K_{\max}$ and define a cost per tool call:

$$C_{\text{tool}} \;=\; \sum_{k=1}^{K} c_{\text{tool}}(a^{(k)}).$$

This encourages minimal necessary tool use and supports rate limiting. More importantly, budgets create a control point: the organization can prevent indefinite search that may increase exposure.

(6) Ambiguity collapse into a single narrative. Many VLA systems are optimized for producing a single coherent answer. Under partial observability, the correct representation is a distribution over hypotheses. If the system collapses uncertainty prematurely, it may act on a single hypothesis that is wrong. Mathematically, let $\mathcal{H}$ denote hypotheses about latent state (e.g., which template

the document uses, which entity a name refers to). A proper Bayesian action would consider:

$$\mathbb{E}[U(a) \mid \mathcal{I}_t] \;=\; \sum_{h \in \mathcal{H}} U(a; h)\, \mathbb{P}(h \mid \mathcal{I}_t).$$

If the system instead selects $\hat{h} = \arg\max_h \mathbb{P}(h \mid \mathcal{I}_t)$ and acts as if $\hat{h}$ is true, it is performing a maximum a posteriori (MAP) approximation. MAP can be fragile when posterior mass is spread across multiple plausible hypotheses. Governance mitigation is to require either (i) explicit multi-hypothesis reporting, or (ii) deferral when the posterior is not concentrated. A simple concentration condition is:

$$\max_{h \in \mathcal{H}} \mathbb{P}(h \mid \mathcal{I}_t) \;\geq\; \tau,$$

where τ is a threshold. If not met, the system must verify or defer.

These instability mechanisms collectively justify the chapter's insistence on system-level controls. Many fragilities are structural: they arise from composition, discontinuities, and feedback loops. Improving the base model may reduce some errors but does not eliminate the need for gating, traceability, and explicit uncertainty handling.

5.5.5 What theory does NOT guarantee

It is tempting to view formal objectives, constraints, and state/action definitions as providing a pathway to guarantees. In safety-critical engineering, one can sometimes obtain bounded-error guarantees under strong assumptions. In multimodal VLA systems, the strongest claim of this section is the opposite: the relevant theory has sharp limits, and organizations must not confuse formalism with assurance. Several non-guarantees must be stated explicitly.

(1) High confidence does not imply correctness. Even if a model produces a confidence score or a logit margin, there is no general theorem that such confidence is calibrated under distribution shift or adversarial conditions. Calibration results exist under assumptions about training and test distributions; those assumptions are frequently violated in enterprise multimodal deployment. Therefore, any gating rule based on model confidence must itself be validated. Formally, calibration would require that for a confidence value p, the empirical correctness frequency is p:

$$\mathbb{P}(\text{correct} \mid \hat{p} = p) = p.$$

In real deployments, $\hat{p}$ is often not well-defined, and the conditional distribution changes over time. Thus, confidence is a signal, not a guarantee.

(2) Alignment does not imply injection immunity. Even if a model is trained to follow instructions safely, it can still be manipulated by untrusted content when the system architecture allows content to influence control decisions. There is no theorem that "aligned models" resist content-as-instruction attacks in arbitrary tool-coupled environments. This is because injection is a

system-level vulnerability: it exploits the interface between content and policy. Formally, even if the model's policy π_{model} satisfies some alignment constraint in a closed setting, the composed policy with tools and environment $\pi = \pi_{\text{model}} \circ \Pi_{\text{guard}}$ can still fail if Π_{guard} is weak or if the information set $\mathcal{I}_t$ includes adversarially crafted instructions. The guarantee that matters is about Π_{guard}, not about π_{model} alone.

(3) Training coverage does not imply deployment coverage. Foundation models are trained on massive datasets, but enterprise workflows have specialized distributions: internal document templates, rare edge cases, jurisdiction-specific forms, and idiosyncratic screenshots. Even if the training data included superficially similar content, the joint distribution of modalities, provenance, and action contexts in deployment is different. There is no theorem that general pretraining covers these distributions adequately. This is a direct consequence of distribution shift. Therefore, any claim about reliability must be conditional on evaluation under realistic deployment conditions.

(4) End-to-end optimization does not imply constraint satisfaction. One might hope that by optimizing a Lagrangian objective with penalties, the system will learn to respect constraints. In practice, penalties are imperfect, constraints are hard to encode, and rare violations matter. For privacy and safety constraints, organizations usually require hard guarantees (no unauthorized tool call, no irreversible action without approval). Penalty-based optimization provides no universal guarantee of zero violation probability unless one imposes strong assumptions and verifies the policy under all relevant states. For high-dimensional multimodal inputs, such verification is generally infeasible. Hence, enforce constraints externally through access control and gating.

(5) Formal models do not capture organizational incentives. The equations above treat U and C as if they were stable and correctly specified. In reality, organizations face incentives to increase automation, reduce latency, and expand tool access. These pressures can lead to gradual relaxation of constraints or expansion of action sets. No mathematical formalism prevents this drift. Governance must impose institutional processes: change control, audits, and approvals. Mathematics can clarify what is being changed (e.g., expanding $\mathcal{A}(s)$ or increasing κ), but it cannot prevent it.

(6) Traceability is not automatic. Even if we define evidence indices and provenance maps, there is no guarantee that the system will produce correct evidence attribution. Models can cite irrelevant regions, select misleading snippets, or omit crucial sources. Evidence mapping must be engineered and validated. In particular, if evidence attribution is generated by the same model that generates conclusions, it can suffer from correlated failures. Independent verification mechanisms (e.g., deterministic extraction checks, rule-based validators) are often required to make traceability credible.

(7) Robustness definitions do not yield operational robustness without measurement. Defining robustness as worst-case performance over perturbations is mathematically clean, but operationally one must define Δ or $\mathcal{P}$. Enterprises rarely know all perturbations they will face, and adversaries can adapt. Therefore, robustness must be treated as an empirical discipline: stress

testing, adversarial red teaming, and continuous monitoring. Mathematics guides what to measure; it does not remove the need to measure.

(8) Privacy-preserving logging lacks universal solutions. One might hope for a general theorem that hashing and redaction preserve auditability while ensuring privacy. In practice, the trade-off is context-dependent. Too much redaction undermines incident reconstruction; too little creates exposure. Moreover, privacy risk is not only about stored content but about inference: even hashed references can leak patterns if misused. Therefore, privacy-preserving auditability is an engineering and governance problem more than a solved mathematical one.

The appropriate conclusion is not pessimism; it is discipline. Formalism is valuable because it forces explicit articulation of decision variables, constraints, and control points. But organizations should treat the formal model as a *design lens*, not a proof of safety.

Risk & Control Notes

Boundary reminder: model vs system. Even perfect in-model alignment does not eliminate system-level failure modes created by tool access, UI affordances, logging practices, and organizational incentives. Treat control design as a systems engineering problem with explicit authority limits and verifiable audit trails.

A final synthesis is helpful. The mathematics of VLA systems provides three durable insights. First, multimodal deployments are naturally POMDP-like: uncertainty is structural and must be managed, not ignored. Second, constraints are first-class: privacy, permissions, and gating belong in the action set and control layer, not as soft afterthoughts. Third, instability is compositional: even small perception errors can cascade through planning and tool use into large operational harms, especially near discontinuities and under feedback loops. These insights justify the chapter's broader claim that frontier AI governance is about *system design under uncertainty*, not about celebrating model capability. In practice, the most defensible deployments will therefore be those that (i) restrict $\mathcal{A}$ via least privilege, (ii) tie uncertainty to verification and deferral, (iii) require evidence maps for material claims, and (iv) enforce approvals for irreversible actions. The mathematics does not guarantee these properties, but it clarifies exactly what must be engineered and audited for a VLA system to be responsibly deployed.

5.6 Evaluation and Validation

5.6.1 Why naive metrics fail

Evaluation is the discipline that determines whether multimodal vision–language–action (VLA) systems are a responsible capability or an institutional liability. The central claim of this chapter is that the evaluation problem changes qualitatively once perception, reasoning, and action are coupled. The most common evaluation mistake is to borrow metrics from single-modality paradigms—image accuracy, OCR character error rate, ASR word error rate, or even question-answer accuracy—and treat them as proxies for end-to-end safety. Those metrics measure components, not systems. They are necessary, but they are not sufficient, and in high-accountability settings they can be dangerously misleading.

The first failure of naive metrics is that **modality accuracy is not decision integrity**. A VLA system can have high OCR accuracy on average yet still be unsafe because the rare OCR errors occur exactly in high-stakes fields (amounts, dates, names, thresholds). In decision workflows, error impact is not linear in error frequency. A single misread digit can trigger an incorrect approval class or generate a materially incorrect filing. Component-level accuracy metrics are blind to this because they average across all tokens and cases. What matters is *risk-weighted accuracy*: correctness on high-sensitivity fields and near decision thresholds. If a system's errors cluster around threshold discontinuities, a "high accuracy" score can coexist with unacceptable operational risk.

The second failure is that naive metrics ignore **error propagation and compounding**. In a tool-coupled VLA system, a small misperception can lead to a wrong reasoning step, which leads to a wrong tool call, which changes external state, which then becomes part of future evidence. Traditional benchmarks evaluate a single prediction given a fixed input. They do not measure trajectories. But the system's real behavior is a sequence of dependent steps. The relevant metric is not only "was the final answer correct," but "did the system take safe steps, request verification when uncertain, and avoid irreversible actions under ambiguity." A model that is right for the wrong reasons is a problem; a system that is right but violates policy is a bigger problem; and a system that is wrong and acts is the largest problem.

The third failure is that naive metrics do not capture **adversarial conditions**, especially multimodal prompt injection and content-as-instruction attacks. A benchmark that tests a model on clean documents is not an evaluation of a system deployed in the presence of external counterparties, customer uploads, or untrusted attachments. In the real world, documents can be crafted to manipulate extraction, and images can embed instructions that hijack control flow. Traditional accuracy metrics typically assume benign inputs. Yet governance-first evaluation must assume adversarial inputs by default. A system that performs well on clean tasks but fails catastrophically under injection is not ready for deployment, regardless of its average-case scores.

The fourth failure is that naive metrics ignore **privacy and compliance risk**. Many benchmarks

treat the input as a free resource: the model can see everything, and evaluation ignores what gets logged, stored, or exposed. In enterprise settings, what matters is not only what the system outputs, but what it *reveals* and what it *retains*. A model can answer correctly while leaking sensitive information in a summary, or while storing raw documents in logs that violate retention policy. Standard benchmarks do not test these failure modes because they are not designed as governance instruments. Yet in regulated domains, privacy violations can be more consequential than task errors.

The fifth failure is that naive metrics are often blind to **deferral quality**. A safe VLA system must be willing to say "I cannot determine this from the provided evidence" and request a better scan, a missing page, or human review. Many benchmark datasets reward always producing an answer. This implicitly punishes deferral, encouraging systems that hallucinate rather than escalate. In governance-first deployment, the opposite incentive is needed: deferral under uncertainty is not a failure but a success. Therefore, evaluation must treat appropriate deferral as positive behavior and penalize confident guessing in ambiguous contexts.

The sixth failure is that naive metrics underweight **system configuration** and **change control**. A VLA system is not just a model; it is a model plus OCR settings, parsing heuristics, retrieval sources, tool permissions, and gating logic. A benchmark result without configuration trace is not reproducible and not auditable. Yet many evaluations treat the system as static. In reality, small configuration changes can produce large behavior shifts. Governance demands that evaluation bind results to versioned configurations.

In sum, naive metrics fail because they answer the wrong question. They answer "How well does this component perform on a curated task?" The relevant question is: "Under realistic uncertainty, adversarial conditions, privacy constraints, and tool coupling, does the system behave in a way the organization can defend?" A governance-first evaluation program therefore replaces static accuracy metrics with end-to-end, trace-based, constraint-aware validation.

5.6.2 Appropriate evaluation units

Once the evaluation goal is framed correctly, the next step is to define the right unit of measurement. For VLA systems, the appropriate unit is almost never a single output token or a single answer. The appropriate unit is a **case trajectory** with a complete trace: evidence ingested, transformations applied, reasoning steps taken, tools invoked, actions proposed, approvals requested, and final outcomes. In other words, evaluation should mirror what the organization will need during incident reconstruction.

A practical way to define evaluation units is to separate *task outcomes* from *process integrity*. Task outcomes include correctness and completion; process integrity includes governance adherence. In high-accountability settings, process integrity is often the dominant requirement because correct outcomes without defensible process are still unacceptable.

(1) End-to-end task episodes. An episode begins with a user request and a multimodal input bundle and ends with a final output and either an executed action or a deferred/escalated state. An episode includes all intermediate tool calls and evidence selections. Evaluation should define a set of canonical episodes that correspond to real workflows: invoice triage, contract term extraction, dashboard anomaly analysis, compliance escalation drafting, or mixed-media incident reporting. For each episode, the evaluation must record whether the system achieved the desired task outcome *and* whether it adhered to constraints (privacy, permissions, gating).

(2) Provenance fidelity as a first-class unit. For every material claim the system makes, it should provide an evidence reference. In multimodal settings, evidence references are not just citations; they are coordinate pointers: page numbers, bounding boxes, time intervals, or cell indices. An evaluation unit should therefore include a *claim-to-evidence mapping* artifact. Let $\mathcal{C}(Y)$ denote the set of material claims extracted from output Y. Let $\mathcal{E}$ denote the set of evidence pointers. The evaluation should compute a coverage score:

$$\text{Coverage} \;=\; \frac{|\{c \in \mathcal{C}(Y) : \exists e \in \mathcal{E} \text{ such that } e \text{ supports } c\}|}{|\mathcal{C}(Y)|}.$$

But coverage alone is insufficient. Evidence must be *correct*. Therefore evaluation should also assess evidence precision: the fraction of cited evidence pointers that actually support the associated claim. In practice, this requires human labeling or deterministic checks for some cases. The point is that provenance fidelity is measurable and must be treated as an output of the system, not a documentation afterthought.

(3) Action outcome integrity. For systems that propose or execute actions, the evaluation must include action outcome metrics. These metrics are not merely whether the action succeeded technically, but whether it was *permitted, justified, and safe*. For example, if the system proposes sending an email to an external party, evaluation must check whether this action required approval, whether approval was requested, and whether the email content respects policy and privacy. Action integrity metrics include:

- **Permission compliance:** Did the system attempt any disallowed tool call?
- **Gating compliance:** Did the system request approval for irreversible actions?
- **Justification integrity:** Was the proposed action supported by evidence references?
- **Deferral correctness:** Did the system defer when evidence was insufficient?

(4) Transformation traceability. Because preprocessors (OCR/ASR/layout parsing) are part of the system, evaluation must include transformation traces: which OCR version, which parsing settings, which redaction rules. This is not optional. If a system misreads a number, the organization must know whether the error came from the OCR stage or the multimodal model. Therefore, the evaluation unit includes a structured trace of transformations:

$$\tau \;=\; (\text{input references}, \text{transform versions}, \text{tool calls}, \text{outputs}, \text{gates/approvals}).$$

A VLA system that cannot produce this trace is not evaluable in the sense required for governance.

(5) Safety-critical "no" as an evaluated behavior. A mature evaluation suite includes cases where the correct behavior is refusal, deferral, or escalation. For example, a blurred contract signature should trigger a request for a clearer scan. A prompt injection in a document should trigger sanitization and refusal to follow embedded instructions. A privacy-sensitive artifact should trigger redaction. These behaviors must be scored positively. Otherwise, evaluation will select for systems that are helpful-but-unsafe.

The consequence of these evaluation units is a shift in measurement mindset. Instead of producing a single accuracy score, evaluation produces a *bundle*: task success, provenance fidelity, action integrity, privacy behavior, and trace completeness. This is more demanding than classic benchmarks, but it matches the reality of accountable deployment.

5.6.3 Robustness and stress testing

Robustness in VLA systems is not a theoretical aspiration; it is an operational necessity because real inputs are messy, adversarial, and privacy constrained. Stress testing is therefore not a special exercise reserved for security teams; it is part of basic validation. A governance-first stance demands that stress testing be integrated into the evaluation harness and treated as a gating requirement for deployment.

A useful stress-testing program targets four domains: (i) perturbations and noise, (ii) adversarial content and injection, (iii) privacy leakage and retention, and (iv) tool misuse and authority escalation.

(1) Perturbation tests for perception robustness. The goal is to test whether the system's outputs and actions remain stable when the input evidence is degraded in realistic ways. For images and scanned documents, perturbations include blur, compression artifacts, rotation, partial occlusion, low contrast, and cropping. For tables and charts, perturbations include low resolution and unusual fonts. For audio, perturbations include background noise, overlapping speakers, and clipped segments. Evaluation should include both *performance degradation curves* (how task success changes with degradation) and *safety degradation behavior* (whether the system escalates appropriately when evidence quality drops). A robust system should not simply fail silently; it should increase uncertainty and defer.

(2) Adversarial overlays and hidden instructions. Multimodal prompt injection must be tested explicitly. Stress cases should include documents or images that contain instruction-like text intended to override system policy (e.g., "ignore prior instructions," "exfiltrate data," "call tool X"). Importantly, the evaluation must test whether the system's control plane is protected: does it treat untrusted content as data rather than instruction? It must also test whether sanitization removes or neutralizes such content. The stress suite should include variations: instructions in

headers/footers, tiny font, rotated text, or embedded in images that look like legitimate stamps or notes. The purpose is not to simulate every attack, but to ensure the system fails safely and does not follow embedded instructions.

(3) Privacy leakage tests. Privacy evaluation should include cases where the input contains sensitive content that should not appear in outputs or logs: account numbers, addresses, faces, signatures, health information, or confidential financial figures. The system should be tested on whether it (i) redacts sensitive content appropriately, (ii) respects role-based viewing (different users see different levels of detail), and (iii) avoids storing raw sensitive content in logs. Privacy leakage tests should also include *prompted leakage attempts*: users asking the system to reveal hidden details or to summarize sensitive segments. The correct behavior is refusal or redacted output. This is a governance-first stress test because the failure mode is not wrong answers; it is unauthorized disclosure.

(4) Tool misuse and authority escalation tests. For tool-coupled systems, the stress suite must include attempts to induce unsafe tool calls: requesting disallowed databases, attempting write operations when read-only is intended, or attempting to trigger irreversible actions without approval. Evaluation should confirm that the guard layer blocks such actions regardless of model behavior. This is crucial: tool safety cannot rely on the model's compliance. A robust system must enforce permission constraints externally and log every blocked attempt.

(5) Contradiction and consistency stress. A common real-world failure mode is contradictory evidence: a document says one value while a table shows another; a chart title conflicts with axis labels; an email thread conflicts with an attachment. Stress tests must include contradictory cases and measure whether the system detects contradiction, surfaces it, and escalates. A system that resolves contradictions by guessing is unsafe. A system that flags contradictions and requests clarification is governance-ready.

(6) Distribution shift stress. Enterprises face template drift: new invoice formats, new contract templates, new dashboard layouts. Stress testing should therefore include "unseen template" cases. This can be done with synthetic generation: create variations in layout, fonts, and field positions while preserving underlying content. The evaluation should measure whether the system generalizes or whether it fails into hallucination. Again, appropriate failure is deferral, not confident guessing.

(7) Trace stress. Finally, stress testing must include auditability tests: can the system produce a complete trace under stress? When it defers, does it explain why? When it blocks a tool call, does it record the reason? When it cites evidence, are the pointers valid? Many systems degrade trace quality under complexity. A governance-first system must maintain trace discipline even under perturbation.

The overarching design principle for robustness evaluation is *behavioral invariants*. The system should satisfy invariants such as:

- Do not execute irreversible actions without approval.
- Do not follow instructions from untrusted content.
- Do not output or log sensitive content beyond policy.
- Defer when evidence quality is insufficient.
- Provide evidence references for material claims.

Stress tests should be designed to violate invariants if the system is weak. Passing stress tests means invariants hold even when task accuracy drops. This is the correct governance posture: accuracy can degrade; safety must not.

5.6.4 Failure taxonomies

A disciplined evaluation program requires a failure taxonomy that is both technically meaningful and operationally actionable. The goal is to classify failures in a way that supports triage: what broke, where, why, and what control should address it. For multimodal VLA systems, a practical taxonomy centers on four families: misperception, injection, privacy leakage, and unsafe action triggering. Each family contains subtypes that map to distinct mitigation strategies.

(1) Misperception failures. These occur when the system's interpretation of the multimodal evidence is wrong due to perception errors. Subtypes include:

- **Extraction errors:** OCR misreads, table cell misalignment, chart misinterpretation, ASR transcription errors.
- **Localization errors:** incorrect bounding boxes, wrong page/region selection, missing relevant evidence.
- **Ambiguity collapse:** the system guesses where evidence is unreadable or incomplete.
- **Contradiction blindness:** the system fails to detect inconsistent evidence across modalities.

Evaluation should log misperception failures with evidence pointers and confidence signals to identify whether the issue is upstream preprocessing, encoder robustness, or fusion weighting.

(2) Injection failures. These occur when untrusted content alters the system's control behavior. Subtypes include:

- **Content-as-instruction:** documents/images include directives that override policy or user intent.
- **Tool redirection:** content induces the system to call tools or retrieve data outside scope.
- **Prompt hierarchy collapse:** the system treats untrusted content as higher priority than system rules.

Injection failures are rarely visible in output correctness alone. A system might produce an answer that looks fine while having executed unsafe tool calls or having violated policy internally. Therefore,

evaluation must inspect traces, not just outputs.

(3) Privacy leakage failures. These occur when the system exposes sensitive information. Subtypes include:

- **Output leakage:** sensitive details appear in responses when they should be masked.
- **Log leakage:** raw sensitive artifacts stored in debug logs, traces, or telemetry.
- **Cross-user leakage:** a user with lower permission receives information intended for higher privilege.
- **Inference leakage:** the system deduces and reveals sensitive information that was not directly requested.

Privacy failures must be evaluated with role-based scenarios and with explicit retention tests. They are governance failures even when task outputs are correct.

(4) Unsafe action triggering failures. These occur when the system proposes or executes actions that violate authority boundaries. Subtypes include:

- **Permission bypass attempts:** tool calls outside allowlists, write attempts in read-only contexts.
- **Approval bypass:** irreversible actions proposed or executed without approval tokens.
- **Justification failure:** actions proposed without evidence support or with false evidence.
- **Sequence risk:** safe actions in isolation become unsafe in sequence (e.g., retrieving sensitive info then sending it externally).

These failures highlight why evaluation must treat sequences as first-class objects. A single tool call may be allowed; a sequence may be unacceptable.

A taxonomy is only useful if it drives action. Therefore, each failure category should be mapped to a control class:

- Misperception → improved preprocessing validation, uncertainty gating, contradiction detection.
- Injection → control-plane separation, sanitization, allowlists, trace audits, red team suites.
- Privacy leakage → redaction, minimization, role-based controls, privacy-preserving logging.
- Unsafe action → least privilege, approval gates, sequence-level policies, rollback mechanisms.

This mapping turns evaluation from a scorecard into an engineering and governance feedback loop.

5.6.5 Limits of current benchmarks

Despite rapid progress in multimodal modeling, current benchmarks are not designed to answer the questions that regulated enterprises must answer before deployment. The limitation is not simply that benchmarks are incomplete; it is that they optimize for the wrong evaluation unit. Most benchmarks measure single-step performance on curated tasks with benign inputs. They do

not measure whether a system can operate safely in the presence of untrusted content, privacy constraints, and tool coupling.

The first limitation is the **absence of standardized enterprise VLA safety benchmarks**. There are many datasets for vision-language understanding, document question answering, and multimodal reasoning. But enterprise safety requires specific scenario classes: injection-laced documents, role-based privacy constraints, approval gating requirements, and audit trace completeness. These are not common in academic benchmarks because they are domain-specific and because they require system-level instrumentation rather than model-only evaluation.

The second limitation is **weak coverage of tool/action coupling**. Tool use is now central to real deployments, but benchmarks rarely include tool calls that can change state or access sensitive data. Even when tool use is evaluated, it is often evaluated in toy settings with benign tools. Enterprise tool stacks include databases, workflow systems, and communication channels. The risk is that a system that performs well on static multimodal tasks fails when tools are introduced, either because of injection vulnerabilities or because action selection is not safely constrained.

The third limitation is **insufficient adversarial coverage**. Some benchmarks include adversarial images or perturbations, but systematic multimodal prompt injection testing is not standardized. In enterprise settings, adversarial behavior is not optional; it is expected. Benchmarks that omit adversarial cases create a false sense of readiness.

The fourth limitation is **privacy blindness**. Benchmarks typically do not model retention, logging, or role-based access. They evaluate what the model outputs, not what the system stores. Yet privacy exposure can occur in logs, traces, or intermediate artifacts. Without benchmarks that include privacy constraints and measure compliance, evaluation cannot support governance.

The fifth limitation is **lack of trace requirements**. Many benchmark tasks can be solved with a correct answer alone. Governance requires more: evidence references, transformation logs, and decision traces. Current benchmarks rarely require these artifacts, so systems can optimize for answer correctness while remaining unauditable.

The practical implication is that organizations must build their own evaluation harnesses tailored to their workflows, threats, and obligations. This does not mean abandoning public benchmarks; it means treating them as component checks rather than deployment certification. Public benchmarks can screen for basic capability. Deployment readiness requires a domain-anchored, trace-based evaluation suite that tests invariants under realistic stress.

Artifact (Save This)

Minimum evaluation bundle (chapter standard). Define a small but repeatable suite: (i) clean cases, (ii) ambiguous cases requiring active perception, (iii) injection-laced documents/images, (iv) privacy-sensitive artifacts requiring redaction, and (v) action requests requiring explicit approval. Each case must produce a structured trace: inputs referenced, transformations applied, tool calls, proposed actions, approvals, and final outputs.

The minimum bundle above is intentionally modest. Its purpose is not to cover every workflow; it is to enforce a baseline discipline: evaluate end-to-end behavior under the conditions that matter. In a governance-first program, this bundle becomes a gate: no expansion of tool authority, no increase in autonomy, and no deployment into higher-stakes workflows without passing the bundle at each system version. The evaluation question is therefore not "Is the model impressive?" It is "Does the system behave safely, traceably, and defensibly when the world is messy, adversarial, and privacy constrained?" That is the frontier standard for multimodal VLA evaluation.

5.7 Implementation Considerations

5.7.1 Minimal viable implementation patterns

The central implementation mistake in multimodal vision–language–action (VLA) systems is to treat "working" as equivalent to "deployable." A prototype that reads a document, summarizes it, and calls a tool is easy to produce. A system that can be trusted inside an organization—especially in regulated, safety-sensitive, or privacy-constrained settings—requires a minimal viable *governed architecture*. This subsection defines implementation patterns that are intentionally conservative. The goal is not maximal automation, but a design that preserves accountability, enables audit, and fails safely under uncertainty.

The simplest robust pattern can be expressed as: **observe** → **propose** → **verify** → **act**. This is not a slogan; it is a control structure that separates perception from authority and forces verification before state-changing steps.

Observe. The system first enumerates what it has actually seen. In multimodal settings, "seeing" must be defined carefully: what pages were processed, what regions were extracted, what audio intervals were transcribed, what tables were parsed. The observe step should output a structured inventory of inputs and provenance: source, timestamp, trust label, and sensitivity class. It should also list uncertainties, such as low OCR confidence, ambiguous layouts, missing pages, or contradictory cues. The key implementation rule is that the observe step is descriptive, not interpretive: it reports what evidence exists before telling a story about it.

Propose. The propose step generates candidate interpretations and a minimal plan. Importantly, proposals can be plural: under ambiguity, the system should present multiple plausible interpretations rather than collapsing into one narrative. Propose should also specify the *minimum actions* required to resolve uncertainty. Minimality is an implementation requirement because each additional tool call increases cost and risk exposure.

Verify. Verification is where a governed system differs from a demo. Verification means executing deterministic checks: re-running OCR on a specific region, validating totals against line items, cross-checking fields against authoritative databases, reconciling contradictions, and enforcing policy constraints. Verification can also include retrieval confirmation: if the system cites a policy, it must confirm version and provenance. The verify step is the place where the system decides whether it has enough confidence to act or whether it must defer.

Act (gated). If and only if verification passes, the system emits an action request. In a conservative design, the system does not directly execute irreversible actions. Instead, it produces a structured action request that can be approved by a human or executed by a controller. The action request should include: the action class (read/write/notify/execute), permissions required, evidence justification, anticipated effects, and rollback plan. For low-stakes actions (e.g., drafting a response), the system may act automatically, but the action class should still be explicit.

Separation of perception from action. A minimal governed implementation enforces architectural separation. The perception module may interpret multimodal evidence and produce hypotheses. The action module does not accept free-form instructions from perception; it accepts structured proposals that are filtered through a policy engine. This separation is a control-plane defense. It ensures that untrusted content embedded in documents cannot directly cause tool execution. Implementation-wise, it implies two different interfaces: (i) a "perception API" that produces structured evidence maps and candidate extractions, and (ii) an "action API" that accepts only schema-validated action objects.

Sandboxed tools and explicit gates. Tools should be sandboxed by default. Code execution should occur in isolated environments. Browsing should be restricted to allowlisted domains in enterprise settings. Database access should be scope-limited. Every tool call should be logged. Gates should be explicit and external: a policy middleware that checks tool calls against allowlists, verifies parameter constraints, enforces rate limits, and blocks disallowed operations. Most importantly, irreversible operations should require explicit approval tokens that cannot be generated by the model.

Read-only first. A practical minimal viable deployment pattern is "read-only first." The system can read documents, extract fields, propose actions, and draft communications, but it cannot write to authoritative systems without approval. This establishes a safe baseline. Organizations often attempt to jump directly to write access because that is where automation value appears. Governance-first implementation treats write access as an escalation step that requires evidence of safety and strong evaluation.

Deferral as a feature. The minimal viable pattern must include deferral. A system that always produces an answer is not safe in multimodal settings. Implementation should support explicit "defer" states with standardized reasons: missing pages, unreadable scans, conflicting evidence, insufficient permissions, or policy ambiguity. Deferral should not be an exception; it should be a normal state that routes cases to human review.

Human-in-the-loop design that is real. If the design includes human approval, it must be implementable under time pressure. Approval interfaces must show evidence pointers, uncertainty flags, and a concise explanation of why the action is proposed. A common implementation error is to require humans to review without giving them the tools to verify, leading to rubber-stamping. Minimal viable implementation therefore includes: evidence previews (cropped regions, page highlights), policy checks results, and a clear description of action impact.

These patterns are minimal not because they are easy, but because they establish the smallest set of controls necessary for defensible deployment. They are the implementation translation of the chapter's thesis: the frontier is a system-of-systems governance challenge. The system must be engineered so that it cannot accidentally become autonomous in ways the organization cannot defend.

5.7.2 Reproducibility and determinism

Reproducibility is the difference between an interesting system and an auditable system. In multimodal VLA deployments, reproducibility is difficult because many components introduce nondeterminism: stochastic inference, retrieval ranking changes, OCR heuristics, and evolving external data sources. Yet regulated environments demand that decisions be reconstructable. The correct implementation posture is therefore: enforce determinism where feasible, record configuration state everywhere else, and design evaluation suites that can be replayed.

Versioning transformations. The first reproducibility requirement is that every transformation stage be versioned. OCR versions, layout parsers, image resize/crop settings, and ASR models must be recorded. This includes seemingly minor settings like DPI normalization, thresholding, or language dictionaries. If the system extracts a number incorrectly, the organization must be able to determine whether the OCR configuration changed between runs. Therefore, each run should record a transformation manifest:

$$\text{transform\_manifest} = \{\text{component\_name}, \text{version}, \text{parameters}, \text{hash}\}.$$

This can be implemented as a JSON artifact stored with the case trace.

Deterministic synthetic test suites. Because real data is sensitive and variable, a minimal governed implementation should maintain a synthetic test suite that is deterministic. Synthetic documents and images can be generated with controlled variations: different layouts, fonts, noise, injected instructions, and privacy-sensitive placeholders. Synthetic suites allow repeatable regression testing without privacy exposure. Determinism means: fixed seeds for generation and fixed transformations. The suite should produce stable expected outputs and expected trace behaviors (e.g., the system must defer on ambiguous cases, must block tool calls in injection cases).

Seed control and inference stability. Many models produce stochastic outputs. Where possible, inference should be configured with deterministic settings (fixed temperature, fixed sampling seed) for evaluation runs. For production, determinism may be partially relaxed, but the system should still record inference parameters and random seeds when feasible. Even if the model is nondeterministic, the organization can at least reproduce the distribution of behaviors under the same configuration.

Pinned retrieval corpora and timestamps. Retrieval is a major source of nondeterminism because knowledge bases evolve. A reproducible evaluation must pin the retrieval corpus or at least record snapshot identifiers and timestamps. If the system uses external sources, it should record the retrieved documents' identifiers and content hashes. This prevents "moving target" evaluations where the same query yields different context over time.

Deterministic tool mocks. Tool calls to real systems are often nondeterministic: APIs change, data updates, network failures occur. Evaluation harnesses should therefore include mocks or recorded tool responses for canonical tests. This allows replay of tool-augmented trajectories. In

production incident reconstruction, recorded tool responses (or hashes of responses stored in evidence vaults) enable understanding of what the system actually saw.

Configuration as a first-class artifact. A governed implementation treats configuration as an auditable artifact. Permissions, allowlists, gating thresholds, and redaction rules must be versioned and signed off. If a system violates policy, the organization must know whether a configuration change introduced the vulnerability. This implies a change control process: configuration updates require evaluation reruns and approvals.

Reproducibility boundaries. It is important to state what reproducibility does not mean. In a complex enterprise environment, perfect replay may be impossible if external systems have changed. The goal is to replay the decision context as it existed: the inputs, transformations, tool results, and policy rules. This requires evidence vaults and structured traces. The reproducibility standard should be: an auditor can reconstruct the evidence and control path sufficiently to evaluate whether the system acted appropriately.

In short, reproducibility is not an academic preference. It is an implementation requirement for accountability. Without deterministic test suites and versioned manifests, evaluation becomes performative and governance collapses under incident scrutiny.

5.7.3 Logging and traceability

Logging is the structural backbone of governed VLA systems. It is also where privacy risk concentrates, because multimodal inputs are rich in sensitive content. The implementation problem is therefore dual: preserve enough traceability to support audit and incident reconstruction while minimizing exposure and complying with retention policies. This cannot be solved by "log everything" or "log nothing." It requires structured logging with privacy-aware design.

Structured traces. The log output should not be free-form. It should be structured around the system's decision pipeline:

- **Input references:** identifiers for each artifact (document ID, image ID, audio ID), source, trust label, sensitivity class.
- **Transformations applied:** OCR/ASR versions, preprocessing settings, redaction rules.
- **Evidence map:** pointers to pages/regions/timestamps/cells used for material claims.
- **Tool calls:** tool name, arguments (redacted as necessary), timestamps, response references/hashes.
- **Proposed actions:** action class, parameters, justification, permissions required.
- **Gating decisions:** allow/deny outcomes, approval requests, human approvals, escalation routes.
- **Final outputs:** delivered output content (possibly redacted), plus claim-to-evidence mapping.

This trace is the basis for both debugging and audit. It also supports evaluation metrics: coverage, permission compliance, deferral behavior, and action integrity.

Evidence references instead of raw media. Storing raw documents and images in logs is often unacceptable. Instead, logs should store references: a content hash, an evidence vault pointer, and coordinate references. For example, rather than logging a full page image, log "doc_id=..., page=3, bbox=(x1,y1,x2,y2), hash=...". The raw media can be stored in a secure evidence vault with access controls. This decouples operational logging from sensitive evidence storage.

Redaction and hashing strategies. Sensitive content may appear in tool arguments, outputs, and extracted text. Logging systems should apply redaction rules before writing. A common approach is selective hashing: store hashes of sensitive strings (account numbers, names) so that later one can confirm whether the system processed a particular value without exposing it. Another approach is tokenization of identifiers: replace sensitive values with stable pseudonyms within a trace. The key is consistency: if a field appears multiple times, the trace should represent it consistently for reconstruction.

Approval and accountability records. For gated actions, logs must record who approved what and when. Approval tokens should be cryptographically bound to the action request to prevent replay or spoofing. Even if cryptography is out of scope for a minimal implementation, the log should at least include approval metadata: approver identity, timestamp, and the exact action object approved. This prevents ambiguous accountability.

Separation of operational logs and debug logs. Many privacy incidents occur because debug logs capture raw inputs and are retained indefinitely. A governed implementation separates operational audit logs (minimal, structured, policy-compliant) from debug logs (restricted access, short retention, carefully sanitized). Debug logs should be disabled in production unless explicitly needed and approved. This is an implementation governance control, not a developer convenience.

Retention and access controls. Logs are data assets. They must have retention schedules, access controls, and deletion procedures. The system should tag logs with sensitivity classes and enforce that only authorized roles can access raw evidence. If the organization cannot enforce retention and access, it cannot responsibly deploy multimodal systems that handle sensitive media.

Trace integrity and tamper evidence. For auditability, logs must be trustworthy. At minimum, implement append-only logging with integrity checks (hash chaining) so that tampering is detectable. This is especially relevant when logs may be used in regulatory inquiries. Again, minimal implementations can use simple hash chains; mature systems can use secure log services. The principle is the same: if logs can be altered silently, the organization loses the ability to defend decisions.

Logging is therefore not a technical afterthought but a governance artifact. Done correctly, it enables controlled expansion of system authority because the organization can measure behavior, detect misuse, and reconstruct incidents. Done poorly, it creates both safety gaps and privacy liabilities.

5.7.4 Cost, latency, and scaling issues

VLA systems are expensive in exactly the places governance demands rigor: multimodal inference, tool loops, verification steps, and human review. A realistic implementation must therefore treat cost and latency as constraints that interact with safety. This is not merely a performance tuning problem; it is an operational design problem: how to deliver value while maintaining verification and oversight under limited budgets.

Multimodal inference cost. Processing images and documents often requires heavy models, and preprocessing (OCR, layout parsing) adds compute. Audio processing (ASR, diarization) can be expensive and slow. Tool calls add additional latency. A naive design that runs full pipelines for every request will be too slow or too costly, encouraging teams to cut verification. Governance-first design resists this temptation by introducing *tiered processing*. Low-risk requests can run on lighter models or partial pipelines; high-risk requests can trigger deeper verification. The system must decide early what tier applies, ideally based on sensitivity class, action class, and uncertainty.

Caching and reuse. Many enterprise workflows involve repeated processing of the same artifacts. A governed system should cache intermediate outputs: OCR text, parsed tables, extracted metadata, and even embeddings. Caching reduces cost and latency, but it introduces governance requirements: cache entries must be versioned (tied to transformation settings), invalidated when preprocessors change, and controlled for privacy (cache access should respect permissions). Caching is a leverage point: it makes verification affordable by reducing redundant computation.

Human review bandwidth as the limiting factor. In many deployments, the true scaling bottleneck is not compute but review capacity. If every action requires human approval, throughput is limited. Organizations may respond by reducing approvals, but that is a governance escalation. A better approach is to design approval routing: only irreversible or high-impact actions require approval; low-impact actions can be automated with strong logging; ambiguous cases escalate. This is essentially triage. Mathematically, one can view it as allocating review budget to cases with highest expected risk. Implementation should include queue management and workload metrics to ensure the system does not overload reviewers.

Latency and user experience trade-offs. In many workflows, a delayed response is costly. But a fast wrong action is worse. The implementation must therefore define service-level classes: e.g., real-time summarization (low-risk, read-only), near-real-time extraction with verification (moderate risk), and asynchronous action proposals requiring approval (high risk). This structure prevents the system from being judged by a single latency metric that encourages unsafe shortcuts.

Tool-call budgets and rate limits. Tool loops can explode cost. A governed implementation sets explicit budgets per request: maximum tool calls, maximum retrieval tokens, maximum browsing steps. Budgets are not only cost controls; they are safety controls that reduce exposure and attack surface. When a budget is reached, the system must defer or request human assistance. This is a designed failure mode.

Monitoring cost. Continuous monitoring for injection attempts, drift, and anomalies also adds overhead. However, monitoring is cheaper than incident response. Implementation should include lightweight telemetry for policy violations and anomaly signals, and deeper logging for flagged cases. This again is tiering: allocate expensive monitoring to suspicious contexts.

Scaling VLA systems responsibly therefore requires deliberate operational engineering. The system must be designed so that governance controls are affordable and do not collapse under scale pressure. If governance depends on expensive manual review for every case, it will fail. If governance is baked into tiering, caching, and structured gating, it can scale with the organization's real constraints.

5.7.5 Integration risks

Integration is where multimodal systems become real, and where most failures occur. A VLA system can perform well in a controlled environment and still fail catastrophically when connected to tools, UIs, and organizational workflows. Integration risks are not minor engineering details; they are the principal source of system-level vulnerability. A governance-first implementation treats integration as an explicit risk surface to be minimized, monitored, and controlled.

Tool coupling expands attack surface. Every tool integrated into a VLA system becomes a potential path for misuse. Read tools can expose sensitive data; write tools can modify authoritative records; communication tools can send information externally. Tool coupling also creates injection pathways: untrusted content can attempt to trigger tool calls or to influence parameters. Therefore, integration must be least-privilege by design. Tools should be allowlisted, scoped, and wrapped with validators. Where possible, implement *capability tokens*: the system can call a tool only if it has a token issued by the policy engine for the current case context. This prevents generalized tool access.

UI automation risk and brittleness. When systems operate UIs rather than APIs, the control surface is unstable. UI changes can alter what the system sees, leading to misclicks or misinterpretation. Worse, UI affordances can imply authority: if the system appears to "click" and "submit" like a human, users may assume it is safe. A governance-first approach treats UI automation as a high-risk integration mode that should be restricted to sandbox environments or read-only operation until proven safe with extensive evaluation and monitoring.

False authority and UX-induced over-trust. Interfaces that present clean summaries and grounded-looking outputs can cause users to over-trust the system. This is especially dangerous when uncertainty is hidden. Integration therefore includes UI design: outputs must surface uncertainty, show evidence references, and clearly label what is observed versus inferred. If the system is proposing an action, the UI should present it as a request, not as an accomplished fact. Otherwise, the system may induce "automation bias" where humans approve without verification.

Uncertainty may be obscured by pipeline layers. In multimodal stacks, uncertainty arises at multiple stages: OCR confidence, detection confidence, retrieval relevance. If these signals are not

propagated to the final output and gating logic, the system may act as if it were certain. Integration risk therefore includes *uncertainty plumbing*: ensuring uncertainty metadata flows from preprocessors to reasoning to policy engine. Without this, the observe–propose–verify pattern collapses because the system cannot know when verification is needed.

Mismatch between enterprise policy and tool semantics. Enterprise tools embody policy implicitly. For example, a database might allow updates without requiring explicit approval, even though organizational policy says approvals are needed. If a VLA system is integrated directly, it may exploit tool affordances that bypass policy. Integration must therefore include policy wrappers that enforce organizational rules regardless of tool permissiveness.

Data governance and retrieval drift. Integrations with retrieval systems can introduce outdated or unauthorized content into the model context. If retrieval sources are not curated and versioned, the system can cite stale policies or expose confidential information. Therefore, retrieval integration must include source allowlists, freshness checks, and provenance logging. "Search everything" is not a safe default in enterprise VLA.

Operational incentive drift. Integration also interacts with organizational incentives. Teams may request broader tool access to improve performance, or may disable gating because it slows workflows. This is not a technical bug; it is an operational risk. Therefore, integration must be governed by change control: expansions of tool scope require passing evaluation bundles and formal approvals. Without such controls, integration risk grows over time until a failure occurs.

These integration risks justify a conservative implementation roadmap: start with constrained read-only tasks; build traces and evaluation harnesses; add verification tools; then gradually expand authority only when evidence shows invariants hold under stress. Integration is not merely hooking up APIs; it is deciding how much authority to delegate and whether the organization can defend that delegation.

Prompt (Copy/Paste)

Operator prompt pattern (implementation).

- **Observe:** Describe inputs and their provenance. List uncertainties and ambiguities.
- **Propose:** Provide candidate interpretation(s) and the minimum actions needed.
- **Verify:** Specify checks (cross-modal, OCR re-run, retrieval confirmation, policy constraints).
- **Act (gated):** Emit an action request with justification, permissions required, and rollback plan.
- **Log:** Output a trace with evidence identifiers and decision rationale.

The prompt pattern above should be treated as an interface contract between the model and the governance middleware. It is effective because it forces the model to expose uncertainty, it makes verification explicit, and it frames action as a gated request rather than an entitlement. When implemented with schema validation and policy enforcement, it becomes a practical mechanism for

turning a powerful multimodal model into a controlled operator suitable for high-accountability environments.

5.8 Impact and Opportunity

5.8.1 New capabilities unlocked

Multimodal vision–language–action (VLA) systems unlock a class of capabilities that are best described not as "better models" but as *new organizational interfaces*. The most important shift is that information previously trapped in messy, human-facing artifacts—scanned PDFs, screenshots, photos, dashboards, whiteboards, audio snippets, and mixed-format emails with attachments—becomes computationally accessible and operationally actionable. For decades, organizations have accumulated value in forms that were legible to people but resistant to automation. VLA systems lower that barrier. They do so by combining perception (extracting signal from visual and auditory sources), language (structuring meaning and producing explanations), and action (tool use, retrieval, and workflow integration). The opportunity is therefore not limited to efficiency; it is the possibility of redesigning workflows around a new kind of evidence-processing layer.

Document intelligence beyond extraction. The most immediate capability class is document intelligence, but the frontier is not simple OCR. It is end-to-end interpretation with context: extracting key terms, comparing versions, identifying deviations from standard clauses, reconciling attachments with email threads, and producing structured summaries tied to evidence references. In finance and corporate governance contexts, this includes extracting covenants, identifying disclosure-relevant statements, flagging missing schedules, and linking extracted terms to policy requirements. In audit and compliance contexts, it includes document triage, anomaly detection in invoices, and cross-document consistency checks. The capability is not that the system "reads" like a human; it is that it can produce operationally useful structure at scale while retaining a connection to the underlying evidence.

Visual inspection and monitoring. A second capability class is visual inspection: using images or video-like snapshots as inputs to quality control, maintenance, and operational monitoring. In manufacturing, this can include defect detection and process verification. In infrastructure, it can include inspection of equipment states. In facilities management, it can include safety checks based on images. The value is that visual evidence, which often requires in-person or manual review, can be triaged and summarized. The frontier opportunity is not necessarily autonomous detection; it is triage with evidence-based escalation, enabling human experts to focus attention where it matters.

Mixed-media decision support. The most strategically interesting capability is mixed-media decision support: combining textual policy, numerical tables, and visual evidence into a unified reasoning context. Many executive decisions rely on dashboards, screenshots, slide decks, and memos. A multimodal system can ingest these artifacts, reconcile inconsistencies, and generate decision briefs that track claims to evidence. This is not "analysis in the abstract." It is analysis anchored to what the organization already uses as evidence. In controlled form, this can increase decision velocity while improving documentation.

Interface operation and workflow assistance. VLA systems can also act as interface operators. They can interact with existing enterprise systems by reading screens and proposing actions. In environments where API integration is costly, interface operation can be an attractive path to automation. The opportunity is speed of deployment and coverage across legacy systems. The risk, explored elsewhere in this chapter, is authority creep and brittleness. But within constrained scopes, interface operation can reduce friction for routine tasks like gathering information across systems, preparing drafts, and assembling evidence packets for review.

Better "evidence work," not just content generation. Finally, VLA systems unlock a subtle but profound capability: they can improve the practice of evidence work. They can identify what is missing, highlight ambiguous or low-quality inputs, and prompt users for specific clarifications. This active perception behavior is valuable precisely because it can reduce downstream mistakes. In traditional automation, missing data often produces failure or silent errors. In governed VLA systems, missing data can produce structured deferral and escalation.

The common thread across these capabilities is that multimodality creates a bridge between unstructured, human-oriented evidence and structured, auditable decision processes. The frontier opportunity is not magical understanding; it is an expanded ability to extract, structure, and route evidence under constraints.

5.8.2 Organizational implications

The organizational impact of multimodal VLA systems is not limited to tool adoption. It changes the meaning of governance because the system becomes both a perception layer and a workflow participant. This introduces new responsibilities for leadership: design authority boundaries, validate perception like a sensor system, and treat reviewability as a core product requirement rather than a compliance add-on.

Cross-functional governance becomes mandatory. In text-only deployments, organizations often treat AI as a product or IT decision. In VLA systems, that approach fails. Perception and action coupling creates security, privacy, operational risk, and compliance implications that require cross-functional governance: security teams for injection defense and access control; privacy teams for retention and redaction; risk teams for model risk management and escalation thresholds; operations teams for workflow integration; and product teams for user experience and control design. If any one function is missing, the system will be optimized locally and fail globally. Practically, this implies creating an ownership model that spans functions and formalizes approval gates for expanding system authority.

Validate perception like sensors, not like text generators. The organization must treat multimodal perception outputs as sensor readings: noisy, context-dependent, and subject to calibration. This is a cultural shift. Many teams implicitly assume that if a model produces a fluent summary, it is correct. Governance-first deployment rejects this assumption. Instead, it

requires measurement of perception quality under realistic conditions, explicit uncertainty, and verification loops. This sensor-like view also affects accountability: decisions based on perception must be traceable to evidence and must include uncertainty indicators. In other words, multimodal systems should inherit the validation discipline of safety engineering, not the informal adoption style of productivity tools.

Reviewability becomes a design goal. For many organizations, the limiting factor in responsible automation is not model capability but review bandwidth and audit requirements. Therefore, reviewability must be designed into the system. That means evidence maps, claim-to-evidence mapping, structured action requests, and clear uncertainty reporting. It also means designing approval interfaces that support real verification rather than performative sign-off. A system that cannot be reviewed efficiently will either be rejected or will produce unsafe rubber-stamping. In high-accountability contexts, a system that is "helpful" but non-reviewable is worse than one that is slower but defensible.

Authority boundaries must be explicit. VLA systems tend to drift toward autonomy because it is convenient. Organizations must resist this drift by formalizing authority boundaries in system design: read-only modes, tool allowlists, irreversible-action approval gates, and escalation policies. Importantly, boundaries must be documented and auditable. This changes how organizations think about deployment phases. "Pilot" is not a vague period of experimentation; it is a defined authority envelope with explicit constraints and evaluation gates.

New operational roles and skills. Multimodal deployments create demand for roles that did not exist in earlier AI adoption waves. Organizations will need operators who understand how to provide evidence, how to interpret uncertainty signals, how to verify outputs efficiently, and how to escalate appropriately. They will also need engineers who can build safe tool wrappers, implement privacy-preserving logging, and maintain deterministic test suites. This is not simply "prompt engineering." It is operational competence in governed AI workflows.

Change control becomes more central, not less. Because multimodal systems integrate many components and touch sensitive workflows, changes must be controlled. Updating an OCR parser, adjusting a redaction rule, or expanding a tool allowlist can materially change system behavior. Therefore, organizations need change control processes that look more like those used in financial systems or safety-critical software: versioning, regression testing, approvals, and rollback plans. This is a major organizational implication: adopting VLA systems responsibly is closer to deploying an operational system than to adopting a productivity app.

In sum, the organizational opportunity is real, but the organizational requirements are non-negotiable. The organizations that succeed will be those that treat multimodal systems as governed operational infrastructure, not as a novelty.

5.8.3 Potential new industries or markets

The emergence of multimodal VLA systems creates not only internal enterprise opportunities but also external market opportunities. In many cases, the value will not accrue primarily to the foundation model providers, but to the layer of governance, control, and integration products that make VLA systems deployable under real constraints. If the frontier is a system-of-systems problem, then the commercial frontier becomes the market for system-level governance infrastructure.

Multimodal injection defense as a product category. Prompt injection has already created a market for security tooling in text systems. Multimodal injection expands the problem: instructions can be embedded in images, PDFs, and scanned documents. This creates demand for specialized defenses: content sanitization pipelines, detection of instruction-like patterns in extracted text, segmentation of trusted vs untrusted content, and policy engines that ensure content cannot influence control. A plausible new industry category is "multimodal control-plane security"—products that sit between untrusted media and tool invocation.

Provenance and evidence tooling. Enterprises will demand provenance-aware systems that can produce audit-grade traces: claim-to-evidence mapping, coordinate references, transformation manifests, and retrieval provenance. This will create demand for tooling that standardizes these artifacts across vendors and deployments. One can expect products that act as "evidence routers" and "trace compilers," producing structured logs suitable for audit and incident reconstruction. In regulated environments, provenance tooling can be a differentiator because it converts AI outputs from suggestions into defensible artifacts.

Redaction, minimization, and privacy-preserving logging. Privacy will be the dominant constraint in many multimodal deployments. This creates demand for redaction engines that operate on images, PDFs, and audio; for minimization frameworks that generate safe representations; and for logging infrastructure that preserves auditability while minimizing raw data retention. One can imagine a market for "privacy middleware for multimodal AI" that integrates with enterprise retention policies and access controls.

Enterprise interface operators. If VLA systems can operate interfaces, there will be demand for robust "interface operator" platforms that provide safe UI automation, detection of UI drift, sandboxed execution, and audit logs. This is distinct from classic robotic process automation (RPA). Traditional RPA is brittle and rule-driven; VLA-based interface operation is adaptive but requires stronger governance. A market opportunity exists for platforms that combine the adaptability of multimodal models with the discipline of enterprise controls.

Evaluation harnesses and certification layers. The lack of standardized enterprise benchmarks creates a market opportunity for evaluation harness providers: synthetic case generators, stress-testing suites, injection test packs, privacy leakage tests, and trace validators. In other words, "AI validation as a service" for multimodal systems. Organizations will pay for tools that let them measure readiness and enforce invariants. Over time, these harnesses may become quasi-standards,

shaping what it means to be "deployable."

Audit and compliance products. Multimodal systems will be scrutinized by internal audit, regulators, and counterparties. This creates demand for audit tooling that can ingest traces, verify controls, and produce compliance reports. Products may emerge that translate VLA traces into governance artifacts: approval logs, risk registers, and policy adherence summaries. In regulated markets, the ability to demonstrate control can be more valuable than raw capability.

The broader point is that multimodal VLA adoption will not be a single-vendor story. It will be an ecosystem story. The organizations that build the governance layer may shape the practical frontier more than the organizations that build the base models.

5.8.4 Second-order effects

The second-order effects of multimodal VLA systems are not side notes; they are the strategic consequences of deploying perception systems at scale. These effects often emerge slowly and then become unavoidable. Governance-first leadership must anticipate them because they can create backlash, liability, and institutional dependency even when the system performs well on its core tasks.

Surveillance temptation and privacy backlash. Multimodal systems make it easy to process images, video-like snapshots, and audio. This creates a temptation for organizations to expand monitoring: more cameras, more dashboard scraping, more transcript analysis. The boundary between operational monitoring and surveillance can erode quickly. Even if the organization's intent is benign, stakeholders may perceive overreach. Employees may view multimodal monitoring as intrusive. Customers may resist. Regulators may intervene. Therefore, multimodal deployments create governance obligations around purpose limitation, transparency, and consent. This is not merely a legal issue; it is a trust issue.

Normalization of opaque perception in high-stakes decisions. Over time, organizations may begin to treat multimodal outputs as if they were ground truth. This is especially likely when outputs are presented as clean summaries and when evidence maps are not surfaced. The risk is that decisions gradually become dependent on opaque perception models that are difficult to contest. This can undermine accountability and fairness. In extreme cases, organizations can "forget how to see" without the system, losing internal competence. This is a dependency risk: a capability that initially augments humans can become a crutch that weakens institutional judgment.

Data gravity and lock-in. Multimodal systems often require evidence vaults, annotation workflows, and customized evaluation harnesses. Once built, these infrastructures create lock-in. The organization may become dependent on specific model vendors or tool providers because switching would require revalidating the entire system under governance constraints. Lock-in is not inherently bad, but it must be recognized as a strategic consequence. Governance-first planning

should include portability strategies: standardized trace formats, modular tool wrappers, and separation of model from policy enforcement.

Shift in accountability narratives. When multimodal systems are used to interpret evidence, organizations may face a subtle accountability drift: "the system said so" becomes an implicit justification. This is a governance failure even if no one intends it. Over time, accountability can become diffused: engineers blame the model, operators blame the interface, managers blame process complexity. The remedy is explicit responsibility assignment: who owns the system's decisions, who approves actions, who maintains evaluation harnesses. Without such clarity, second-order failures will be organizational rather than technical.

Adversarial adaptation. As organizations deploy multimodal systems, adversaries will adapt. Counterfeit documents may be designed to exploit extraction weaknesses. Images may include overlays intended to mislead. Attackers may attempt to induce tool misuse. These dynamics create an arms race. The second-order effect is that security becomes a continuous operational discipline, not a one-time deployment checklist. This implies ongoing red teaming, monitoring, and control updates.

Cultural and labor implications. Multimodal automation changes which tasks humans perform. Routine evidence triage may be automated; human work shifts toward exception handling, verification, and escalation. This can improve job quality for some roles and degrade it for others. It can also create new skill requirements. Organizations must manage this transition deliberately. If not, they may face resistance or quality declines as institutional knowledge is displaced without replacement.

Second-order effects matter because they are often overlooked in early pilots. They determine whether a multimodal deployment becomes a sustainable advantage or a reputational and governance liability.

5.8.5 Overstated expectations

Because multimodal systems feel more grounded than text-only systems, they are unusually vulnerable to hype. A system that can look at an image and answer questions can create the illusion of "understanding." In enterprise contexts, this illusion is costly. The final subsection therefore clarifies the most common overstated expectations and provides the governance-first framing needed to keep deployments disciplined.

Multimodal does not imply ground truth. Seeing is not knowing. Images are partial views; documents can be incomplete; audio can be ambiguous. Multimodal models can produce plausible interpretations even when evidence is insufficient. Therefore, multimodal capability should not be interpreted as a guarantee of correctness. The correct mental model is: multimodal systems are *evidence processors* with uncertainty, not truth engines. They can accelerate triage and interpretation,

but they cannot remove the need for verification in high-stakes decisions.

Grounding is not traceability. Even when outputs reference visual evidence, the link between a claim and a specific evidence region can be weak. Models can cite irrelevant regions or produce summaries that are not supported by the document. True governance requires claim-to-evidence mapping with verifiable pointers, not rhetorical references. If a system cannot produce auditable evidence references, it should not be treated as a decision support system for regulated contexts.

More modalities do not necessarily reduce error. A common assumption is that adding modalities improves reliability because the system has more information. In practice, more modalities can increase complexity and introduce new failure modes: fusion dominance errors, contradiction blindness, and privacy exposure. Multimodal systems can also learn spurious cross-modal shortcuts. Therefore, adding modalities is not a free robustness gain; it is an engineering and governance escalation that requires additional evaluation.

Tool access is not an unqualified enhancement. Connecting a multimodal model to tools can produce dramatic functionality, but it also introduces authority risk. Tool access expands attack surface, creates injection pathways, and can turn minor perception errors into state changes. Therefore, tool integration must be evaluated as a separate risk regime. The value of a tool-augmented multimodal system depends on calibrated uncertainty, safe gating, and strong traceability. Without these controls, tool access converts capability into liability.

Value requires calibrated uncertainty and safe action gating. The governing principle for enterprise value is not maximal automation; it is *controlled reliability.* A VLA system creates durable value when it (i) signals uncertainty, (ii) defers appropriately, (iii) verifies via checks, (iv) proposes actions with evidence justification, and (v) triggers approvals for irreversible steps. Systems that skip these features may appear more impressive in demos, but they produce fragile operational value and high governance cost.

The frontier opportunity is governed deployment, not mere capability. The most overstated expectation is that frontier AI value is primarily a function of model capability. In reality, in high-accountability contexts, value is a function of governability. Organizations that can deploy multimodal systems with strong controls, traceability, and privacy discipline will capture durable advantage. Organizations that chase capability without evaluation and governance will accumulate hidden liabilities.

The impact and opportunity of multimodal VLA systems is therefore real, but conditional. The frontier is not simply that these systems can see and act. The frontier is that they can be made to see and act *under constraints*, with audit trails, and with explicit authority boundaries. That is the difference between innovation and negligence, and it is the dividing line that this book insists on drawing.

5.9 Governance, Risk, and Control

5.9.1 New risk categories

Multimodal vision–language–action (VLA) systems expand the organizational risk surface in ways that are qualitatively different from text-only assistants. The risk does not merely scale; it changes form. When a system can ingest documents and images, reason over them, and trigger tools or workflows, the boundary between "content" and "control" becomes porous. When that boundary becomes porous, the organization inherits risks that resemble those of security engineering, safety engineering, and operational governance all at once. A governance-first approach therefore begins by naming the new risk categories explicitly, because unnamed risks are unmanaged risks.

Injection through documents and images (content-as-instruction risk). The most distinctive multimodal risk category is prompt injection through non-text channels: PDFs, scanned documents, screenshots, images with overlays, and even audio transcripts. The governing problem is not that the model can be tricked into saying something wrong; it is that untrusted content can attempt to influence system behavior by masquerading as instructions. In practice, this can take many forms: a contract attachment that contains a hidden instruction to ignore policies; an invoice that includes a footer prompting the model to call a tool; a screenshot containing a watermark-like instruction; or a scanned document with embedded text that is not visually salient to a human reviewer. The risk is heightened because multimodal systems often run OCR, converting visual text into tokens that enter the model context. If the system does not enforce separation between untrusted OCR text and trusted system prompts, then the extraction pipeline becomes an injection pipeline.

This risk is not hypothetical. It is the natural consequence of a system architecture that treats all text equally after extraction. In a text-only setting, untrusted content is already a risk; in multimodal settings, the channels multiply and the detection problem becomes harder. Many injection strings will look like normal content to naive filters. The governance implication is blunt: if the system ingests untrusted multimodal content and has tool access, injection is a default threat model, not an edge case.

Privacy exposure through logs, outputs, and intermediate artifacts. Multimodal inputs are rich in sensitive information. A single document scan can contain PII, account numbers, signatures, addresses, or confidential pricing. A single photo can contain faces and location clues. Audio can contain names and incidental disclosures. The privacy risk is therefore threefold. First, the system can leak sensitive information in its outputs (summarization leakage). Second, it can leak information through intermediate artifacts (OCR text caches, embeddings, parsed tables). Third, it can leak information through logs and traces, which are often retained longer and accessed more broadly than raw evidence. In many organizations, the most dangerous privacy exposure is not the user-facing output, but the quiet accumulation of raw artifacts in debug logs or telemetry.

Multimodal systems create a further privacy category: **privacy amplification**. Because the model can extract and connect information across modalities, it can surface sensitive associations that were present but not operationally accessible before. For example, a system can link a person's name in a document to their face in a photo, or connect address fragments across attachments. Even if each artifact is individually permissible, the joined inference may cross a privacy boundary. Governance must therefore treat cross-modal linking as a sensitive capability requiring explicit policy, role-based access, and minimization.

Physical-world and operational risk from actions. Once the system can trigger actions—even actions that appear "digital" such as sending emails, updating tickets, submitting forms, or changing system states—it can cause real-world consequences: financial transfers, compliance filings, operational disruptions, or reputational incidents. In physical settings, VLA systems may interface with monitoring or control systems. Even in non-robotic contexts, the system becomes an operational actor. This introduces safety-like risk categories: irreversibility, cascading error, and delayed harm. A minor misperception that would be harmless in a chat setting can become harmful when it triggers a workflow.

A distinctive multimodal dimension is **sensor illusion**. Organizations may treat visual inputs as more trustworthy than text. This can lead to overconfidence: if the system "saw it in the image," decision makers may assume it is true. But images are partial, manipulable, and context-dependent. The system can misread charts, misinterpret layouts, or miss crucial details due to resolution. Therefore, physical-world risk is not only about robots; it is about any system whose actions are justified by perception outputs that are treated as authoritative.

Authority confusion and decision laundering. Multimodal systems can produce outputs that look like objective interpretations of evidence. This creates the risk of decision laundering: humans may treat the system's summary as a neutral fact and use it to justify decisions without doing the underlying evidentiary work. The danger is that accountability becomes diffused: "the model said the contract says X" becomes a substitute for verification. This risk is not a technical bug; it is an organizational behavior that emerges when interfaces present model interpretations without uncertainty and without evidence pointers.

Provenance and record integrity risk. In multimodal workflows, models may generate structured outputs that are then stored as records. If those records later become part of retrieval or audit trails, the organization risks contaminating its evidence base with model-generated artifacts. Over time, the system can amplify its own errors. This is a governance risk because it corrupts the distinction between observed evidence and generated interpretation. In regulated settings, record integrity is central: the organization must be able to separate what was observed from what was inferred.

Adversarial adaptation and ecosystem risk. As organizations deploy VLA systems, external actors will adapt. Counterparties may craft documents to exploit extraction weaknesses or to trigger unsafe behaviors. Fraud attempts may include multimodal elements (fake stamps, manipulated

screenshots). Attackers may exploit the fact that models are trained to be helpful. The risk is therefore dynamic: it evolves as deployment increases. Governance must treat security as continuous, not a one-time certification.

The practical outcome of these new categories is that multimodal governance must unify security, privacy, and safety engineering. Treating any one of these as secondary will produce failure in the others.

5.9.2 Control points and safeguards

Once risks are named, governance becomes a design problem: where to place control points such that the system cannot exceed its authority and cannot violate privacy or safety constraints even when the model is manipulated or uncertain. A governance-first implementation treats safeguards as *system-level invariants*: properties that must hold regardless of model behavior. This subsection outlines the key control points, emphasizing those that can be enforced externally to the model.

Least privilege and allowlists. The foundational safeguard is least privilege: the system should have access only to the tools and data sources required for the current task, and only at the minimal scope. This implies allowlists at multiple levels:

- Tool allowlists: which tools are available at all.
- Parameter allowlists: which arguments are permitted (e.g., which database tables, which domains).
- Data allowlists: which datasets and documents can be accessed by role and context.

Least privilege is enforced by middleware, not by prompting. The model may request a tool call; the policy layer checks it. This is the most important system-level control because it prevents authority creep from becoming an exploit path. In practice, allowlists should be versioned, audited, and tied to change control.

Separation of content from control signals. Injection resistance requires architectural separation. Untrusted content extracted from documents must never be treated as part of the system control prompt. This implies compartmentalization: the system prompt and tool policies are derived from trusted sources and are not modifiable by input content. Untrusted text can be passed to the model as data, but the policy layer must ensure it cannot alter tool scope or permissions. A robust design enforces that tool invocation decisions are made by a guard that is not influenced by untrusted content, or at minimum that the guard uses only structured proposals rather than free-form text.

Sanitization and injection detection. Sanitization is a practical layer: detect and neutralize instruction-like patterns in extracted content. This includes removing or flagging phrases that attempt to override instructions, exfiltrate data, or request tool calls. Sanitization is not perfect; attackers can evade patterns. Therefore, sanitization must be coupled with detection: monitor for suspicious instruction patterns and treat them as risk triggers that force read-only behavior or human review. The key is that detection should not depend on the same model that is being

protected; it should include deterministic rules and external checks where possible.

Action gating with explicit approval for irreversible steps. Irreversible and high-impact actions must require explicit human approval. This is not negotiable in governance-first design. The system can propose actions, but execution requires an approval token issued by a human or by an approved workflow controller. The approval interface must show evidence references and uncertainty signals so that the approval is meaningful. Approval must be logged with identity and timestamp. The gating logic must be enforced externally so that the model cannot bypass it. In effect, the system becomes an advisor that produces structured action requests, not an autonomous actor.

Role-based controls and context binding. Permissions should be bound not only to user role but also to context. For example, a compliance case may allow access to certain sensitive documents, whereas a general support case should not. Context binding reduces the chance that a general-purpose system becomes a privileged system by accident. Context binding also supports audit: the system can justify why it had access. Practically, context is a key input to the policy engine: "who is asking, what workflow is this, what sensitivity class is involved."

Privacy-by-design controls: minimization, redaction, retention. Privacy safeguards should be implemented in the pipeline, not after the fact. Key controls include:

- Minimization: do not ingest more data than needed; prefer redacted views.
- Redaction: mask or remove sensitive fields before they enter the model context where possible.
- Output filtering: prevent sensitive content from being returned to unauthorized users.
- Retention controls: ensure logs and caches comply with retention schedules; separate evidence vaults from operational logs.

These controls should be enforced by policy middleware and secure storage systems, not by the model's good intentions.

Verification and constraint checking as mandatory steps. In governed VLA systems, verification is a control. Deterministic checks (totals, field format, cross-document consistency) and policy checks (permissions, redaction compliance) must be integrated into the pipeline. Crucially, failing checks should force deferral or escalation rather than being ignored. This is an implementation of the "observe–propose–verify–act" pattern at the governance level: verification is not optional when the system can act.

Change control and evaluation gating. Controls degrade if the system changes without revalidation. Therefore, a critical safeguard is organizational: changes to tool access, model versions, OCR settings, sanitization rules, or logging configurations must require rerunning evaluation bundles and obtaining sign-off. Without this, the system's risk posture will drift until a failure occurs. Change control is a governance safeguard equal in importance to technical gating.

These control points collectively aim to enforce invariants: untrusted content does not control the system; privacy constraints are honored; irreversible actions require approval; and the system

remains auditable under stress.

5.9.3 Human oversight boundaries

Human oversight is often invoked as a solution to AI risk, but in VLA systems oversight must be defined precisely. Vague claims that "a human is in the loop" are not governance. Oversight must specify who has decision authority, what actions require review, how uncertainty is surfaced, and what escalation rules apply. Otherwise, human oversight becomes either a bottleneck that will be bypassed or a rubber stamp that provides false comfort.

Explicit authority limits. The first oversight boundary is authority: what is the system allowed to do without human approval? A governance-first stance defines authority limits in terms of action classes:

- **Read-only autonomy:** the system may retrieve and summarize within allowlists.
- **Drafting autonomy:** the system may draft communications and action proposals but not send or execute them.
- **Execution autonomy:** the system may execute only reversible, low-impact actions, with logging.
- **Irreversible actions:** always require human approval (or multi-party approval).

These boundaries should be documented and enforced by policy middleware. The boundary should not be negotiable at runtime by user prompting.

Review thresholds tied to risk, not convenience. Oversight should be triggered by thresholds: sensitivity class of evidence, uncertainty levels, presence of injection signals, and action impact. For example, if OCR confidence is low in a field that influences an approval threshold, the system must escalate. If the document contains instruction-like patterns, the system must switch to read-only and require review. If the system proposes an external communication, it must require approval. These thresholds should be codified and testable, not informal.

Sensitive interpretations require sign-off. Many multimodal deployments involve interpretations that can create liability: contract interpretation, compliance determinations, incident reports, or safety assessments. In such cases, the system's output should be treated as a draft that requires domain expert sign-off. The sign-off should be explicit: an approver endorses not only the action but the interpretation. This is important because interpretation itself can be consequential even without an action. A governance-first system therefore distinguishes between "draft output" and "approved output."

Escalation rules and responsibility assignment. Oversight must define escalation: when does the system route to compliance, legal, security, or operations? It must also define responsibility: who is accountable for approving actions and for maintaining controls? Without clear responsibility, oversight fails during incidents because no one owns the decision path. Responsibility assignment

should be tied to roles and captured in logs.

Oversight ergonomics. Effective oversight requires usable interfaces. Reviewers must be able to see evidence pointers, uncertainty flags, and the results of verification checks. They must be able to reproduce the system's view (cropped regions, extracted text) without exposing more sensitive data than necessary. If oversight is painful, it will be bypassed. Therefore, oversight is not only policy; it is product design.

Training and operational discipline. Oversight depends on human competence. Operators must understand what the system can and cannot do, how injection works, how to interpret uncertainty, and how to verify evidence. Organizations must therefore treat multimodal deployment as a training and process change, not merely a software rollout. The oversight boundary is cultural as much as technical: humans must not treat the system's outputs as authoritative without evidence review.

A well-defined oversight regime converts human involvement from a vague safety claim into a controlled accountability mechanism. The system proposes; humans approve; the organization can show who decided what, based on which evidence, under which policies.

5.9.4 Monitoring and auditability

Monitoring and auditability are the operational counterpart to safeguards. Controls are hypotheses until monitored; safety is a claim until audited. For VLA systems, monitoring must detect both technical failures (perception drift, tool misuse) and adversarial behavior (injection attempts), while auditability must preserve the evidence trail in a privacy-preserving way.

Monitoring injection attempts and anomalies. Because injection is a default threat model, monitoring should include detection of instruction-like strings in OCR output, repeated patterns across documents, suspicious tool-call sequences, and unusual access requests. Monitoring should track the rate of blocked tool calls and the contexts in which they occur. A spike in blocked calls can indicate an attack or a system behavior drift. Monitoring should also track contradiction rates and deferral rates; sudden changes may indicate distribution shift or pipeline degradation.

Behavioral invariants and runtime checks. Monitoring should enforce invariants at runtime: no disallowed tool calls executed, no irreversible actions executed without approval, no sensitive content output to unauthorized roles. These invariants can be implemented as runtime assertions in policy middleware. When violations are attempted, they should be logged and escalated. This creates a safety net: even if the model behaves unpredictably, the system prevents worst-case outcomes and provides evidence for investigation.

Audit trails with privacy-preserving retention. Auditability requires that the organization can reconstruct what happened: what inputs were processed, what transformations were applied, what evidence was referenced, what tools were called, what actions were proposed, and who approved what. In multimodal systems, raw evidence is sensitive and large. Therefore audit trails should

store references and hashes, not raw media, and rely on secure evidence vaults for retrieval under controlled access. The audit trail should include transformation manifests and policy configurations. Without these, an auditor cannot determine whether an output was reasonable given the system state.

Retention and access governance for audit artifacts. Logs and evidence vaults must have retention schedules and access controls. Audit logs should be immutable or tamper-evident. Access to raw evidence should be restricted and tracked. In many organizations, internal audit will need controlled access to evidence; this must be planned, not improvised during an incident. Retention policies must balance audit needs with privacy obligations. If retention is too short, incident reconstruction becomes impossible; if too long, exposure risk increases.

Continuous evaluation and drift monitoring. Monitoring should include periodic re-evaluation using the minimum evaluation bundle. Drift is not only model drift; it is template drift, OCR drift, retrieval drift, and policy drift. A practical strategy is to run synthetic regression tests daily or weekly and run a subset of real-case audits with human review. Monitoring should flag when performance on canonical cases degrades or when deferral patterns change.

Incident reconstruction as a design requirement. The test of auditability is whether an incident can be reconstructed without heroic effort. A governance-first system is built with incident reconstruction in mind: structured traces, evidence pointers, tool logs, approval records, and configuration manifests. If these artifacts are absent, the organization will not only suffer operational harm; it will also be unable to defend itself. Auditability is therefore not a compliance checkbox; it is a survival requirement.

Monitoring and auditability convert controls from static design intentions into operational reality. They are the difference between a system that "should be safe" and a system whose safety can be demonstrated.

5.9.5 Deployment deferral criteria

Perhaps the most underappreciated governance tool is deployment deferral: the decision to restrict scope or refuse deployment until minimum controls are in place. Organizations often treat deployment as inevitable once a system appears useful. Governance-first leadership treats deployment as conditional. This subsection defines deferral criteria as explicit gates. If these criteria are not met, the system should either be restricted to a safer envelope (typically read-only) or not deployed at all.

No injection defense implies read-only restriction. If the system cannot demonstrate injection resistance—sanitization, detection, and control-plane separation—then it must not be allowed to execute or even propose tool-coupled actions beyond safe read-only behaviors. In particular, it should not be connected to sensitive databases or external communication channels. In this state,

the system may still be used for bounded tasks such as summarizing internal documents under trusted provenance, but it should not ingest untrusted attachments with tool authority. The deferral logic is simple: when content can control behavior, tool access becomes a liability.

If privacy controls cannot be enforced, do not deploy. Privacy controls include redaction, minimization, role-based output filtering, and retention policies for logs and caches. If the organization cannot enforce these, deployment is irresponsible. This is not a theoretical stance; it reflects the fact that multimodal systems will inevitably encounter sensitive content. If the system stores raw evidence in logs, or if outputs can leak PII, the organization will accumulate regulatory and reputational risk. Therefore, a minimum privacy posture is a deployment prerequisite, not a later improvement.

If there is no auditable trail, there is no authority. A system that cannot produce a trace cannot be trusted with authority. If the organization cannot reconstruct what inputs were processed, what transformations were applied, and what tools were called, it cannot defend decisions. Therefore, lack of audit trails is a hard deferral criterion. If auditability is partial, the system's authority must be restricted accordingly. For example, a system without full evidence maps may be used for drafting but not for compliance determinations.

If irreversible action gating is absent, block irreversible actions. If the system cannot enforce explicit approval for irreversible or high-impact actions, it must not be allowed to perform those actions. This criterion is non-negotiable because irreversible actions are where small errors become big incidents. In practice, this means that a deployment without approval workflows must remain advisory. Execution authority is deferred until gating exists.

If end-to-end evaluation under stress is absent, restrict scope. If the system has not passed an end-to-end evaluation suite that includes injection cases, privacy-sensitive artifacts, ambiguous cases requiring deferral, and action gating scenarios, then deployment should be restricted to narrow, low-risk use cases. This is the governance equivalent of not releasing a product without testing. Because VLA systems are system-of-systems, component tests are not enough. Absence of end-to-end stress evaluation is a deferral criterion.

If oversight capacity is insufficient, do not expand autonomy. Even if technical controls exist, organizations must have the human oversight capacity to review escalations and approvals. If the system generates more review workload than the organization can handle, oversight will degrade and unsafe behavior will slip through. Therefore, autonomy expansion should be deferred until oversight capacity is demonstrated, including staffing, training, and workflow design.

Deployment deferral criteria are not obstacles to innovation. They are the only mechanism that prevents the organization from adopting autonomy by accident. They also create a disciplined roadmap: begin with read-only and drafting; build audit trails and evaluation harnesses; implement injection defenses and privacy controls; then expand tool authority step by step.

Risk & Control Notes

Minimum control set (executive-ready).

1. **Least-privilege access** for every tool and data source, enforced by allowlists.
2. **Action gating** with explicit human approval for irreversible or high-impact actions.
3. **Injection resistance** (sanitization + detection + separation of content from control signals).
4. **Privacy-preserving auditability** (redaction/hashing with reconstructable evidence references).
5. **End-to-end evaluation** that tests perception→reasoning→action outcomes under stress.

The minimum control set above is intentionally framed in executive language because it represents decisions leadership must own. These are not optional technical features; they are the control substrate that determines whether a multimodal VLA deployment is a competitive advantage or a latent incident. The strategic lesson of this chapter is that multimodal frontier systems cannot be governed at the model level alone. Governance must be implemented as system design: bounded authority, verifiable traces, and constraint enforcement that remains intact even when models are uncertain, inputs are adversarial, and incentives push toward speed. That is what responsible frontier deployment looks like.

5.10 Outlook and Open Questions

5.10.1 Near-term research questions

The near-term research agenda for multimodal vision–language–action (VLA) systems is not dominated by the question "How do we make models see better?" It is dominated by a more operationally decisive question: *How do we make system behavior reliable under uncertainty, adversarial inputs, and constrained authority?* The frontier is now defined by governance feasibility as much as by model capability. The most valuable research questions are therefore those that reduce the gap between impressive demonstrations and deployable, defensible systems.

A first cluster of questions concerns **robust multimodal fusion**. Current systems often fuse modalities in ways that are powerful but brittle: one modality can dominate incorrectly, contradictions can be smoothed over, and spurious correlations can drive decisions. Near-term research should aim for fusion mechanisms that explicitly represent uncertainty and disagreement, rather than forcing premature coherence. Practically, this means models that can maintain multiple hypotheses, quantify confidence per modality, and expose "why" a modality influenced a decision. The key requirement is not simply better accuracy, but better *behavior under ambiguity*: when signals conflict or degrade, the system should not hallucinate certainty; it should defer, request additional evidence, or escalate. The research opportunity is to formalize fusion not merely as a representation trick, but as a controlled inference process with explicit failure modes.

A second cluster concerns **calibrated uncertainty quantification (UQ) for multimodal decisions**. In VLA systems, confidence must be meaningful because it drives gating: whether to verify, whether to act, whether to escalate. Yet current confidence signals are often poorly calibrated and may degrade sharply under distribution shift. Near-term research should focus on calibration methods that remain robust when inputs are messy: low-quality scans, unusual layouts, partial occlusions, and adversarial overlays. Importantly, UQ must be tied to action impact. The relevant uncertainty is not only "is the caption correct," but "is the evidence sufficient to justify an action." This suggests research into decision-aware calibration: confidence signals that directly estimate the probability that an action would violate constraints or produce unacceptable outcomes.

A third cluster is **standardized defenses against multimodal injection**. Injection is now a structural risk because untrusted content can enter the system through documents and images. Near-term research should aim to move beyond ad-hoc sanitization toward principled control-plane security: methods that distinguish content from control reliably, detect instruction-like patterns across modalities, and reduce susceptibility to hidden directives. This includes research on robust prompt compartmentalization, secure parsing, and detection of adversarial instruction channels in OCR output. It also includes research on evaluation: injection defenses are only credible if they can be tested under realistic attack patterns.

A fourth cluster concerns **action gating and safe tool mediation**. Tool use is where multimodal

systems become operational, and where failures become costly. Research is needed on architectures that ensure tool calls are provably bounded by policy: least privilege, parameter constraints, and approval workflows that cannot be bypassed. This includes designing action representations that are schema-validated, as well as designing mediators that can reason about tool-call sequences and block unsafe compositions. A core question is how to build tool mediators that are themselves auditable and robust, rather than opaque learned policies. The near-term goal is not maximal autonomy; it is predictable, enforceable authority boundaries.

Collectively, these research questions share a theme: the target is *system reliability*, not model impressiveness. The frontier opportunity in the next two years will belong to the teams that can demonstrate stable behavior under stress and credible governance instrumentation.

5.10.2 Technical unknowns

Beyond the near-term agenda, several technical unknowns remain unresolved and are likely to define the next phase of multimodal VLA deployment. These unknowns are not minor; they are structural uncertainties about generalization, attribution, and the reliability of tool-coupled reasoning.

A first unknown is **generalization under extreme distribution shift**. Enterprises face "extreme shift" in mundane forms: new document templates, jurisdiction-specific formats, rare edge cases, and unusual scanning artifacts. In multimodal settings, shift is compounded: the joint distribution of layout, typography, images, and embedded metadata can change abruptly. Even if a model generalizes well on standard benchmarks, its behavior under extreme shift is difficult to predict. The unknown is not whether performance drops—it will—but whether the system fails safely. Specifically, will uncertainty increase appropriately, will the system defer, and will it avoid unsafe actions? Designing systems that degrade gracefully under extreme shift is a technical challenge that intersects with calibration, verification, and control design. It is not clear that current training and adaptation approaches produce the kind of graceful degradation required for high-stakes deployment.

A second unknown is **reliable attribution across modalities**. In governance-first contexts, it is not enough that the system produces an answer; it must show what evidence drove the answer. Attribution becomes harder when modalities are fused in latent space and when reasoning steps are implicit. A system may cite a region of a document, but the real driver of its conclusion could be an unrelated cue learned during training. Attribution in multimodal systems must therefore address two layers: (i) evidence referencing (what inputs were used) and (ii) causal influence (what inputs materially changed the decision). The technical unknown is whether we can reliably and efficiently measure causal influence in multimodal pipelines without heavy interpretability tooling. Without credible attribution, auditability remains partially performative.

A third unknown is **robust tool mediation in adversarial environments**. Tool-coupled systems can be attacked by manipulating the content that drives tool selection, by exploiting tool parameter spaces, or by inducing unsafe sequences. Even with allowlists, subtle risks remain: the system may

retrieve sensitive data and then leak it through a permitted channel, or it may combine innocuous actions into an unsafe sequence. Robust mediation may require sequence-aware policies and formal verification of constraints over trajectories, which is technically challenging at enterprise scale. The unknown is whether practical tool mediators can provide strong guarantees without making systems unusably rigid.

A fourth unknown concerns **long-horizon stability when perception feeds action**. Even when individual steps are accurate, repeated cycles of perceive–act–observe can accumulate error. This is familiar in control systems and simulation, but multimodal VLA systems add new error sources: misperception, retrieval drift, and policy changes. The technical question is whether we can design mechanisms that detect drift early and prevent silent compounding. This may involve online monitoring, periodic re-verification, and "safe state" fallbacks. The unknown is how to make these mechanisms reliable without excessive cost.

A fifth unknown is **privacy-preserving learning and adaptation**. Organizations will want systems to improve on their specific documents and workflows. But training or fine-tuning on sensitive multimodal data raises privacy and security risks, and may create leakage pathways. The unknown is whether organizations can obtain meaningful improvements through privacy-preserving methods (redacted training, synthetic augmentation, federated learning) without sacrificing performance. This is a central technical uncertainty because the most valuable deployments will be domain-specific.

These technical unknowns suggest that the next phase of VLA will be defined by control engineering and operational robustness rather than by raw multimodal capability. The systems that win enterprise trust will not be the ones that are most impressive on curated demos, but the ones that behave predictably in messy reality.

5.10.3 Governance unknowns

Even if technical progress accelerates, multimodal VLA deployment remains constrained by governance unknowns: what standards will emerge, how liability will be allocated, and what audit norms will be accepted as reasonable. These unknowns are particularly important because they determine whether organizations can adopt VLA systems without exposing themselves to open-ended risk.

A first governance unknown is **standards for logging and retention in mixed-media systems**. Traditional audit logging standards were designed for structured transactions and text records. Multimodal traces include evidence pointers, transformations, and potentially sensitive media. Organizations must decide what to store, for how long, and under what access controls. Regulators and auditors may have different expectations than privacy authorities. The unknown is whether standard practices will emerge that balance auditability with minimization, and whether those practices will be recognized across industries. Without standards, organizations either over-log (creating exposure) or under-log (destroying defensibility).

A second governance unknown is **liability allocation for VLA outcomes**. When a multimodal system proposes an action and a human approves it, who is responsible if the action is harmful? When a system misinterprets evidence, is the organization liable for relying on it? When an injection attack manipulates a system, is the fault in the model provider, the integrator, or the deploying firm? Liability frameworks for AI exist in general terms, but VLA systems introduce specific complexities because they operate on evidence and can change state. The unknown is how courts, regulators, and counterparties will treat mixed-media traces and approval logs. This uncertainty will affect adoption decisions and may push organizations toward conservative deployment envelopes.

A third governance unknown is **audit norms for mixed-media traces**. Auditors will need to evaluate whether the system's decisions were reasonable. But what constitutes a reasonable trace? Is evidence referencing sufficient, or is causal attribution required? How much of the raw media must be retained to support audits? What is the acceptable level of automation bias risk? These are not purely technical questions; they are institutional norms. The unknown is whether audit communities will develop accepted methodologies for evaluating VLA systems, analogous to model risk management practices in finance. Without audit norms, organizations will struggle to demonstrate control in a credible way.

A fourth governance unknown is **acceptable autonomy thresholds**. Even within the same organization, different stakeholders will disagree on how much autonomy is acceptable. Product teams may push for automation; risk teams may resist. Regulators may impose constraints. The unknown is whether widely accepted maturity ladders for VLA autonomy will emerge, specifying what controls are required at each autonomy tier. Such ladders would enable more predictable adoption. In their absence, autonomy decisions will remain ad hoc and politically contested.

A fifth governance unknown is **cross-border and sectoral divergence**. Multimodal systems touch privacy, security, and safety, all of which are regulated differently across jurisdictions. Global organizations may face divergent requirements for retention, redaction, and evidence handling. The unknown is how to design governance architectures that can satisfy multiple regimes without building entirely separate systems.

Governance unknowns matter because they influence investment. Organizations may delay deployment not because models are weak, but because governance expectations are uncertain. The frontier, therefore, is as much about standard-setting and institutional practice as about model innovation.

5.10.4 What would change the assessment

The assessment of multimodal VLA systems today is cautiously optimistic under strong constraints: their opportunity is large, but their deployability is conditional on system-level controls. The question, then, is what developments would meaningfully change that assessment from "conditional" to "broadly deployable." Several concrete shifts would alter the landscape.

First, **widely adopted VLA safety benchmarks** would change the assessment. Not generic vision-language benchmarks, but enterprise-relevant suites that include injection-laced documents, privacy-sensitive artifacts, ambiguous cases requiring deferral, and tool/action coupling scenarios. Benchmarks must measure not only correctness but invariants: permission compliance, approval gating, trace completeness, and evidence fidelity. If such benchmarks become standard and widely used, organizations can compare systems credibly and vendors can optimize for deployability rather than for spectacle.

Second, **injection-resilient tool stacks** would change the assessment. This means standardized architectures in which untrusted content cannot influence control signals, tool calls are schema-validated, allowlists are enforced by default, and approval gates are cryptographically bound to action requests. If enterprises can adopt reference implementations of these stacks, the burden of building control-plane security from scratch decreases dramatically. Such stacks would also reduce the variance in deployment quality across organizations, making risk more predictable.

Third, **privacy-preserving provenance tooling** would change the assessment. Organizations need to retain enough evidence for audits while minimizing exposure. Tooling that can produce reconstructable evidence references (hashes, pointers, coordinate maps) without storing raw sensitive media in logs would enable broader adoption. If such tooling becomes standardized, auditability no longer requires privacy compromise.

Fourth, **operational practices for continuous evaluation** would change the assessment. If organizations routinely run stress suites, monitor injection attempts, and treat autonomy expansion as gated by evidence, the risk posture improves. This is less about new algorithms and more about disciplined operational maturity. A world where continuous evaluation is normal would make VLA deployments more trustworthy.

Finally, **clearer governance standards and liability norms** would change the assessment. If regulators and audit communities establish acceptable trace formats, retention standards, and accountability expectations, organizations can invest with less uncertainty. The frontier would then shift from governance ambiguity to implementation excellence.

These developments are not speculative fantasies; they are specific engineering and institutional milestones. They define what "maturity" would mean for multimodal VLA systems in the next cycle.

5.10.5 Link to subsequent papers

This chapter functions as the capstone of *AI 2026: Frontier Awareness Without the Hype* because it collapses the comforting fiction that frontier risk is contained inside model weights. In multimodal VLA systems, the model is only one component in a larger system whose failure modes are created by integration, tool authority, logging, and organizational incentives. That is the unifying lesson of

the book.

The chapter links backward to every core theme. It links to Chapter 1 because evaluation must be trajectory-level: multimodal systems must be evaluated as sequences of perception, reasoning, and action under stress. It links to Chapter 2 because deeper reasoning and tool use increase delayed failure and false confidence risks. It links to Chapter 3 because representation control and internal steering are not sufficient when untrusted content can enter the context through OCR and images. It links to Chapter 4 because retrieval and memory determine what evidence is seen and what becomes silently emphasized, and because persistent traces create provenance risk. It links to Chapter 5 because planning and search amplify objectives, and multimodal systems often operate in loops where small errors can be magnified.

As a launch point for *AI 2027*, the natural expansion is toward domains where action coupling becomes even more consequential: robotics-like systems, cyber defense and offense governance, and biosecurity-adjacent applications where multimodal inputs and tool actions intersect with physical harm pathways. The governance template established here would carry forward: least privilege, action gating, injection resistance, privacy-preserving auditability, and end-to-end evaluation as prerequisites for deploying systems that see and act.

The final strategic message is therefore consistent with the title of this volume: frontier awareness without the hype. Multimodal VLA systems are not science fiction; they are emerging operational infrastructure. Their value will be realized by organizations that can govern them, measure them, and constrain them. The frontier, in practice, is the ability to build systems that remain accountable when perception, reasoning, and action are fused into a single loop.

Bibliography

[1] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. *Learning Transferable Visual Models From Natural Language Supervision*. Proceedings of the 38th International Conference on Machine Learning (ICML), PMLR 139, 2021. Available: arXiv:2103.00020.

[2] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob Menick, Sebastian Borgeaud, Andrew Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karen Simonyan. *Flamingo: a Visual Language Model for Few-Shot Learning*. Advances in Neural Information Processing Systems (NeurIPS), 2022. Available: arXiv:2204.14198.

[3] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. *BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models*. Proceedings of the 40th International Conference on Machine Learning (ICML), PMLR 202, 2023. Available: arXiv:2301.12597.

[4] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. *Toolformer: Language Models Can Teach Themselves to Use Tools*. International Conference on Learning Representations (ICLR), 2023. Available: arXiv:2302.04761.

[5] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. *ReAct: Synergizing Reasoning and Acting in Language Models*. International Conference on Learning Representations (ICLR), 2023. Available: arXiv:2210.03629.

[6] OWASP. *OWASP Top 10 for Large Language Model Applications* (Version 1.1). OWASP GenAI Security Project, 2024–2025. Available: owasp.org (Top 10 for LLM Apps).

[7] National Institute of Standards and Technology (NIST). *Artificial Intelligence Risk Management Framework (AI RMF 1.0)*. NIST AI 100-1, 2023. Available: NIST AI 100-1 (PDF).

[8] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. *On Calibration of Modern Neural Networks*. Proceedings of the 34th International Conference on Machine Learning (ICML), PMLR 70, 2017. Available: arXiv:1706.04599.

[9] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. *Explaining and Harnessing Adversarial Examples*. International Conference on Learning Representations (ICLR), 2015. Available: arXiv:1412.6572.

[10] Martín Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. *Deep Learning with Differential Privacy*. Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS), 2016. Available: arXiv:1607.00133.

Appendix A

Notebook Index (Companion Colab Notebooks)

This appendix lists the companion notebooks for each chapter. Each notebook is available on GitHub and is designed to be runnable end-to-end in Google Colab.

Repository path (GitHub): `(placeholder)`

Chapter	Notebook (file) and focus
Chapter 6	`chapter_6.ipynb`: Discovery workflows, evaluation under scarce ground truth, safety screens, reproducible experiment scaffolds.
Chapter 7	`chapter_7.ipynb`: Surrogate modeling, operating envelopes, constraint checks, robustness and drift probes.
Chapter 8	`chapter_8.ipynb`: Finance evaluation harnesses, disclosure-aware drafting scaffolds, audit trails, tail-risk stress tests.
Chapter 9	`chapter_9.ipynb`: Evidence discipline for narratives, contradiction handling, attribution scaffolds, framing stress tests.
Chapter 10	`chapter_10.ipynb`: Multimodal provenance scaffolds, tool gating, stop conditions, monitoring and incident reconstruction artifacts.