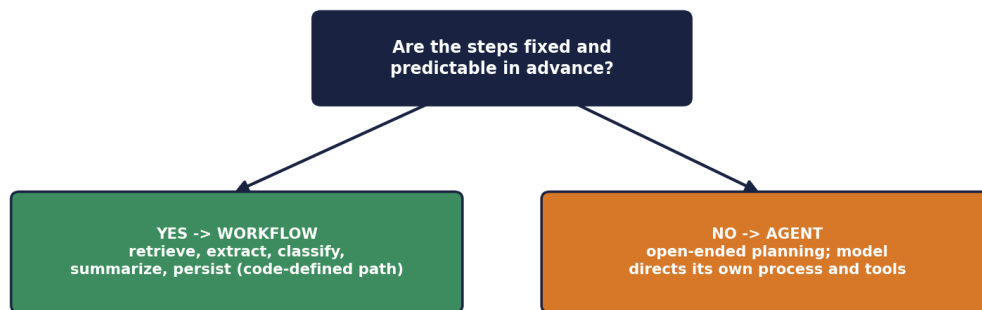


CLAUDE CERTIFIED ARCHITECT PROFESSIONAL (CCAR-P)

The Ultimate Study Guide

Enterprise-scale solution architecture with Claude: design, models and context, integration, evaluation, governance, and stakeholder leadership.

Workflow or Agent?



Start with the simplest solution; add autonomy only when needed.

How to Use This Guide

The Professional exam rewards architectural judgment: nearly every item presents constraints (latency, cost, security, compliance, audience) and asks for the best trade-off. Parts II–VIII teach the concepts; Part IX gives answer patterns and a cheat sheet for practice questions.

Part	Title	Maps to blueprint domain
I	Exam Facts & Blueprint	Overview
II	Solution Design & Architecture	1. Solution Design & Architecture
III	Models, Prompting, Context & Cost	2. Claude Models, Prompting & Context Engineering
IV	Integration: MCP, Tools, Claude Code	3. Integration
V	Evaluation, Testing & Optimization	4. Evaluation, Testing & Optimization
VI	Governance, Safety & Risk	5. Governance, Safety & Risk Management
VII	Stakeholders & Lifecycle	6. Stakeholder Communication & Lifecycle Management
VIII	Developer Productivity & Enablement	7. Developer Productivity & Operational Enablement
IX	Answer Patterns & Cheat Sheet	Review

Part I — Exam Facts & Blueprint

Exam details below come from third-party summaries of Anthropic's program (published around mid-2026) and may change. Always confirm on Anthropic's official certification page before registering.

Item	Reported detail
Exam	Claude Certified Architect, Professional (CCAR-P); advanced tier above Architect, Foundations (CCAR-F)
Format	Scenario-based multiple choice and multiple response (each item states how many to select); online proctored or test center via Pearson VUE
Length	Reported as 63 questions in 120 minutes (one source)
Passing score	720 on a 1000 scale; compensatory scoring (no per-domain minimum); score report shows percent correct by domain
Fee	Reported as \$175 USD per attempt
Recommended experience	About 3+ years in systems architecture or platform engineering and 6+ months with Claude/LLM systems in production
Validity	Reported as 12 months, renewed via a free non-proctored assessment
Retakes	Reported waits of 14, 30 and 90 days; up to four attempts per rolling 12 months

The seven blueprint domains

1. Solution Design & Architecture: turning business problems into Claude-based solutions and choosing patterns.
2. Claude Models, Prompting & Context Engineering: model choice, prompts, context and token optimization.
3. Integration: MCP, direct APIs, RAG pipelines, protocols.
4. Evaluation, Testing & Optimization: metrics, test frameworks, accuracy/latency/cost.
5. Governance, Safety & Risk Management: guardrails, compliance, ethics.
6. Stakeholder Communication & Lifecycle Management: requirements, documentation, feedback loops.
7. Developer Productivity & Operational Enablement: tooling for teams, debugging, operations.

Exam takeaway: Because scoring is compensatory, strong domains can offset weak ones, but scenario judgment appears in every domain, so practice the patterns in Part IX.

Part II — Solution Design & Architecture

2.1 Workflows vs. Agents

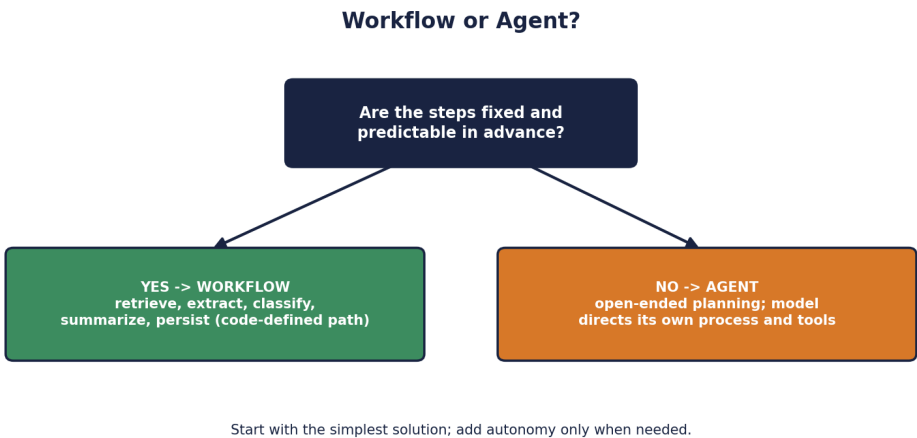


Figure 2.1 — Predictable path = workflow; open-ended = agent.

Anthropic's "Building Effective Agents" guidance distinguishes workflows (LLMs and tools orchestrated through predefined code paths) from agents (LLMs that dynamically direct their own process and tool use). Its core advice is to find the simplest solution that works and add complexity only when needed. A fixed batch pipeline (retrieve, extract, classify, summarize, persist) is a workflow, not an agent.

The five workflow patterns

Pattern	Use when	Example
Prompt chaining	Task decomposes cleanly into fixed sequential steps; each call uses the prior output (add programmatic checks between)	Draft, then translate; outline, then write
Routing	Distinct input categories deserve specialized prompts/handlers	Support triage to refund, technical, or billing handler; easy queries to a small model
Parallelization	Independent subtasks (sectioning) or multiple attempts for confidence (voting)	Guardrail check in parallel with answer; multi-reviewer code scan
Orchestrator-workers	Subtasks cannot be predicted in advance	Multi-file code changes; research across variable sources
Evaluator-optimizer	Clear evaluation criteria and iterative refinement adds value	Generate, critique, revise until the evaluator is satisfied

2.2 The Execution Layers

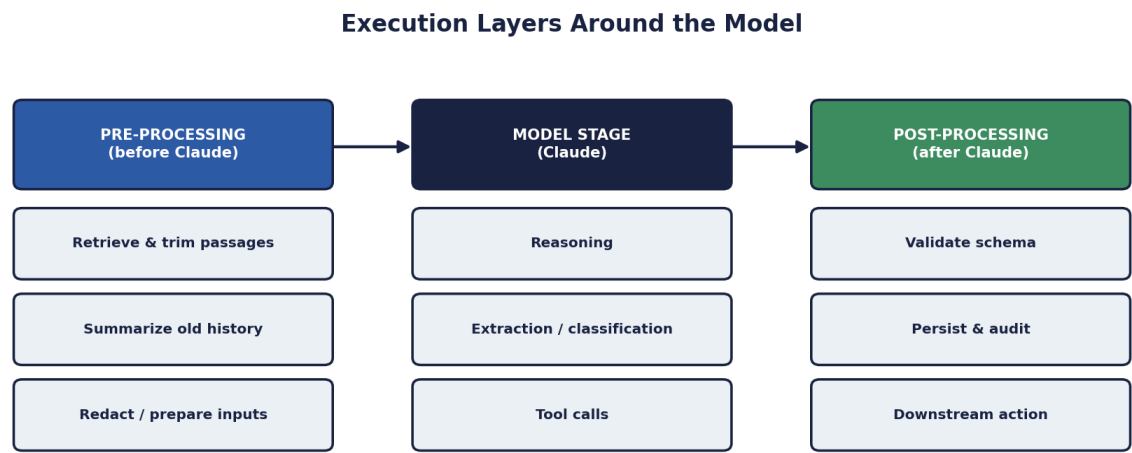


Figure 2.2 — Before Claude, Claude, after Claude.

Layer	What belongs here
Pre-processing	Retrieval, trimming retrieved passages, summarizing history, redaction, preparing inputs
Model stage	Reasoning, extraction, classification, tool calls
Post-processing	Schema validation, persistence for downstream use and audit, triggering actions

2.3 Decomposing Capabilities (When to Split an Agent)

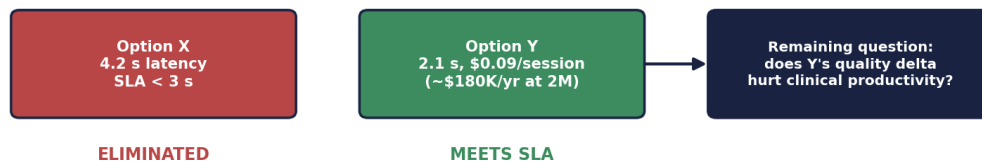
Before adding a router, look for evidence the agent should be decomposed: tools that cluster into distinct domains, and capabilities that are rarely or never used (unnecessary breadth). The router's extra call adds latency and cost, but that is a consequence, not the strongest evidence for decomposition. Splitting also supports least privilege and focused prompts.

2.4 Structured Discovery

Start with stakeholder workshops to surface latency, accuracy, and compliance constraints before scoping, and document explicit success criteria and failure thresholds that will gate production. Choosing a model tier from past experience, building a prototype first, or deferring constraints to integration design all come too early or too late.

2.5 Deciding Under Hard Constraints

Eliminate Hard-Constraint Violators First



Hard SLA / security / compliance constraints dominate; compare survivors on value.

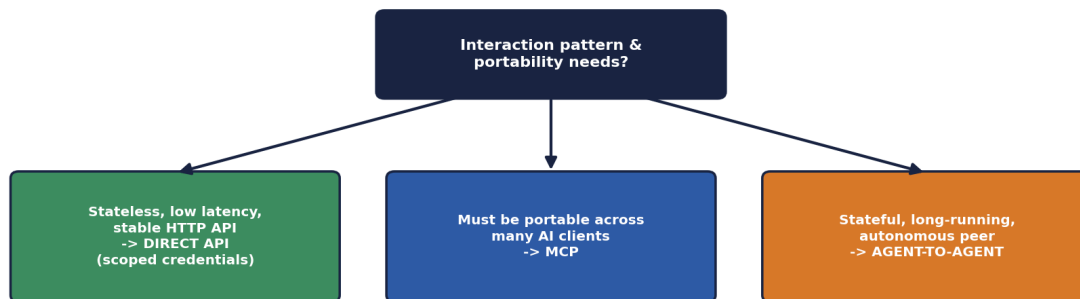
Figure 2.3 — Eliminate violators of hard constraints first.

Case: Option X scores better on quality but takes 4.2 seconds against an under-3-second SLA. Option Y meets it at 2.1 seconds and about \$0.09 per session (roughly \$180,000 per year at 2 million sessions). Correct approach: eliminate X on the SLA, then evaluate Y against efficiency and solution-cost pillars, and judge whether its quality delta materially affects clinical productivity. Do not choose by cost alone, recommend a hybrid with no analysis, or pick the premium option and fix the SLA later.

Similarly, if managed agent infrastructure would move financial data outside the approved network boundary, that hard security constraint dominates the recommendation regardless of convenience.

2.6 Choosing the Integration Protocol

Choosing Direct API, MCP, or Agent-to-Agent



Do not add agent/session layers where a direct call suffices.

Figure 2.4 — Protocol follows interaction pattern and portability.

- Stateless, low-latency service with a stable HTTP API: call it directly with appropriately scoped credentials. Do not add another Claude agent, a bespoke streaming protocol, or a long-lived session.

- The two factors that drive protocol choice: whether the integration must be portable across multiple AI clients, and whether the interaction is stateless/latency-sensitive or stateful/longer-running. Naming conventions, team familiarity, and transport encryption are not the deciding factors.

Part III — Models, Prompting, Context & Cost

3.1 Model Selection

Claude is offered in tiers. Lighter models (Haiku) are fast and cost-efficient for high-volume, well-defined work; mid-tier (Sonnet) balances capability and cost for most production workloads; the most capable tiers (Opus and above) serve complex, multi-step reasoning at higher latency and cost. Check Anthropic's current model list, since names and versions change.

- High-volume content moderation or other well-defined classification: Haiku.
- Choose by measured workload fit across quality, latency, cost, and requirements. Model size or recency alone is not a justification.
- Tiered routing: send straightforward query types to a smaller faster model and reserve the full model for complex cases.

3.2 Prompt Caching

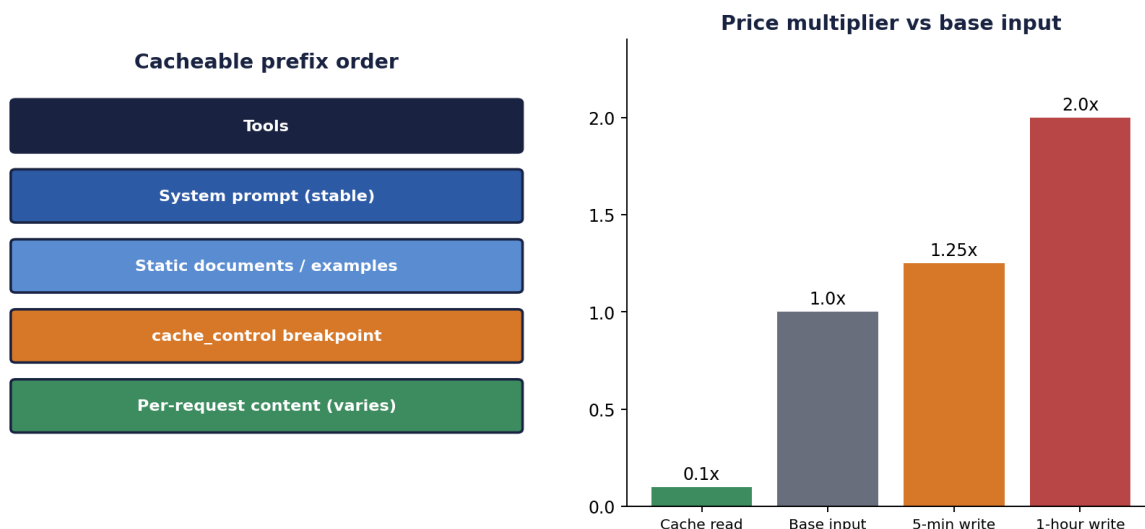


Figure 3.1 — Put stable content first; reads cost a fraction of base input.

Prompt caching lets Anthropic reuse the processed prefix of a prompt across requests. Per Anthropic's documentation:

- The cache references the whole prefix in order: tools, then system, then messages, up to the block marked with `cache_control`. Stable content must come before variable per-request content.
- Pricing multipliers: 5-minute cache writes cost 1.25x base input, 1-hour writes 2x, and cache reads 0.1x. Default TTL is 5 minutes; a 1-hour TTL is available.
- Minimum cacheable prompt length applies (reported as about 1,024 tokens for most Sonnet/Opus models and higher for some smaller models); up to four breakpoints per request; the API reports `cache_creation_input_tokens` and `cache_read_input_tokens` so you can verify hits.
- Caching pays off only if the same prefix is reused within the TTL; otherwise you pay the write premium without reads.

Exam cases:

- Same 80,000-token filing queried 14 times per case: cache the filing as the prompt prefix. Summarizing, vector retrieval, or a smaller model risks losing source fidelity. Keep cached content within the approved tenancy for security.
- Interactive assistant with per-turn latency limits: reduce unnecessary retrieval work and cache stable prompt content.
- Cost optimization while accuracy must stay intact: cache repeated content; do not remove information or downgrade the model.

3.3 Cost and Latency Levers

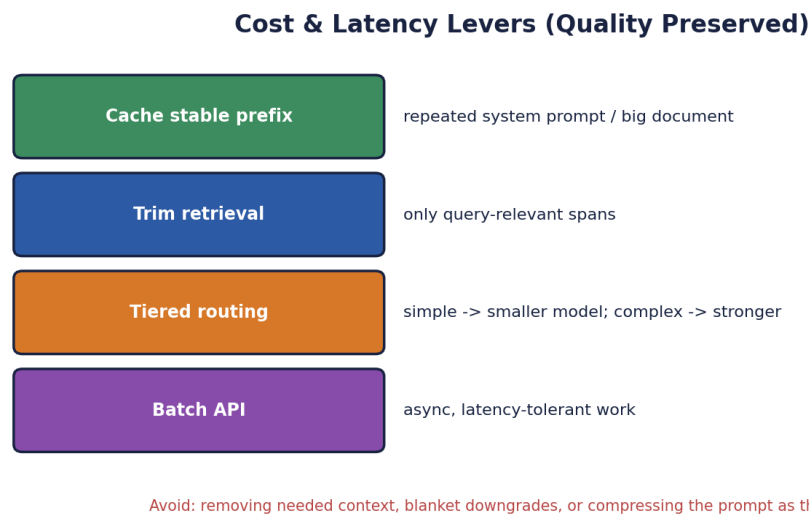


Figure 3.2 — Levers that preserve quality.

- Dominant cost from a long repeated prompt plus lengthy retrieval: move the repeated prompt into a cacheable prefix and trim retrieved context to query-relevant spans.
- High-volume pipeline needing lower cost and p95 latency: cache the static system-prompt prefix and route simple work to a smaller model. Raising max tokens, retrieving the full corpus, or adding more few-shot examples makes cost and latency worse.
- Latency-tolerant bulk work can use the Message Batches API for lower cost.

3.4 Token-Management Lifecycle

Stage	Activities
Input preparation	Trim retrieved passages to relevant spans; summarize prior history when full history is unnecessary
Prompt construction	Move stable repeated content into a cacheable prefix; order sections so cacheable content precedes per-request content
Output handling	Validate structured output against schema; persist validated output for downstream use and audit

Exam takeaway: Input = prepare/reduce context. Prompt = arrange/cache. Output = validate/persist.

Part IV — Integration: MCP, Tools & Claude Code

4.1 Model Context Protocol (MCP)

MCP is an open standard through which AI clients connect to external tools, data, and prompts exposed by servers. Its main value is portability: one server can serve many clients. The Claude API also offers an MCP connector for remote servers, with per-server toolsets that can allowlist, denylist, or defer tools.

4.2 Many Tools: Tool Search

Tool Search: On-Demand Discovery of Many MCP Tools



Fixes context exhaustion from upfront tool definitions while keeping every capability reachable.

Figure 4.1 — Deferred loading plus on-demand discovery.

When Claude Code sessions load 60+ MCP tools and run out of context, the fix is on-demand tool discovery. Tool definitions are expensive (reported at roughly 10–20K tokens for ~50 tools). With the tool search tool and `defer_loading`, Claude initially sees only the search tool (and in Claude Code, server instructions and tool names), searches when needed, and loads only the few relevant definitions. In current Claude Code releases this is on by default. Pre-trimming to fewer servers or shrinking responses does not address the cause.

4.3 Claude Code Permissions

Claude Code Permission Evaluation



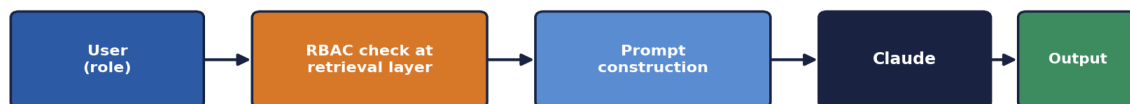
Enterprise rule: least privilege + explicit denies (defense in depth).

Figure 4.2 — Deny first, then ask, then allow.

Claude Code evaluates permission rules in deny, ask, allow order, and deny always wins. For enterprise rollout, apply least privilege and defense in depth: scope allows narrowly to each workflow and add explicit denies for sensitive paths and commands. Blanket allow-all or bypass modes, and relying on prompts to behave, are wrong. Claude Code also asks for approval for new MCP servers and caps MCP tool output by default.

4.4 Retrieval and RBAC

Enforce Authorization BEFORE Data Reaches the Model



Prompt-only rules are not an authorization boundary; post-filtering is too late.

Figure 4.3 — Authorize before context is built.

For finance analysts and external auditors with different permissions, enforce role-based access control at the retrieval layer so restricted content never reaches prompt construction or the model. System-prompt rules are not an authorization boundary, post-generation filtering is too late, and a shared organization-wide token removes accountability.

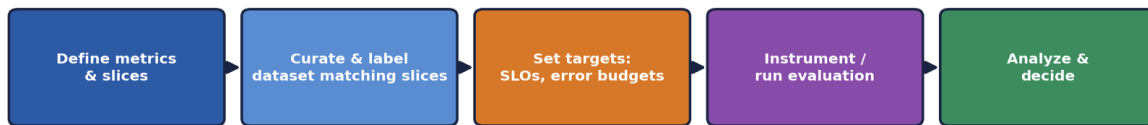
4.5 Tool Scope for Agents

Procurement agent: keep task-relevant tools (read-only policy lookup, draft-writing into a review queue, validation helpers) and remove direct external sending (emailing vendors) and unrelated administrative access (production database admin). The agent drafts; humans and controlled systems act.

Part V — Evaluation, Testing & Optimization

5.1 Specify Before You Execute

Specify Before You Execute (Evaluation & Monitoring)



Instrumentation and evaluation are driven by predefined metrics, slices and operational targets.

Figure 5.1 — Define, label, set targets, then run.

- Evaluation pipeline: before running, define what will be measured (metrics and slices) and curate a properly labeled dataset matching those slices.
- Monitoring: before instrumentation, define metrics/slices and operational targets such as SLOs and error budgets.
- Choose metrics that match the stated requirement: RBAC enforcement measures unauthorized-access prevention; per-query token cost measures the cost ceiling. Generic quality scores do not.

5.2 Controlled Prompt Experiments

Controlled Prompt Experiment

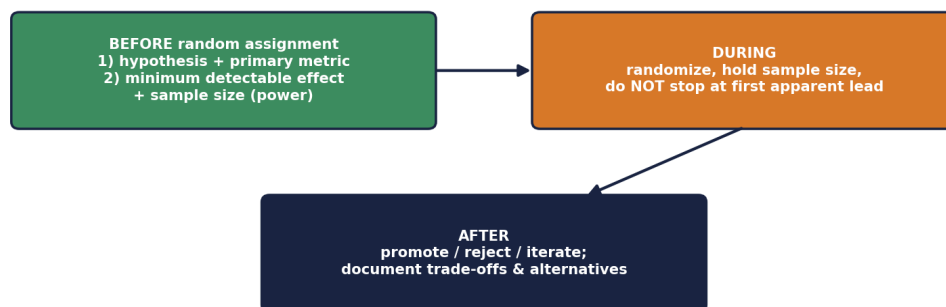


Figure 5.2 — Pre-register the design; do not peek and stop.

Before random assignment: define the hypothesis and primary success metric, and determine the minimum detectable effect and the sample size needed for statistical power. Do not monitor interim metrics and stop as soon as one prompt

looks ahead, since that biases results. Afterward, decide to promote, reject, or iterate, and document recommendation, trade-offs, and alternatives.

5.3 Test Strategy

- Safety under attack: adversarial testing (prompt injection, jailbreaks, data exfiltration attempts).
- Cross-component behavior: integration testing.
- Sensitive financial pipelines need both; unit tests alone are insufficient.

5.4 Investigating Incidents and Regressions

- Fabricated claims in research summaries: inspect grounding safeguards, and sample recent outputs to measure whether the incident is isolated or systemic.
- Assistant now contradicts retrieved sources with retrieval unchange: test whether the failure reproduces on the previous model version and inspect whether the system prompt's grounding and citation constraints remain intact. Bigger context windows, new embedding models, or lower temperature do not target the likely change.

5.5 Fairness Evaluation

For loan pre-qualification, approval-rate parity alone is not enough; analyze error rates (false approvals/denials) across groups. Agreement with human decisions is not proof of fairness because humans may be biased and the model can reproduce it.

5.6 Reading Pilot Metrics

A procurement assistant with high acceptance but frequent specialist overrides: high acceptance does not establish correctness after specialist review, and repeated overrides based on omitted criteria show a material scope/input gap. Pilot length or response rate are secondary.

5.7 Reviewing AI-Generated Code

AI-generated pull requests still need verification, and a small line count can hide a high-impact security or correctness issue. Risk is not proportional to diff size.

Part VI — Governance, Safety & Risk Management

6.1 Layered Guardrails

Guardrails are layered controls plus validation before action: defense in depth with input controls, constrained tools, and output validation that gates downstream actions. A single prompt instruction is never the only control.

Regulated investment recommendations: define explicit out-of-scope categories with fixed refusal behavior in the system prompt, and add a runtime output classifier that blocks regulated recommendation language before delivery. Logging to a SIEM is detective, not preventive; session limits and higher temperature are irrelevant or harmful.

6.2 Identity, Credentials and Attribution

Security audit finding a shared API key and unattributed calls: replace the shared key with per-user OAuth tokens carrying scope-restricted permissions, and add actor attribution so tool-call logs record the initiating user. Moving secrets to a vault, RBAC at retrieval, and output validation are good practices but do not directly fix shared identity and missing attribution.

6.3 Human Oversight and Delegation

Financial decisions such as refunds carry real consequences; delegate to Claude with appropriate human oversight rather than assuming small amounts are automatically low-risk. Match oversight to impact and reversibility.

6.4 Transparency

For customer-facing AI, disclose that users are interacting with AI and provide a clear human escalation path.

6.5 Regulatory Scoping

Before mapping data flows to regulatory clauses: (1) identify which regulations apply based on data types, jurisdiction, and audience, and (2) inventory vendor-provided compliance tooling and which capabilities are in place. Only then compare controls to requirements, document gaps and residual risk, and schedule remediation. Common frameworks to recognize include GDPR (personal data in the EU), HIPAA (US protected health information), and FedRAMP (US federal cloud authorization).

6.6 Common Risks to Recognize

Prompt injection, data leakage, hallucination or fabrication, bias and unfair outcomes, excessive agency (over-privileged tools), and insecure output handling (OWASP's LLM risk themes). Each maps to a control already covered: isolation, least privilege, grounding, fairness evaluation, scoped tools, and validation.

Part VII — Stakeholders & Lifecycle Management

7.1 Tailoring Communication

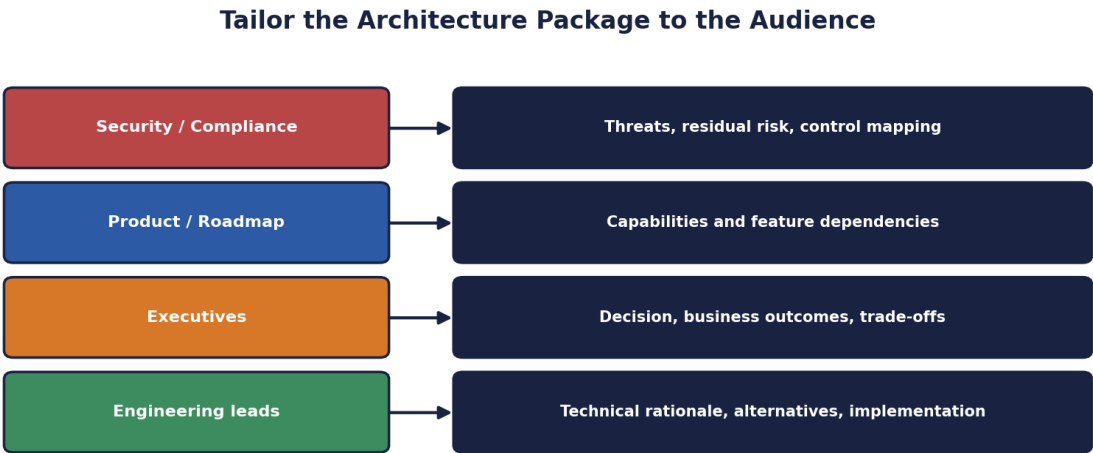


Figure 7.1 — Different audiences, different packages.

- Security and compliance review: threats, residual risk, and control mapping. Product and roadmap: capabilities and feature dependencies.
- Mixed audience decision presentation: lead with the decision, business outcomes, and trade-offs, then technical rationale, alternatives, and implementation implications. Do not open with a deep technical dive or announce and move straight to implementation.

7.2 Lifecycle Continuity

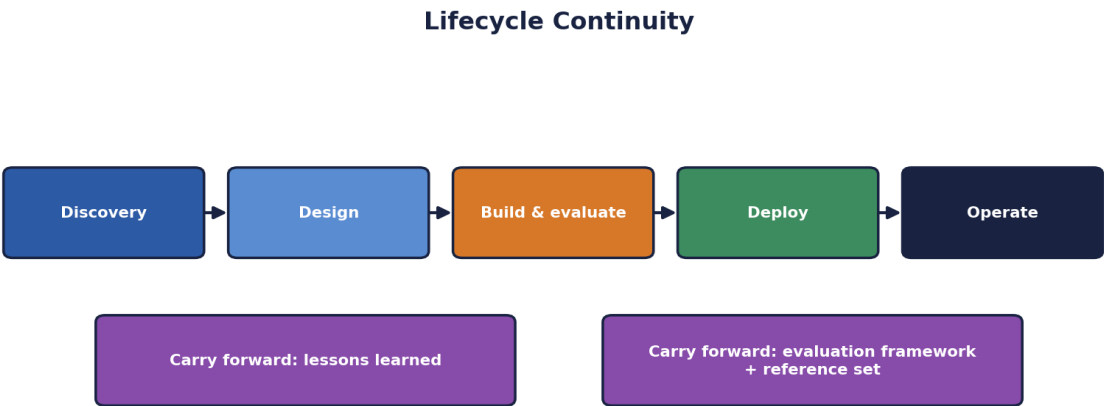


Figure 7.2 — Carry learning and evaluation standards forward.

Across phases, capture lessons learned at the end of each phase as inputs to the next, and carry the evaluation framework and reference set forward across iterations. Locking early decisions, notifying every stakeholder at every transition, and archiving everything are not continuity.

7.3 Deployment Handoff Documentation

A complete handoff includes the architecture overview, architecture decision records (ADRs), component diagrams, operational runbooks, an on-call playbook, and known limitations. Retrospectives, marketing one-pagers, and old design proposals do not enable operation or troubleshooting.

7.4 Discovery Revisited

Discovery sets the contract for the whole lifecycle: stakeholder-surfaced constraints and measurable success and failure thresholds (see 2.4).

Part VIII — Developer Productivity & Operational Enablement

- Keep Claude Code sessions efficient: rely on tool search for large MCP estates, scope permissions per workflow, and use project-level configuration (CLAUDE.md, settings) for shared standards.
- Standardize verification for AI-generated changes: tests, review checklists, and security scanning regardless of diff size.
- Debugging and operations: capture traces of prompts, tool calls, and validations; compare against previous model versions when behavior shifts; maintain runbooks and on-call playbooks.
- Enablement: document architecture decisions, known limitations, and escalation paths so teams can operate and extend the system.

Part IX — Answer Patterns & Cheat Sheet

9.1 Patterns That Recur

- Constraints first: hard SLA, security, or compliance constraints eliminate options before value comparison.
- Specify before executing: metrics, slices, datasets, SLOs, hypotheses, and sample sizes come before running anything.
- Enforce, don't request: authorization at retrieval, validation before action, deterministic blocking; prompts alone are not controls.
- Cache and trim, don't cut: preserve information while reducing repeated tokens.
- Least privilege: only task-relevant tools and scopes; explicit denies.
- Evidence over comfort: acceptance, agreement, and averages do not prove correctness or fairness.
- Right-size complexity: direct API for stateless HTTP, workflow for fixed steps, agent only for open-ended work.
- Audience-aware communication and complete handoffs.

9.2 Cheat Sheet by Question

Q	Topic	Answer / rule
1	Refund delegation	Human oversight for financial decisions
2	Fabrication incident	Inspect grounding safeguards + sample outputs for frequency
3	Eval pipeline	Define metrics/slices + labeled dataset
4	Monitoring spec	Metrics/slices + SLOs/error budgets
5	60+ MCP tools	On-demand tool discovery (tool search)
6	Clinical summary metrics	RBAC enforcement metric + per-query cost
7	Invoice layers	Pre / Model / Post
8	Per-turn latency	Reduce unnecessary retrieval + cache stable prompt
9	Customer-facing AI	Disclose AI + human escalation
10	Communication packages	Security: threats/controls; Product: capabilities/dependencies
11	Claude Code permissions	Least privilege + explicit deny
12	Fixed batch steps	Workflow
13	Managed agents, data boundary	Hard security constraint evaluated first
14	AI-generated PRs	Keep verification; risk is not line count
15	Loan fairness	Error-rate analysis; agreement is not fairness
16	Guardrails	Layered defense + validate before action
17	Model claims	Measured workload fit, not size/recency

18	Revenue projection cost	Cache repeated content; keep context and model
19	Before adding router	Distinct domains + unused tools -> decompose
20	Sensitive pipeline tests	Adversarial + integration
21	Procurement tools	Keep draft/validate/read tools; remove external send and DB admin
22	Stateless pricing service	Direct API call
23	Prompt experiment	Hypothesis + metric, MDE/sample size; no early stopping
24	Lifecycle continuity	Lessons learned + carry evaluation framework
25	Decision presentation	Decision/outcomes/trade-offs first, then technical
26	High acceptance, overrides	Acceptance is not correctness; input gap
27	Handoff docs	Architecture, ADRs, diagrams, runbooks, on-call, limitations
28	Regulated recommendations	Prompt refusal boundary + runtime classifier
29	Shared key audit	Per-user scoped OAuth + actor attribution
30	Discovery	Stakeholder workshops + success/failure criteria
31	Moderation classifier	Haiku
32	Repeated 80K filing	Prompt caching
33	Cost + p95 latency	Cache prefix + tiered routing
34	Repeated prompt + retrieval	Cacheable prefix + trimmed retrieval
35	MCP vs API vs A2A	Portability + interaction pattern
36	RBAC	Enforce before retrieval/context
37	Clinical architecture	Eliminate SLA violator, then value analysis
38	Before clause mapping	Applicability + compliance-tool inventory
39	Contradicts sources	Previous model version + grounding instructions
40	Token lifecycle	Input prep / Prompt construction / Output handling

Good luck on the Claude Certified Architect Professional exam.