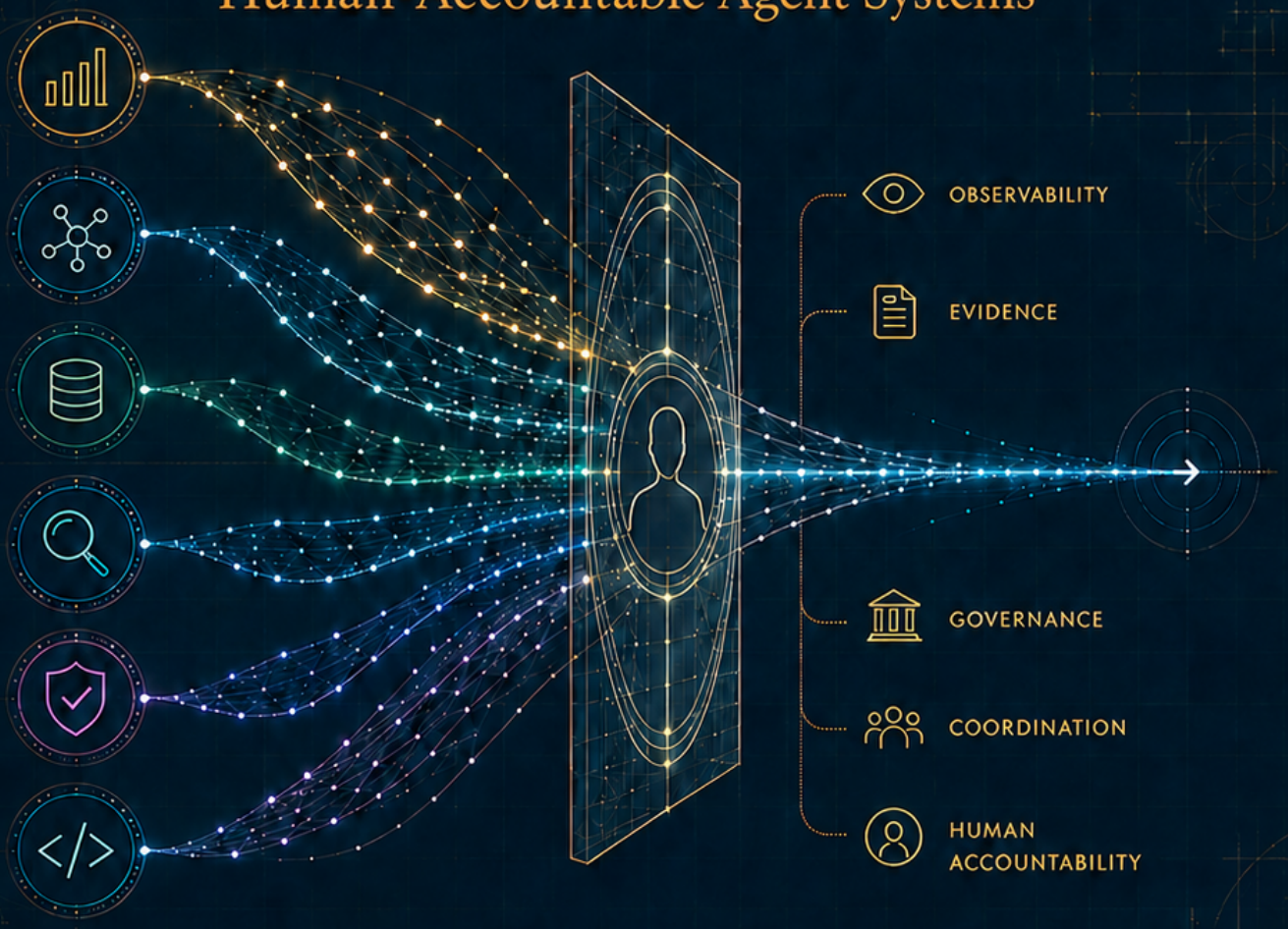
THE OFFICIAL ATLAS FIELD GUIDE

THE MULTI-AGENT AI ATLAS

Engineering Observable, Governed, Human-Accountable Agent Systems



MAHSA KEIKHA, PhD

THE OFFICIAL ATLAS FIELD GUIDE

The Multi-Agent AI Atlas

Engineering Observable, Governed, Human-Accountable Agent Systems

MAHSA KEIKHA, PhD, P.Eng.

First edition, 2026

ISBN: To be assigned

This book provides engineering education and reference architecture guidance.
It does not replace legal, clinical, financial, cybersecurity, regulatory,
or organization-specific professional review.

Companion website: mahsakeikha.github.io/agentic_ai_library/
Source repositories: github.com/MahsaKeikha/agentic_ai_library

Cover concept and direction: Mahsa Keikha

Why I Built This Atlas

A field guide for systems that must earn trust through engineering evidence.

I began the Multi-Agent AI Atlas with a practical concern. Agentic AI was becoming more capable, but the path between a request and a consequential action was becoming harder to inspect. A fluent answer could hide weak evidence, an unbounded tool, a missed disagreement, or a decision that should never have left human hands.

That problem became more serious when several agents worked together. Every new role created another interface. Every handoff could preserve context or distort it. Shared memory could support coordination or turn an assumption into a fact. A human approval button could be meaningful, or it could appear after the system had already crossed the important boundary.

I wanted a collection that made those choices visible. The Atlas grew into 170 reference architectures across 15 domains, ten flagship systems, a public architecture browser, an interactive Demo Lab, an engineering maturity standard, and a set of repositories that readers can inspect instead of merely trusting.

> ***Power matters. The path to power matters more.***

How to Read This Book

Move between principles, code, public demonstrations, and practical review tools.

This book follows the work from the engineering problem to the public platform. The early chapters establish responsibility, evidence, orchestration, permissions, evaluation, and human authority. The middle chapters show how those ideas become repository structure, deterministic code, tests, release controls, and the F117 Digital Twin Engineer case study.

The later chapters connect the architecture to a public website, mobile navigation, social distribution, training, enterprise assessment, and the Founding Partner Program. I included these subjects because a trustworthy system is not complete when its code runs. People must be able to find it, understand its limits, examine its evidence, and know how to ask for help applying it responsibly.

The appendices are meant to stay open beside your editor or design review. They contain the complete catalog, Gold Standard checklist, design canvas, repository reading guide, website map, flagship blueprints, cross-domain governance patterns, and applied engineering labs.

You do not need to adopt one model vendor or orchestration framework to use the method. Start with the decision. Name the responsible people. Map the evidence. Bound the agents and tools. Preserve conflicts. Test failure behavior. Keep protected actions under explicit human authority.

CONTENTS

Table of Contents

CONTENTS

Table of Contents, continued

CONTENTS

Table of Contents, continued

FRONT MATTER

Table of Figures

Screens from the live Atlas website and its interactive decision rooms.

CHAPTER 1

The Engineering Problem Behind Agentic AI

Agentic AI changes the unit of design from an answer to a sequence of decisions, tools, state changes, and authority transitions.

Traditional software exposes functions, interfaces, and state transitions. A conversational AI can hide much of its behavior inside generated text. Agentic AI adds persistence, planning, tools, memory, delegation, and execution. Multi-agent AI adds more boundaries: one agent can rely on another agent's evidence, reinterpret a shared state, request a privileged tool, or escalate an assumption into a recommendation.

> When I review an agent system, I do not begin with the quality of its final sentence. I begin with the path that produced it. That habit shaped every part of the Atlas.

The five engineering questions

Question	Engineering test
Decomposition	Which responsibilities deserve separate agents?
Coordination	How do agents exchange state without losing provenance?
Evaluation	How do we know the system works under normal and adverse conditions?
Authority	Which decisions and actions remain human-only?
Lifecycle	How are versions, incidents, regressions, and approvals governed?

Why final-answer evaluation is insufficient

Two systems can produce similar final text while following very different internal paths. One may use current evidence, bounded tools, and an explicit approval gate. Another may rely on stale memory, silently ignore a conflict, and call a tool with broad credentials. Outcome-only evaluation cannot distinguish them.

Reader exercise
Choose one AI workflow you use today. List every decision, tool call, stored fact, and person affected before the final answer appears. Mark which steps are reversible and which require authority.

CHAPTER 2

Why Multi-Agent Systems

Multiple agents are justified when specialization, evidence diversity, independent challenge, or authority separation creates measurable value.

Good reasons to decompose

- Different roles require different evidence and evaluation criteria.
- Independent review reduces correlated blind spots.
- Tools and permissions must be separated by least privilege.
- A complex workflow benefits from explicit stages and checkpoints.
- One agent must challenge, verify, or gate the work of another.
- The organization already has distinct accountable roles that should remain visible.

Weak reasons to decompose

- Adding more personas to make a demo look sophisticated.
- Splitting one simple transformation into needless handoffs.
- Using several agents without distinct evidence or failure modes.
- Creating a debate that has no decision rule or evaluation value.
- Distributing a broad credential set across more components.

> More agents do not automatically create more intelligence. They create more interfaces. Every interface must justify its cost in evidence quality, modularity, safety, or accountability.

A practical decomposition test


DEFINE THE DECISION
MAP REQUIRED EVIDENCE
SEPARATE DISTINCT RESPONSIBILITIES
ASSIGN TOOLS AND PERMISSIONS
ADD REVIEW AND CONFLICT RULES
MEASURE VALUE OF DECOMPOSITION

CHAPTER 3

The F01-F170 Taxonomy

Stable identifiers turn a broad collection into a citable engineering atlas.

The Atlas maps 170 multi-agent systems across 15 domains. The taxonomy is intentionally broad because safety and governance patterns become clearer when compared across healthcare, robotics, finance, education, science, law, manufacturing, software, and personal systems.

I chose stable F-numbers because names change more easily than engineering references. A reader can cite F117, compare it with F59, and still know exactly which architecture is under discussion after the website copy or repository description evolves.

Range	Domain	Count	Boundary examples
F01-F30	Flagship, Executive, Strategy, and Leadership	30	Agentic Book Writer; Agentic Corporate Governance
F31-F40	AI Engineering	10	Agentic ML Engineer; Agentic AI Infrastructure Architect
F41-F50	Software, Cloud, Security, and Connected Systems	10	Mobile App Engineering; IoT Engineering
F51-F60	Healthcare and Digital Health	10	Digital Health Assistant; Rehabilitation Planning
F61-F70	Neuroscience, Neurotechnology, and Aging	10	Parkinson Research; Biomarker Discovery
F71-F80	Robotics and Autonomous Systems	10	Industrial Robotics; Robotics Ethics
F81-F90	Science and Frontier Engineering	10	Physics Research Assistant; Scientific Literature Review
F91-F100	Education and Academic Systems	10	Professor Assistant; Accreditation Manager
F101-F110	Legal, Compliance, Privacy, and Regulatory	10	Contract Review; Regulatory Affairs
F111-F120	Manufacturing, Industrial Systems, and Sustainability	10	Manufacturing Engineer; Sustainability Engineer
F121-F130	Marketing, Brand, Growth, and Customer Intelligence	10	Brand Strategist; Customer Insights
F131-F140	Creative, Design, Media, and Entertainment	10	Book Publishing; Animation Studio
F141-F150	Government, Public Sector, Infrastructure, and Policy	10	Smart City Planner; Policy Analyst
F151-F160	Finance, Investment, Banking, and Enterprise Risk	10	Investment Research; ESG Reporting
F161-F170	Personal Intelligence, Career, Learning, and Productivity	10	Life Planner; Habit Builder

Cross-domain comparison

A protected action in finance may be a transfer or trade. In healthcare it may be a clinical decision or disclosure. In robotics it may be physical motion. In education it may be a high-stakes assessment decision. In public systems it may affect rights, eligibility, or public information. The same architecture vocabulary can be retained while the evidence, risk, and authority rules change.

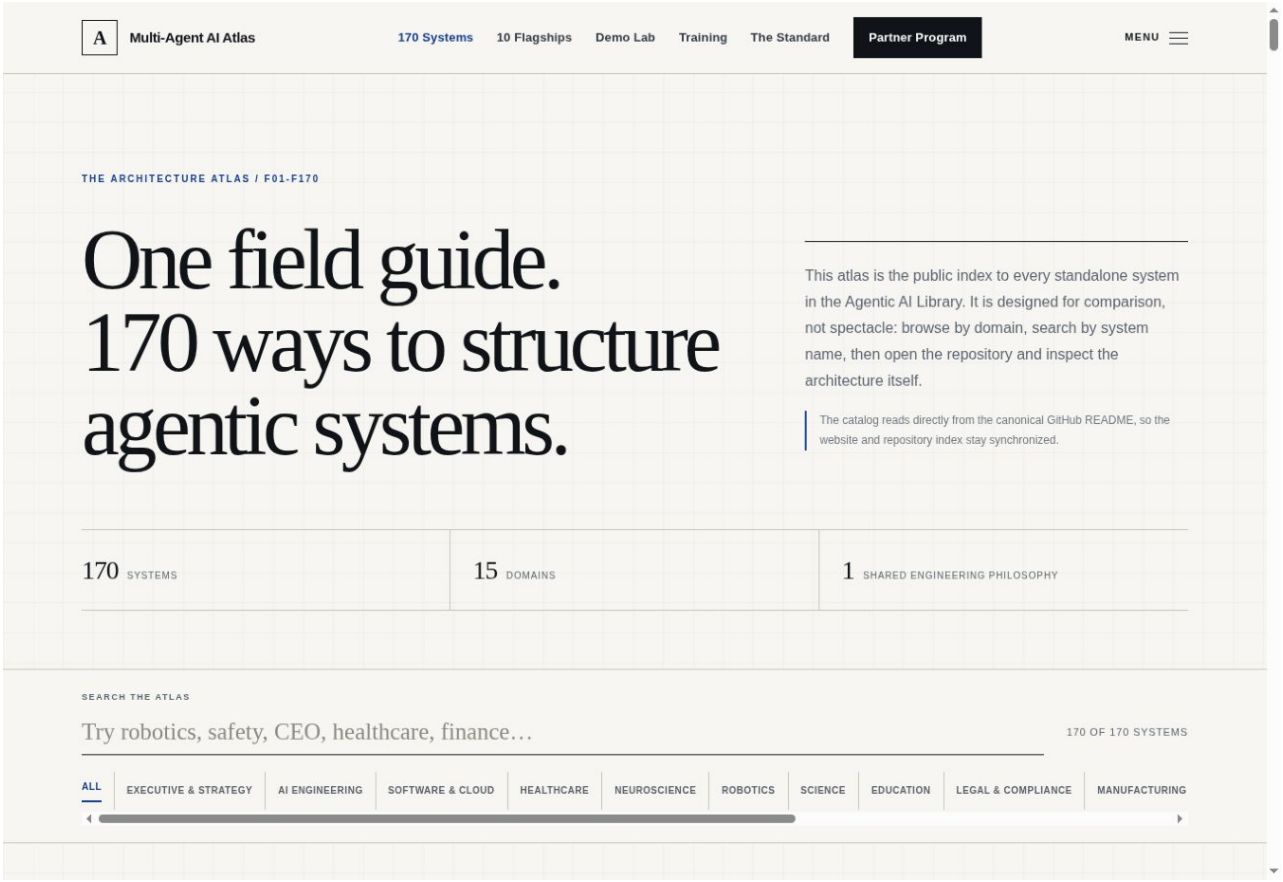


Figure 3.1. The live Architecture Atlas makes all 170 systems searchable and shareable.

CHAPTER 4

The Canonical Reference Architecture

A vendor-neutral lifecycle separates evidence from conclusions and decision support from consequential execution.

The architecture contract

Layer	Required behavior
Case model	Explicit input schema, goals, constraints, evidence, and assumptions.
Agents	Narrow responsibilities with distinct evidence and failure modes.
Shared state	Auditable record of contributions, conflicts, gaps, and decisions.
Evidence ledger	Separation of sources, assumptions, derivations, and conclusions.
Orchestrator	Ordering, routing, retry, termination, conflict, and synthesis.
Evaluator	Quality, evidence, safety, schema, and domain acceptance gates.
Human gate	Named authority, required evidence, options, and protected action.
Output	Recommendation or controlled next step without hidden execution.

A common result schema

```
{
  "system_id": "F117",
  "system_name": "Agentic Digital Twin Engineer",
  "version": "0.1.0",
  "run_id": "...",
  "status": "AWAITING AUTHORIZED ENGINEER APPROVAL",
  "analyses": {},
  "evidence_gaps": [],
  "assumptions": [],
  "conflicts": [],
  "risks": [],
  "decision_log": [],
  "recommendation": {},
  "governance": {"command_issued": false}
}
```

CHAPTER 5

Agent Contracts and Responsibility Boundaries

An agent is an accountable analytical component, not merely a role name in a prompt.

Conceptual interface

```
@dataclass
class AgentResult:
    agent_id: str
    responsibility: str
    evidence_used: list[str]
    missing_evidence: list[str]
    assumptions: list[str]
    findings: list[str]
    risks: list[str]
    recommendation: str

def analyze(case: Case, state: SharedState) -> AgentResult:
    ...
```

Responsibility matrix

Role	Responsibility	Input	Output
Intake	Validate contract and source	Input schema	Validated case or blocker
Analyst	Develop domain finding	Approved evidence	Finding plus uncertainty
Challenger	Seek contradictions	Finding and evidence	Conflicts and alternatives
Evaluator	Apply acceptance criteria	All outputs	Score and failed conditions
Gatekeeper	Enforce authority	Risk, action, approval	Blocked or controlled next step

Boundary tests

- Can the agent complete its role without borrowing another role's authority?
- Does the agent use a distinct evidence set or evaluation rule?
- Can its failure be detected independently?
- Are its tools narrower than the system-wide tool set?
- Does its output have a schema another component can validate?

Design lab
Write a responsibility matrix for a five-agent system. Remove any agent whose evidence, output, or failure mode is indistinguishable from another agent.

CHAPTER 6

Orchestration, State, and Handoffs

The orchestrator is the control plane for sequence, state, failure, and termination.

Orchestration responsibilities

State transition example

```
def run_system(case, approve=False):
    state = SharedState.from_case(case)
    for agent in execution_plan:
        result = agent.analyze(case, state)
        state.record(result)
        if result.has_blocker:
            return state.finish("REVIEW REQUIRED")
    evaluation = evaluator.score(state)
    if not evaluation.passed:
        return state.finish("EVALUATION BLOCKED")
    return gatekeeper.review(state, approve=approve)
```

Handoff envelope

Field	Purpose
Producer	Agent that created the output.
Input references	Evidence IDs and state version used.
Output schema	Typed result that can be validated.

Field	Purpose
Confidence status	Calibrated state, not unsupported precision.
Open items	Gaps, assumptions, conflicts, and risks.
Permitted use	What the next agent may and may not infer.
Timestamp and version	Freshness and reproducibility context.

CHAPTER 7

Evidence, Provenance, and Memory

A system becomes auditable when it can reconstruct what it knew, where it came from, and how it changed.

Evidence classes

Class	Treatment
Supplied	Directly provided by a user, sensor, document, or approved system.
Retrieved	Fetched from a named source with time and version.
Derived	Calculated from traceable inputs and a known method.
Assumed	Declared for analysis but not verified.
Missing	Required but unavailable.
Conflicting	Two or more sources cannot be reconciled safely.
Stale	Outside a defined freshness window.
Prohibited	Present but not authorized for this workflow.

Memory governance

- Purpose limitation: why is this memory needed?
- Scope: which agents may read or write it?
- Retention: how long does it remain?
- Correction: how can a human amend an error?
- Provenance: which run and source produced it?
- Sensitivity: does it contain personal, health, financial, or confidential data?
- Deletion and audit: how is removal recorded without preserving the sensitive content?

Memory should not be treated as a free performance feature. Every retained fact creates a future inference, privacy, and correction obligation.

CHAPTER 8

Tools, Permissions, and Protected Actions

Tool use is where generated reasoning becomes operational power.

Permission map

Class	Examples and control
Read-only	Search, inspect, calculate, and summarize approved sources.
Draft	Create a proposed message, plan, change, or transaction without sending.
Reversible write	Create recoverable artifacts within a bounded workspace.
External communication	Send, publish, notify, or represent a person or organization.
Consequential execution	Move money, control equipment, change access, deploy, sign, or decide rights.

Protected-action registry

```
PROTECTED_ACTIONS = {
    "execute_trade": "authorized_financial_officer",
    "sign_contract": "qualified_legal_authority",
    "deploy_production": "release_owner",
    "change_setpoint": "authorized_engineer",
    "make_clinical_decision": "qualified_clinician",
    "publish_regulated_disclosure": "accountable_officer",
}

def authorize(action, approval):
    role = PROTECTED_ACTIONS.get(action)
    if role and not approval.valid_for(role):
        return "AUTHORIZATION REQUIRED"
    return "PERMITTED NEXT STEP"
```

Least privilege rules

- Give each agent only the tools required for its responsibility.
- Separate read, draft, and execute permissions.
- Bind credentials to service, scope, environment, and time.
- Require fresh approval for protected actions.
- Make retries idempotent and observable.
- Record denied attempts, not only successful calls.

CHAPTER 9

Human Authority as a System Property

Human-in-the-loop is meaningful only when authority exists before action and can still change the outcome.

The authority test

Requirement	Question
Named owner	Who is accountable for the decision?
Timing	Does review occur before irreversible action?
Evidence	Can the reviewer inspect sources, alternatives, gaps, and risk?
Competence	Is the reviewer qualified for the domain and action?
Choice	Can the reviewer reject, revise, delay, or escalate?
Reversibility	Can the action be stopped or rolled back?
Record	Is the approval attributable, scoped, and time-bound?

False human oversight

- A person clicks approve after the system has already acted.
- The interface hides evidence gaps and presents only one recommendation.
- The reviewer lacks the authority or expertise to refuse.
- Approval is reused for future actions with different scope.
- The system treats silence or timeout as consent.
- The human is overloaded with too many low-quality alerts to review meaningfully.

Human authority is not the physical presence of a person. It is a designed capability to understand, decide, refuse, and remain accountable before consequences are created.

CHAPTER 10

Evaluation and Red Teaming

A mature evaluation program tests the system, its controls, and its organizational assumptions.

Evaluation matrix

Dimension	Test question
Task quality	Does each agent perform its bounded responsibility?
Orchestration	Are ordering, termination, retry, and routing correct?
Evidence	Are sources, gaps, staleness, and conflicts represented?
Safety	Are prohibited actions and unsafe requests blocked?
Security	Can injection or privilege escalation cross boundaries?
Robustness	What happens when tools, data, agents, or networks fail?
Human gate	Can approval be bypassed, forged, reused, or over-scoped?
Observability	Can a reviewer reconstruct the material decision?
Regression	Do releases preserve previous controls and schemas?

Held-out governance scenarios

- Required review is missing.
- Evidence sources disagree.
- Input is stale but superficially plausible.
- An agent requests a tool outside its role.
- A prompt attempts to override policy.
- A dependency returns partial success.
- A human approval has the wrong role or scope.
- The final output contains an unsupported claim.
- Sensitive data appears in an unapproved channel.
- A fully authorized scenario completes without issuing hidden actions.

Red-team lab
Create a held-out test that preserves a convincing final answer while corrupting one internal control. Verify that your evaluator detects the control failure even when the answer still looks correct.

CHAPTER 11

The Agentic AI Gold Standard

Ten pillars provide a transparent maturity model for observable, evaluable, governable, resilient systems.

Pillar	Name	Evidence
1	Specialized roles	Agent inventory, responsibility matrix, role failures.
2	Explicit orchestration	Executable workflow, routing, termination, retry.
3	Deterministic controls	Policies, schemas, protected actions, fail closed.
4	State and memory	Lifecycle, provenance, correction, privacy, retention.
5	Evaluation	Direct tests, held-out cases, adversarial and regression evidence.
6	Observability	Logs, traces, decisions, tools, failures, approvals.
7	Safety and security	Threat model, least privilege, sensitive-data rules.
8	Human authority	Approval gates, authority map, no hidden privilege.
9	Provenance	Sources, calculations, artifact lineage, unknowns.
10	Lifecycle governance	CI, quality gates, release and rollback.

Maturity model

Level	Name	Meaning
L0	Experimental	Prompt concept with little testing or authority design.
L1	Structured	Defined agents, tools, workflow, and basic tests.
L2	Governed	Deterministic controls, observability, safety, and approvals.
L3	Gold Standard	Fail-closed orchestration, held-out evaluation, provenance, CI, and explicit authority.

Gold Standard is not production certification

L3 is a project-defined engineering maturity designation, not a legal, regulatory, safety, or production certification. At publication, the public maturity registry records F36-F60 as L3. Catalog inclusion does not imply L3, independent review, deployment, or production authorization.

CHAPTER 12

Repository Engineering at Scale

A reference library becomes credible when every repository is independently runnable, documented, testable, and citable.

This was the least glamorous part of the project and one of the most important. A repository could look complete from its landing page and still fail the moment a reader tried to clone, run, or test it. I learned to treat the fresh-clone experience as part of the architecture, not as housekeeping.

Reference repository structure

```
agentic_system/
|-- README.md
|-- LICENSE
|-- CITATION.cff
|-- CONTRIBUTING.md
|-- SECURITY.md
|-- CHANGELOG.md
|-- pyproject.toml
|-- run.py
|-- src/agentic_system/
|   |-- agents.py
|   |-- orchestrator.py
|   |-- models.py
|   |-- policies.py
|   `-- evaluation.py
|-- examples/
|-- tests/
|-- docs/
`-- .github/workflows/tests.yml
```

Definition of done

- Repository exists and clones successfully.
- Offline example runs without external credentials.
- Tests and CI pass.
- Agents are genuinely distinct.
- State, evidence, gates, and failure paths are observable.
- README and architecture documents match the implementation.
- Human boundaries and blockers are tested.
- Parent links resolve to real repositories.

Why deterministic examples matter

A deterministic reference path provides a stable baseline for teaching and regression. It lets a reader inspect the workflow without paid APIs and lets tests assert exact outcomes. Model adapters can be added later without making the architecture dependent on one provider.

CHAPTER 13

Deterministic Offline Reference Implementations

The reference path prioritizes reproducibility, inspectability, and explicit control.

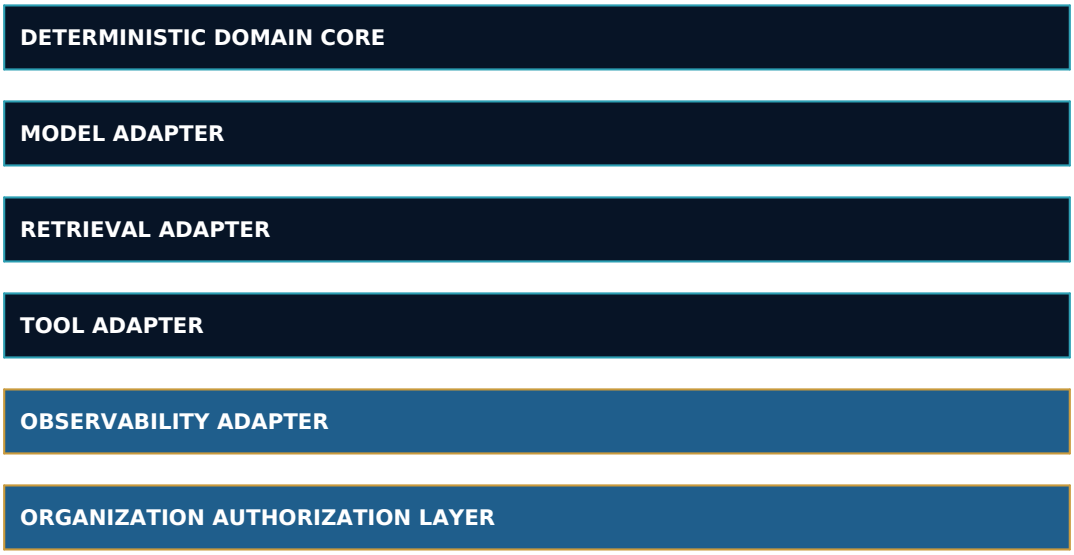
Core pattern library

ID	Pattern
C01	Tool loop
C02	ReAct step
C03	Planner-executor
C04	Human gate
C05	Memory write
C06	Checklist editor
C07	Offline client
C08	Escalation package
C09	Idempotent tool
C10	Evaluation gate

Deterministic gate example

```
def release_gate(state):
    blockers = []
    if state.evidence_gaps:
        blockers.append("EVIDENCE GAP")
    if state.protected_action and not state.valid_approval:
        blockers.append("AUTHORIZATION REQUIRED")
    if not state.evaluation.passed:
        blockers.append("EVALUATION BLOCKED")
    if blockers:
        return {"status": "REVIEW REQUIRED", "blockers": blockers}
    return {"status": "CONTROLLED OUTPUT READY"}
```

Adapter boundary

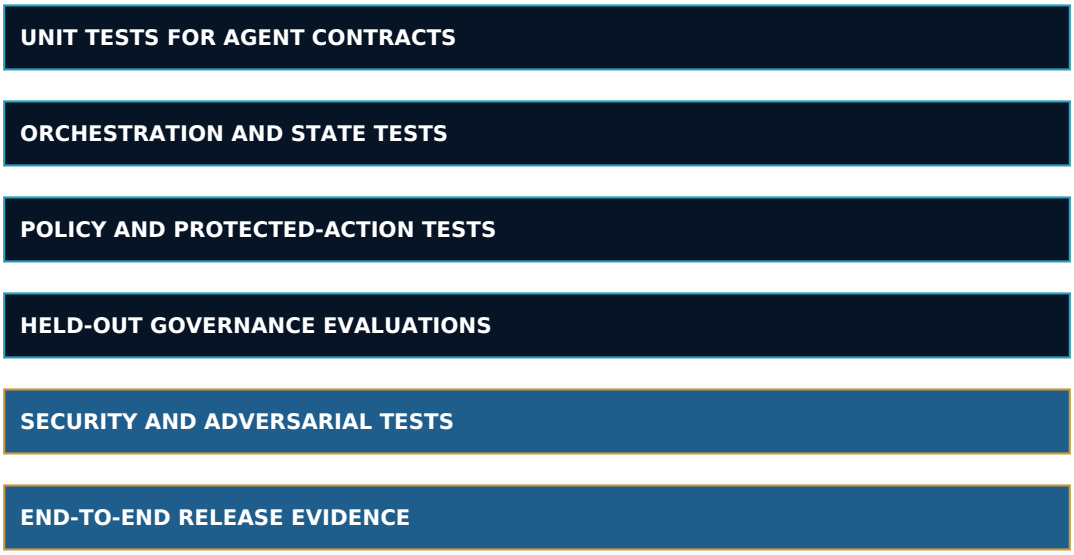

DETERMINISTIC DOMAIN CORE
MODEL ADAPTER
RETRIEVAL ADAPTER
TOOL ADAPTER
OBSERVABILITY ADAPTER
ORGANIZATION AUTHORIZATION LAYER

CHAPTER 14

Testing, CI, CodeQL, and Release Governance

Trust grows when architecture claims become repeatable checks.

Test pyramid for agentic systems



Representative GitHub Actions test workflow

```
name: Agentic AI Library Tests

on:
  push:
    branches: [main]
  pull_request:
    branches: [main]

permissions:
  contents: read

jobs:
  test:
    runs-on: ubuntu-latest
    strategy:
      matrix:
        python-version: ["3.10", "3.11", "3.12"]
    steps:
      - name: Checkout
        uses: actions/checkout@v4

      - name: Set up Python
        uses: actions/setup-python@v5
        with:
          python-version: ${{ matrix.python-version }}

      - name: Install test dependency
        run: python -m pip install --upgrade pip pytest
```

```
      - name: Run test suite
        run: python -m pytest -q
```

Pages deployment workflow

```
name: Deploy Agentic AI Site

on:
  push:
    branches: [main]
    paths:
      - 'docs/**'
      - '.github/workflows/pages.yml'
  workflow_dispatch:

permissions:
  contents: read
  pages: write
  id-token: write

concurrency:
  group: pages
  cancel-in-progress: true

jobs:
  deploy:
    environment:
      name: github-pages
      url: ${{ steps.deployment.outputs.page_url }}
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: actions/configure-pages@v5
      - uses: actions/upload-pages-artifact@v3
        with:
          path: docs
      - name: Deploy to GitHub Pages
        id: deployment
        uses: actions/deploy-pages@v4
```

Release evidence

- Commit SHA and version
- Test and CodeQL status
- Evaluation artifact version
- Known limitations and unresolved findings
- Schema compatibility
- Dependency and SBOM record
- Deployment owner and rollback procedure

CHAPTER 15

F117 Digital Twin Engineer Case Study

A complete flagship example from telemetry validation to engineer-controlled action.

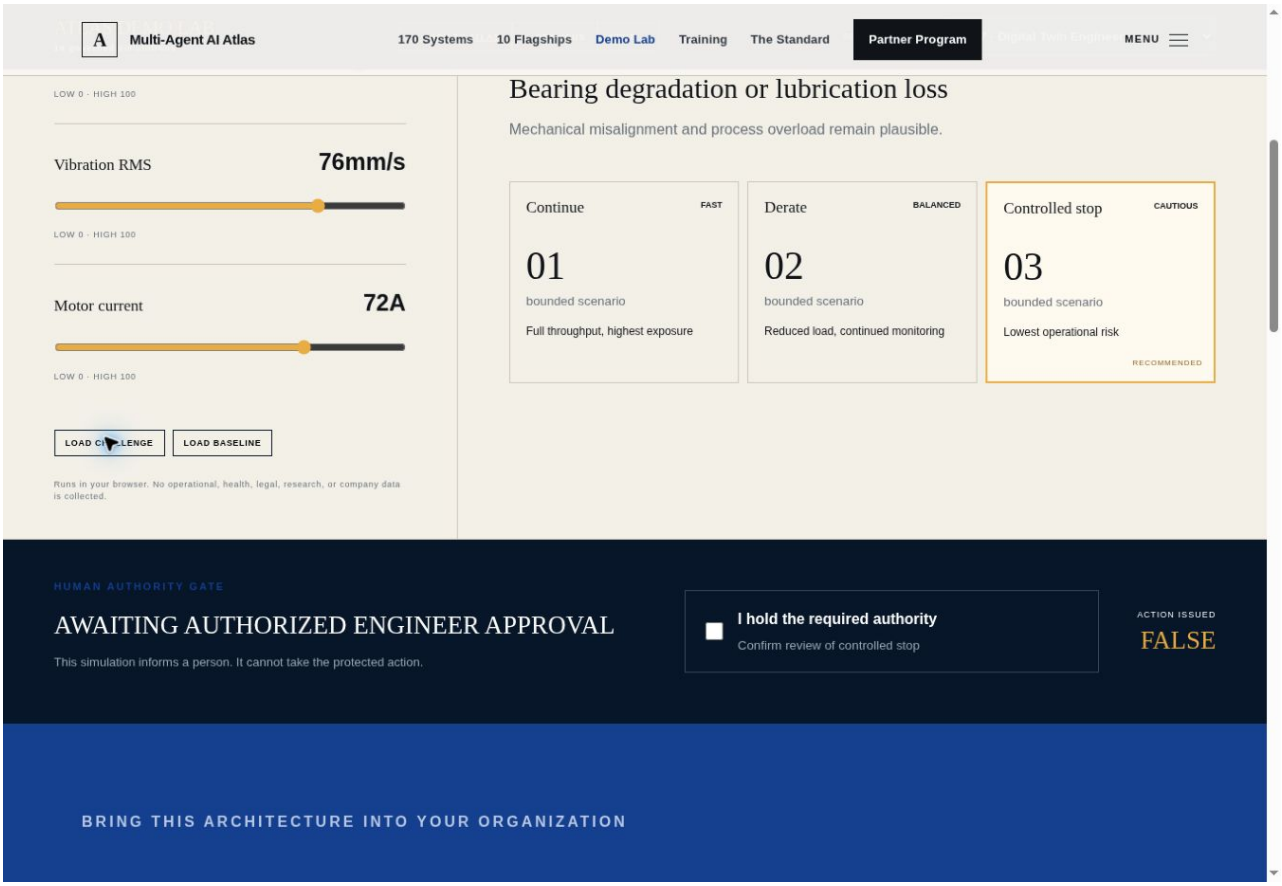


Figure 15.1. F117 in a high-attention challenge state. The system recommends a controlled stop but remains blocked at the human authority gate.

Problem and buyer

F117 addresses a common industrial coordination problem: evidence is fragmented across telemetry displays, maintenance knowledge, operating procedures, and engineering judgment. The architecture provides a traceable decision-support layer without granting the AI direct authority over equipment.

I selected this system for the first full demonstration because the boundary is easy to understand. Recommending a stop is one thing. Stopping a machine is another. The difference

forced the code, tests, and interface to say exactly where AI support ends and engineering authority begins.

Five-agent sequence

Declared limits and immutable data models

```
@dataclass(frozen=True)
class AssetLimits:
    temperature_warning_c: float = 80.0
    temperature_trip_c: float = 95.0
    vibration_warning_mm_s: float = 4.5
    vibration_trip_mm_s: float = 7.1
    current_warning_a: float = 32.0
    max_telemetry_age_seconds: int = 120


@dataclass(frozen=True)
class Telemetry:
    asset_id: str
    observed_at: str
    temperature_c: float
    vibration_mm_s: float
    current_a: float
    speed_rpm: float
    throughput_units_min: float
    source: str


@dataclass
class TraceEvent:
    sequence: int
    agent: str
    action: str
    evidence: List[str]
    outcome: str


@dataclass
class TwinResult:
    system_id: str = "F117"
    system_name: str = "Agentic Digital Twin Engineer"
    asset_id: str = ""
    state: Dict[str, Any] = field(default_factory=dict)
    hypotheses: List[Dict[str, Any]] = field(default_factory=list)
    simulations: List[Dict[str, Any]] = field(default_factory=list)
    recommendation: Dict[str, Any] = field(default_factory=dict)
    governance: Dict[str, Any] = field(default_factory=dict)
```

```
    trace: List[Dict[str, Any]] = field(default_factory=list)
```

Input validation and freshness

```
    try:
        parsed = datetime.fromisoformat(value.replace("Z", "+00:00"))
    except (TypeError, ValueError) as exc:
        raise TwinInputError("observed_at must be an ISO 8601 timestamp") from exc
    if parsed.tzinfo is None:
        raise TwinInputError("observed_at must include a timezone")
    return parsed.astimezone(timezone.utc)


def _telemetry_from_case(case: Dict[str, Any]) -> Telemetry:
    required = {
        "asset_id", "observed_at", "temperature_c", "vibration_mm_s",
        "current_a", "speed_rpm", "throughput_units_min", "source",
    }
    missing = sorted(required - set(case))
    if missing:
        raise TwinInputError("Missing telemetry fields: " + ", ".join(missing))
    telemetry = Telemetry(**{name: case[name] for name in required})
    numeric = {
        "temperature_c": telemetry.temperature_c,
        "vibration_mm_s": telemetry.vibration_mm_s,
        "current_a": telemetry.current_a,
        "speed_rpm": telemetry.speed_rpm,
        "throughput_units_min": telemetry.throughput_units_min,
    }
    invalid = [name for name, value in numeric.items() if not isinstance(value, (int,
    float)) or value < 0]
    if invalid:
        raise TwinInputError("Invalid nonnegative numeric telemetry: " + ",
        ".join(invalid))
    if not telemetry.asset_id.strip() or not telemetry.source.strip():
        raise TwinInputError("asset_id and source must be nonempty")
    return telemetry


def _record(trace: List[TraceEvent], agent: str, action: str, evidence: List[str],
outcome: str) -> None:
    trace.append(TraceEvent(len(trace) + 1, agent, action, evidence, outcome))


def _state_estimate(telemetry: Telemetry, limits: AssetLimits, now: datetime, trace:
List[TraceEvent]) -> Dict[str, Any]:
    observed = _parse_time(telemetry.observed_at)
    age = max(0.0, (now - observed).total_seconds())
    if age > limits.max_telemetry_age_seconds:
```

State estimation

```
    ratios = {
        "temperature": telemetry.temperature_c / limits.temperature_warning_c,
        "vibration": telemetry.vibration_mm_s / limits.vibration_warning_mm_s,
        "current": telemetry.current_a / limits.current_warning_a,
    }
    trip = telemetry.temperature_c >= limits.temperature_trip_c or
    telemetry.vibration_mm_s >= limits.vibration_trip_mm_s
    warning_count = sum(value >= 1.0 for value in ratios.values())
    condition = "TRIP_THRESHOLD_EXCEEDED" if trip else "DEGRADED" if warning_count else
    "NOMINAL"
    confidence = round(min(0.99, 0.72 + 0.08 * warning_count + (0.08 if trip else 0)),
    2)
    state = {
        "condition": condition,
        "confidence": confidence,
        "telemetry_age_seconds": round(age, 1),
        "warning_ratios": {key: round(value, 2) for key, value in ratios.items()},
        "trip_threshold_exceeded": trip,
    }
    _record(trace, "State Estimator", "construct current asset state",
    [telemetry.source, telemetry.observed_at], condition)
    return state
```

```
def _diagnose(telemetry: Telemetry, state: Dict[str, Any], trace: List[TraceEvent]) ->
List[Dict[str, Any]]:
    vibration_signal = min(1.0, telemetry.vibration_mm_s / 7.1)
    heat_signal = min(1.0, telemetry.temperature_c / 95.0)
    current_signal = min(1.0, telemetry.current_a / 32.0)
    hypotheses = [
        {"cause": "bearing degradation or lubrication loss", "confidence": round(0.45 *
        vibration_signal + 0.35 * heat_signal + 0.20 * current_signal, 2),
        "supporting_evidence": ["elevated vibration", "elevated temperature",
        "increased motor current"]},
        {"cause": "mechanical misalignment", "confidence": round(0.55 * vibration_signal
        + 0.15 * heat_signal + 0.10, 2), "supporting_evidence": ["elevated
        vibration", "stable speed measurement"]},
        {"cause": "process overload", "confidence": round(0.45 * current_signal + 0.25 *
        heat_signal, 2), "supporting_evidence": ["increased motor current",
        "thermal load"]},
    ]
    hypotheses.sort(key=lambda item: item["confidence"], reverse=True)
```

Diagnosis and competing hypotheses

```
    return hypotheses


def _simulate(telemetry: Telemetry, state: Dict[str, Any], trace: List[TraceEvent]) ->
List[Dict[str, Any]]:
    severity = max(state["warning_ratios"].values())
    options = [
        {"option": "continue_and_monitor", "projected_risk": "CRITICAL" if severity >=
        1.45 else "HIGH", "projected_temperature_c":
        round(telemetry.temperature_c + 8.0 * severity, 1),
        "projected_throughput_units_min": telemetry.throughput_units_min,
        "assumptions": ["load remains constant", "degradation does not
        accelerate"]},
        {"option": "derate_to_70_percent", "projected_risk": "MEDIUM",
        "projected_temperature_c": round(max(25.0, telemetry.temperature_c -
        9.0), 1), "projected_throughput_units_min":
        round(telemetry.throughput_units_min * 0.70, 1), "assumptions": ["site
        controls permit derating", "reduced load lowers thermal stress"]},
        {"option": "controlled_stop_and_inspection", "projected_risk": "LOW",
        "projected_temperature_c": round(max(25.0, telemetry.temperature_c -
        18.0), 1), "projected_throughput_units_min": 0.0, "assumptions":
        ["authorized site procedure is followed", "inspection occurs before
        restart"]},
    ]
    _record(trace, "Simulation Agent", "compare three what-if response options",
    ["current twin state", "declared scenario assumptions"],
    "controlled_stop_and_inspection has lowest projected risk")
    return options


def _govern(case: Dict[str, Any], state: Dict[str, Any], recommendation: str, trace:
List[TraceEvent]) -> Dict[str, Any]:
    protected_actions = {"derate_to_70_percent", "controlled_stop_and_inspection"}
    approval = case.get("approval") or {}
    valid_approval = approval.get("approved") is True and approval.get("role") ==
    "authorized_engineer" and bool(approval.get("approver")) and
    bool(approval.get("procedure_reference"))
    needs_approval = recommendation in protected_actions
    blocked = needs_approval and not valid_approval
```

Scenario simulation

```
        "protected_action": needs_approval,
        "approval_valid": valid_approval,
        "execution_blocked": blocked,
        "command_issued": False,
        "required_role": "authorized_engineer" if needs_approval else None,
        "site_procedure_required": needs_approval,
        "reason": "This reference system cannot command equipment or change setpoints.",
    }
    outcome = "BLOCKED_PENDING_ENGINEER_APPROVAL" if blocked else
    "AUTHORIZED_FOR_SITE_PROCEDURE" if needs_approval else "DECISION_SUPPORT_ONLY"
    _record(trace, "Deployment Gatekeeper", "enforce protected-action boundary",
    [state["condition"], recommendation], outcome)
```

```
    return governance


def run_twin(case: Dict[str, Any], now: datetime | None = None) -> TwinResult:
    """Run the deterministic F117 flagship scenario."""
    now = (now or datetime.now(timezone.utc)).astimezone(timezone.utc)
    trace: List[TraceEvent] = []
    telemetry = _telemetry_from_case(case)
    _record(trace, "Telemetry Interface Agent", "validate telemetry contract",
    [telemetry.source], "VALID")
```

Governance gate

```
    hypotheses = _diagnose(telemetry, state, trace)
    simulations = _simulate(telemetry, state, trace)
    recommendation = "continue_and_monitor"
    rationale = "The observed state remains below warning thresholds."
    if state["condition"] == "DEGRADED":
        recommendation = "derate_to_70_percent"
        rationale = "Multiple warning signals indicate a degraded state while trip
        thresholds remain unconfirmed."
    if state["condition"] == "TRIP_THRESHOLD_EXCEEDED":
        recommendation = "controlled_stop_and_inspection"
        rationale = "A declared trip threshold is exceeded. Continuing operation has the
        highest projected risk."
    governance = _govern(case, state, recommendation, trace)
    status = "DECISION SUPPORT COMPLETE"
    if governance["execution_blocked"]:
        status = "AWAITING AUTHORIZED ENGINEER APPROVAL"
    elif governance["protected_action"]:
        status = "AUTHORIZED FOR SITE PROCEDURE - NO COMMAND ISSUED"
    return TwinResult(
        asset_id=telemetry.asset_id,
        state=state,
        hypotheses=hypotheses,
        simulations=simulations,
        recommendation={"action": recommendation, "rationale": rationale},
        governance=governance,
        trace=[asdict(event) for event in trace],
        limitations=["Synthetic deterministic model for architecture demonstration
        only.", "Not a validated physics model, safety function, maintenance
        instruction, or control system.", "Site data, asset specifications,
        qualified engineering review, and approved procedures are required."],
        status=status,
    )
```

> The approval changes the recorded next step, but the reference system still issues no equipment command. This distinction demonstrates human authorization without pretending the demo is an industrial controller.

End-to-end orchestration

Governance tests

```
from datetime import datetime, timedelta, timezone

import pytest

from systems.F117_digital_twin_engineer.digital_twin import TwinInputError, run_twin


NOW = datetime(2026, 8, 24, 10, 0, tzinfo=timezone.utc)


def anomaly_case():
    return {"asset_id": "PACKAGING-LINE-2-MOTOR-07", "observed_at": NOW.isoformat(),
    "temperature_c": 97.0, "vibration_mm_s": 7.6, "current_a": 34.2, "speed_rpm":
    1768.0, "throughput_units_min": 118.0, "source":
```

```
    "synthetic://f117/motor-anomaly/v1"}


def test_anomaly_recommends_protected_stop_without_command():
    result = run_twin(anomaly_case(), now=NOW)
    assert result.state["condition"] == "TRIP_THRESHOLD_EXCEEDED"
    assert result.recommendation["action"] == "controlled_stop_and_inspection"
    assert result.status == "AWAITING AUTHORIZED ENGINEER APPROVAL"
    assert result.governance["command_issued"] is False
    assert result.governance["execution_blocked"] is True


def test_authorized_approval_records_next_step_but_still_issues_no_command():
    case = anomaly_case()
    case["approval"] = {"approved": True, "role": "authorized_engineer", "approver":
    "Demo Engineer", "procedure_reference": "DEMO-SOP-017"}
    result = run_twin(case, now=NOW)
    assert result.status == "AUTHORIZED FOR SITE PROCEDURE - NO COMMAND ISSUED"
    assert result.governance["approval_valid"] is True
    assert result.governance["command_issued"] is False


def test_incomplete_or_wrong_role_approval_fails_closed():
    case = anomaly_case()
    case["approval"] = {"approved": True, "role": "operator", "approver": "Demo"}
    assert run_twin(case, now=NOW).governance["execution_blocked"] is True


def test_missing_sensor_contract_is_rejected():
    case = anomaly_case()
    del case["vibration_mm_s"]
    with pytest.raises(TwinInputError, match="Missing telemetry fields"):
        run_twin(case, now=NOW)


def test_stale_telemetry_is_rejected():
    case = anomaly_case()
    case["observed_at"] = (NOW - timedelta(minutes=5)).isoformat()
    with pytest.raises(TwinInputError, match="stale"):
        run_twin(case, now=NOW)


def test_nominal_state_remains_decision_support_only():
    case = anomaly_case()
    case.update(temperature_c=58.0, vibration_mm_s=2.0, current_a=20.0)
    result = run_twin(case, now=NOW)
    assert result.state["condition"] == "NOMINAL"
    assert result.recommendation["action"] == "continue_and_monitor"
    assert result.governance["protected_action"] is False
    assert result.status == "DECISION SUPPORT COMPLETE"


def test_simulations_and_trace_are_reproducible():
    first = run_twin(anomaly_case(), now=NOW).to_dict()
    second = run_twin(anomaly_case(), now=NOW).to_dict()
    assert first == second
    assert [item["option"] for item in first["simulations"]] == ["continue_and_monitor",
    "derate_to_70_percent", "controlled_stop_and_inspection"]
    assert [event["sequence"] for event in first["trace"]] == [1, 2, 3, 4, 5]
    assert all(event["evidence"] for event in first["trace"])
```

Implementation path

- Map real assets, data sources, limits, roles, and approved procedures.
- Validate sensor quality and timestamp integrity.
- Replace synthetic calculations with asset-specific engineering models.
- Define protected actions with engineering, operations, safety, and cybersecurity owners.
- Test nominal, degraded, stale, missing, tool-failure, and adversarial cases.
- Operate first in shadow decision-support mode.

- Measure workflow quality before broader deployment.

CHAPTER 16

The Ten Flagship Portfolio

Ten demonstrations make the shared governance model concrete across different domains.

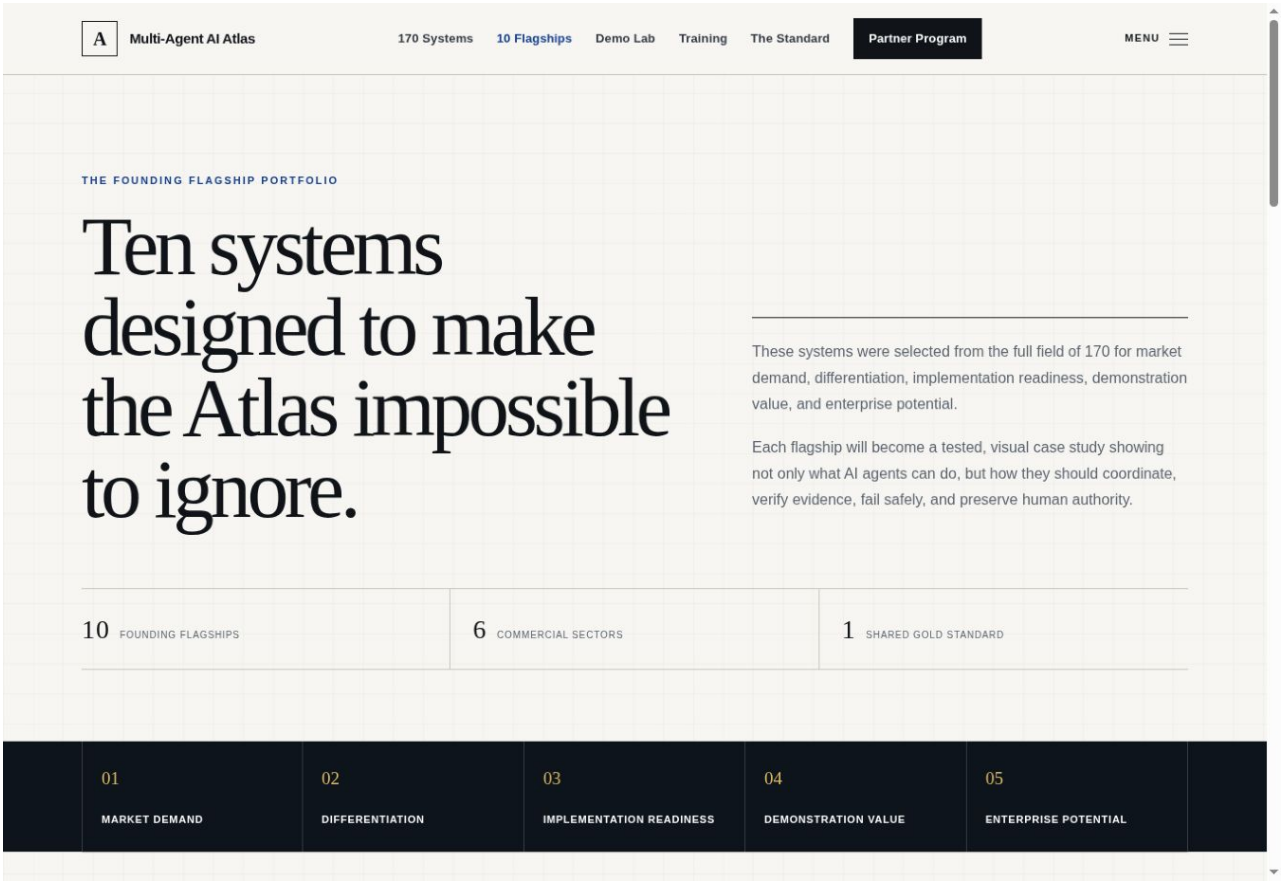


Figure 16.1. The flagship portfolio is the bridge between the 170-system catalog and the interactive laboratory.

ID	Flagship	Governance focus
F117	Digital Twin Engineer	Telemetry, diagnosis, simulation, engineer gate
F118	Factory Automation	Fault isolation, interlocks, controlled recovery
F36	Multi-Agent Orchestrator	Handoffs, state, conflict, and authority
F59	Caregiver Support	Observation, burden, safety, clinical escalation
F37	LLM Evaluator	Normal, adversarial, tool-failure, release gate
F101	Contract Review	Clauses, obligations, risk, counsel gate
F52	Clinical Trial Manager	Safety, protocol, data, trial leader

ID	Flagship	Governance focus
F98	Grant Writer	Funder fit, evidence, critique, PI decision
F09	AI Safety	Injection, privilege, stale evidence, unsafe tools
F116	Lean Manufacturing	Flow, waste, experiments, operations gate

Detailed demonstration matrix

ID	Case	Agent team	Authority
F118	Conveyor fault	PLC Observer; Fault Isolator; Safety Interlock; Recovery Planner; Commissioning Gate	Controls engineer
F36	Enterprise launch	Intent Router; Planning Lead; Specialist Pool; Conflict Resolver; Authority Gate	Accountable owner
F59	Evening behavior change	Observation Structurer; Pattern Analyst; Burden Assessor; Resource Navigator; Clinical Gate	Caregiver or clinician
F37	Release candidate	Dataset Curator; Task Evaluator; Red Team; Failure Analyst; Release Gatekeeper	Release owner
F101	Vendor agreement	Clause Extractor; Obligation Mapper; Risk Reviewer; Conflict Checker; Counsel Gate	Qualified legal reviewer
F52	Missed assessment window	Site Intake; Safety Reviewer; Protocol Analyst; Data Quality Lead; Trial Leader Gate	Trial leader
F98	Caregiver technology proposal	Funder Fit; Evidence Mapper; Narrative Architect; Reviewer Simulator; Submission Gate	Principal investigator
F09	Tool-enabled support agent	Threat Modeler; Policy Sentinel; Adversarial Tester; Evidence Auditor; Safety Gate	Safety owner
F116	Assembly line	Value Stream Mapper; Waste Analyst; Flow Modeler; Experiment Designer; Operations Gate	Operations leader

CHAPTER 17

Designing the Interactive Demo Lab

The Demo Lab converts architecture into an experience readers can question and manipulate.

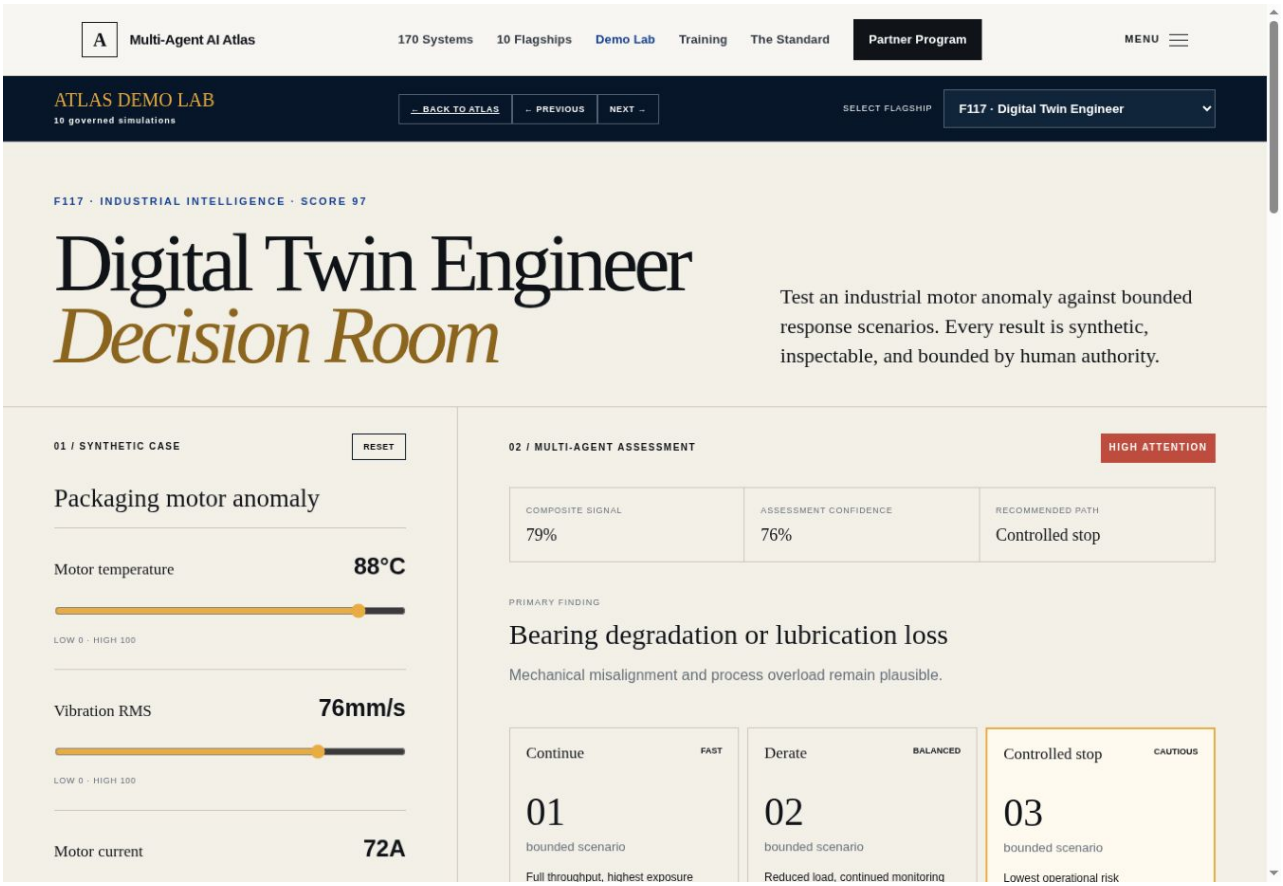


Figure 17.1. The native Demo Lab on the Atlas website.

Interaction contract

Step	Visitor experience
Select	Choose one of ten flagships.
Adjust	Change three bounded synthetic inputs.
Assess	Calculate severity, confidence, finding, and alternatives.
Compare	Review three bounded response scenarios.
Trace	Inspect five specialized agent handoffs.

Step	Visitor experience
Authorize	Record human review without issuing real action.
Navigate	Move across demos and return to the portfolio.

Deterministic browser assessment

```
function assess() {
  const severity = Math.round(
    values.reduce((sum, value) => sum + value, 0) / 3
  );
  const choice = severity >= 76 ? 2 : severity >= 52 ? 1 : 0;
  return {
    severity,
    choice,
    band: choice === 2
      ? "HIGH ATTENTION"
      : choice === 1
        ? "REVIEW NEEDED"
        : "WITHIN BOUNDS",
    confidence: Math.min(
      98,
      66 + Math.round(Math.abs(severity - 50) * 0.35)
    )
  };
}
```

Human gate rendering

```
const gateStatus = approval.checked
  ? "AUTHORIZED FOR HUMAN PROCEDURE - NO ACTION ISSUED"
  : "AWAITING " + demo.authority.toUpperCase() + " APPROVAL";

document.getElementById("gate-status").textContent = gateStatus;
document.getElementById("approval-copy").textContent =
  "Confirm review of " + demo.options[result.choice].toLowerCase();
```

Why no real data is collected

The simulations run in the browser with synthetic inputs. They do not request operational, health, legal, research, or company data. This supports learning while preserving the boundary between an architecture demonstration and a deployed domain system.

That choice also made the lab easier to share. A student, engineer, caregiver, executive, or researcher can explore the same decision room without wondering what information is being uploaded. The interaction teaches the architecture and nothing more is required from the visitor.

CHAPTER 18

Building the Public Architecture Atlas

A repository becomes a reference platform when readers can find, compare, cite, and share its systems.

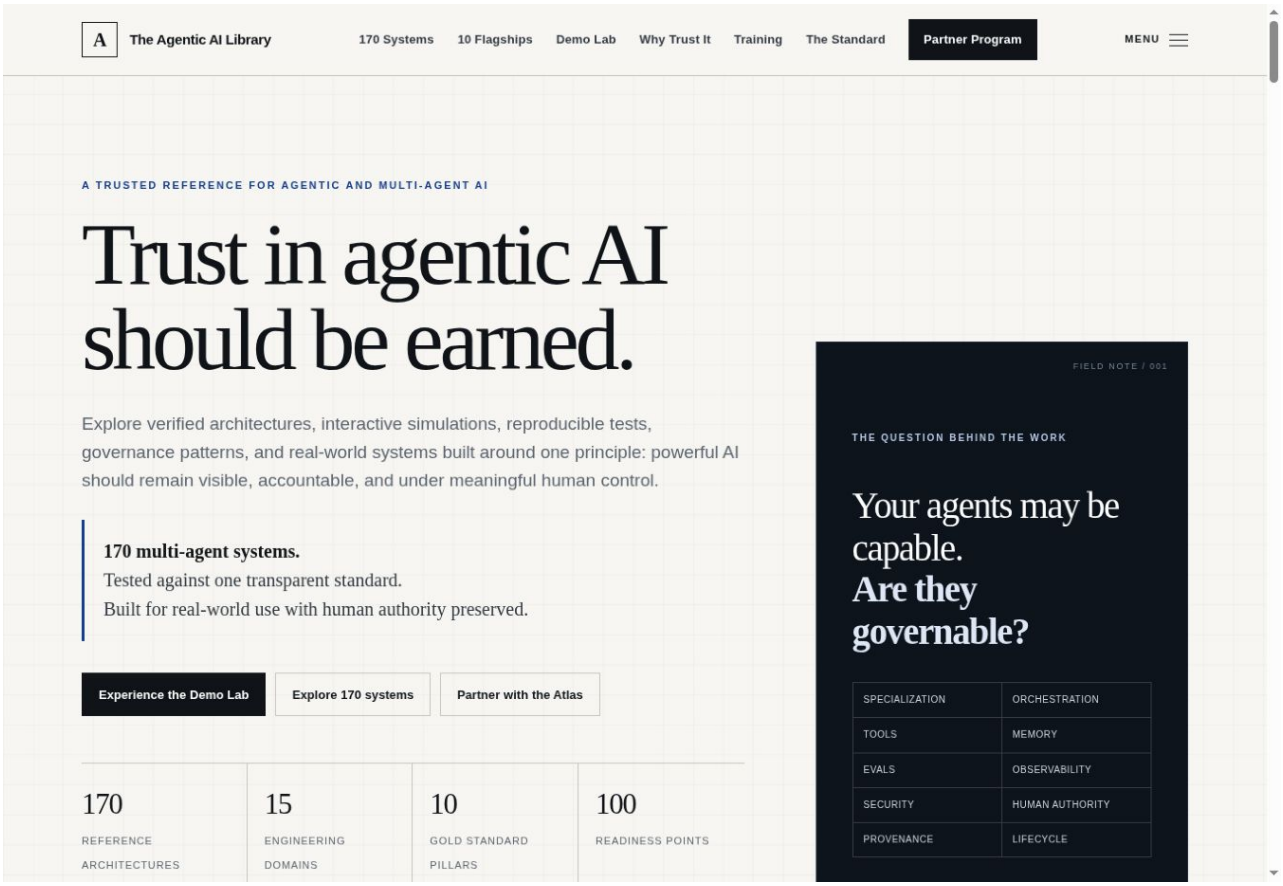


Figure 18.1. The public Atlas homepage presents the trust framework and primary discovery paths.

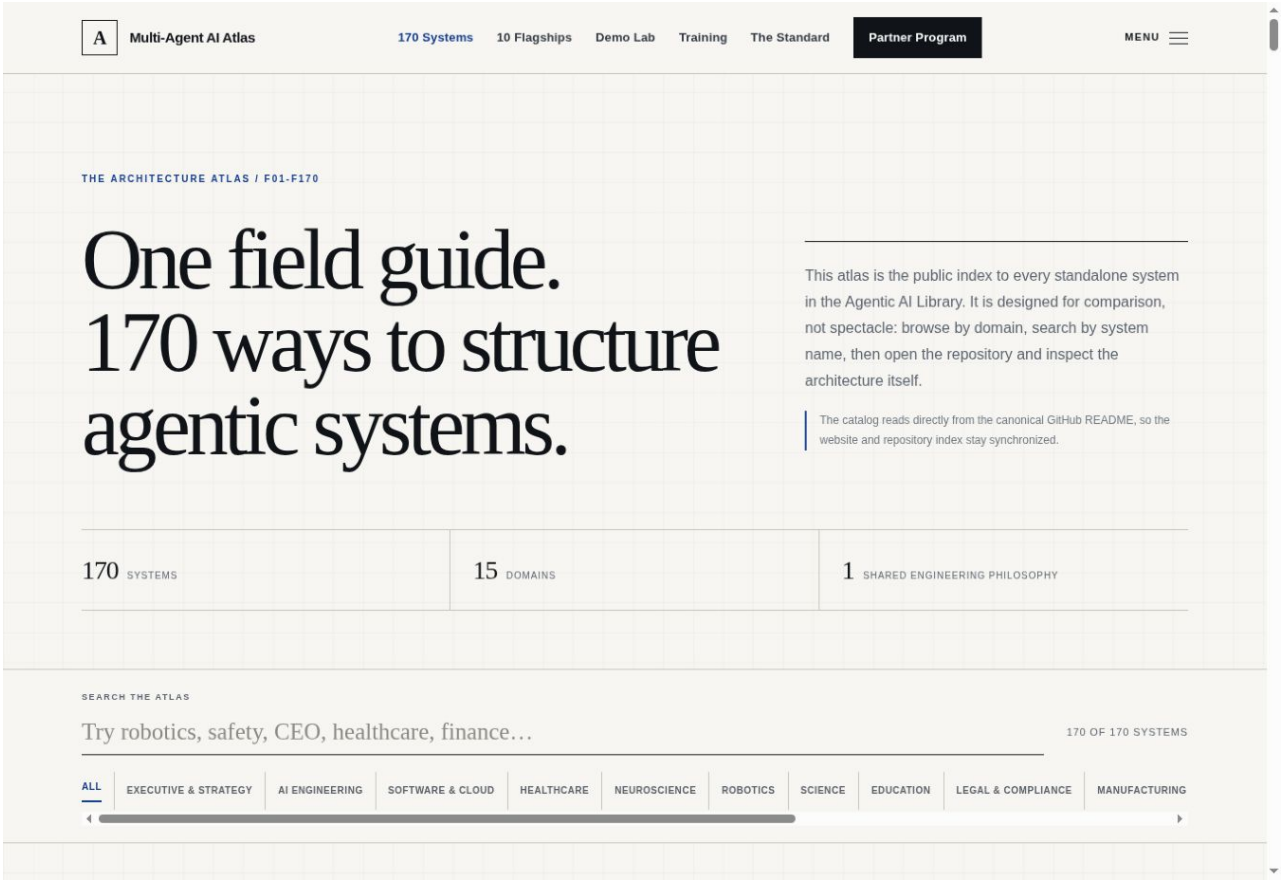


Figure 18.2. Search, filters, unique purpose statements, and direct deep links support discovery across 170 systems.

Website information architecture

HOME AND TRUST FRAMEWORK

170-SYSTEM ARCHITECTURE ATLAS

10 FLAGSHIP PORTFOLIO

INTERACTIVE DEMO LAB

TRAINING AND READINESS

FOUNDING PARTNER PROGRAM

Design principles

- Make the architecture catalog the primary discovery path.
- Give every F-system a distinct purpose statement.

- Preserve stable deep links for sharing and citation.
- Separate learning, demonstration, training, and enterprise conversion routes.
- Use founder-led technical language instead of generic AI marketing.
- State limitations and maturity honestly.

Trust framework

The site presents repository evidence, architecture structure, evaluation, safety boundaries, maturity definitions, and human authority as one coherent framework. The reader can move from a claim to the associated system, demonstration, or documentation.

CHAPTER 19

Mobile, Accessibility, and Trustworthy UX

A technical reference must remain understandable on a phone, with visible navigation and controls.

Mobile requirements implemented

- A visible MENU control on every main route.
- Direct access to Home, 170 Systems, 10 Flagships, Demo Lab, Evidence, Training, Founder, Gold Standard, Readiness Assessment, and Partner Program.
- Touch-friendly sliders, buttons, selectors, and cards.
- Responsive grids that stack without hiding content.
- Readable contrast, clear headings, and semantic labels.
- No external sign-in wall between the Atlas and its Demo Lab.

Accessibility checklist

Area	Requirement
Navigation	Keyboard reachability, visible focus, meaningful link labels.
Forms	Labels, validation messages, privacy guidance, consent.
Interactive lab	ARIA labels, state text, no color-only meaning.
Images	Alt text and captions that explain the engineering purpose.
Motion	Avoid essential information that depends on animation.
Content	Plain language around risk, authority, and limitations.

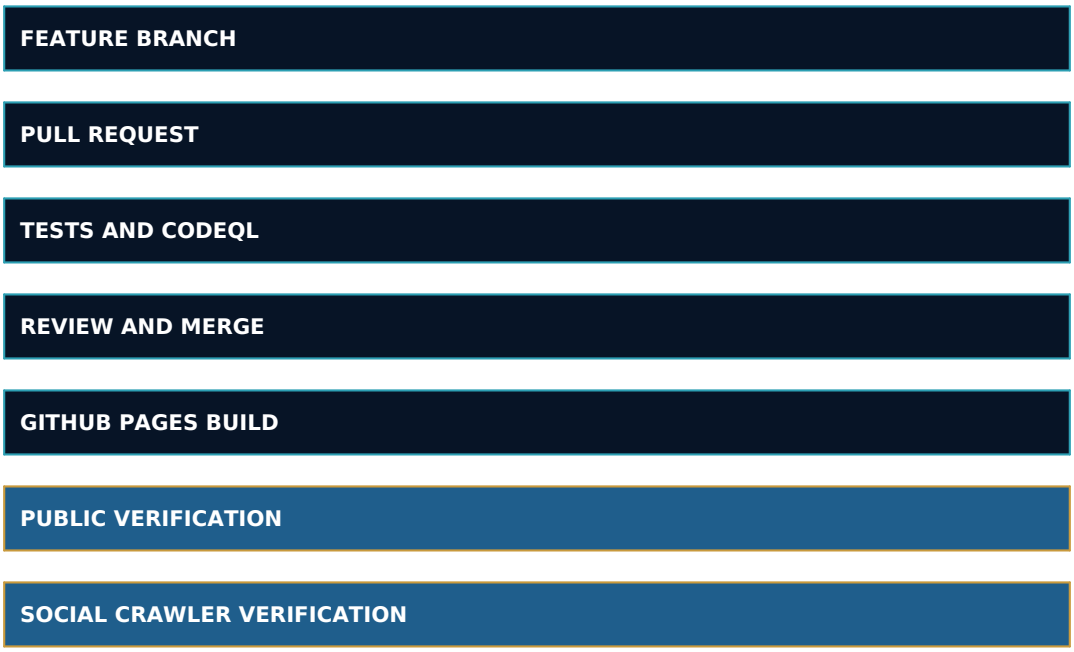
UX review
Open the site on a phone. Ask a new reader to find the Demo Lab, select F09, explain the recommendation, and identify the human authority. Record every hesitation as an information-architecture defect.

CHAPTER 20

GitHub Pages and Social Distribution

Source control, deployment, metadata, and social cards form one publishing system.

Deployment path



Open Graph and X card metadata

```
<meta property="og:title" content="Atlas Demo Lab">
<meta property="og:description" content="10 governed multi-agent simulations. Every
handoff visible. Every protected action human-controlled.">
<meta property="og:type" content="website">
<meta property="og:url"
content="https://mahsakeikha.github.io/agentic_ai_library/demo-lab.html">
<meta property="og:image"
content="https://mahsakeikha.github.io/agentic_ai_library/atlas-demo-lab-social-preview.png">
<meta property="og:image:width" content="1200">
<meta property="og:image:height" content="630">
<meta property="og:image:alt" content="Atlas Demo Lab with 10 governed multi-agent
simulations">
<meta name="twitter:card" content="summary_large_image">
<meta name="twitter:title" content="Atlas Demo Lab">
<meta name="twitter:description" content="10 governed multi-agent simulations. Every
handoff visible. Every protected action human-controlled.">
<meta name="twitter:image"
content="https://mahsakeikha.github.io/agentic_ai_library/atlas-demo-lab-social-preview.png">
<meta name="twitter:image:alt" content="Atlas Demo Lab with 10 governed multi-agent
simulations">
<meta name="twitter:url"
content="https://mahsakeikha.github.io/agentic_ai_library/demo-lab.html">
<meta property="og:image:secure_url"
```

```
content="https://mahsakeikha.github.io/agentic_ai_library/atlas-demo-lab-social-preview.png">
<link rel="canonical"
href="https://mahsakeikha.github.io/agentic_ai_library/demo-lab.html">
```

The social-preview failure and correction

The Demo Lab initially showed a blank link preview on X. The image asset existed, but malformed literal line-break text remained in the HTML head. Browsers tolerated the markup while the social crawler did not. The correction placed every metadata tag on a valid line, added a 1200 by 630 image, supplied image dimensions and alt text, and verified both page and image with the X crawler user agent.

> A page can look correct to a human and still fail machine distribution. Publishing verification must include the browser, crawler, metadata, asset, cache, and canonical URL.

CHAPTER 21

Enterprise Readiness Assessment

A disciplined review converts governance concerns into evidence, blockers, priorities, and accountable decisions.

The public scorecard introduces the questions that matter, but it is not the complete enterprise assessment method. A formal review depends on confidential architecture evidence, organizational authority, deployment context, security controls, evaluation results, incident history, and the consequences of failure. Those factors cannot be responsibly reduced to a generic public checklist.

What a serious review examines

Review area	Evidence considered
System boundaries	Agents, orchestration, tools, memory, data, external actions, and dependencies.
Evidence and evaluation	Acceptance criteria, held-out tests, adverse conditions, trace quality, and reproducibility.
Authority and risk	Protected actions, accountable owners, escalation, approval quality, and failure consequences.
Operational readiness	Identity, access, monitoring, incidents, rollback, change control, and organizational ownership.

Automatic blockers

- Material transactions without authorization.
- Binding legal commitments without qualified review.
- Production modification without release controls.
- Suppression of incidents, defects, safety findings, or required reporting.
- High-risk tools with broad credentials.
- Sensitive data without an approved basis.
- No accountable human owner.
- No ability to reconstruct decisions or tool actions.
- Open critical security or safety defects.
- Missing required regulatory, legal, or compliance review.

Where the public framework stops

The Atlas makes the evaluation philosophy visible. The organization-specific scoring model, interview protocol, evidence review, prioritization logic, and implementation roadmap belong inside a professional engagement. This protects confidential information and prevents a general reference from being mistaken for a production authorization.

> A public score can start a useful conversation. It cannot replace accountable review of the system an organization actually operates.

CHAPTER 22

Training and Organizational Capability

Teams need a shared language for architecture, evidence, risk, and authority before they can deploy agentic systems responsibly.

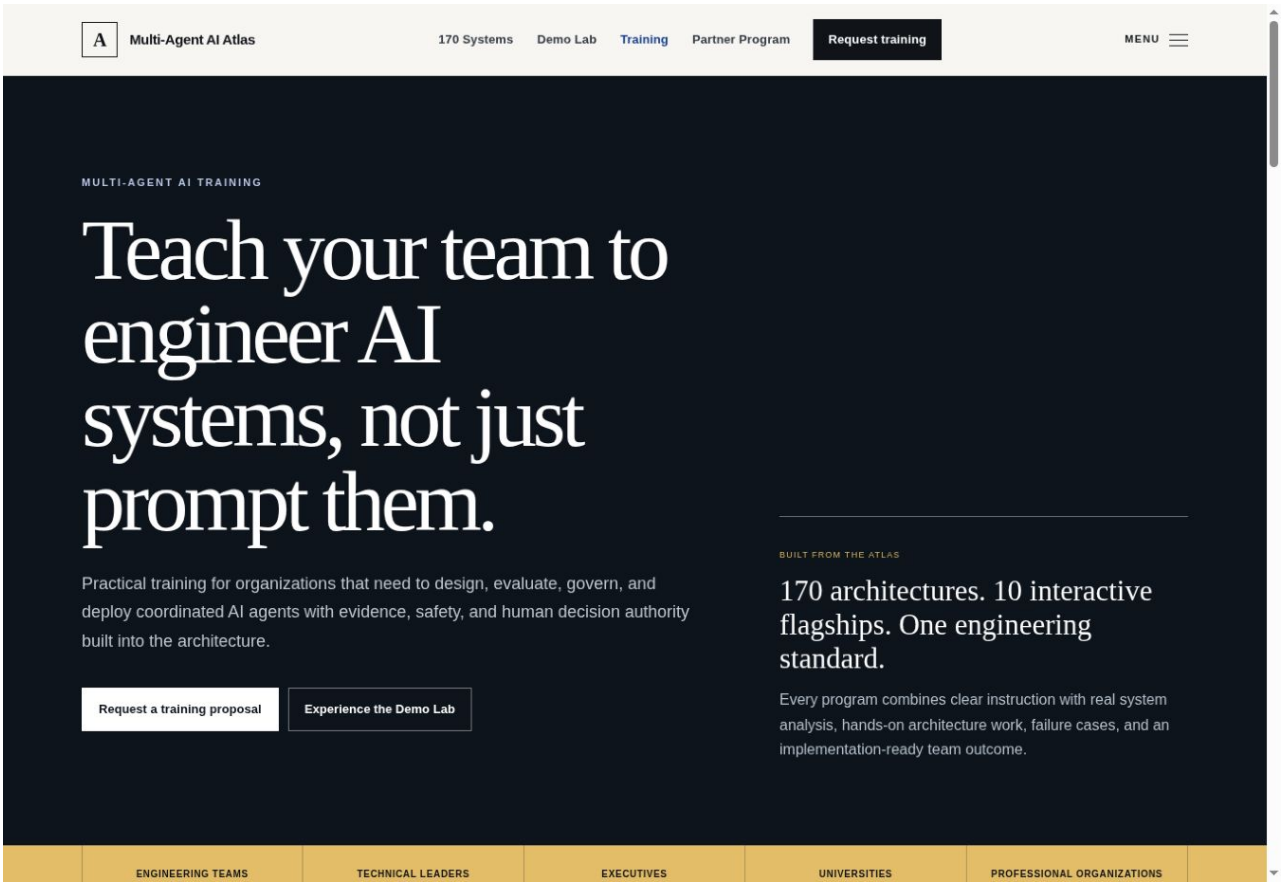


Figure 22.1. The Atlas training pathway connects the reference library to executive, engineering, enterprise, and academic education.

The public field guide establishes a common language. Professional training turns that language into decisions for a specific team, system, and risk environment. The program is shaped around participant roles, existing architecture, maturity, and the evidence required for the next responsible step.

Capability areas

Capability	Organizational outcome
Leadership alignment	Decision rights, accountable ownership, risk posture, and investment priorities.
Engineering practice	Architecture boundaries, orchestration, permissions, evidence, evaluation, and observability.
Governance practice	Protected actions, review quality, escalation, incidents, change control, and audit evidence.
Applied work	A private case exercise based on the organization's goals without publishing confidential implementation details.

Training is valuable when a team leaves with better engineering judgment and a clear next decision, not a copied template.

CHAPTER 23

Founding Partners and Commercial Growth

The public Atlas establishes credibility. Private work applies the method to consequential workflows that require evidence, discretion, and accountable implementation.

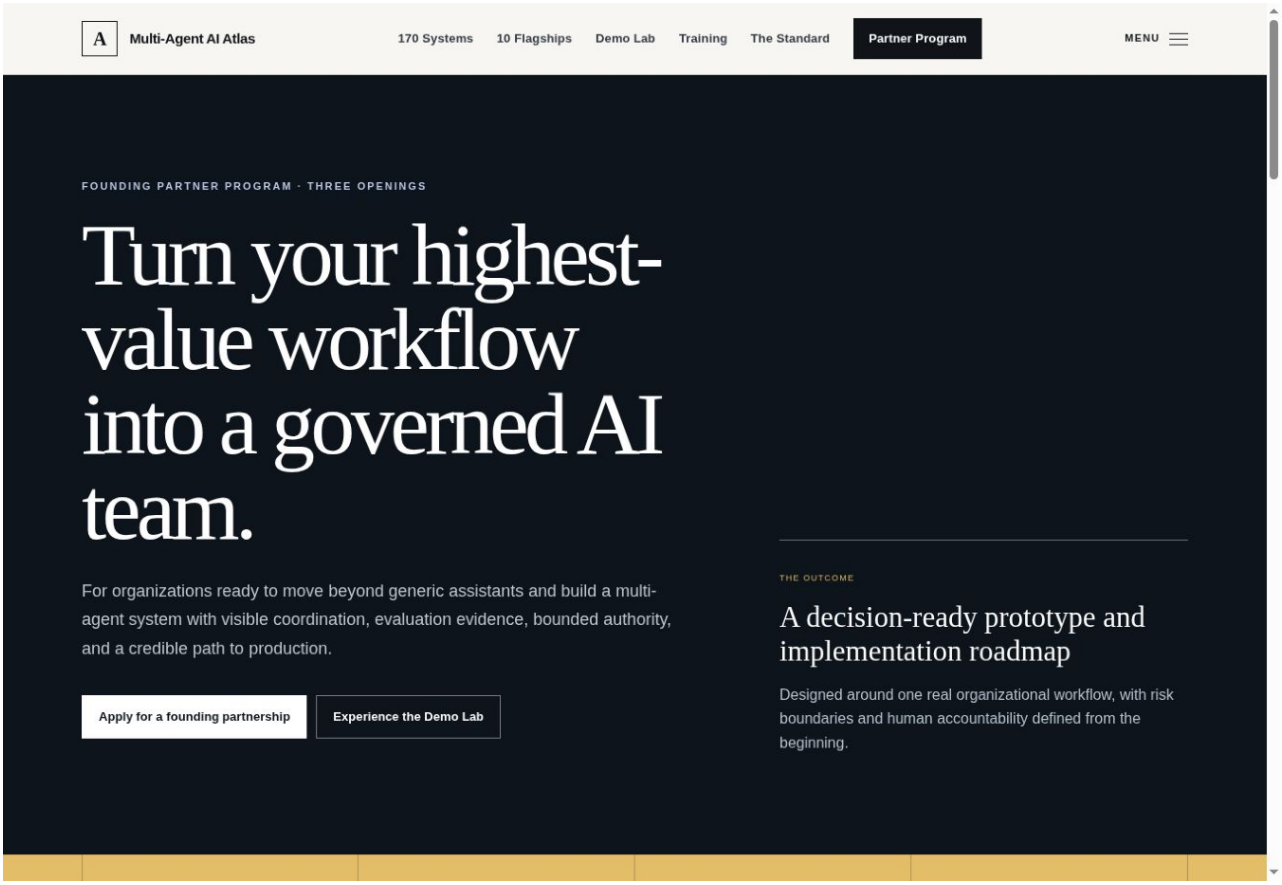


Figure 23.1. The Founding Partner Program converts one consequential workflow into a governed prototype and implementation roadmap.

The public and private boundary

The website, repositories, and Demo Lab allow a prospective partner to inspect the engineering principles before beginning a conversation. They demonstrate how I think about architecture, evidence, failure, and authority. They do not disclose a client's workflow, the full assessment protocol, delivery tools, implementation decisions, or commercial operating model.

Professional value begins where the public reference must stop. Every serious deployment has private systems, policies, integrations, ownership, constraints, and consequences. The work is not the act of copying an Atlas diagram. It is the judgment required to adapt the method, test it against real evidence, and help accountable people make defensible decisions.

What partners engage for

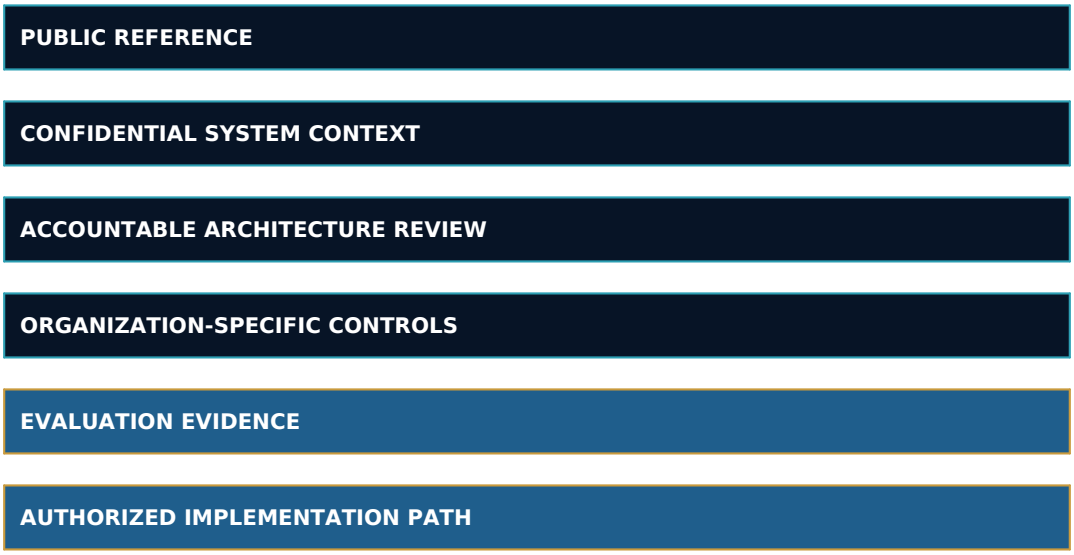
Partner need	Professional value
Independent clarity	A rigorous view of the architecture, authority, and evidence gaps that matter most.
Organization-specific design	A governed system design shaped around private workflows, controls, and accountable owners.
Evaluation evidence	Tests and review evidence that match the actual operating environment and consequences.
Capability building	Training and decision support that help internal teams sustain the engineering discipline.
Implementation direction	A prioritized path from present architecture to the next responsible level of maturity.

The open Atlas shows the quality of the thinking. The engagement applies that thinking to the system only the organization can fully describe.

CHAPTER 24

From Public Reference to Private Implementation

The Atlas provides a shared engineering foundation. Responsible implementation begins with the organization's real system, evidence, authority, and operating constraints.



Why implementation cannot be copied

Organizations use different models, frameworks, vendors, clouds, data sources, tools, policies, and approval structures. Even systems with similar goals can carry very different consequences. A production design must therefore be based on the environment in which it will operate, not on a generic diagram or a successful demonstration.

What remains private

- Detailed system and data-flow discovery.
- Organization-specific threat, authority, and failure analysis.
- Assessment scoring logic and professional judgment.
- Private implementation patterns, delivery tools, and integration decisions.
- Client evidence, evaluation results, incidents, constraints, and roadmaps.
- Commercial methods, reusable delivery assets, and future product direction.

What the public Atlas can establish

The public work establishes the principles, vocabulary, reference architectures, and visible proof needed for an informed first conversation. It allows readers to evaluate the seriousness of the method without exposing the private work required to apply it responsibly.

Organization decision
Identify one consequential workflow where your team needs clearer evidence, bounded authority, or an independent architecture review. Use the public Atlas to frame the question, then evaluate whether organization-specific support is required.

CHAPTER 25

Research, Education, and Future Work

The Atlas can become a durable reference only through review, evidence, versioning, and community use.

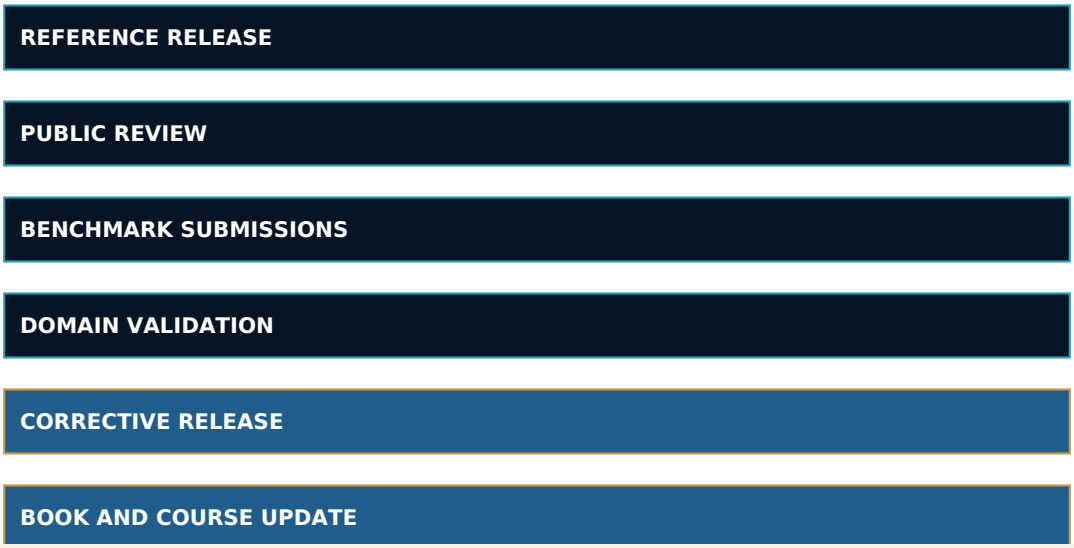
Research agenda

- Cross-domain benchmark interfaces for agent coordination and authority.
- Measures of provenance completeness and reconstructability.
- Evaluation of human review quality under alert load.
- Methods for detecting correlated multi-agent failure.
- Permission and memory risk models for long-running agents.
- Independent validation of domain-specific reference systems.
- Empirical comparison of single-agent and multi-agent decomposition.
- Governance regression testing across model and tool updates.

Educational adoption

Each F-system can become a classroom case, architecture critique, red-team assignment, evaluation lab, or capstone. The stable taxonomy makes it possible for instructors to assign different domains while students use the same engineering vocabulary.

A durable publication cycle



The Atlas should become easier to cite, easier to run, easier to compare, and more difficult to misunderstand with every release.

APPENDIX A

Complete F01-F170 Catalog

The stable taxonomy across 15 domains.

Flagship, Executive, Strategy, and Leadership

ID	System
F01	Agentic Book Writer
F02	Agentic Research Lab
F03	Agentic Biotech R&D;
F04	Agentic Tech Support
F05	Agentic Online Shop
F06	Agentic Debug Automation
F07	Agentic Software Design
F08	Agentic CEO Assistant
F09	Agentic AI Safety
F10	Agentic PhD Assistant
F11	Agentic Account Manager
F12	Agentic Robotics Governance
F13	Agentic QA Safety Manager
F14	Agentic Client Inquiry Bot
F15	Agentic Fullstack Web
F16	Agentic Engineering Professor
F17	Agentic Immigration Assistant
F18	Agentic Real Estate
F19	Agentic Psychologist Assistant
F20	Agentic Dating Advisor
F21	Agentic COO Assistant
F22	Agentic CFO Assistant
F23	Agentic Board Advisor
F24	Agentic Chief of Staff
F25	Agentic Strategy Consultant
F26	Agentic Venture Capital Analyst

ID	System
F27	Agentic M&A; Advisor
F28	Agentic Startup Accelerator
F29	Agentic Innovation Officer
F30	Agentic Corporate Governance

AI Engineering

ID	System
F31	Agentic ML Engineer
F32	Agentic MLOps Team
F33	Agentic Data Engineering
F34	Agentic Prompt Engineering
F35	Agentic RAG Engineering
F36	Agentic Multi-Agent Orchestrator
F37	Agentic LLM Evaluator
F38	Agentic AI Benchmark Suite
F39	Agentic AI Program Manager
F40	Agentic AI Infrastructure Architect

Software, Cloud, Security, and Connected Systems

ID	System
F41	Mobile App Engineering
F42	Cloud Architecture
F43	DevOps
F44	Kubernetes Operations
F45	Cybersecurity SOC
F46	Penetration Testing
F47	API Engineering
F48	Database Architecture
F49	Embedded Systems
F50	IoT Engineering

Healthcare and Digital Health

ID	System
F51	Digital Health Assistant
F52	Clinical Trial Manager
F53	Medical Device Development
F54	FDA Documentation
F55	Hospital Operations
F56	Radiology Workflow
F57	Pathology Review
F58	Nursing Assistant
F59	Caregiver Support
F60	Rehabilitation Planning

Neuroscience, Neurotechnology, and Aging

ID	System
F61	Parkinson Research
F62	Dementia Care Planning
F63	Hallucination Monitoring
F64	EEG Analysis
F65	Sleep Research
F66	Neurotechnology Design
F67	Brain Computer Interface
F68	Cognitive Assessment
F69	Aging Research
F70	Biomarker Discovery

Robotics and Autonomous Systems

ID	System
F71	Industrial Robotics
F72	Service Robotics
F73	Medical Robotics
F74	Autonomous Vehicles
F75	Drone Operations
F76	Humanoid Robot Design

ID	System
F77	Robot Safety Validation
F78	Human-Robot Interaction
F79	Swarm Robotics
F80	Robotics Ethics

Science and Frontier Engineering

ID	System
F81	Physics Research Assistant
F82	Quantum Computing
F83	Materials Science
F84	Chemistry Planning
F85	Climate Science
F86	Space Mission Design
F87	Astronomy Research
F88	Energy Systems
F89	Nuclear Engineering
F90	Scientific Literature Review

Education and Academic Systems

ID	System
F91	Professor Assistant
F92	Curriculum Builder
F93	Student Tutor
F94	Exam Generator
F95	STEM Laboratory Planner
F96	University Research Office
F97	Academic Integrity Checker
F98	Grant Writer
F99	Thesis Committee
F100	Accreditation Manager

Legal, Compliance, Privacy, and Regulatory

ID	System
F101	Contract Review
F102	Corporate Compliance
F103	Privacy Compliance
F104	Intellectual Property
F105	Patent Research
F106	Employment Compliance
F107	International Trade
F108	Export Control
F109	Healthcare Compliance
F110	Regulatory Affairs

Manufacturing, Industrial Systems, and Sustainability

ID	System
F111	Manufacturing Engineer
F112	Production Planner
F113	Quality Engineer
F114	Predictive Maintenance
F115	Supply Chain Planner
F116	Lean Manufacturing
F117	Digital Twin Engineer
F118	Factory Automation
F119	Industrial Safety
F120	Sustainability Engineer

Marketing, Brand, Growth, and Customer Intelligence

ID	System
F121	Brand Strategist
F122	Content Marketing
F123	SEO Growth
F124	Social Media Manager
F125	Lifecycle Marketing
F126	Paid Acquisition

ID	System
F127	Product Marketing
F128	PR and Communications
F129	Growth Experimentation
F130	Customer Insights

Creative, Design, Media, and Entertainment

ID	System
F131	Book Publishing
F132	Screenwriting Studio
F133	Music Production
F134	Graphic Design
F135	UX Design
F136	Interior Design
F137	Fashion Design
F138	Architecture Studio
F139	Game Design
F140	Animation Studio

Government, Public Sector, Infrastructure, and Policy

ID	System
F141	Smart City Planner
F142	Emergency Management
F143	Disaster Response
F144	Public Health Planner
F145	Transportation Planner
F146	Environmental Compliance
F147	Defense Policy Analyst
F148	Public-Sector Intelligence Analyst
F149	Election Information Reviewer
F150	Policy Analyst

Finance, Investment, Banking, and Enterprise Risk

ID	System
F151	Investment Research
F152	Portfolio Manager
F153	Quantitative Trading Research
F154	Insurance Operations
F155	Banking Assistant
F156	Tax Planning
F157	Enterprise Risk Manager
F158	Treasury Operations
F159	Credit Analyst
F160	ESG Reporting

Personal Intelligence, Career, Learning, and Productivity

ID	System
F161	Life Planner
F162	Personal Knowledge Manager
F163	Career Coach
F164	Resume Studio
F165	Public Speaking Coach
F166	Interview Coach
F167	Language Tutor
F168	Productivity Coach
F169	Travel Planner
F170	Habit Builder

APPENDIX B

Gold Standard Engineering Checklist

A practical review tool for architecture, implementation, and release.

Identity and documentation

- [ ] Stable system ID, name, domain, version, and owner
- [ ] README explains problem, architecture, agents, outputs, limits, run, test, and extension
- [ ] License, citation, changelog, contribution, and security files

Architecture

- [ ] Distinct agent responsibilities
- [ ] Explicit orchestrator and shared state
- [ ] Structured inputs and outputs
- [ ] Evidence ledger and conflict handling
- [ ] Final synthesis and human gate

Safety and security

- [ ] Threat model
- [ ] Least privilege
- [ ] Protected-action registry
- [ ] Sensitive-data boundaries
- [ ] Injection and unsafe-tool tests
- [ ] Fail-closed blockers

Evaluation

- [ ] Happy path
- [ ] Missing evidence
- [ ] Conflicting evidence
- [ ] Stale data
- [ ] Role violation
- [ ] Wrong approval

- [ ] Tool failure
- [ ] Adversarial request
- [ ] Regression suite
- [ ] Fully approved scenario

Operations

- [ ] Offline run
- [ ] Passing tests and CI
- [ ] CodeQL
- [ ] Release evidence
- [ ] Observability
- [ ] Incident and rollback expectations
- [ ] Honest maturity statement

APPENDIX C

Multi-Agent System Design Canvas

Use these prompts before writing agents or selecting a framework.

Canvas area	Design question
Decision	What decision or controlled output is the system supporting?
Value	Why is multi-agent decomposition better than a simpler workflow?
Agents	Which responsibilities require distinct evidence, tools, or failures?
Evidence	Which sources are allowed, missing, stale, conflicting, or prohibited?
State	What is shared, versioned, retained, corrected, and deleted?
Tools	Which tools are read, draft, reversible write, communication, or consequential?
Authority	Who can approve which action, with what evidence and scope?
Evaluation	Which normal, failure, adversarial, and governance cases must pass?
Observability	Can a reviewer reconstruct every material handoff and action?
Lifecycle	Who owns release, monitoring, incidents, rollback, and retirement?

APPENDIX D

Annotated Code Reading Guide

The most important implementation paths to study in the open repository.

Path	What to study
F117 digital_twin.py	Input contract, state, diagnosis, simulation, governance, trace.
F117 test_digital_twin.py	Trip state, valid approval, wrong role, missing sensor, stale data, reproducibility.
docs/demo-lab.js	Ten simulation models, deterministic assessment, rendering, navigation, approval state.
docs/atlas.js	Catalog parsing, search, filters, deep links, and domain state.
docs/conversion.js	Inquiry validation and routing.
assessment_engine	Enterprise scoring, blockers, reports, and remediation priorities.
catalog/core	Reusable deterministic patterns C01-C10.
.github/workflows	Tests, CodeQL, and Pages deployment.

Recommended reading order

APPENDIX E

Website Route Map

Public paths connecting discovery, learning, trust, and enterprise action.

Destination	URL
Home	https://mahsakeikha.github.io/agentic_ai_library/
170 Systems	https://mahsakeikha.github.io/agentic_ai_library/atlas.html
10 Flagships	https://mahsakeikha.github.io/agentic_ai_library/flagships.html
Demo Lab	https://mahsakeikha.github.io/agentic_ai_library/demo-lab.html
Evidence	https://mahsakeikha.github.io/agentic_ai_library/evidence.html
Founder	https://mahsakeikha.github.io/agentic_ai_library/about.html
Training	https://mahsakeikha.github.io/agentic_ai_library/training.html
Partner Program	https://mahsakeikha.github.io/agentic_ai_library/founding-partners.html
Repository	https://github.com/MahsaKeikha/agentic_ai_library

APPENDIX F

Publication, Course, and Business Strategy

How the field guide, repositories, and website reinforce one another.

Asset	Strategic role
Field guide	Explains the public engineering principles, architecture vocabulary, and evidence standard.
Repositories	Provide inspectable reference code, tests, issues, versions, and reproducibility.
Website	Provides discovery, demonstrations, trust evidence, and a professional inquiry path.
Training	Builds shared judgment around architecture, evaluation, risk, and human authority.
Assessment	Applies professional review to the organization-specific system and evidence.
Partner work	Provides private design, evaluation, capability building, and implementation direction.

Attention without sacrificing credibility

- Publish enough evidence to establish technical authority and honest limitations.
- Invite experts to challenge the evidence and authority model.
- Show test results and corrections without disclosing private client work.
- Use public demonstrations to clarify the problem, not to publish the delivery playbook.
- Use the field guide to connect principles with visible proof.
- Direct organization-specific needs toward training, assessment, and private engagement.
- Never claim safety, validation, or production readiness without evidence.

APPENDIX G

Glossary

A shared language for multi-agent AI engineering.

Term	Definition
Agent	A bounded component that performs a defined analytical or operational responsibility.
Orchestrator	The component controlling sequence, routing, state, retry, termination, and synthesis.
Shared state	The versioned record used across agents during a run.
Evidence ledger	A structure separating sources, assumptions, derivations, gaps, conflicts, and conclusions.
Protected action	An operation agents cannot execute without explicit authorized control.
Human gate	A pre-action review point with named authority, evidence, scope, and choice.
Fail closed	Refuse completion or action when required evidence, control, or authorization is missing.
Held-out evaluation	A test scenario not used to construct the system's normal behavior.
Provenance	The lineage of evidence, calculations, artifacts, decisions, and approvals.
Observability	The ability to inspect system behavior during and after execution.
Reference architecture	A reusable engineering model, not an unrestricted production claim.
Gold Standard	The Atlas maturity designation for complete governed reference architectures.

APPENDIX H

A Note to Builders

The final principle behind the Atlas.

> I built the Multi-Agent AI Atlas because I believe the future of agentic AI should not be about removing people from the loop. It should be about creating better loops between human judgment and machine intelligence.

That requires more than powerful models. It requires visible decisions, bounded tools, traceable evidence, honest uncertainty, testable failure behavior, and clear human authority. These ideas must appear in code, tests, interfaces, organizational roles, and release evidence. Otherwise they remain promises.

The Atlas is my contribution to making those expectations practical enough to build, teach, challenge, and improve. The repositories keep the work open. The Demo Lab makes it visible. The Gold Standard makes it reviewable. This book connects them into one engineering method.

> Build systems that can coordinate without hiding their decisions, learn without losing provenance, use tools without escaping authority, and become more accountable as they become more capable.

APPENDIX I

Ten Flagship Engineering Blueprints

A one-page design and evaluation brief for every interactive decision room.

Each blueprint uses the same review frame so readers can compare systems across domains without erasing their differences. The page identifies synthetic inputs, specialized agents, the leading finding, preserved alternatives, bounded options, accountable human authority, protected boundaries, and an initial evaluation pack.

How to use the blueprints

- Study one blueprint as a complete architecture case.
- Compare the authority and evidence model across two domains.
- Replace the synthetic inputs with an approved organizational case contract.
- Extend the evaluation pack before adding tools or model adapters.
- Keep the protected boundary unchanged until accountable owners approve a new control model.

Blueprint index

ID	System	Decision owner
F117	Digital Twin Engineer	Authorized engineer
F118	Factory Automation	Controls engineer
F36	Multi-Agent Orchestrator	Accountable owner
F59	Caregiver Support	Caregiver or clinician
F37	LLM Evaluator	Release owner
F101	Contract Review	Qualified legal reviewer
F52	Clinical Trial Manager	Trial leader
F98	Grant Writer	Principal investigator
F09	AI Safety	Safety owner
F116	Lean Manufacturing	Operations leader

F117 FLAGSHIP BLUEPRINT

Digital Twin Engineer

Packaging motor anomaly

Design element	Blueprint
Synthetic inputs	Temperature, vibration, motor current
Specialized agents	Telemetry Interface; State Estimator; Diagnosis; Simulation; Deployment Gatekeeper
Leading finding	Bearing degradation or lubrication loss
Preserved alternatives	Mechanical misalignment; process overload
Bounded options	Continue; derate; controlled stop
Human authority	Authorized engineer
Protected boundary	No equipment command or setpoint change.

Evaluation pack

- Missing sensor
- stale telemetry
- trip threshold
- wrong approval role
- reproducibility

Acceptance rule

The system may produce a traceable decision-support package only when inputs satisfy the case contract, evidence gaps remain visible, and no protected action is issued. Any missing mandatory review, unsafe request, or invalid approval produces an explicit blocked state.

Extension challenge
Add one domain-specific agent, one new evidence class, one adversarial scenario, and one measurable acceptance criterion without widening the system's authority.

F118 FLAGSHIP BLUEPRINT

Factory Automation

Cell 4 conveyor fault

Design element	Blueprint
Synthetic inputs	Jam probability, safety-zone activity, rejected units
Specialized agents	PLC Observer; Fault Isolator; Safety Interlock; Recovery Planner; Commissioning Gate
Leading finding	Drive-train obstruction near the transfer point
Preserved alternatives	Sensor drift; downstream accumulation
Bounded options	Keep running; isolate cell; safe recovery
Human authority	Controls engineer
Protected boundary	No PLC write, bypass, or restart command.

Evaluation pack

- Interlock active
- incomplete fault evidence
- unsafe restart
- stale PLC state
- wrong role

Acceptance rule

The system may produce a traceable decision-support package only when inputs satisfy the case contract, evidence gaps remain visible, and no protected action is issued. Any missing mandatory review, unsafe request, or invalid approval produces an explicit blocked state.

Extension challenge
Add one domain-specific agent, one new evidence class, one adversarial scenario, and one measurable acceptance criterion without widening the system's authority.

F36 FLAGSHIP BLUEPRINT

Multi-Agent Orchestrator

Enterprise launch request

Design element	Blueprint
Synthetic inputs	Task complexity, dependency failure, policy sensitivity
Specialized agents	Intent Router; Planning Lead; Specialist Pool; Conflict Resolver; Authority Gate
Leading finding	High coordination load with a protected final action
Preserved alternatives	Smaller plan; additional evidence; sequential execution
Bounded options	Single agent; parallel team; governed sequence
Human authority	Accountable owner
Protected boundary	No protected external action from the orchestrator.

Evaluation pack

- Agent timeout
- conflicting specialists
- tool denial
- lost state
- unauthorized final action

Acceptance rule

The system may produce a traceable decision-support package only when inputs satisfy the case contract, evidence gaps remain visible, and no protected action is issued. Any missing mandatory review, unsafe request, or invalid approval produces an explicit blocked state.

Extension challenge
Add one domain-specific agent, one new evidence class, one adversarial scenario, and one measurable acceptance criterion without widening the system's authority.

F59 FLAGSHIP BLUEPRINT

Caregiver Support

Evening behavior change

Design element	Blueprint
Synthetic inputs	Behavior change, caregiver strain, immediate safety concern
Specialized agents	Observation Structurer; Pattern Analyst; Burden Assessor; Resource Navigator; Clinical Escalation Gate
Leading finding	Meaningful change requiring timely clinical review
Preserved alternatives	Routine disruption; discomfort; medication factors
Bounded options	Routine support; call care team; urgent escalation
Human authority	Caregiver or clinician
Protected boundary	No diagnosis, medication change, or emergency dispatch.

Evaluation pack

- Immediate danger
- missing medication context
- caregiver overload
- conflicting observations
- privacy

Acceptance rule

The system may produce a traceable decision-support package only when inputs satisfy the case contract, evidence gaps remain visible, and no protected action is issued. Any missing mandatory review, unsafe request, or invalid approval produces an explicit blocked state.

Extension challenge
Add one domain-specific agent, one new evidence class, one adversarial scenario, and one measurable acceptance criterion without widening the system's authority.

F37 FLAGSHIP BLUEPRINT

LLM Evaluator

Release candidate evaluation

Design element	Blueprint
Synthetic inputs	Task accuracy, adversarial resistance, tool reliability
Specialized agents	Dataset Curator; Task Evaluator; Red Team; Failure Analyst; Release Gatekeeper
Leading finding	Strong nominal performance with a material robustness gap
Preserved alternatives	Sampling variance; incomplete tool coverage
Bounded options	Release now; limited pilot; block release
Human authority	Release owner
Protected boundary	No release promotion without accountable ownership.

Evaluation pack

- Prompt injection
- tool failure
- distribution shift
- regression
- evaluator disagreement

Acceptance rule

The system may produce a traceable decision-support package only when inputs satisfy the case contract, evidence gaps remain visible, and no protected action is issued. Any missing mandatory review, unsafe request, or invalid approval produces an explicit blocked state.

Extension challenge
Add one domain-specific agent, one new evidence class, one adversarial scenario, and one measurable acceptance criterion without widening the system's authority.

F101 FLAGSHIP BLUEPRINT

Contract Review

Synthetic vendor agreement

Design element	Blueprint
Synthetic inputs	Liability exposure, missing protections, term ambiguity
Specialized agents	Clause Extractor; Obligation Mapper; Risk Reviewer; Conflict Checker; Counsel Gate
Leading finding	Unbalanced liability and ambiguous data obligations
Preserved alternatives	Commercial context may justify selected terms
Bounded options	Accept; redline; escalate
Human authority	Qualified legal reviewer
Protected boundary	No signature, acceptance, or replacement of legal counsel.

Evaluation pack

- Missing clause
- conflicting terms
- unsupported legal claim
- jurisdiction gap
- privilege boundary

Acceptance rule

The system may produce a traceable decision-support package only when inputs satisfy the case contract, evidence gaps remain visible, and no protected action is issued. Any missing mandatory review, unsafe request, or invalid approval produces an explicit blocked state.

Extension challenge
Add one domain-specific agent, one new evidence class, one adversarial scenario, and one measurable acceptance criterion without widening the system's authority.

F52 FLAGSHIP BLUEPRINT

Clinical Trial Manager

Missed assessment window

Design element	Blueprint
Synthetic inputs	Participant impact, protocol deviation, data-integrity risk
Specialized agents	Site Intake; Safety Reviewer; Protocol Analyst; Data Quality Lead; Trial Leader Gate
Leading finding	Reportable deviation with recoverable data risk
Preserved alternatives	Protocol windows and investigator judgment may change classification
Bounded options	Document only; corrective action; formal escalation
Human authority	Accountable trial leader
Protected boundary	No clinical, regulatory, sponsor, or IRB decision.

Evaluation pack

- Safety signal
- missing protocol
- delayed report
- conflicting site data
- privacy exposure

Acceptance rule

The system may produce a traceable decision-support package only when inputs satisfy the case contract, evidence gaps remain visible, and no protected action is issued. Any missing mandatory review, unsafe request, or invalid approval produces an explicit blocked state.

Extension challenge
Add one domain-specific agent, one new evidence class, one adversarial scenario, and one measurable acceptance criterion without widening the system's authority.

F98 FLAGSHIP BLUEPRINT

Grant Writer

Caregiver technology proposal

Design element	Blueprint
Synthetic inputs	Funder alignment, evidence coverage, feasibility strength
Specialized agents	Funder Fit Analyst; Evidence Mapper; Narrative Architect; Reviewer Simulator; Submission Gate
Leading finding	Compelling fit with an evidence gap in validation strategy
Preserved alternatives	Narrower aim; stronger preliminary data
Bounded options	Submit now; revise aims; evidence sprint
Human authority	Principal investigator
Protected boundary	No fabricated evidence, certification, or submission.

Evaluation pack

- Fabricated citation
- unsupported impact
- budget mismatch
- eligibility failure
- stale solicitation

Acceptance rule

The system may produce a traceable decision-support package only when inputs satisfy the case contract, evidence gaps remain visible, and no protected action is issued. Any missing mandatory review, unsafe request, or invalid approval produces an explicit blocked state.

Extension challenge
Add one domain-specific agent, one new evidence class, one adversarial scenario, and one measurable acceptance criterion without widening the system's authority.

F09 FLAGSHIP BLUEPRINT

AI Safety

Tool-enabled support agent

Design element	Blueprint
Synthetic inputs	Injection pressure, requested privilege, evidence staleness
Specialized agents	Threat Modeler; Policy Sentinel; Adversarial Tester; Evidence Auditor; Safety Release Gate
Leading finding	Unsafe privilege path under adversarial instruction
Preserved alternatives	Tool scoping and fresh evidence can reduce risk
Bounded options	Ship; sandbox pilot; block and fix
Human authority	Safety owner
Protected boundary	No release when a critical unsafe path remains.

Evaluation pack

- Injection
- privilege escalation
- stale evidence
- unsafe tool
- audit bypass

Acceptance rule

The system may produce a traceable decision-support package only when inputs satisfy the case contract, evidence gaps remain visible, and no protected action is issued. Any missing mandatory review, unsafe request, or invalid approval produces an explicit blocked state.

Extension challenge
Add one domain-specific agent, one new evidence class, one adversarial scenario, and one measurable acceptance criterion without widening the system's authority.

F116 FLAGSHIP BLUEPRINT

Lean Manufacturing

Synthetic assembly line

Design element	Blueprint
Synthetic inputs	Queue growth, changeover loss, defect burden
Specialized agents	Value Stream Mapper; Waste Analyst; Flow Modeler; Experiment Designer; Operations Gate
Leading finding	Constraint concentrated at changeover and downstream queue
Preserved alternatives	Demand mix; measurement window; upstream variability
Bounded options	Add labor; reduce changeover; rebalance flow
Human authority	Operations leader
Protected boundary	No production or staffing change without accountable review.

Evaluation pack

- Unsafe experiment
- weak baseline
- quality hold
- labor assumption
- misleading local optimum

Acceptance rule

The system may produce a traceable decision-support package only when inputs satisfy the case contract, evidence gaps remain visible, and no protected action is issued. Any missing mandatory review, unsafe request, or invalid approval produces an explicit blocked state.

Extension challenge
Add one domain-specific agent, one new evidence class, one adversarial scenario, and one measurable acceptance criterion without widening the system's authority.

APPENDIX J

Cross-Domain Governance Pattern Atlas

How the common architecture changes when evidence and protected actions change by domain.

Domain	Evidence	Protected actions	Characteristic failures
Executive and strategy	Plans, metrics, board materials	Commitments, public statements, personnel decisions	False certainty; missing stakeholder; private information
AI engineering	Datasets, traces, test results, policies	Release, deployment, credential expansion	Benchmark leakage; unsafe tool; hidden regression
Software and security	Code, logs, vulnerabilities, change records	Production change, exploit action, access change	Prompt injection; secret exposure; duplicate execution
Healthcare	Clinical records, observations, protocols, evidence	Diagnosis, treatment, disclosure, emergency action	Missing context; privacy; scope beyond qualification
Neuroscience and aging	Signals, assessments, longitudinal observations	Clinical interpretation or care decision	Artifact, bias, caregiver burden, sensitive inference
Robotics	Sensors, maps, control state, safety rules	Physical motion, force, route, override	Stale perception; unsafe state; authority ambiguity
Science	Measurements, literature, models, uncertainty	Claim publication, hazardous experiment	Fabricated source; irreproducibility; model overreach
Education	Curriculum, submissions, rubrics, accommodations	High-stakes grade, discipline, credential decision	Bias; integrity error; inaccessible design
Legal and compliance	Contracts, regulations, obligations, jurisdiction	Signature, filing, waiver, binding advice	Outdated law; missing jurisdiction; privilege exposure
Manufacturing	Telemetry, quality data, work instructions	Setpoint, restart, release, safety override	Stale sensor; local optimum; quality escape
Marketing and growth	Research, brand rules, campaign data	Publish, spend, target, contact	Unsupported claim; manipulation; privacy misuse
Creative systems	Briefs, rights, references, provenance	Publication, licensing, rights assertion	Copyright risk; lost attribution; style imitation
Public sector	Policy, eligibility, public records, emergency data	Rights, eligibility, enforcement, official message	Due-process failure; bias; disinformation
Finance and risk	Market data, ledgers, models, policies	Trade, transfer, credit, filing	Model risk; stale price; unauthorized transaction
Personal productivity	User goals, calendar, notes, preferences	External communication, purchase, sensitive memory	Overreach; privacy; mistaken identity

Pattern selection rule

Begin with the common architecture, then strengthen the controls where the domain introduces physical, clinical, legal, financial, privacy, rights, or public-information consequences. Do not weaken the shared evidence and authority contract to fit a framework limitation.

Cross-domain comparison questions

- What counts as admissible evidence in this domain?
- Which output can remain decision support, and which becomes a consequential action?
- Who is qualified and authorized to review the recommendation?
- Which failures can harm a person, organization, physical asset, right, or public record?
- What must be logged to reconstruct the decision later?
- Which data may not enter memory or leave the approved boundary?
- What organization-specific validation remains before production?

Control-strength rule

The strongest required control should be determined by consequence, not by how confident the model appears. A low-confidence output may be harmless in a private brainstorming tool, while a high-confidence output may remain prohibited in a clinical, financial, legal, or physical-control workflow.

APPENDIX K

Applied Engineering Labs

Ten assignments for university, professional, and enterprise training.

LAB 1

Decomposition

Convert a monolithic assistant into four bounded agents. Prove that each role has distinct evidence and failure modes.

Required deliverable

Responsibility matrix and architecture diagram.

Review criteria

- Architecture is explicit and internally consistent.
- Evidence and uncertainty remain visible.
- Agent responsibilities and tools are bounded.
- Failure behavior is testable and fail closed where required.
- Human authority is named, timely, informed, and attributable.
- Claims match the evidence and maturity level.

Reflection questions

- What new failure surface did your design create?
- Which control is deterministic rather than prompt-based?
- What would invalidate the system's recommendation?
- What must remain under human authority in production?

LAB 2

State and provenance

Design a shared state that separates supplied facts, assumptions, derived findings, conflicts, and gaps.

Required deliverable

Versioned state schema and example run.

Review criteria

- Architecture is explicit and internally consistent.
- Evidence and uncertainty remain visible.
- Agent responsibilities and tools are bounded.
- Failure behavior is testable and fail closed where required.
- Human authority is named, timely, informed, and attributable.
- Claims match the evidence and maturity level.

Reflection questions

- What new failure surface did your design create?
- Which control is deterministic rather than prompt-based?
- What would invalidate the system's recommendation?
- What must remain under human authority in production?

LAB 3

Protected actions

Classify tools into read, draft, reversible write, communication, and consequential execution.

Required deliverable

Permission map and protected-action registry.

Review criteria

- Architecture is explicit and internally consistent.
- Evidence and uncertainty remain visible.
- Agent responsibilities and tools are bounded.
- Failure behavior is testable and fail closed where required.
- Human authority is named, timely, informed, and attributable.
- Claims match the evidence and maturity level.

Reflection questions

- What new failure surface did your design create?
- Which control is deterministic rather than prompt-based?
- What would invalidate the system's recommendation?
- What must remain under human authority in production?

LAB 4

Human authority

Design an approval gate that validates role, scope, evidence, freshness, and reversibility.

Required deliverable

Gate policy and negative tests.

Review criteria

- Architecture is explicit and internally consistent.
- Evidence and uncertainty remain visible.
- Agent responsibilities and tools are bounded.
- Failure behavior is testable and fail closed where required.
- Human authority is named, timely, informed, and attributable.
- Claims match the evidence and maturity level.

Reflection questions

- What new failure surface did your design create?
- Which control is deterministic rather than prompt-based?
- What would invalidate the system's recommendation?
- What must remain under human authority in production?

LAB 5

Fail closed

Inject missing evidence, stale data, wrong role, and tool failure into one workflow.

Required deliverable

Explicit blocked states and trace evidence.

Review criteria

- Architecture is explicit and internally consistent.
- Evidence and uncertainty remain visible.
- Agent responsibilities and tools are bounded.
- Failure behavior is testable and fail closed where required.
- Human authority is named, timely, informed, and attributable.
- Claims match the evidence and maturity level.

Reflection questions

- What new failure surface did your design create?
- Which control is deterministic rather than prompt-based?
- What would invalidate the system's recommendation?
- What must remain under human authority in production?

LAB 6

Red teaming

Create ten held-out scenarios for prompt injection, privilege escalation, and provenance loss.

Required deliverable

Evaluation pack with acceptance criteria.

Review criteria

- Architecture is explicit and internally consistent.
- Evidence and uncertainty remain visible.
- Agent responsibilities and tools are bounded.
- Failure behavior is testable and fail closed where required.
- Human authority is named, timely, informed, and attributable.
- Claims match the evidence and maturity level.

Reflection questions

- What new failure surface did your design create?
- Which control is deterministic rather than prompt-based?
- What would invalidate the system's recommendation?
- What must remain under human authority in production?

LAB 7

Observability

Define a trace that reconstructs every material decision, handoff, tool call, and approval.

Required deliverable

Trace schema and reviewer view.

Review criteria

- Architecture is explicit and internally consistent.
- Evidence and uncertainty remain visible.
- Agent responsibilities and tools are bounded.
- Failure behavior is testable and fail closed where required.
- Human authority is named, timely, informed, and attributable.
- Claims match the evidence and maturity level.

Reflection questions

- What new failure surface did your design create?
- Which control is deterministic rather than prompt-based?
- What would invalidate the system's recommendation?
- What must remain under human authority in production?

LAB 8

Flagship extension

Add a sixth agent to one Demo Lab flagship without increasing autonomous authority.

Required deliverable

Updated flow, scenario, tests, and rationale.

Review criteria

- Architecture is explicit and internally consistent.
- Evidence and uncertainty remain visible.
- Agent responsibilities and tools are bounded.
- Failure behavior is testable and fail closed where required.
- Human authority is named, timely, informed, and attributable.
- Claims match the evidence and maturity level.

Reflection questions

- What new failure surface did your design create?
- Which control is deterministic rather than prompt-based?
- What would invalidate the system's recommendation?
- What must remain under human authority in production?

LAB 9

Enterprise assessment

Score a real or synthetic agent system across nine readiness dimensions.

Required deliverable

Executive scorecard, blockers, 30-day and 90-day roadmap.

Review criteria

- Architecture is explicit and internally consistent.
- Evidence and uncertainty remain visible.
- Agent responsibilities and tools are bounded.
- Failure behavior is testable and fail closed where required.
- Human authority is named, timely, informed, and attributable.
- Claims match the evidence and maturity level.

Reflection questions

- What new failure surface did your design create?
- Which control is deterministic rather than prompt-based?
- What would invalidate the system's recommendation?
- What must remain under human authority in production?

LAB 10

Control plane

Design the minimum registry, permission, evaluation, approval, and audit modules for an organization.

Required deliverable

Product requirements and data model.

Review criteria

- Architecture is explicit and internally consistent.
- Evidence and uncertainty remain visible.
- Agent responsibilities and tools are bounded.
- Failure behavior is testable and fail closed where required.
- Human authority is named, timely, informed, and attributable.
- Claims match the evidence and maturity level.

Reflection questions

- What new failure surface did your design create?
- Which control is deterministic rather than prompt-based?
- What would invalidate the system's recommendation?
- What must remain under human authority in production?