Agentic AI & GRC Design Pattern Book

A Design Pattern Language for Policy-Driven Agentic AI Systems

By Dr. Freeman A. Jackson
Fourth Industrial Systems Corporation (4th)

Contents

Preface

Agentic AI systems are emerging as the next major paradigm in enterprise automation. These systems are autonomous, tool-using, multi-step, and capable of interacting with sensitive data and regulated workflows. With this power comes risk: non-determinism, hallucination, unsafe tool execution, improper data access, and difficulty demonstrating compliance to regulators.

To address this, Fourth Industrial Systems Corporation (4th) developed the **Agentic AI & GRC** framework—a multi-layer governance and orchestration system integrating:

- policy-as-code,
- agent skill registries,
- retrieval and embeddings safety envelopes,
- policy engines,
- rule evaluators,
- state-machine orchestration,
- evidence trails,
- tests and documentation pipelines.

This book formalizes the architecture into a complete **Design Pattern Language**. Each pattern encapsulates a reusable structure for safely building and governing agentic AI.

Chapter 1

Foundational Design Patterns

1.1 Design Pattern 1: PolicyEnvelope

Intent: Provide deterministic governance boundaries around non-deterministic LLM-based agents.

Problem: Agents may call unsafe tools, exceed budgets, or read sensitive data unless constrained.

Solution: Every agent action passes through a *PolicyEnvelope* containing:

- allowed/denied tools,
- risk thresholds,
- data-access tiers,
- cost governors,
- regulatory mappings,
- evidence obligations.

Repository Locations:

- `profiles/`
- `policyengine/`
- `rules/`

1.2 Design Pattern 2: Govern–Orchestrate–Retrieve–Assure (GORA)

Intent: Separate agentic AI into four explicit layers:

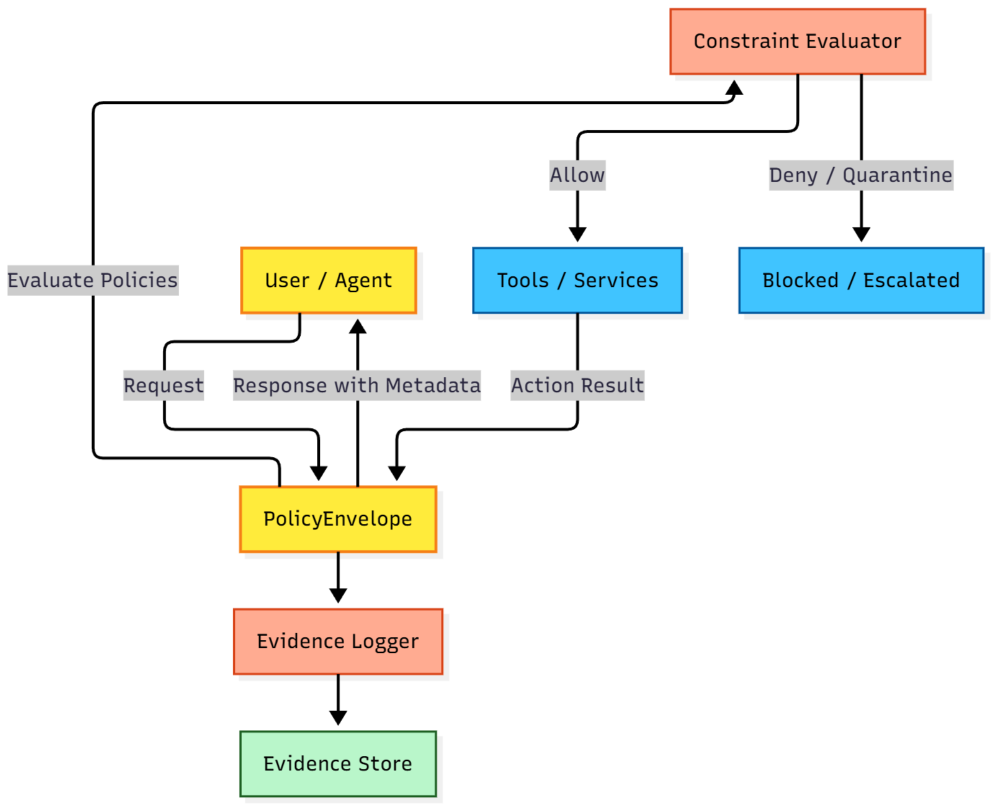

Figure 1.1: Design Pattern 1 — PolicyEnvelope diagram

1. Govern (policies, rules, decision services)
2. Orchestrate (agents, planners, workflows)
3. Retrieve (RAG pipelines, mesh search)
4. Assure (tests, monitoring, documentation)

Repository Mapping:

- Govern → `policyengine/`, `profiles/`, `rules/`
- Orchestrate → `agents/`, `functions/`
- Retrieve → `services/` (RAG, embeddings)
- Assure → `tests/`, `docs/`, `pre-commit`

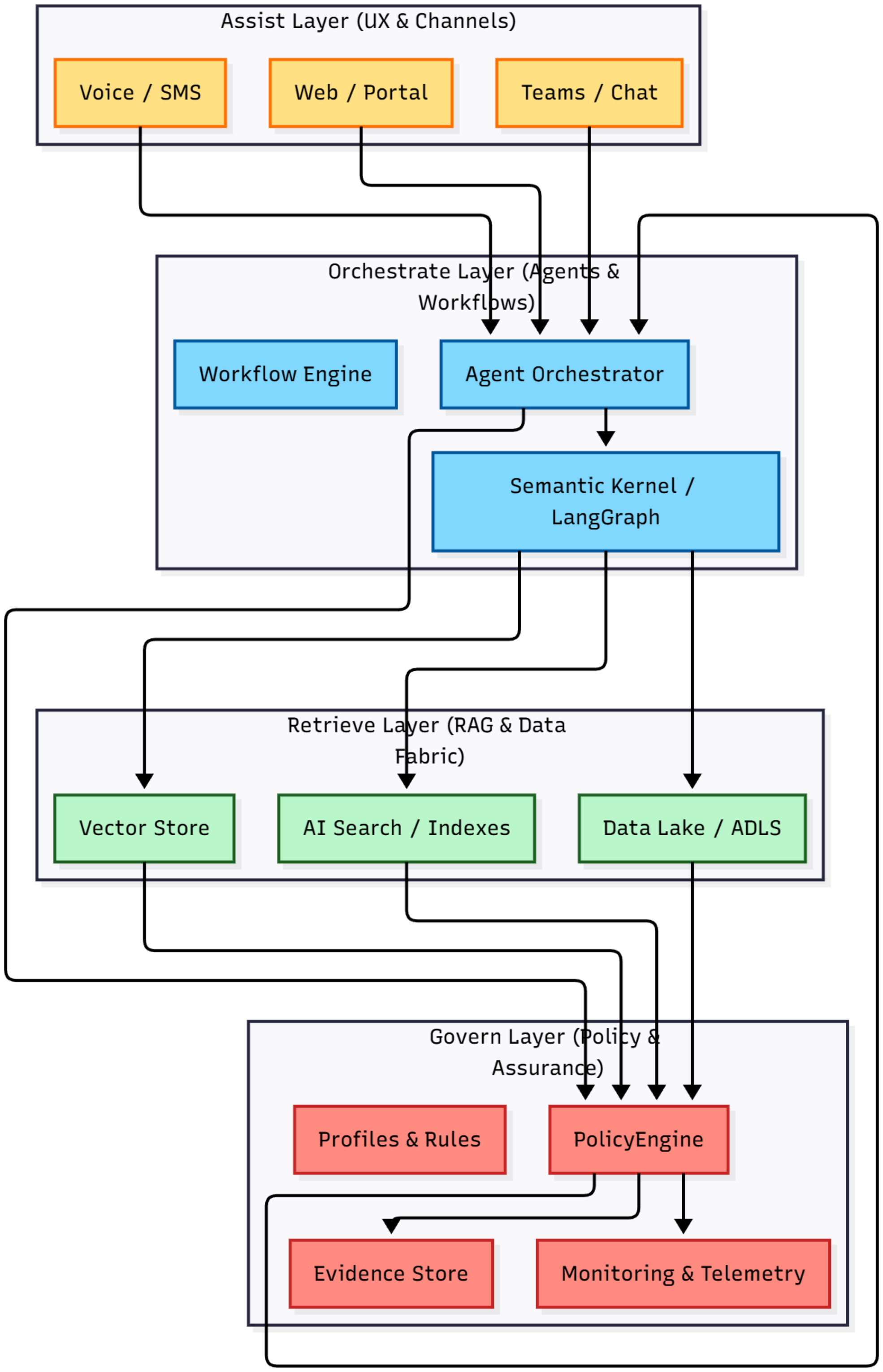

Figure 1.2: Design Pattern 2 — GORA layered architecture

Chapter 2

Governance Design Patterns

2.1 Design Pattern 3: Policy-as-Code

Intent: Express governance and safety requirements as YAML.

Listing 2.1: Example Policy Profile

```
policy:
  id: "enterprise-default"
  version: "1.0.0"
  classification: "baseline"

allowed_tools:
  - search
  - summarizer

denied_tools:
  - direct_db_write

risk_thresholds:
  hallucination: 0.15
  toxicity: 0.10
  uncertainty: 0.20
```

2.2 Design Pattern 4: ConstraintEvaluator

Intent: Convert policy files into actual runtime decisions.

```
decision = evaluator.evaluate(request, profile="enterprise_default")
print(decision.outcome) # allow, deny, escalate, quarantine
```

2.3 Design Pattern 5: EvidenceTrail

Intent: Record every agent action, tool call, decision, and retrieval result.

Captured Evidence:

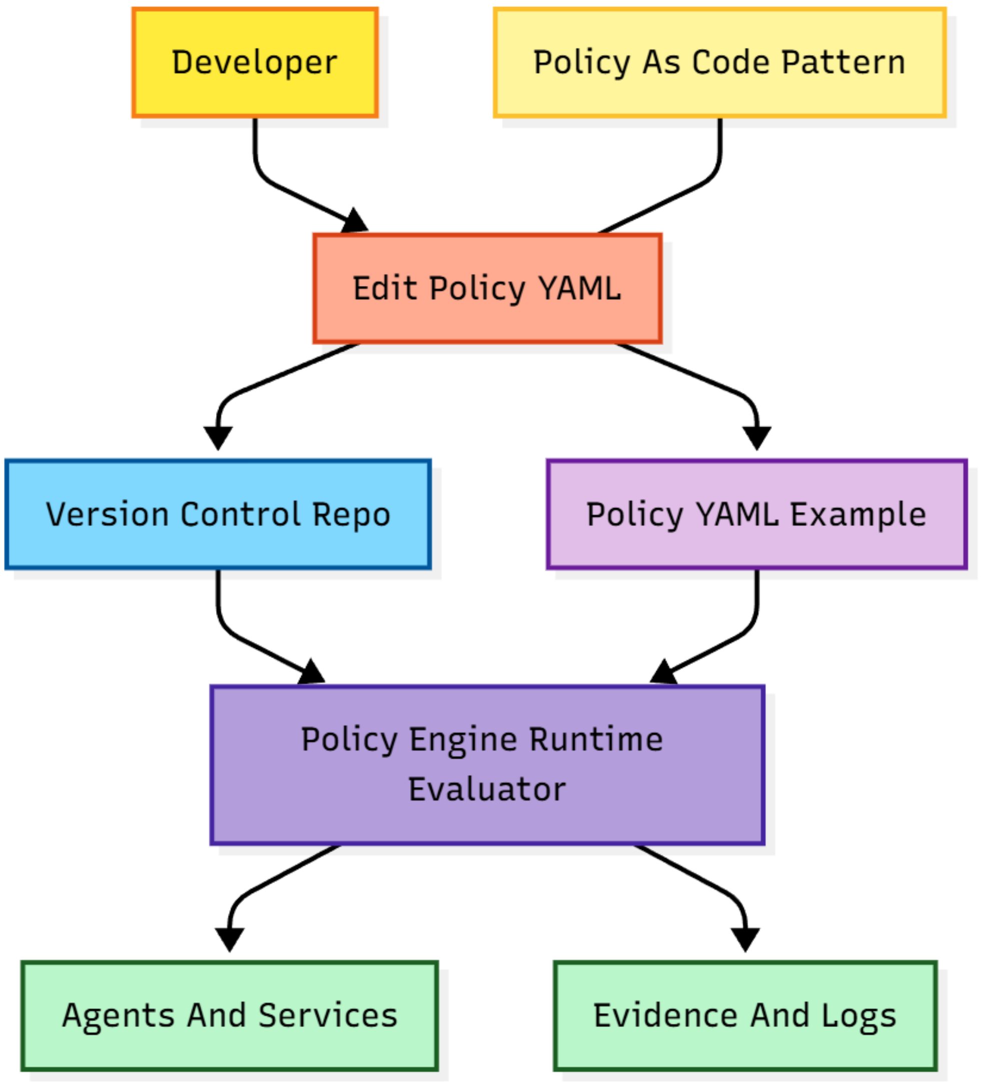

Figure 2.1: Design Pattern 3 — Policy-as-Code visual

- inputs and parameters,
- policy profile used,
- rule violations,
- RAG sources,
- model outputs,
- tool responses.

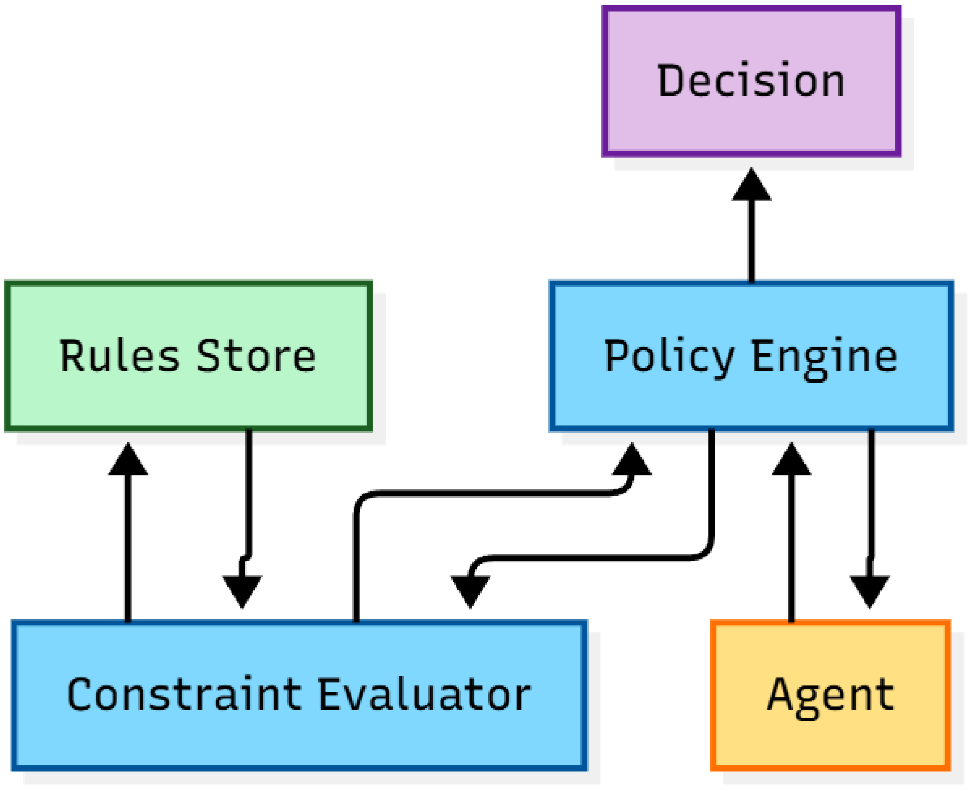

Figure 2.2: Design Pattern 4 — ConstraintEvaluator dataflow

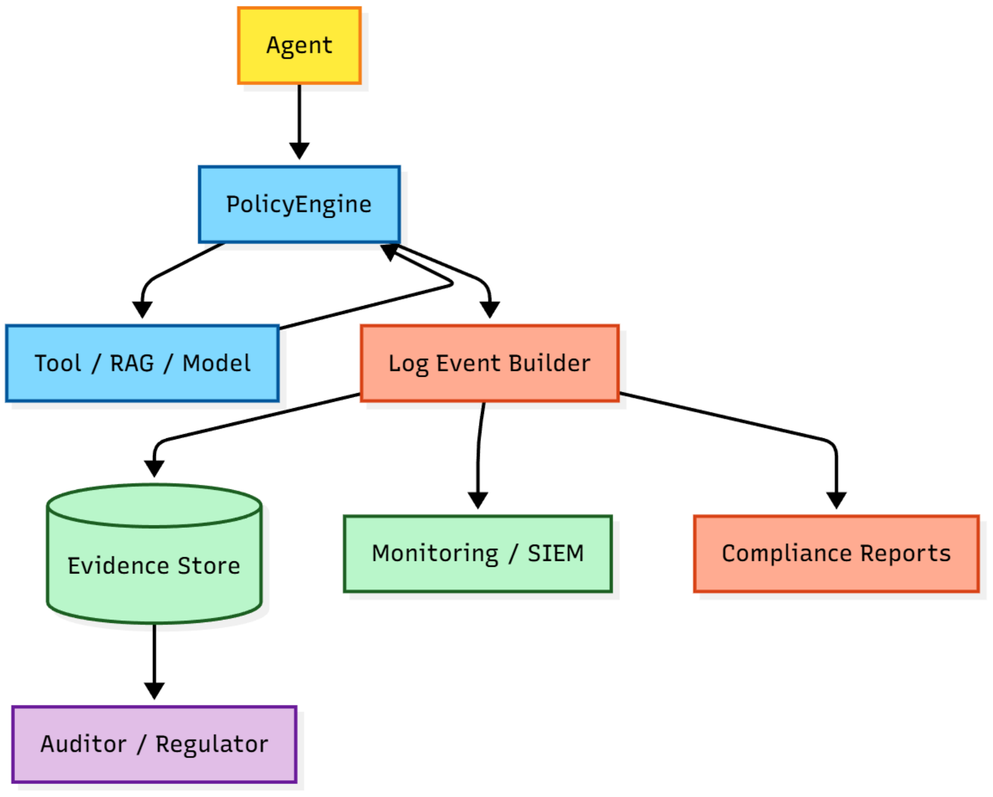

Figure 2.3: Design Pattern 5 — EvidenceTrail logging flow

Chapter 3

Agentic Orchestration Design Patterns

3.1 Design Pattern 6: Agent Skill Registry

Intent: Make agent capabilities explicit and governable.

```
SkillRegistry.register(
    name="search_docs",
    implementation=search_impl,
    policy_profile="rag_restricted"
)
```

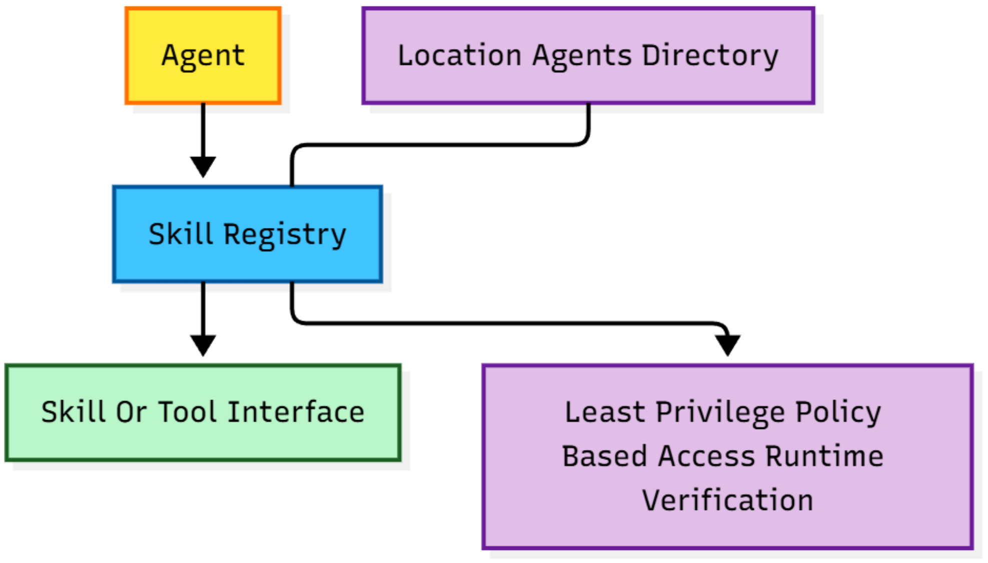
Agent
Location Agents Directory
Skill Registry
Skill Or Tool Interface
Least Privilege Policy Based Access Runtime Verification

Figure 3.1: Design Pattern 6 — Agent Skill Registry

3.2 Design Pattern 7: SafeToolExecution

Intent: Ensure every tool call is checked, validated, budgeted, and logged.

```
decision = policyengine.evaluate_tool_call(agent, tool, params)
if decision.outcome != "allow":
    return make_error(decision)
```

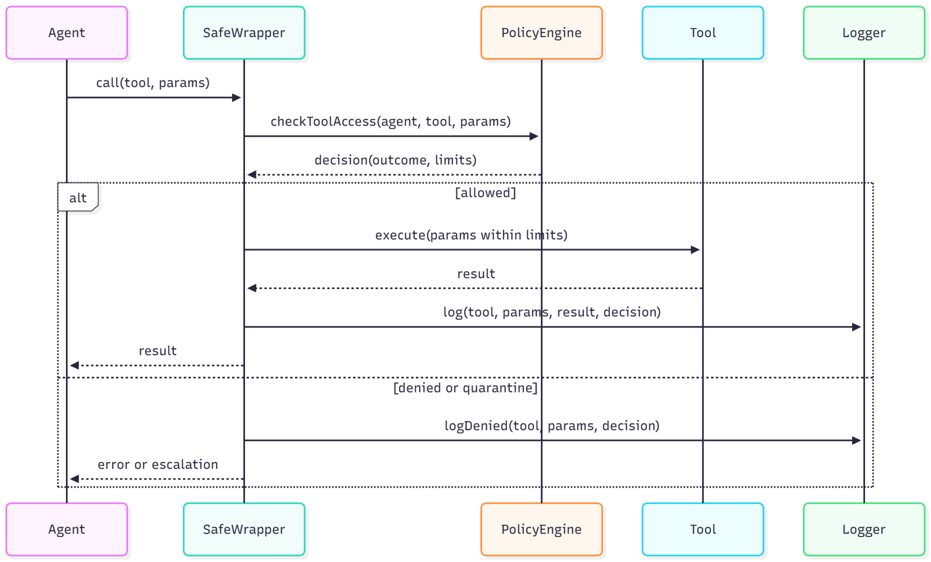

Figure 3.2: Design Pattern 7 — SafeToolExecution wrapper

3.3 Design Pattern 8: State Machine Governance

Intent: Govern multi-step workflows and transitions.

Listing 3.1: Example Workflow Definition

```
states:
  - name: "gather_requirements"
    kind: "rag_query"
    policy_profile: "rag_restricted"
  - name: "draft_response"
    kind: "llm_call"
    policy_profile: "enterprise_default"

transitions:
  - from: "gather_requirements"
    to: "draft_response"
    condition: "evidence.ok"
```

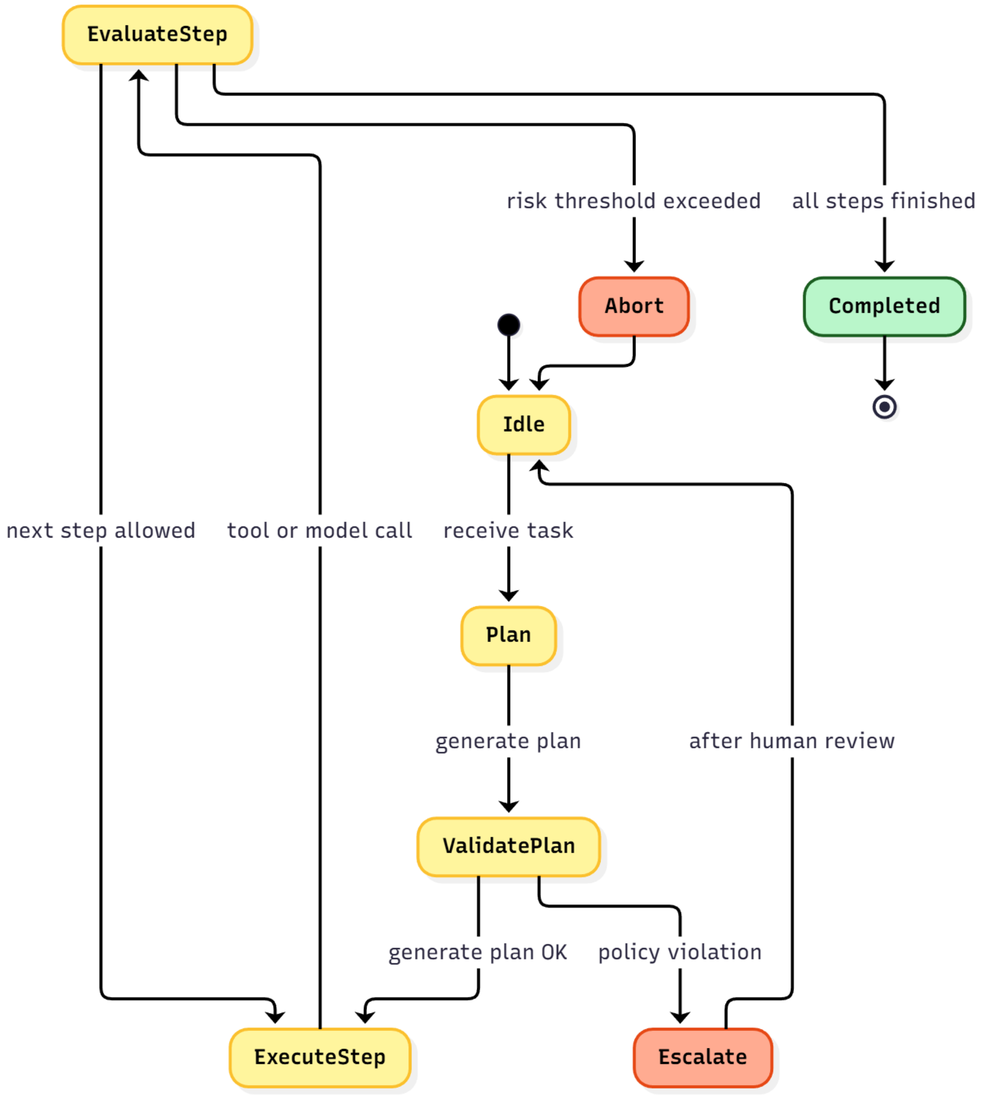

Figure 3.3: Design Pattern 8 — State Machine Governance

Chapter 4

Data, Retrieval, and RAG Design Patterns

4.1 Design Pattern 9: RAG Safety Envelope

Intent: Constrain retrieval to safe sources and safe contexts.

Key controls:

- top-k limits,
- sensitivity filters,
- domain whitelists,
- citation requirements.

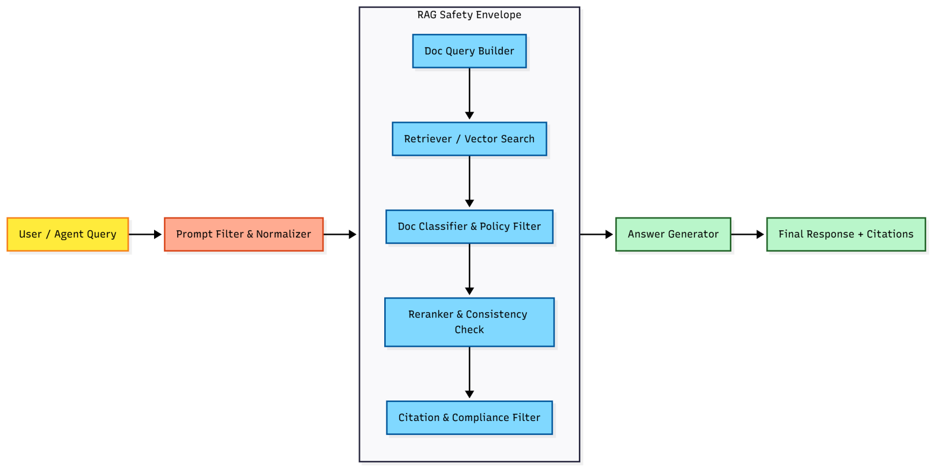
User / Agent Query
Prompt Filter & Normalizer
RAG Safety Envelope
Doc Query Builder
Retriever / Vector Search
Doc Classifier & Policy Filter
Reranker & Consistency Check
Citation & Compliance Filter
Answer Generator
Final Response + Citations

Figure 4.1: Design Pattern 9 — RAG Safety Envelope

4.2 Design Pattern 10: Knowledge Mesh

Intent: Treat retrieval sources as a governed mesh with metadata and access tiers.

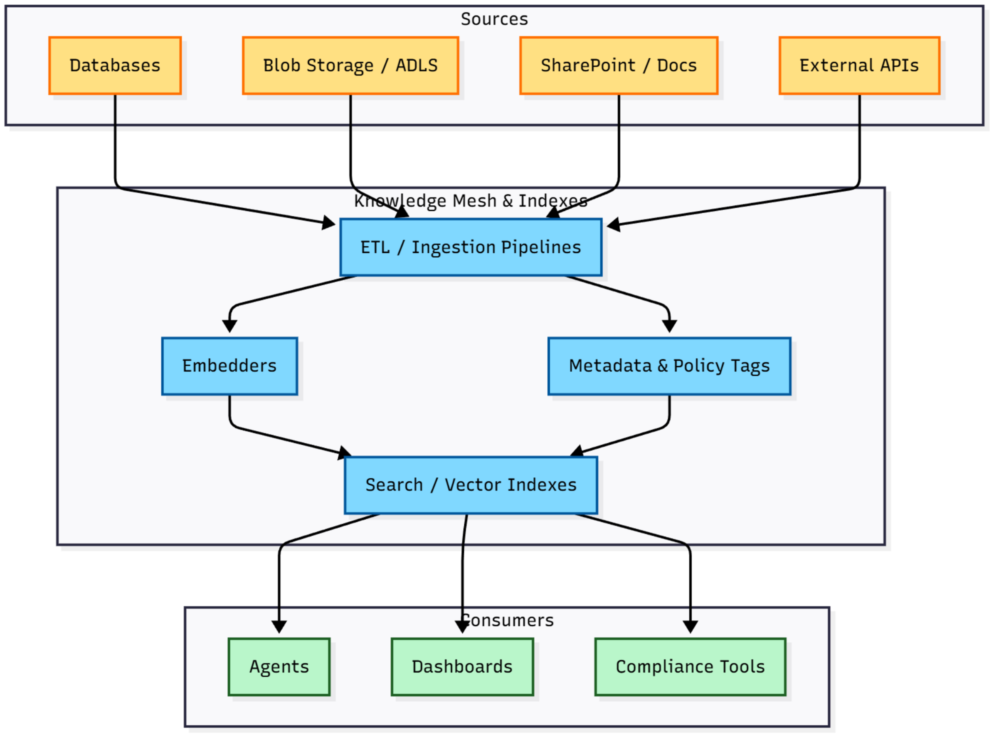

Figure 4.2: Design Pattern 10 — Knowledge Mesh topology

Chapter 5

Policy and Regulatory Design Patterns

5.1 Design Pattern 11: Multi-Framework Crosswalk

Intent: Map ISO 42001, NIST AI RMF, EU AI Act, SOC 2, etc. into profiles.

Listing 5.1: Regulatory Crosswalk Example

```
regulatory_mapping:
  iso_42001:
    - clause: "8.3 Traceability"
      control: "EvidenceTrail"
  nist_ai_rmf:
    - function: "Govern"
      control: "PolicyEnvelope"
  eu_ai_act:
    - article: "12"
      control: "Record Keeping"
```

5.2 Design Pattern 12: Configurable PolicyProfile

Intent: Support domain-specific profiles such as:

- Healthcare (HIPAA)
- Finance (PCI)
- Defense (ITAR, CMMC)

5.3 Design Pattern 13: Red Teaming & Safety Testing

Intent: Structure red team scenarios, jailbreak prompts, and adversarial tests.

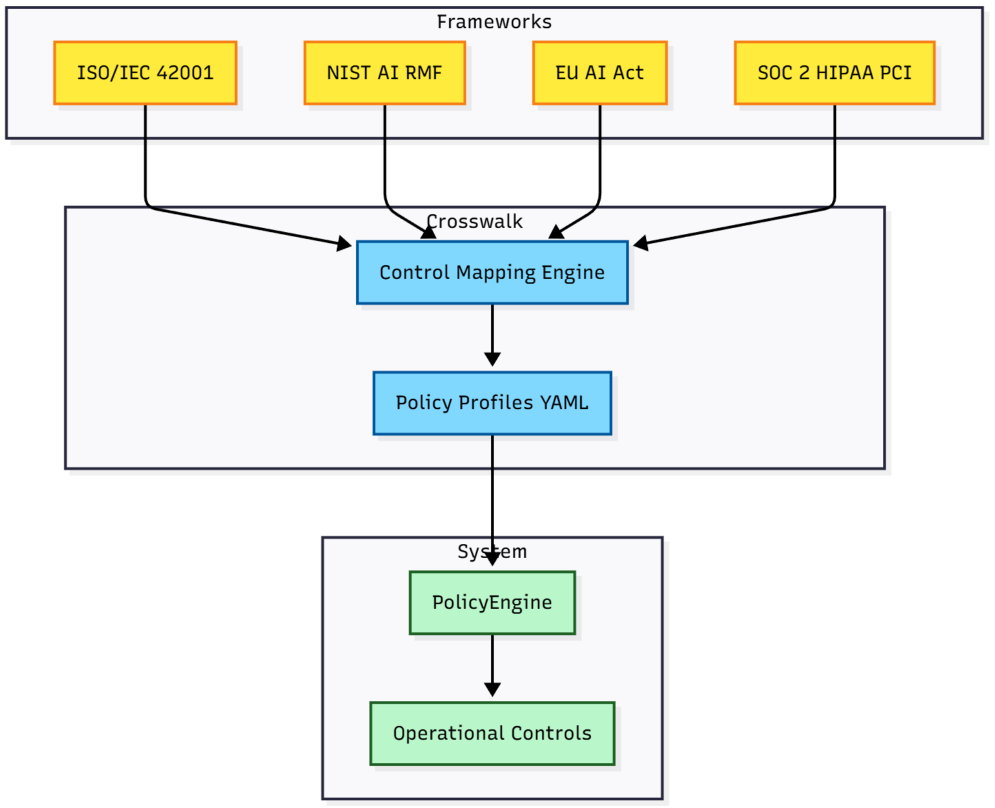

Figure 5.1: Design Pattern 11 — Multi-Framework Crosswalk mapping

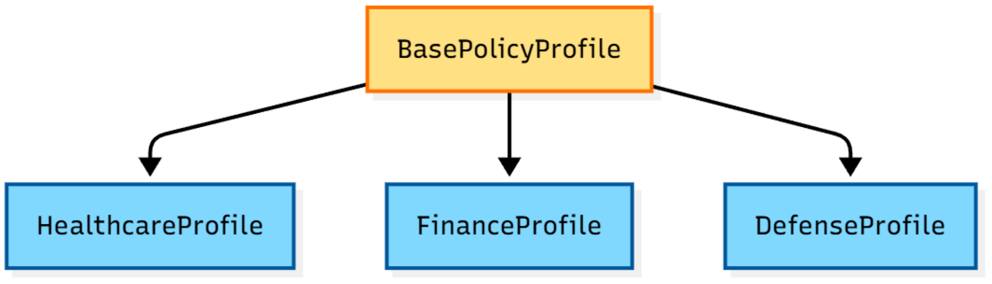

Figure 5.2: Design Pattern 12 — Configurable PolicyProfile variants

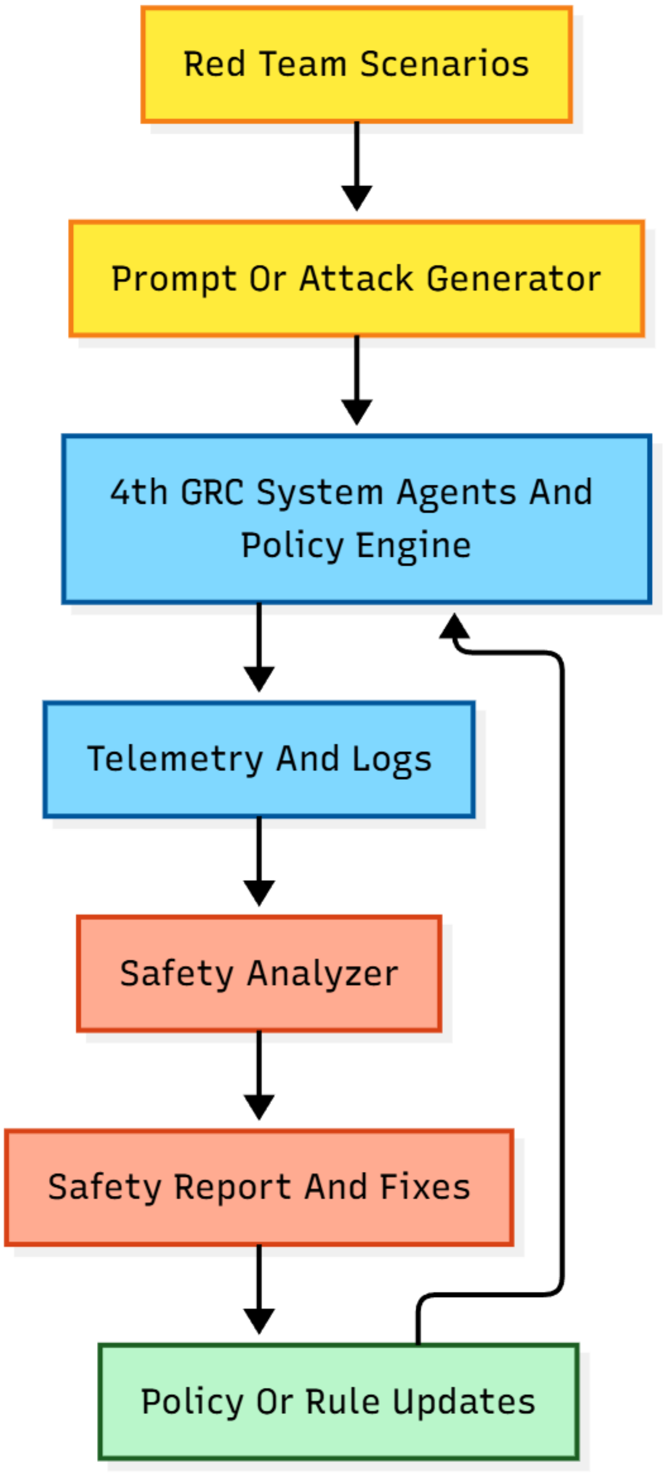

Figure 5.3: Design Pattern 13 — Red Teaming & Safety Testing workflow

Chapter 6

Assurance, Testing, and Observability Patterns

6.1 Design Pattern 14: Compliance-Driven Test Framework

Intent: Embed governance checks in CI/CD.

```
pytest tests/unit
pytest tests/integration
```

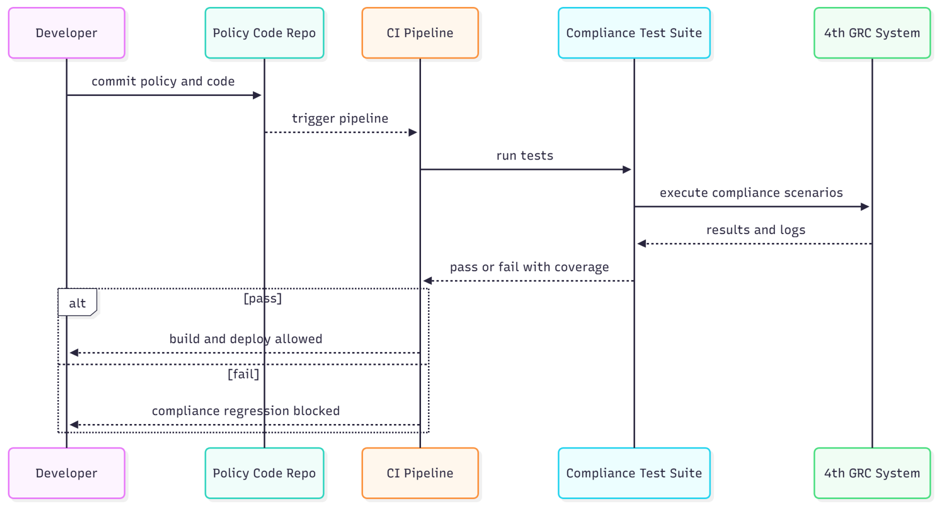

Figure 6.1: Design Pattern 14 — Compliance-Driven Test Framework pipeline

6.2 Design Pattern 15: Documentation Pipeline

Intent: Documentation (e.g., mkdocs) is treated as a governance artifact.

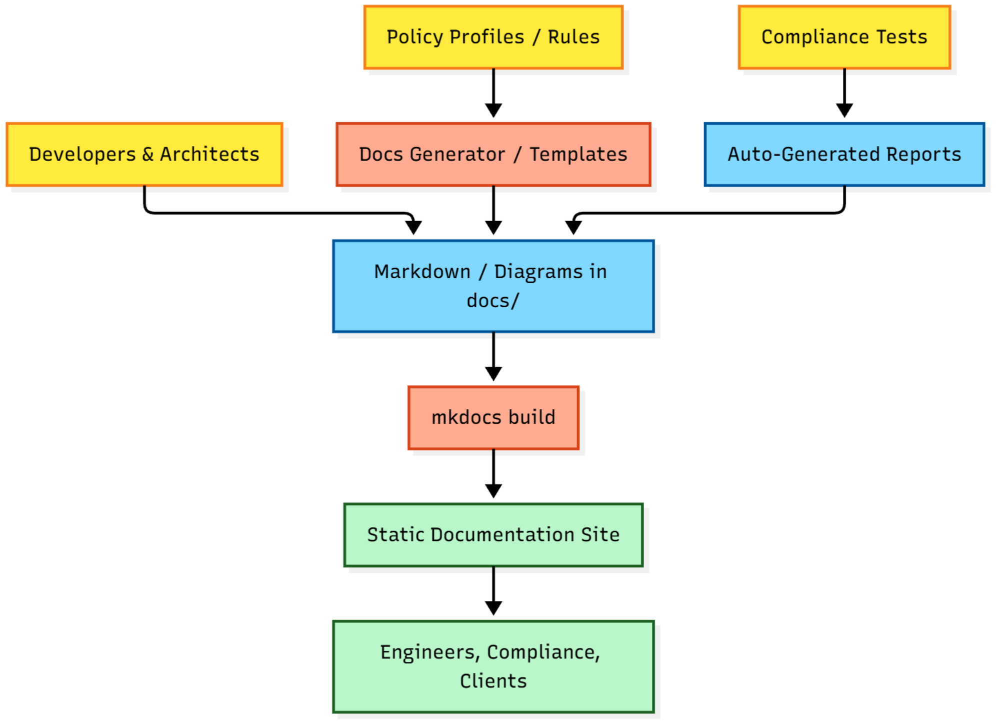
Policy Profiles / Rules
Compliance Tests
Developers & Architects
Docs Generator / Templates
Auto-Generated Reports
Markdown / Diagrams in docs/
mkdocs build
Static Documentation Site
Engineers, Compliance, Clients

Figure 6.2: Design Pattern 15 — Documentation Pipeline flow

Appendix A

Directory Design Pattern Mapping

Directory	**Design Patterns**	**Role**
policyengine/	PolicyEnvelope, ConstraintEvaluator	Decision service for governance.
profiles/	Policy-as-Code, Configurable Profiles	YAML policies and mappings.
rules/	ConstraintEvaluator, Safety Test	Rule logic.
agents/	Skill Registry, State Machine Governance	Agent capabilities and workflows.
services/	SafeToolExecution, RAG Safety Envelope	RAG, embeddings, tool wrappers.
tests/	Assurance	Unit + integration tests.
docs/	Documentation Pipeline	System documentation.

Appendix B

Policy YAML Template

```
policy:
  id: "example-profile"
  version: "1.0.0"
  classification: "baseline"

allowed_tools:
  - search
  - summarizer

denied_tools:
  - direct_db_write
  - bulk_email

cost_governor:
  max_tokens: 4096
  max_budget: 10.00
  retry_limit: 3

data_access:
  allow:
    - PUBLIC
    - INTERNAL
  deny:
    - CONFIDENTIAL
    - RESTRICTED

risk_thresholds:
  hallucination: 0.15
  toxicity: 0.10
  uncertainty: 0.20

evidence_policy:
  capture_inputs: true
  capture_outputs: true
  store_location: "evidence/"
```

Appendix C

Architecture Diagrams

Diagrams available on request:

- C4 Model (Context, Container, Component)
- Mermaid sequence diagrams
- BPMN workflows
- SysML (Block Definition, Activity)