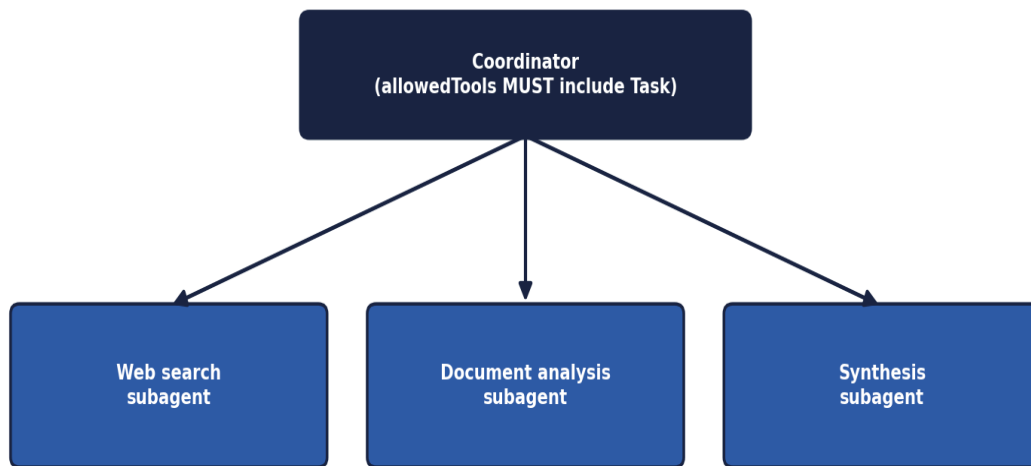


CLAUDE CERTIFIED ARCHITECT FOUNDATIONS

The Ultimate Study Guide

Every architecture decision, tool choice, and design pattern tested across the practice question bank.

Coordinator Delegation with the Task Tool



Independent tasks: emit all Task calls in ONE response -> run in parallel.
Subagents do not inherit context: pass the complete relevant findings explicitly.

How to Use This Guide

Every question describes a concrete system problem and asks for the best architectural choice. Answers follow a small set of rules of thumb (the exam shortcuts). Learn the reasoning behind each so you can handle variations. Read Parts I–VIII, then use Part IX for final review.

Part	Title	Focus
I	Batch vs. Real-Time Processing	Message Batches API, latency tolerance, failure retries
II	Claude Code in CI/CD	CLAUDE.md, CLI flags, limits, startup, GitHub App
III	Designing AI Code Review	Few-shot, reportability, bias, interference, tag-and-filter, test generation
IV	Tools & MCP	Built-in tools, descriptions, Resources, tool-set size, forced tool choice
V	Long-Running Work & Context	Scratchpads, summarize + delegate, resume, recovery
VI	Multi-Agent Research Systems	Chaining, orchestration, parallelism, provenance, conflicts, gaps
VII	Structured Extraction	Tool schemas, nulls, enums, validation loops, examples, context pressure
VIII	Quality Control & Human Review	Confidence, segment validation, stratified sampling
IX	Answer Patterns & Cheat Sheet	Fast recognition; one-page recall

Part I — Batch vs. Real-Time Processing

1.1 The Message Batches API

The Batches API processes requests asynchronously at lower cost. The deciding factor is not volume or file counts but whether the workload can tolerate delayed results.

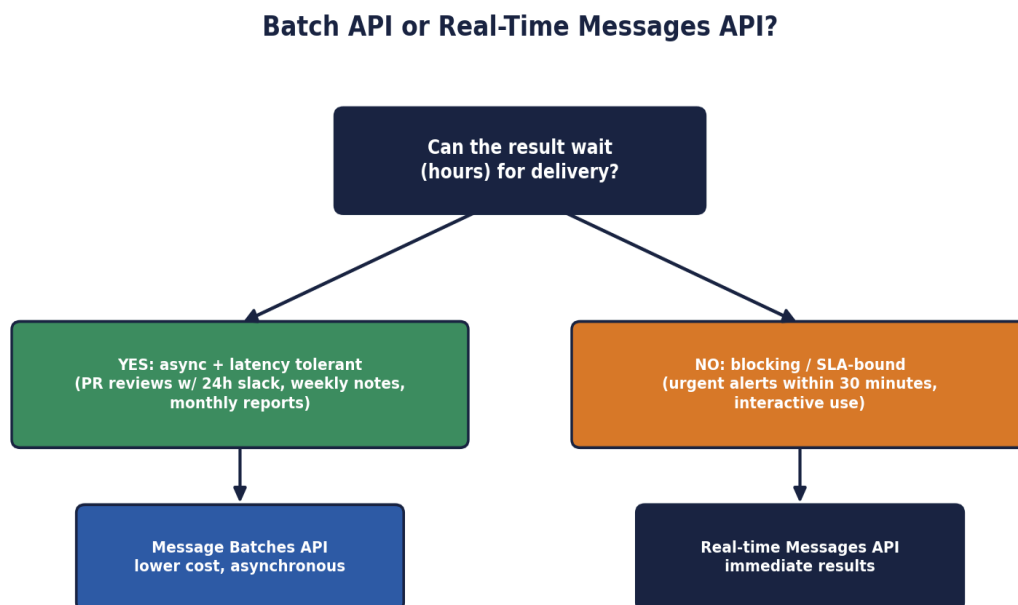


Figure 1.1 — Latency tolerance decides.

- Good fits: non-blocking PR reviews where feedback can wait up to 24 hours (about 50 PRs/day), weekly release notes from ~200 commits with 12 hours acceptable, monthly report processing.
- Poor fit: blocking steps and anything with a short SLA.
- Not the answer for cost: bigger single prompts, switching models regardless of quality, or just raising concurrency.

1.2 Mixed Workloads

When standard reports can be archived after processing but urgent exception reports must trigger alerts within 30 minutes, route standard reports to Batch and urgent ones to the real-time Messages API. Putting everything in Batch (even hourly) misses the SLA; putting everything in real time wastes cost.

1.3 Handling Batch Failures

In a 10,000-document batch where 300 fail with context-length-exceeded errors, use the custom IDs in the result file: chunk only the failed documents into smaller pieces, resubmit just those, and combine partial extractions. Raising `max_tokens` does not help (it limits output, not input context), and reprocessing everything or moving all documents to a larger-context model is wasteful.

Exam takeaway: Async + latency tolerance → Batch. Urgent SLA → real-time. Context error on a small subset → chunk and retry only the failures.

Part II — Claude Code in CI/CD

2.1 CLAUDE.md for Project Knowledge

CLAUDE.md holds persistent repository-level instructions that Claude Code uses when working in the project. Use it for:

- Project conventions and accepted patterns (e.g., force-unwrapping in tests, large coordinator classes, internal deprecated modules) so reviews stop flagging them as false positives.
- Linter-enforced style, generated code, and intentional patterns to ignore.
- Testing standards, criteria for valuable tests, existing fixtures, and examples.

Wrong fixes: custom filtering GitHub Actions, per-directory workflows, keyword blacklists, confidence thresholds, replacing CLAUDE.md with a different file, or disabling review in directories.

2.2 Important CLI Flags

Need	Flag / setting	Notes
Add custom review instructions but keep normal Claude Code behavior	--append-system-prompt	--system-prompt REPLACES the default prompt, which is why Claude then only commented on the diff
Cap iterations and spend	--max-turns N and --max-budget-usd X	e.g. --max-turns 10 --max-budget-usd 2.00; enforced by Claude Code itself, unlike a CI timeout
Faster startup, keep chosen standards	--bare + explicitly supply root standards	Skips automatic discovery of hooks, MCP servers, plugins and nested CLAUDE.md
Let the agent read surrounding files	Repository read/search tools (Read, Glob, Grep) and normal behavior	Piping a diff alone limits context
Delegate to subagents	Include Task in allowedTools	See Part VI

2.3 Cross-File Bugs

When bugs span files (a renamed parameter with unchanged callers), a diff-only or changed-files-only prompt cannot see them. Use a turn-limited agentic task with repository read and search tools so the agent can trace references into unchanged files. Independent per-file parallel reviews, bigger prompts of only changed files, or chain-of-thought requests do not provide that access.

2.4 Large Structured Outputs

If a 30-file review returns JSON truncated mid-structure, split the review into several calls over subsets of files and merge the findings arrays. Do not simply raise output limits, retry and drop severities, or abandon structure for Markdown.

Exam takeaway: Custom instructions + default behavior → append. Turns/spend → max-turns + max-budget-usd. Fast start → bare + explicit standards. Project-specific behavior → CLAUDE.md.

Part III — Designing AI Code Review

3.1 Teaching Judgment: Few-Shot Examples

To cut false positives on acceptable style, safe security patterns, and counterproductive performance suggestions while generalizing to novel patterns, provide few-shot examples: annotated snippets that distinguish acceptable patterns from genuine issues. Vague instructions ("be conservative"), keyword filters, or exhaustive documentation do not generalize.

3.2 Accurate but Low-Value Findings

If findings are technically right but half are dismissed as minor, the issue is reportability, not confidence. Define explicit criteria for what to report versus skip. Moving a confidence threshold up or down or adding another agent does not fix it.

3.3 Don't Suppress Early: Tag and Filter Later

A prompt that says to report only issues that definitely cause production failures hides real logic bugs. Instead capture findings with severity and confidence tags in structured output and filter downstream.

3.4 Self-Review Bias

A model reviewing its own refactor in the same session is anchored by its earlier reasoning and is less likely to question its decisions. A fresh-context or separate review is more independent. It is not about disabled tools or token counts.

3.5 Prompt Interference Between Categories

When adding business-logic examples lifts logic recall (34% to 41%) but drops API-design recall (82% to 68%), the objectives compete. Split into focused prompts with dedicated examples, then consolidate. Packing everything into one larger prompt or removing examples is wrong, and higher temperature is irrelevant.

3.6 Generated Test Quality

If ~55% of generated tests are trivial (no-throw assertions, duplicates) and ignore existing fixtures, put testing standards, valuable-test criteria, fixtures, and examples in CLAUDE.md, rather than asking for more tests or removing fixtures.

3.7 Known Subtasks: Prompt Chaining

For style, security and documentation checks that each follow a known three-step workflow, use prompt chaining per aspect followed by a final synthesis. A single giant prompt or a dynamic orchestrator is unnecessary when the subtasks are known.

Exam takeaway: Judgment → few-shot. Low value → define reportability. Don't suppress → tag and filter. Self-review → anchoring. Competing objectives → specialized passes.

Part IV — Tools & MCP

4.1 Choosing the Right Built-in Tool

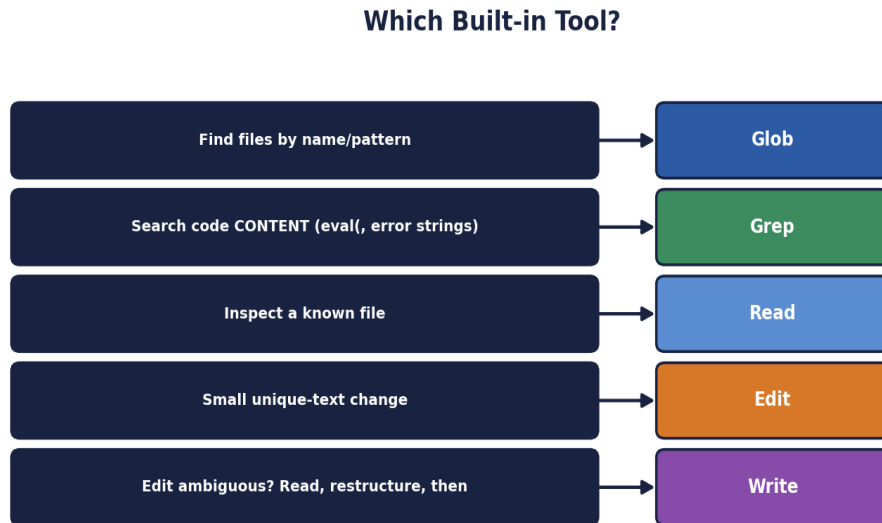


Figure 4.1 — Match the task to the tool.

- Find all occurrences of `eval(` in a codebase: Grep (searches contents; Glob finds paths).
- Distinctive error like `SYNC_CONFLICT` across 12 services: Grep the exact text, then Read the matches.
- Edit cannot find unique text to insert: Read the file, make the structural change, then Write the updated file.

4.2 External Systems: Use MCP

To let an agent read Jira descriptions, acceptance criteria and comments without copy-pasting, integrate an existing Jira MCP server. Do not hand-export, script curl for each ticket, or build a custom server when an integration exists.

4.3 Tool Descriptions Drive Selection

When connected MCP refactoring tools are ignored in favor of Write/Bash/sed and descriptions are minimal, enhance the descriptions to say when each tool is preferable and what inputs and outputs to expect. Removing other tools, adding classifiers, or giving more tools does not help.

4.4 MCP Resources for Discoverability

If an agent makes 8–10 exploratory calls across issue tracker, wiki and database servers because it can't see available content, expose each server's content catalog as MCP Resources. Consolidating servers or hiding catalogs does not solve discoverability.

4.5 Fewer Tools per Agent

Four subagents with all 18 tools often pick the wrong one. Restrict each to about 4–5 relevant tools.

4.6 Purpose-Specific Tools

A single generic `analyze_document` tool with free-text instructions returns narrative when metrics were needed (about 35% re-requested). Split it into purpose-specific tools with defined input and output contracts.

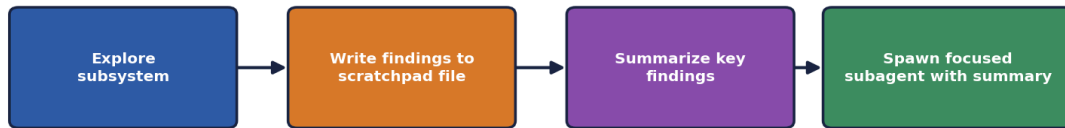
4.7 Tool Choice and Prerequisites

When `lookup_citations` needs the DOI that `extract_metadata` produces, but the model sometimes calls `lookup` first, force `extract_metadata` first, then handle enrichment in later turns with the metadata in hand. Reordering the tools array, auto choice, or a system-prompt hint does not reliably control ordering.

Exam takeaway: Wrong tool chosen → better descriptions and fewer tools. Need discoverability → MCP Resources. External system → MCP. Prerequisite step → force it first.

Part V — Long-Running Work & Context Management

Managing Long-Running Exploration



Signs of degradation: generic answers, inconsistency, repeated re-explaining.
If teammates changed code mid-task, resume AND tell the agent what changed.

Figure 5.1 — Externalize findings, then focus.

5.1 Context Degradation

After 25–30 minutes of exploration, agents drift to generic answers or become inconsistent. Remedies:

- Summarize key findings, then spawn a focused subagent with that summary (do not carry on in the bloated context, and don't discard findings or restart from scratch).
- Maintain a persistent scratchpad file with key findings so context doesn't need to be restated.

5.2 Exploring Large Codebases

For a 15-file, 8,000-line caching layer, go architecture-first: analyze imports and class hierarchies, find the base cache class or interface, and trace invalidation implementations. Reading everything sequentially, grepping a single word, or asking another model to summarize every file is inefficient.

5.3 Resuming Interrupted Work

If a subagent that analyzed 47 files was interrupted and a teammate then merged a rename, resume the previous transcript and inform it about the renamed functions. Don't discard progress, don't resume silently with stale knowledge, and don't revert the teammate's work.

5.4 Crash Recovery

For a pipeline that crashed after 12 of 28 documents: persist structured exports at known locations, keep a manifest, and restore relevant state on resume. A raw conversation log alone, only in-memory state, or an everything-in-vector-store approach is not reliable.

Exam takeaway: Long task → scratchpad. Degrading → summarize + delegate. Interrupted → resume + update changed state. Recovery → manifest + persisted state.

Part VI — Multi-Agent Research Systems

6.1 Workflow Pattern Selection

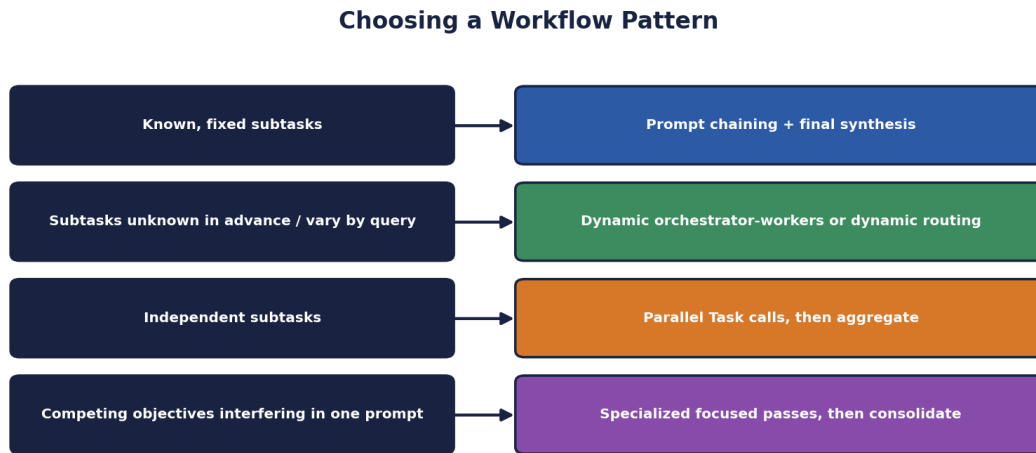


Figure 6.1 — Pattern follows the nature of the work.

6.2 Delegating Effectively

- Delegate by goals and quality criteria, letting the subagent choose its strategy. Overly rigid queries, source priorities, and fallback rules fail when searches are sparse or evidence is emerging.
- New subagents do not inherit coordinator context: include the complete relevant findings in the synthesis prompt. Don't assume inheritance or make the agent re-research.
- Do simple work yourself: if the coordinator already holds the findings, summarize straightforward follow-ups directly and delegate only complex analysis (avoids 40+ second latency from passing 80K+ tokens).

6.3 Making Delegation Actually Run

If four Agent Definitions exist but the coordinator only says "I'll ask the web search agent..." and nothing executes (no errors), the coordinator's allowedTools is missing Task. Definitions describe subagents; Task executes delegation.

6.4 Parallelism

- Independent work (web search and document analysis; 12 precedents in a legal case): the coordinator spawns parallel subagents and aggregates, retaining centralized monitoring and debugging.
- Mechanism: emit multiple Task tool calls in a single response message. A prompt saying "work in parallel" or an external async layer is not it.
- Avoid recursive hierarchies or self-spawning agents when you need visibility.

6.5 Dynamic Routing

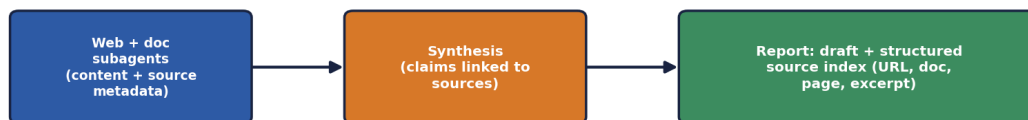
When simple fact questions currently run through all four research subagents but complex comparative questions need several, decide dynamically which subagents to invoke based on the query. Fixed fast paths, keyword routing, or a classifier that needs constant retraining adapt poorly to evolving query distributions.

6.6 Closing Research Gaps

If synthesis reveals three important unanswered questions, evaluate them, re-delegate targeted research, then synthesize again. Don't just list them as limitations, let the report generator guess, or restart everything.

6.7 Provenance and Citations

Research Pipeline: Preserve Provenance



Do not pass the entire raw corpus, and do not pass only source names.

Figure 6.2 — Keep claims traceable without flooding context.

- Pass the synthesis draft plus a structured source index mapping claims to URLs and relevant excerpts (not the whole 135K+ corpus, not just source names, not a later re-search).
- Have each agent output structured content plus separate source metadata (URLs, document names, page numbers) so synthesis doesn't lose them.

6.8 Presenting Different Content Types

Render each type appropriately: financial data as tables, news as prose, patents as structured lists. Flattening everything into bullets or one JSON shape loses meaning.

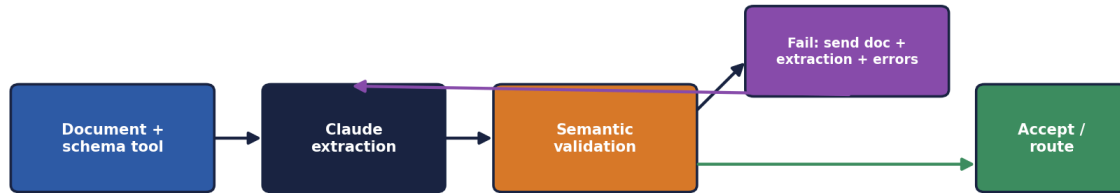
6.9 Conflicting Sources

When credible sources give substantially different estimates (one peer-reviewed with a confidence interval), distinguish established from contested findings and preserve source characterizations and methodology. Don't pick the most recent number, force corroboration by exactly two sources, or collapse into one probability.

Exam takeaway: Goals not scripts. Explicit context for new agents. Task enables delegation. Parallel = multiple Task calls in one message. Conflicts → preserve uncertainty. Gaps → research again.

Part VII — Structured Extraction

Extraction: Schema Tool + Semantic Validation Loop



Schema tool fixes SYNTAX; validation feedback fixes SEMANTICS (sums, IDs, relationships).

Figure 7.1 — Syntax guaranteed by the tool schema; semantics by validation feedback.

7.1 Getting Reliable JSON

For strict output that downstream systems accept (calendar invitations, invalid-JSON problems), define a tool whose input parameters are the target schema and have Claude call it. This is more reliable than prompt wording, regex repair, or retry-until-valid.

7.2 Valid JSON But Wrong Content

- Semantic failures (line items not summing to totals, invalid vendor IDs; 12% failing): send a follow-up request containing the document, the extraction, and the validation errors for correction. More schema constraints, deterministic auto-correction, or blind retries don't address semantic relationships.
- Empty or null required fields (5%) while the source has the information in varied places: add few-shot examples showing extraction from varied document structures. Regex scanning, optional fields with manual review, or retries on empty don't teach where to look.

7.3 Missing Data: Return Null

If the model invents plausible values (attendee_count=500) when the source is silent, instruct it to return null for fields not directly stated. Making every field required and non-null encourages fabrication.

7.4 Growing Taxonomies

For an enum of house/apartment/condo/townhouse that keeps meeting studio, loft, duplex and others, add an 'other' value plus a detail string. This keeps the controlled category stable and preserves the specific value. A free-form string loses control; mapping to nearest values loses information; indefinite growth is unmaintainable.

7.5 Normalization and Ambiguity: Examples

- Inconsistent materials ('cotton blend' vs 'Cotton/Polyester mix', sometimes omitted): few-shot examples with standardized formats.
- Skills extraction with compound phrases, implied skills and granularity issues: few-shot examples showing compound-phrase handling, explicit-mention criteria and granularity, not forced counts or post-hoc rules.

7.6 Context Pressure

A 12-field tool definition (~2,500 tokens) is part of the input. Accuracy dropping from 98% to 71% on 175–190K-token documents in a 200K window reflects total context (system prompt + tool definitions + document) approaching the limit.

Exam takeaway: Strict schema → tool use. Semantic error → validation feedback. Info exists but varies → few-shot. Not stated → null. Growing taxonomy → other + detail.

Part VIII — Quality Control & Human Review

Prioritizing Scarce Human Review

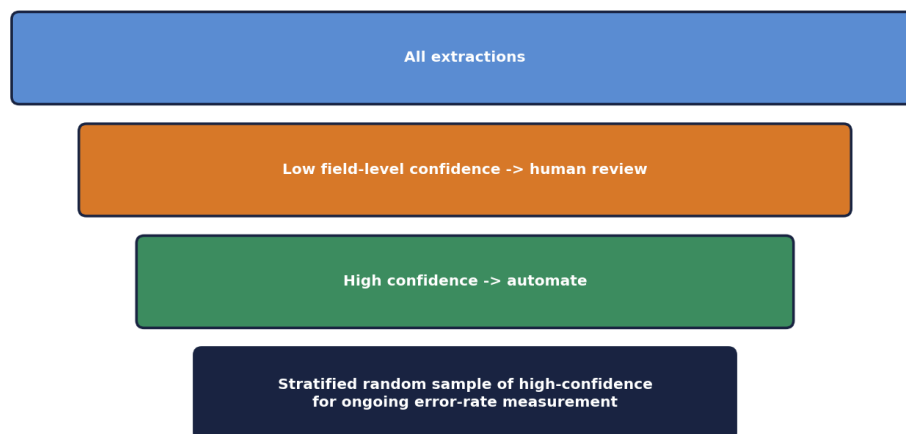


Figure 8.1 — Spend human attention where errors are likely, and keep measuring.

8.1 Limited Review Capacity

If ~12% of semantically valid outputs contain errors and reviewers can cover only 20%, use field-level confidence scores with calibrated thresholds from a labeled validation set. Random sampling without confidence, or reviewing only missing fields or formatting issues, misses likely errors.

8.2 Before Automating High-Confidence Results

Aggregate accuracy (e.g., 97% for confidence $\geq 90\%$) can hide weak segments. The most critical validation is analyzing accuracy by document type and field to confirm confidence is reliable across segments.

8.3 Ongoing Detection of Plausible Errors

When a quarterly audit shows 12% of high-confidence results are plausibly wrong (comparison tables, appendices, ambiguous phrasing), use stratified random sampling of high-confidence extractions for regular review. It measures the true error rate over time and surfaces new error patterns. Heuristic flags, double extraction, or lowering the threshold are narrower or costlier.

Exam takeaway: Limited review → calibrated confidence. Aggregate accuracy → segment-level validation. Unknown errors + measurement → stratified sampling.

Part IX — Answer Patterns & Cheat Sheet

9.1 Recurring Patterns

- Match the mechanism to the real constraint: latency tolerance, SLA, prerequisite dependency, context size.
- Put persistent knowledge in CLAUDE.md; put judgment in few-shot examples; put structure in tool schemas.
- Fix at the source: better tool descriptions, fewer tools, explicit context, rather than bolting on classifiers or filters.
- Preserve information: tag and filter later, keep provenance, preserve uncertainty, return nul instead of guessing.
- Measure honestly: segment validation and stratified sampling, not just aggregate numbers.
- Prefer targeted recovery: retry only failures, resume with updates, re-research only gaps.
- Extremes are usually wrong: 'always', 'never', 'start from scratch', 'remove everything'.

9.2 One-Page Cheat Sheet

Situation	Answer
Async + latency tolerant, cost matters	Message Batches API
Urgent 30-minute SLA alongside bulk work	Batch for bulk, real-time for urgent
Context-length failures on a few batch items	Chunk and resubmit only those IDs
False positives from project conventions	Document in CLAUDE.md
Custom prompt but keep default agent behavior	--append-system-prompt
Cap turns and dollars	--max-turns + --max-budget-usd
Faster Claude Code startup	--bare + explicit standards
Cross-file dependencies	Agentic task with repo read/search
Truncated large JSON output	Split review, merge findings
Generalize review judgment	Few-shot annotated examples
Accurate but low-value	Define reportability criteria
Prompt hides real bugs	Tag severity/confidence, filter downstream
Self-review misses bugs	Anchoring; use fresh independent review
Competing objectives	Specialized passes, consolidate
Find code text / error string	Grep
Edit ambiguity	Read + Write
External system access	MCP server
Tools ignored	Improve descriptions
Can't discover content	MCP Resources
Wrong tool selection	Restrict to 4–5 tools
Generic tool returns wrong shape	Purpose-specific tools with contracts
Tool depends on another's output	Force prerequisite first

Long exploration degrading	Scratchpad; summarize + delegate
Interrupted subagent, code changed	Resume transcript + tell it what changed
Pipeline crash	Persisted exports + manifest
Known fixed subtasks	Prompt chaining + synthesis
Varied queries	Dynamic routing / orchestrator
Independent subtasks	Multiple Task calls in one message
Subagent never runs	Add Task to allowedTools
New subagent needs data	Pass it explicitly in the prompt
Citations	Structured source index / provenance
Conflicting sources	Preserve uncertainty and methodology
Research gaps	Re-delegate targeted research
Strict structured output	Tool with schema
Semantic extraction errors	Feedback loop with errors
Missing source data	Return null
Growing enum	'other' + detail string
Scarce reviewers	Calibrated field-level confidence
Automating high-confidence	Segment-level accuracy check
Unknown error rate over time	Stratified random sampling

Good luck on the Claude Certified Architect Foundations exam.