Software/System Architecture Document (SAD)

Project: Design Lab

Date: 2026-02-06

Format baseline: IEEE 1016-aligned structure (adapted).

Contents

1. Introduction

1.1 Purpose
This document describes the architecture of the Design Lab system, including its major components, responsibilities, interfaces, key design decisions, and verification approach. It also incorporates the Constitutional Build Pipeline workflow for reliable, test-driven package assembly.

1.2 Scope
The SAD covers runtime architecture (services, orchestration, routing, memory/data, multimodal pipeline), development/build architecture (package workflow), and security/governance. It is intended to be the primary reference for engineers and stakeholders building and operating the system.

1.3 Definitions, Acronyms, Abbreviations

SAD:	Software/System Architecture Document
PRD:	Product Requirements Document
TDD:	Test-Driven Development
CI/CD:	Continuous Integration / Continuous Delivery
LLM:	Large Language Model

1.4 References
- 20260205_SystemArchitectureDocument.pdf
- 20260205_PRD.pdf
- 20260206_workflow_SAD.docx
- 20260205_Model_Serving_Orchestration.md
- 20260205_Multimodal_Pipeline_Design.md
- 20260205_SecurityAndGovernanceDesign.pdf
- 20260205_DataArchitectureAndMemoryDesignDocument.pdf
- 20260205_RoutingLogicSpecification.pdf

2. Architectural Representation
The architecture is represented using: (a) component decomposition views, (b) interface and dependency descriptions, (c) behavioral/workflow descriptions for routing and build pipelines, and (d) cross-cutting views for security, governance, and observability.

3. Architectural Goals and Constraints

3.1 Goals
- High-quality responses via robust routing logic and model orchestration.

- Deterministic, testable package builds using the Constitutional Build Pipeline.
- Secure handling of data with governance controls and least privilege.
- Scalable multimodal processing for text/image/video inputs where supported.
- Operational observability for latency, cost, quality, and safety metrics.

3.2 Constraints

- Tooling and connectors limited to supported services; external service connections must be controlled.
- Credits/cost budgeting and rate limits influence routing decisions.
- Security requirements constrain storage and access patterns for sensitive data.
- Reproducibility: builds and evaluations must be repeatable and auditable.

4. System Overview

Design Lab is an AI-assisted system comprised of a chat interface, orchestration layer, routing logic, model-serving layer, memory/data services, and security/governance controls. It supports multimodal inputs and integrates a structured software build workflow for producing reliable packages.

5. Architectural Viewpoints

5.1 Logical View (Components)

- Client/UI: Chat-based interaction surface for users.
- Orchestrator: Coordinates tool calls, planning, and execution steps.
- Router: Selects models/tools and policies based on task and constraints.
- Model Serving Layer: Executes model inference; manages providers/engines.
- Memory/Data Layer: Stores and retrieves conversation artifacts and knowledge.
- Multimodal Pipeline: Normalizes and processes text/image/video inputs.
- Security & Governance: Policy enforcement, audit logging, access control.
- Observability: Metrics, traces, logs, evaluation hooks.

5.2 Deployment View

Services may be deployed as separate scalable components. The orchestrator and router are latency-critical and should be deployed close to model-serving endpoints; memory/data may be deployed with appropriate isolation based on sensitivity tiers.

6. Component Descriptions

6.1 Orchestrator

Responsibilities: plan execution steps, invoke tools, handle retries, manage context windows, and enforce user/developer/system constraints.

6.2 Router

Responsibilities: classify tasks, choose model/tool routes, manage cost/latency tradeoffs, and apply policy constraints from governance.

6.3 Model Serving & Orchestration

Responsibilities: execute inference requests, manage batching/caching where applicable, provide structured outputs, and integrate with evaluation and safety filters.

6.4 Memory/Data Architecture

Responsibilities: store conversation state, long-term memory (where enabled), document indices, and retrieval mechanisms with tiered access controls.

6.5 Multimodal Pipeline

Responsibilities: ingestion and normalization of modalities, extraction (e.g., OCR/transcription), and passing structured representations to the orchestrator/router.

6.6 Security & Governance

Responsibilities: access control, secrets management, audit trails, policy enforcement (data retention, PII handling), and model/tool allowlists/denylists.

7. Interfaces and Data Flows

7.1 Primary Request Flow

- User submits request via UI.
- Orchestrator receives request and consults Router.
- Router selects plan: model call(s), retrieval, tool invocation.
- Orchestrator executes plan, calling Model Serving and tools.
- Memory/Data layer supports retrieval and persistence as allowed.
- Security & Governance checks occur at each boundary.
- Response returned to UI with any generated artifacts.

8. Behavioral Descriptions

8.1 Routing Behavior

Routing behavior follows the routing logic specification, combining task intent classification, constraint evaluation (latency/cost/policy), and selection of model/tool sequences. The router must be auditable: each route decision should be logged with rationale and inputs.

8.2 Constitutional Build Pipeline (Development & Build Architecture)

The system's package-development workflow is a structured, TDD-aligned pipeline to ensure correctness and reproducibility of software packages and deliverables.

8.2.1 Workflow Stages

- Stage 1: Identify package boundaries and contract (inputs/outputs, invariants).
- Stage 2: Define the public API and acceptance criteria.
- Stage 3: Draft test plan and harness (unit/integration/property tests).
- Stage 4: Implement minimal functionality to satisfy tests.
- Stage 5: Run full test suite and static checks; fix failures.
- Stage 6: Assemble/package (build artifacts, versioning, metadata).
- Stage 7: Post-integration validation (smoke tests, dependency checks).
- Stage 8: Release gates and documentation updates.

8.2.2 Quality Gates

- Constraints & invariants checkpoint before full test implementation.
- Coverage thresholds and deterministic test execution.
- Reproducible builds with pinned dependencies and artifact hashes.
- Audit trail for changes, approvals, and release decisions.

9. Cross-Cutting Concerns

9.1 Observability

- Metrics: latency, cost, token usage, tool-call counts, error rates.
- Quality signals: user feedback, eval scores, safety incidents.
- Tracing across orchestrator-router-model-serving boundaries.

9.2 Security

- Least-privilege access to data and tools.
- Encryption in transit and at rest where applicable.
- Policy enforcement for sensitive documents and connectors.
- Comprehensive audit logging of tool invocations and data access.

9.3 Governance

- Model/tool allowlists by environment and role.
- Change management for routing policies and prompts.
- Data retention and deletion policies.

10. Verification and Validation

Verification and validation combine automated tests, evaluation harnesses, and operational monitoring. The Constitutional Build Pipeline enforces TDD and release gates. Routing and safety policies are validated through offline test suites and online canaries.

11. Risks and Mitigations

- Model regressions: mitigate with eval suites, canaries, rollback.
- Cost overruns: enforce budgets and route constraints.
- Security incidents: continuous audits, least privilege, monitoring.
- Tool reliability: retries, fallbacks, and circuit breakers.

12. Open Issues

- Finalize deployment topology and scaling parameters per environment.
- Define precise routing policy matrices for common task classes.
- Specify memory retention tiers and user controls for memories.
- Formalize release governance for routing/prompt changes.

Appendix A. Glossary

See Definitions, Acronyms, Abbreviations; expand as the project evolves.

Appendix B. Document History

Initial generated SAD draft compiled from referenced design documents and workflow SAD.