# The PID (Project Initiation Document) Is the Most Important Artifact in Building New Agentic Enterprise Workflows

*A Theoretical White Paper on Governance-as-Code, the Upstream Governance Gap, and the Case for Structured Project Initiation (illustrated by PMOMax) as Foundational for New Enterprise Workflows*

**Mulugeta Semework, PhD (Technical Lead)**
**Carolina Baez, Engr (Product Lead)**
**Andrew Igharo (Commercial Lead)**

**Katalyst Street, Inc.**
**Date: May 2026**

---

# Introduction: The Governance Problem That Runtime Alone Cannot Fully Solve

The agentic enterprise is arriving faster than most governance models can accommodate. Platforms such as Google's emerging enterprise agent ecosystems now provide sophisticated runtime controls: centralized agent identity, semantic gateways, prompt injection defense, anomaly detection, and unified telemetry. These are necessary capabilities.

They are not sufficient on their own.

Every runtime governance system shares a foundational dependency: it requires policies to enforce. Those policies must come from somewhere. They must define which agents can do

what, under which conditions, based on which approved project parameters, authorized by whom, and traceable to which governing artifact.

Most organizations do not answer these questions in the semantic or architectural layer of the data their business generates. Their project initiation processes produce static documents—not structured, testable, auditable governance artifacts. Scope lives in paragraphs, not boundaries. Approvals are informal. Risk mitigations are written once and never encoded. Decision rights are assumed, not delegated.

The 3 base questions and the resolution of related conflicts -

a. What does the AI touch?

b. What does the AI decide?

c. Who owns the outcome?

This in essence is the upstream governance gap.  It exists before any agent runs. Runtime governance alone cannot fully close it because runtime platforms generally assume the gap is already closed.

This white paper explores the nature of that gap, the operational compromises that sustain it, and the architectural pattern—Governance-as-Code (GaaC)—required to reduce it. It then presents PMOMax as one instantiation of that pattern: an AI-assisted Project Initiation Document platform built on Google Cloud, designed to produce structured governance artifacts that downstream runtime layers can consume.

---

# Part One: The Theory of Governance in Agentic Systems

## 1.1 Governance as a Binding Constraint

Governance, in its simplest form, is the imposition of binding constraints on action. In human organizations, these constraints take many forms: policies, procedures, approval thresholds, segregation of duties, audit requirements. Their purpose is to make action predictable, auditable, and corrigible.

In traditional project delivery, governance constraints are expressed in natural language documents. A project charter states the scope. A risk register lists mitigations. An approval matrix specifies sign-offs. These documents are read by humans and interpreted into action.

In large-scale agentic systems, this model becomes increasingly difficult to govern reliably at enterprise scale. Agents do not interpret natural language in the same way humans do. They cannot reliably infer missing governance constraints suitable for audit-grade enforcement. They cannot consistently recognize when a policy is ambiguous. They execute what they are explicitly permitted to execute, within the boundaries their code and tool definitions allow.

The architectural requirement is therefore straightforward: for an agent to be governed reliably, constraints should be expressible in a form that is machine-readable, testable, and auditable. If a constraint cannot be encoded or operationalized, it becomes increasingly difficult to enforce consistently across autonomous workflows.

---

## 1.2 The Inevitable Compromise: Expressiveness vs. Enforceability

Every governance system faces a fundamental trade-off. Natural language is highly expressive. It can capture nuance, exceptions, and contextual judgment. But it is not inherently machine-enforceable. Code is machine-enforceable but less expressive. Complex human judgments resist clean encoding.

Organizations routinely compromise toward expressiveness. They write detailed policies in prose, then rely on human auditors to verify compliance after the fact. This works imperfectly in human-only systems because people can interpret ambiguity contextually. In agentic systems, that ambiguity creates increasing operational and audit challenges because there may be no human in the loop at the exact moment of execution.

The operational compromise required is not the elimination of nuance, but the structured separation of what can be encoded from what requires human judgment. Governance must move toward enforceability without losing necessary expressiveness.

This means:

1. Encoding what can be encoded as structured, testable rules.
2. Explicitly identifying what cannot be encoded and designating those decisions as human-only with clear escalation and approval handoffs.

---

## 1.3 The Front-of-Loop Principle

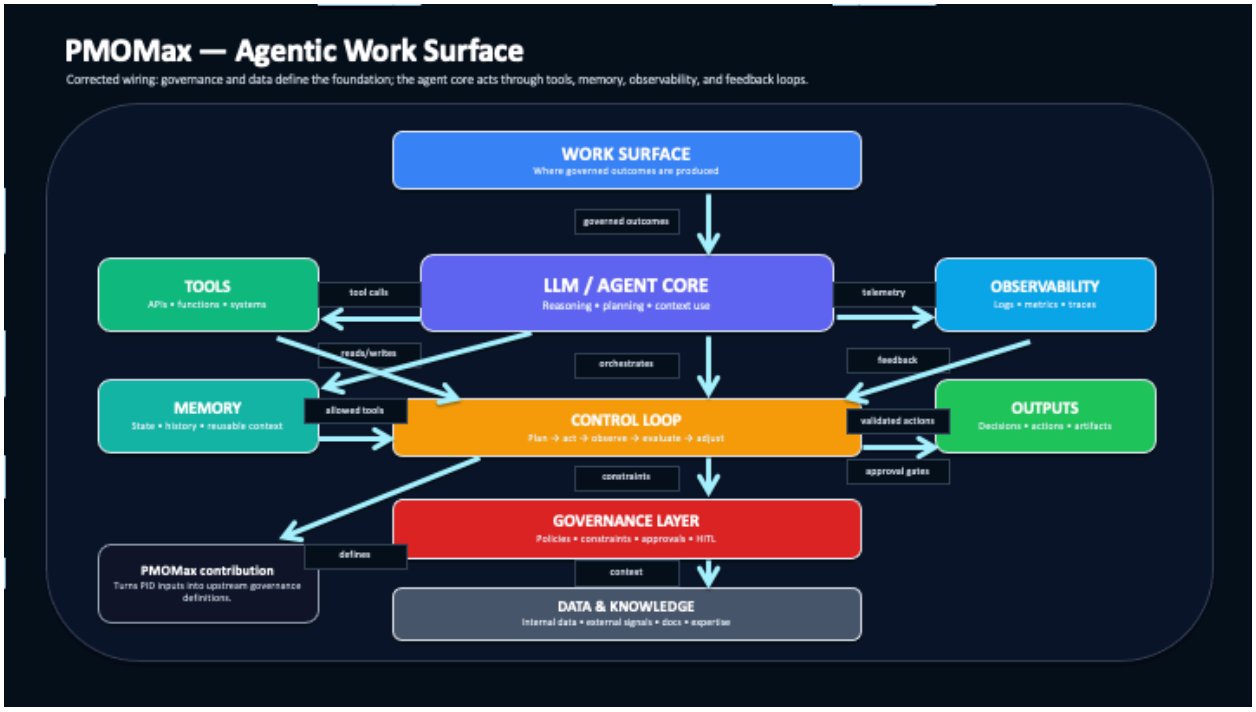
In most AI workflow designs, the human sits at the end of the loop: the AI proposes, the human approves or rejects. This is reactive governance. It assumes the human can evaluate quality and compliance after the fact.

A more scalable model places the human at the front of the loop. The human defines the rules, boundaries, risk tolerances, and decision criteria first. AI then operates within those boundaries. Human review shifts from evaluating every output to validating constraints and handling exceptions.

The operational principle is that governance should be as ex ante as possible, not purely ex post. Constraints defined before execution are generally more reliable than judgments rendered after execution. In this model, the Project Initiation Document is no longer merely a static artifact. It becomes the ex ante governance contract.

**Conceptual Illustration:** Agentic Work Surface and Governance Context

The diagram below presents a simplified view of an “agentic work surface,” where a central AI model operates within a structured execution environment composed of tools, memory, observability, and iterative control loops. Rather than functioning as an isolated reasoning engine, the model exists within a system that continuously integrates context, executes actions, evaluates outcomes, and adapts behavior over time.




***Enhanced Agentic Work Surface illustrating how an LLM operates within a structured execution environment, integrating tools, memory, observability, and control loops under explicit governance constraints.***

Several key characteristics of modern agentic systems are reflected in this structure:

• Tool integration, where external systems and APIs extend the capabilities of the model beyond language generation
• Memory and context propagation, allowing continuity across steps, sessions, and workflows
• Observability, enabling logging, tracing, and auditability of decisions and actions
• Control loops, where outputs are evaluated, refined, and iterated upon
• Governance layers, where constraints, policies, and human oversight shape execution behavior

While these systems are powerful, they depend fundamentally on the quality and structure of the constraints that guide them. Without clearly defined scope boundaries, decision rights, risk tolerances, and approval conditions, agentic systems operate with implicit or incomplete governance.

This is where PMOMax becomes critical. By transforming project initiation into structured, machine-readable governance artifacts, PMOMax provides the upstream definitions required for these systems to operate safely, predictably, and in alignment with enterprise intent.

In this sense, the “work surface” is not only a runtime environment—it is the downstream expression of upstream governance. The quality of outcomes produced within this surface is directly determined by the completeness and clarity of the governance structures defined before execution begins.

PMOMax effectively shifts governance from implicit interpretation to explicit definition, ensuring that agentic execution environments are not only intelligent, but also constrained, auditable, and aligned with organizational objectives.

---

# Part Two: The Compromises That Sustain the Upstream Gap

## 2.1 The PID as Static Artifact

Most Project Initiation Documents are produced once, read a few times, and then archived. They are not updated as projects evolve. They are not linked to execution systems. They are not testable against actual project actions.

This is a design compromise, not a technical necessity. Organizations accept static PIDs because producing structured, machine-readable PIDs requires additional effort without immediate visible reward. The cost of poor initiation is distributed and delayed. The cost of structured initiation is immediate and visible.

Result: governance artifacts that cannot be operationalized effectively by downstream runtime systems.

---

## 2.2 The Stacked Role of the Project Manager

The traditional project manager performs multiple distinct functions: scope definition, risk management, compliance oversight, stakeholder alignment, communications, delivery coordination, and governance. In human-only systems, stacking these roles is efficient.

In agentic systems, it becomes increasingly fragile. When an agent executes a task, it is not consulting the project manager's tacit knowledge. It is following explicit instructions, permissions, tools, and policies. If governance remains stacked inside a single human role, there is no durable machine-readable governance artifact for agents to follow.

Result: agents operate with incomplete governance context and rely indirectly on human oversight.

---

## 2.3 The Runtime Governance Illusion

Emerging runtime governance platforms create the appearance of complete governance. They provide agent identity, gateway enforcement, anomaly detection, telemetry, and policy execution. These are genuine and important capabilities.

But they enforce policies. They do not inherently generate organizational intent, scope definitions, approval logic, risk acceptance, or project governance structure. Those policies still originate from humans, based on project context, scope, approvals, and risk assessments.

Where do those governance definitions come from?

In many organizations, the answer remains fragmented emails, PDFs, slide decks, meeting notes, and tacit organizational knowledge.

Result: organizations deploy runtime governance and discover that their upstream project definitions are inconsistent, unstructured, or incomplete.

---

# Part Three: Governance-as-Code as the Closing Pattern

## 3.1 Defining Governance-as-Code

Governance-as-Code (GaaC) is the practice of expressing governance requirements in structured, machine-readable, testable, version-controlled forms. It borrows from Infrastructure-as-Code the core insight:

> If you cannot represent a constraint in a structured operational form, it becomes increasingly difficult to enforce reliably at scale.

For project initiation, GaaC means:

| Governance Element | Traditional Form | Governance-as-Code Form |
|---|---|---|
| Scope | Prose description | Structured inclusions/exclusions, testable boundaries |
| Risks | Paragraphs in a register | Risk records with owners, impacts, mitigations, review cadences |
| Approvals | Email threads or sign-off sheets | Structured approval gates with conditions and evidence requirements |
| Decisions | Meeting minutes | Decision records with rationale, authorizer, and effective dates |
| Assumptions | Bulleted list | Assumption records with validation criteria and expiration |
| Success criteria | Vague statements | Measurable conditions with verification methods |

---

## 3.2 The Properties of a GaaC Artifact

A PID expressed as Governance-as-Code has four essential properties:

### Testability

A machine can check whether an action falls within encoded scope boundaries, whether a risk mitigation has been reviewed, whether an approval has been obtained.

### Auditability

Every change to the governance rules is versioned. Every agent action referencing the rules leaves a trace linking back to the specific rule version.

### Delegability

Decision rights are explicitly encoded. A machine can determine whether a workflow is authorized for a given action.

### Evolvability

Governance rules can evolve through controlled processes with versioning, approval tracking, and propagation to dependent systems.

---

## 3.3 Relationship to Runtime Governance

GaaC does not replace runtime governance. It supplies what runtime governance requires: structured policies to enforce.

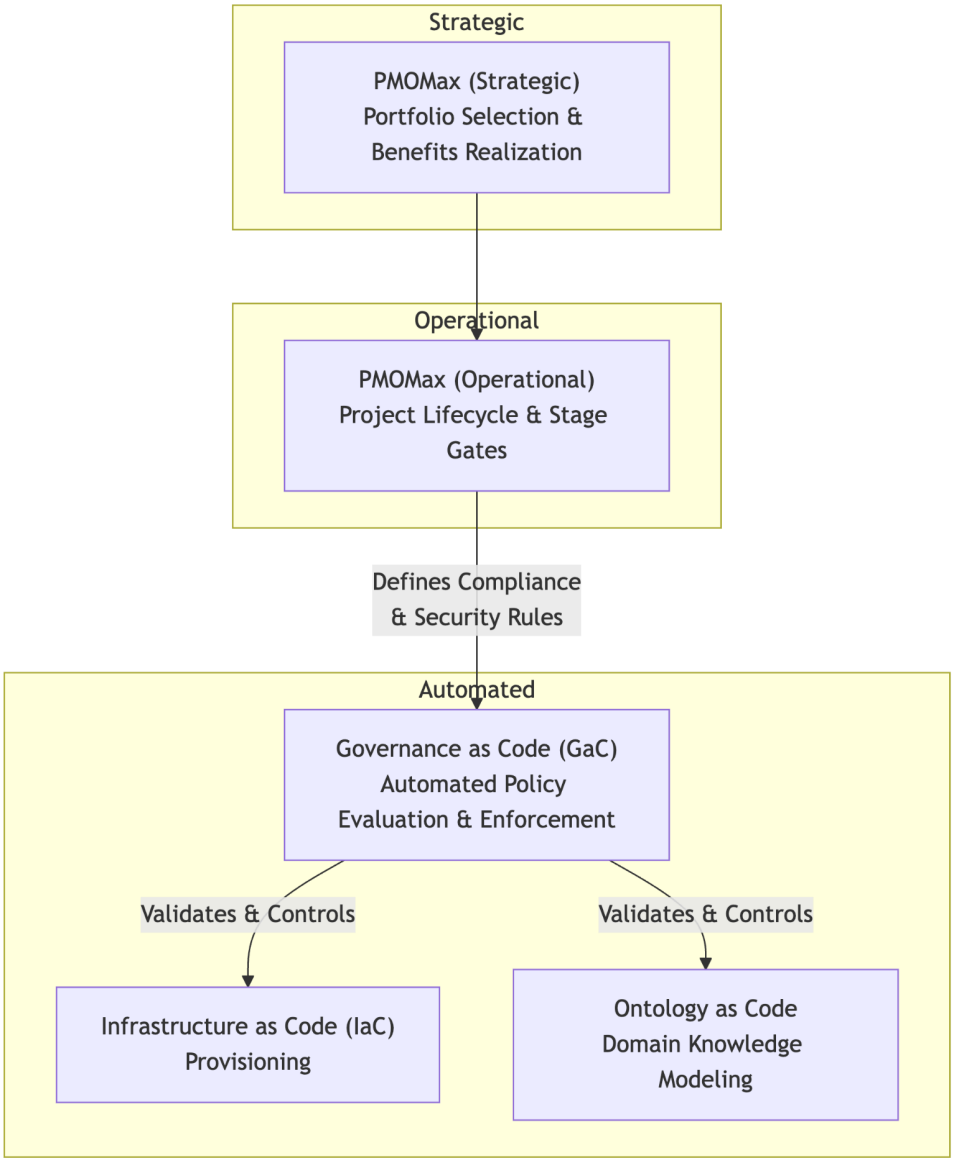
| Layer | Function | Artifact |
|---|---|---|
| Initiation (GaaC) | Define scope, risks, approvals, decision rights | Structured PID |
| Translation | Convert PID into executable policies | Policy definitions |
| Runtime Governance | Enforce policies at execution | Agent gateway, telemetry |

The critical insight is that runtime governance is downstream from initiation governance. You cannot achieve highly effective runtime governance without structured upstream governance artifacts.

Organizations that implement both achieve end-to-end governance.

**Illustrative Governance Stack:** From Project Initiation to Execution Enforcement

The diagram below provides a simplified view of how structured project initiation connects to downstream execution systems through a layered governance model.




***Layered governance model showing PMOMax defining upstream constraints that propagate into Governance-as-Code and downstream execution systems.***

At the top, PMOMax operates at both the strategic and operational levels:
• Strategic: portfolio selection and benefits realization
• Operational: project lifecycle definition and stage-gate governance

These layers define the compliance, security, and execution constraints that are subsequently expressed as Governance-as-Code (GaaC).

The GaaC layer acts as a bridge between human-defined governance and machine-enforceable execution, translating structured project intent into policy evaluation and enforcement mechanisms.

Downstream, these policies can validate and control:
• Infrastructure-as-Code (IaC) provisioning
• Ontology-driven domain modeling and knowledge systems

This layered model reinforces a central argument of this paper: governance must originate upstream, at the point of project definition, before it can be reliably enforced at runtime.

---

# Part Four: The PMOMax Instantiation

## 4.1 Why an Instantiation Matters

Governance-as-Code is a pattern, not a product. Enterprises need concrete tools that lower the operational cost of structured initiation while producing artifacts that downstream systems can consume.

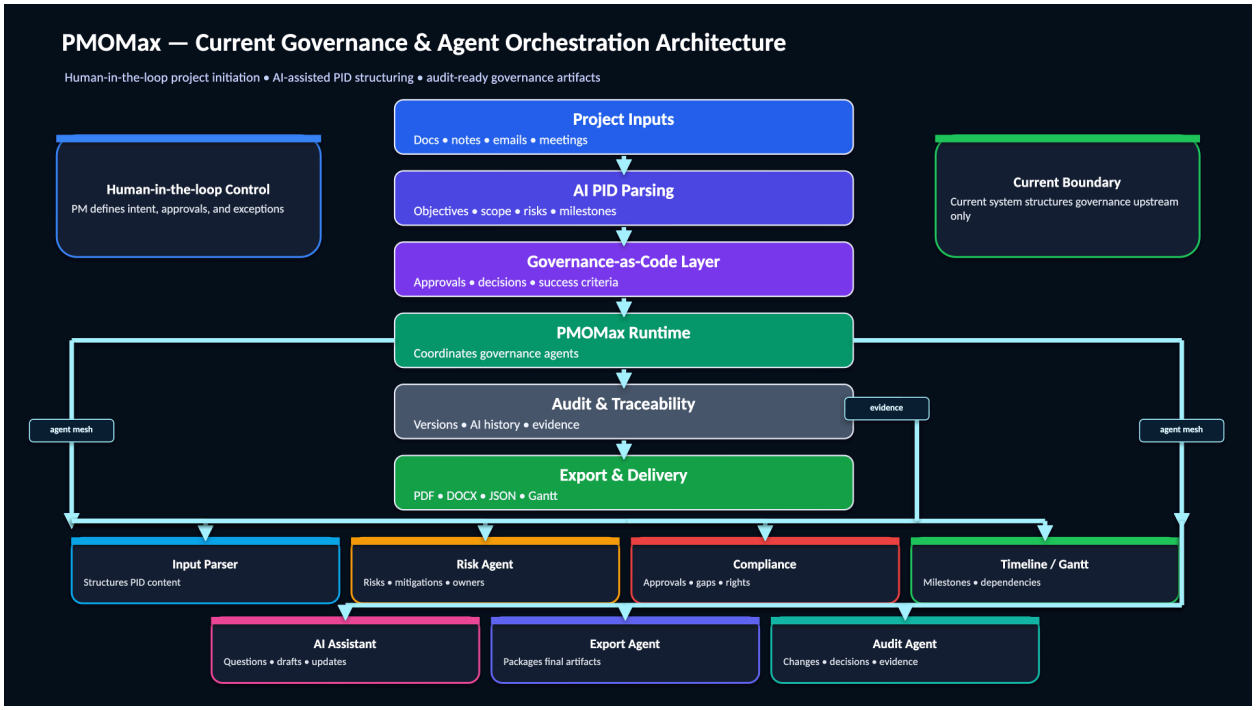
PMOMax is one such instantiation. It is an AI-assisted Project Initiation Document and governance platform built on the Google Cloud stack, designed to produce structured, testable, auditable PIDs that can feed downstream governance layers. Said differently, PMOMax is an AI-assisted, agent-oriented PID governance framework that connects technical architecture with strategic and operational alignment.

---

## 4.2 What PMOMax Does

PMOMax helps teams:

- Ingest unstructured project information from documents, emails, notes, and conversations
- Structure that information into canonical PIDs
- Identify governance gaps and ambiguity
- Analyze risks and approval requirements
- Produce traceable AI-assisted outputs with audit metadata attached
- Export structured governance artifacts for downstream operational systems




***Current PMOMax orchestration flow showing structured PID ingestion, governance processing, AI assistance, risk analysis, timeline generation, auditability, and export workflows.***

---

# 4.3 What PMOMax Does Not Do

PMOMax is not currently a runtime enforcement platform. It does not intercept agent calls. It does not directly enforce gateway policies. It does not provide runtime prompt injection defense or anomaly detection.

Those capabilities belong to downstream runtime governance systems.

The boundary is intentional.

PMOMax focuses on the upstream governance problem. Runtime governance platforms focus on downstream execution governance.

They are complementary rather than competitive.

---

# 4.4 Anthos / GKE Enterprise and Hybrid Governance Deployment

PMOMax is deployed through Google Cloud Marketplace and is associated with Anthos / GKE Enterprise deployment patterns.

Anthos (now aligned under GKE Enterprise) is Google Cloud's hybrid and multi-cloud Kubernetes management platform. In practice, this means PMOMax deployment architectures are not conceptually restricted to a single Google Cloud region or infrastructure boundary.

Enterprise customers may deploy governance workloads:

- within Google Cloud
- across hybrid environments
- inside private data centers
- on VMware or bare metal infrastructure
- across AWS and Azure environments
- or at the edge on localized infrastructure

while still maintaining centralized operational governance.

For governance systems, this flexibility is operationally significant because governance frequently spans organizational, infrastructural, regulatory, and geographic boundaries.

The Anthos / GKE Enterprise alignment also reinforces several architectural characteristics relevant to governance-oriented systems:

## Consistent Policy Management

Anthos Config Management enables centralized policy consistency across distributed environments.

## Fleet-Scale Governance Operations

Organizations can manage many Kubernetes clusters as coordinated operational fleets rather than isolated infrastructure silos.

## Enterprise Security Expectations

Anthos environments are generally associated with stricter operational expectations around:

- vulnerability management
- runtime consistency
- workload governance
- policy enforcement

- operational reliability

### Governance Portability

Structured governance systems become portable across infrastructure boundaries rather than tightly coupled to a single deployment environment.

Importantly, the Anthos designation should not be interpreted as implying formal endorsement of every architectural claim in this paper. Rather, it reflects that PMOMax's deployment architecture aligns with enterprise-oriented Kubernetes operational patterns associated with hybrid and multi-cloud governance environments.

---

## 4.5 The Human Role in PMOMax

PMOMax implements the front-of-loop principle. Humans define project intent, scope boundaries, risk tolerances, approval thresholds, and decision rights.

AI assists by:

- structuring inputs,
- detecting governance gaps,
- drafting sections,
- improving consistency,
- and helping operationalize governance metadata.

Human judgment remains responsible for:

- final approvals,
- governance overrides,
- exception handling,
- and escalation decisions.

The project manager is therefore not replaced. Governance responsibilities become externalized into structured, auditable artifacts that humans, systems, and future workflows can reference consistently.

---

# Part Five: Future A2A and Ontology-Driven Governance Possibilities

## 5.1 Early Workflow Routing and Multi-Agent Orchestration Inspiration

Several orchestration and workflow-routing concepts discussed in this paper were influenced by earlier multi-agent coordination and orchestration research explored in the **AiMedOrchestra** project. AiMedOrchestra is an experimental multi-agent AI platform designed around the idea of specialized AI agents cooperating through an orchestrated workflow rather than relying on a single monolithic model. Reference article:AI Med Orchestra: open-source, multi-agent platform.

AiMedOrchestra explored how multiple specialized agents can collaborate inside a coordinated execution framework. In the platform, each agent is responsible for a focused domain-specific task — such as diagnostics, imaging analysis, genomics interpretation, literature retrieval, ethics auditing, treatment planning, or clinical-trial matching — while an orchestration layer routes requests, coordinates execution order, aggregates outputs, and manages workflow transitions.

The project helped explore several concepts that later influenced workflow thinking in PMOMax, including:

- Context-aware workflow routing
- Specialized agent delegation
- Multi-step orchestration pipelines
- Deterministic versus AI-assisted execution paths
- Shared context propagation between agents
- Structured execution traces
- Human-readable orchestration flows
- Coordinated task sequencing and branching logic

In AiMedOrchestra, the orchestration layer acts similarly to a conductor coordinating an orchestra: individual agents operate independently with specialized capabilities, while the orchestrator manages how information moves between them and determines which agents participate in solving a request. Rather than forcing one model to solve every problem, the system distributes responsibilities across specialized components and recombines the results into a unified response.

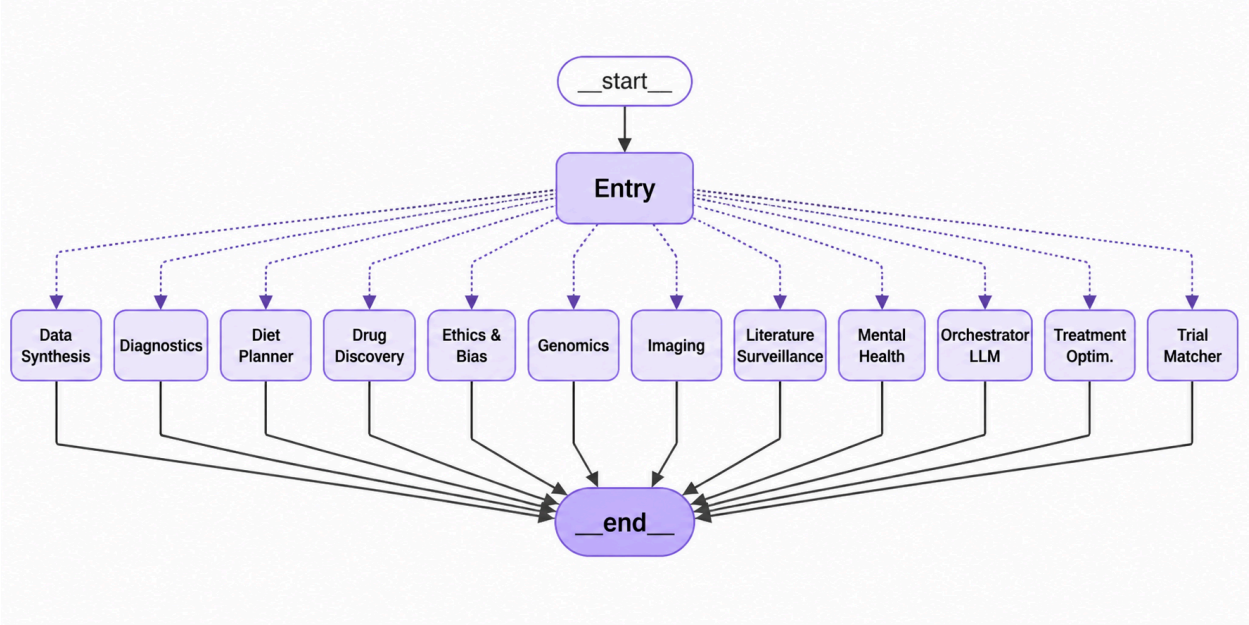
This orchestration-oriented design philosophy influenced several architectural directions later explored in PMOMax, particularly around:

- AI-assisted workflow routing
- governance-aware execution
- contextual decision flows
- structured task decomposition
- audit-oriented execution tracing
- modular agent interaction patterns

The concepts were especially influential in shaping ideas around how PMOMax could evolve from a traditional project-initiation platform into a broader orchestration-aware system capable of

coordinating specialized AI services while preserving governance, traceability, and contextual continuity.

The AiMedOrchestra project also demonstrated how multi-agent systems can remain modular and extensible while still operating within a coherent execution framework. That principle later informed discussions around PMOMax's governance-aware orchestration model and configurable AI decision traceability layer.




***Example multi-agent orchestration and workflow-routing concepts from the AiMedOrchestra project, where specialized agents collaborate through a centralized orchestration and routing layer.***

---

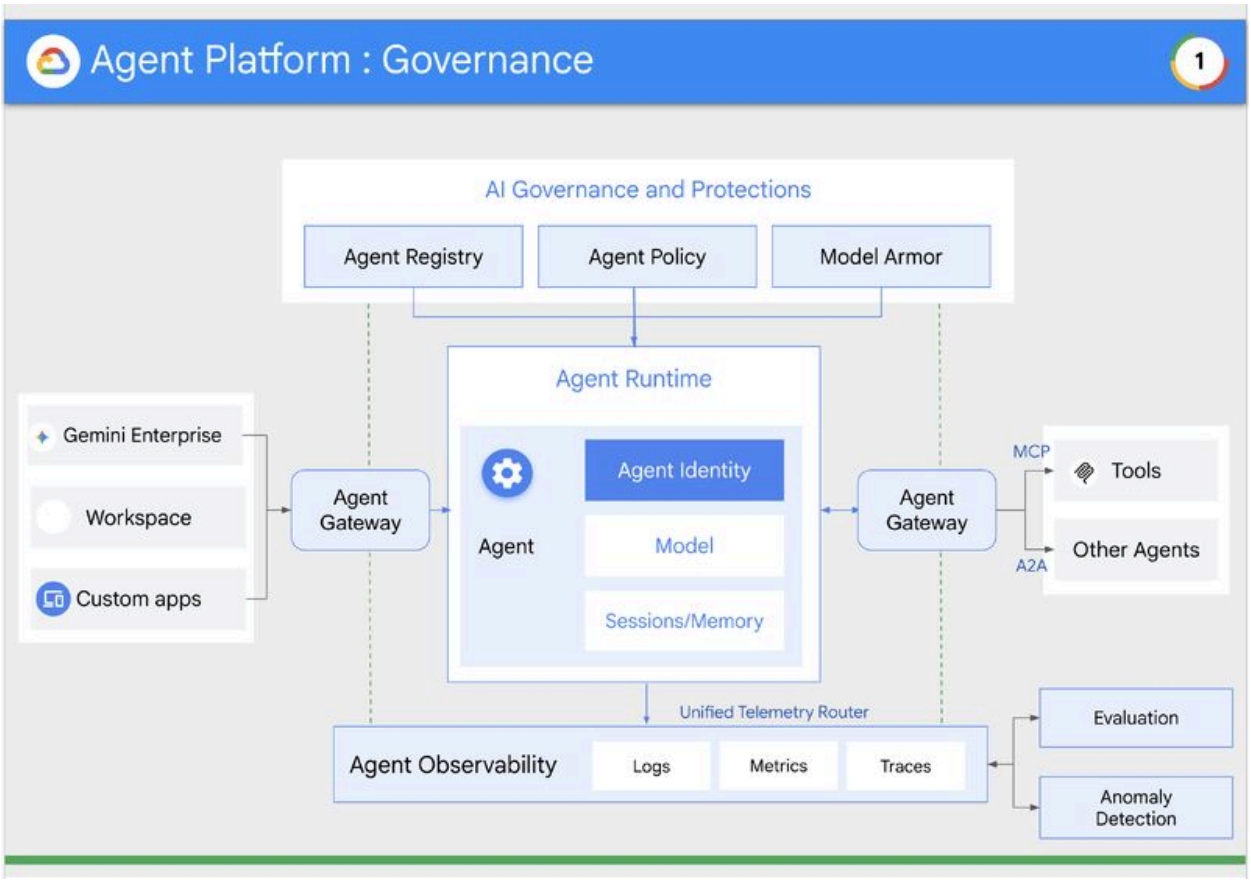
## 5.2 Future Human-in-the-Loop A2A Governance Layer

A future evolution of PMOMax could potentially introduce a human-governed orchestration layer above A2A (Agent-to-Agent) ecosystems and enterprise runtime systems.

This section is informed by emerging industry discussions on agent governance and A2A runtime architectures. Here, we used a description by Eric Dong on his LinkedIn article and the image below came from it.

Under this future-oriented architecture, PMOMax could potentially:

- tie governance data structures to enterprise ontologies
- encode project-level permissions and delegation rights
- coordinate governance-aware workflow routing
- provide human-approved orchestration boundaries
- connect runtime telemetry back to upstream governance definitions
- maintain traceability between business intent and downstream agent actions




***Illustrative governance/runtime ecosystem showing policy layers, runtime orchestration, telemetry, observability, and future human-governed A2A coordination.***

---

## 5.3 PMOMax as an Ontology-Aware Governance Layer

In this future conceptual model, PMOMax would not replace runtime governance systems.

Instead, it could potentially function as a human-in-the-loop orchestration layer connecting:

- governance data structures

- enterprise ontologies
- permissioning systems
- approval workflows
- runtime policy translation
- telemetry
- auditability
- agent-to-agent orchestration

Importantly, these concepts are future-oriented architectural possibilities and should not be interpreted as currently deployed PMOMax production functionality.

---

# Conclusion: Closing the Upstream Gap

Modern runtime governance systems represent genuine progress. They provide telemetry, observability, enforcement, gateway controls, and execution governance mechanisms.

But they still depend on a foundational assumption:

That sufficiently structured governance artifacts already exist upstream.

For many organizations, that assumption remains incomplete.

The upstream governance gap persists because project initiation remains fragmented, static, and weakly operationalized.

Governance-as-Code attempts to reduce this gap by transforming project initiation from static documentation into structured, testable, auditable governance artifacts.

PMOMax represents one possible implementation of this pattern on Google Cloud.

Its future trajectory may eventually include:

- ontology-aware orchestration,
- human-governed A2A coordination,
- governance-aware workflow routing,
- and hybrid enterprise governance architectures spanning runtime and upstream operational layers.

The question is therefore no longer whether runtime governance matters.

It does.

The deeper question is whether runtime governance has sufficiently reliable upstream governance artifacts to enforce.

That is the upstream governance challenge.

And increasingly, it may become one of the defining organizational architecture problems of the agentic enterprise era.