

✦ GUIDE · AI ARCHITECTURE

AI Architecture Patterns

A catalogue of the patterns behind production AI systems — RAG, routers, tool-using agents, planner-executor, evaluator-optimizer, gateways, semantic caches, review queues and batch enrichment — with when to use each and what it costs you.

FORMAT	LEVEL	READ TIME	UPDATED
Guide	Advanced	15 min	May 2026

OVERVIEW

What this guide covers

Most production AI systems are assembled from a small number of recurring patterns. Knowing them by name makes design discussions shorter and trade-offs explicit: what each pattern is for, where it breaks, and what it adds in latency, cost and operational burden. This guide catalogues the patterns we use most, groups them by the problem they solve, and shows how they combine into a typical system.

KEY TAKEAWAYS

- Start with the simplest pattern that solves the problem; add agents only when the path is genuinely dynamic.
- Retrieval, routing and evaluation patterns usually deliver more reliability than a bigger model.
- A model gateway is the foundation for cost control, provider choice and observability across every other pattern.
- Caching and batch patterns change the economics of a system more than prompt tuning does.
- Human review queues are an architectural component, with their own data model and SLAs.

WHO IT'S FOR

- Software and solution architects
- AI / ML and platform engineers
- Technical leads planning AI features
- CTOs reviewing system designs

HOW TO USE IT

Use the pattern map in section 01 to find candidates, read the relevant pattern cards, then check the trade-off table in section 06 before committing to a design.

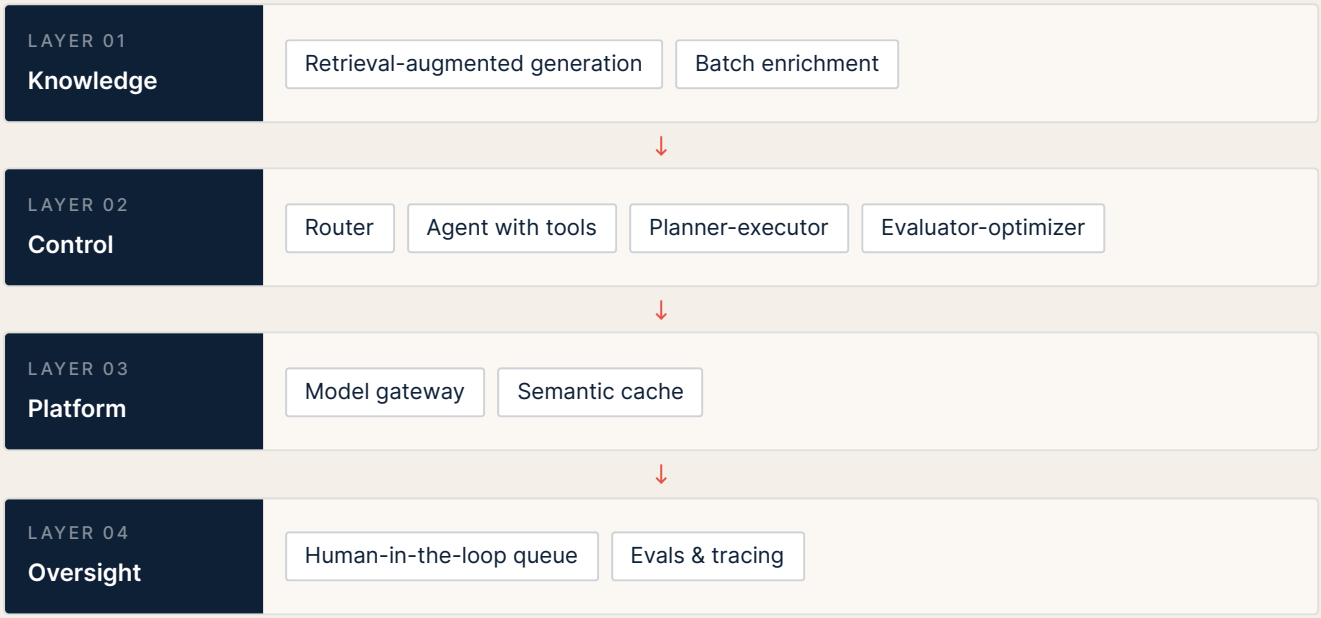
CONTENTS

01 The pattern map
02 Knowledge patterns
03 Control-flow patterns
04 Platform patterns
05 Human-in-the-loop queue
06 Comparing patterns and combining them

01 / MAP

The pattern map

Patterns fall into four groups by the problem they solve. Most real systems use one pattern from each of the first two groups and several from the last two.



Quick selector

IF YOU NEED TO…	START WITH	THEN CONSIDER
Answer from your own documents	RAG	Router for query types, semantic cache
Handle very different request types	Router	Specialized pipelines per route
Take actions in other systems	Agent with tools	HITL queue for high-impact actions
Complete long multi-step tasks	Planner-executor	Checkpointing and evaluator-optimizer
Raise quality on hard outputs	Evaluator-optimizer	Human review on persistent failures
Process large backlogs cheaply	Batch enrichment	Smaller models, offline evals

✦ 02 / KNOWLEDGE

Knowledge patterns

Ground the model in your data — either at request time or ahead of time.

PATTERN

Retrieval-augmented generation

Retrieve relevant, permission-filtered passages at query time and have the model answer from them with citations.

PATTERN

Batch enrichment

Run models offline over records or documents to add summaries, tags, entities or embeddings that online systems then use cheaply.

RAG — USE WHEN

- Answers must reflect current, changing documents
- Users need citations to trust answers
- Access differs per user and must be enforced
- Trade-off: retrieval quality caps answer quality; adds latency

BATCH ENRICHMENT — USE WHEN

- The same derived data is read many times
- Latency at request time matters
- Volume is high and freshness can lag by hours
- Trade-off: stale until the next run; needs re-run on model change

The two combine well: enrich documents in batch with summaries, keywords and classifications, then let online RAG filter and rank on that metadata.

✦ 03 / CONTROL

Control-flow patterns

These patterns decide who chooses the next step: your code, a classifier or the model itself. Move down the list only when you need to.

01 Router

A classifier or small model labels the request and sends it to a specialized pipeline, prompt or model. Cheap, predictable and easy to evaluate; fails when categories overlap or new intents appear.

02 Agent with tools

The model loops: choose a tool, observe the result, decide again. Flexible for open-ended tasks; harder to test, bound and secure, and costlier per task.

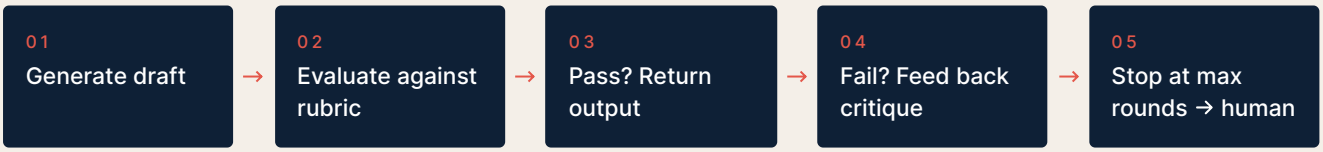
03 Planner-executor

One step writes an explicit plan; executors carry out each step, possibly with different models or tools. Good for long tasks and auditability; plans go stale and need re-planning.

04 Evaluator-optimizer

A generator produces output; an evaluator scores it against criteria and returns feedback until it passes or a limit is hit. Raises quality on checkable outputs; multiplies latency and cost.

Evaluator-optimizer loop



✦ RULE OF THUMB

Deterministic first, autonomous last

If you can write the steps as code with model calls inside, do. Routers and workflows are easier to test, cheaper to run and simpler to explain to auditors than open-ended agents.

04 / PLATFORM

Platform patterns

Shared infrastructure that every AI feature benefits from. Build these once.

PATTERN

Model gateway

A single service in front of all model providers that handles auth, routing, quotas, retries, fallbacks, logging and cost attribution.

PATTERN

Semantic cache

Store answers keyed by query embedding and scope; return a cached answer when a new query is close enough and the sources have not changed.

CONCERN	MODEL GATEWAY	SEMANTIC CACHE
Main benefit	Provider choice, cost control, one place for observability	Lower latency and cost on repeated questions
Main risk	Single point of failure; needs high availability	Serving a wrong or stale answer to a similar-looking query
Key design choice	Per-team keys, budgets and model allow-lists	Similarity threshold, scope (tenant, permissions, language), invalidation
Do not use when	A single prototype with one provider	Answers are personalized or depend on live data

FIELD NOTE

Scope every cache key

A semantic cache must include tenant, permission scope, language and source version in its key. Without them, it becomes a data-leak and staleness bug waiting to happen.

05 / OVERSIGHT

Human-in-the-loop queue

When outputs need review, the queue is a product in its own right: it decides what people see, in what order, and how their decisions feed back.



Design decisions

- Routing rules: confidence thresholds, risk tier, amount limits, customer segment
- What the reviewer sees: input, proposed output, sources, reason for review
- Service levels and what happens when they are missed (escalate, default, hold)
- How edits are captured as structured corrections, not just free text
- How thresholds are tuned over time as quality data accumulates

06 / TRADE-OFFS

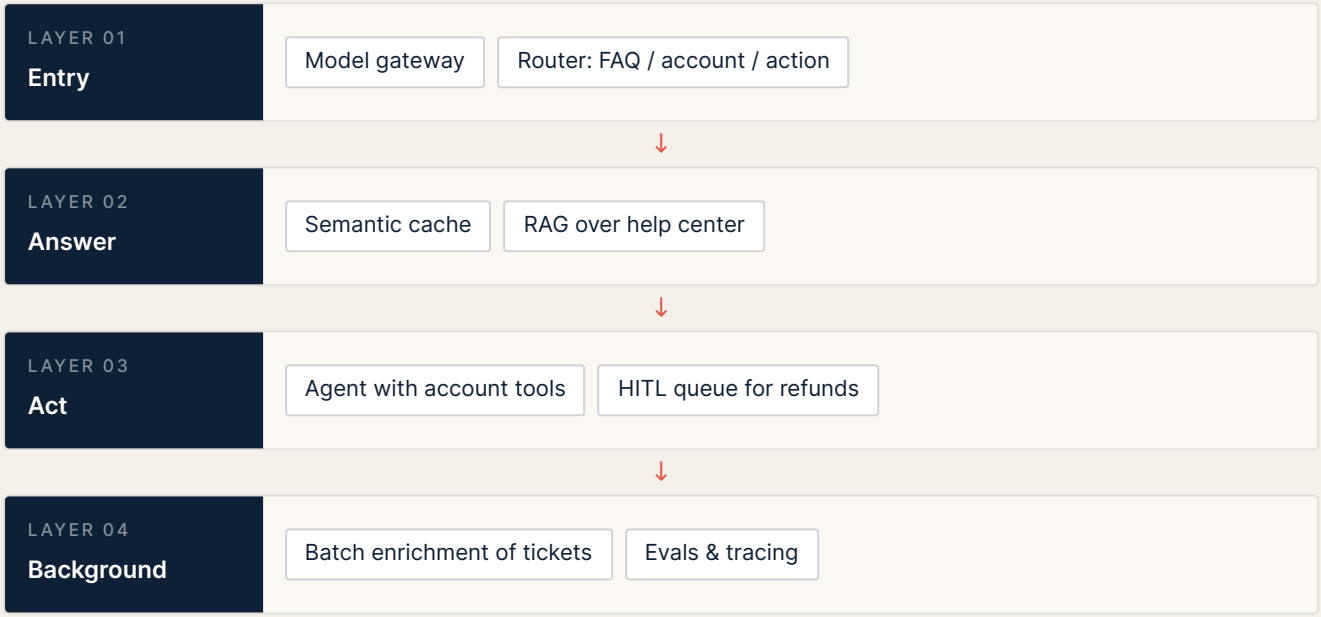
Comparing patterns and combining them

Every pattern trades something. Make the trade explicit in the design review.

PATTERN	LATENCY	COST	PREDICTABILITY	TESTING EFFORT
RAG	Medium	Medium	High with good retrieval	Golden questions + retrieval metrics
Router	Low	Low	High	Labeled classification set
Agent with tools	High, variable	High, variable	Low–medium	Scenario and trajectory tests
Planner-executor	High	High	Medium	Plan and step-level tests

PATTERN	LATENCY	COST	PREDICTABILITY	TESTING EFFORT
Evaluator-optimizer	Multiplied by rounds	Multiplied by rounds	Medium–high	Rubric calibration
Batch enrichment	None at request time	Low per item	High	Offline sampled review

A typical combined system: support assistant



> Name the pattern, state the trade-off, then write the code.
>
> SPLENDENSLABS ENGINEERING PRINCIPLE

✦ GET IN TOUCH

Let's Talk

We design AI systems from proven patterns and build them to production standards. Bring us your use case and we will sketch the architecture with you.

- An architecture review with an engineer
- Pattern selection for your use case
- Gateway, cache and eval foundations
- A build plan with trade-offs made explicit

BOOK AN AI STRATEGY SESSION →

EMAIL
hello@splendenslabs.com

WEB
splendenslabs.com

OFFICE
Sona Incubation Foundation, Salem