From Chatbots to Coordinated Agents: A Construction Management (CM) Taxonomy of Agentic AI Design Patterns

Reihaneh Samsami, Ph.D., PE*
Assistant Professor and MSCM Program Coordinator
Department of Civil Engineering
University of New Haven, 300 Boston Post Road, West Haven, CT 06516
Email: rsamsami@newhaven.edu

Total Number of Pages: 18

Submitted 7/30/2026

*Corresponding Author

ABSTRACT

Objectives. Construction management (CM) tools are increasingly marketed as “agentic”, but the term covers systems with very different levels of autonomy and reliability, leaving CM organizations without a shared vocabulary for evaluating what a tool does or where it may fail. This paper translates a general-purpose taxonomy of agentic artificial intelligence (AI) design patterns into construction-specific vocabulary so practitioners can reason about pattern selection deliberately rather than adopting agentic tooling as a black box.

Methods. Twenty patterns were extracted from a practitioner reference on agentic system architecture and mapped onto recurring CM information workflows, each paired with a brief illustrative case grounded in transportation construction. One pattern, Retrieval, was validated empirically using a synthetic bridge rehabilitation specification corpus (16 base sections, 6 addenda) and a 24-question test set (stable, addendum-affected, and distractor types), evaluated across three indexing conditions and two chunking strategies.

Findings. The twenty patterns were organized into four functional groups, each illustrated with an original architecture diagram. The retrieval case study found that an index built only from original specifications answered 0 percent of addendum-affected questions correctly; adding current addenda raised this to 50 percent, and supersession-aware re-ranking raised it to 100 percent but reduced distractor accuracy from 87.5 to 62.5 percent, showing that grounding requires more than re-indexing alone.

Novelty. This is the first systematic translation of the general-purpose agentic AI design pattern literature into CM vocabulary, and the first empirical test of the Retrieval pattern's addenda-grounding claim against a working pipeline.

Practical Applications. The taxonomy gives CM digital transformation leads and procurement teams a common language for evaluating vendor claims, while the case study offers a reproducible method for auditing whether a retrieval tool is actually grounded in current project documents.

INTRODUCTION

Software vendors serving the architecture, engineering, and construction industry (AEC) now routinely describe their products as agentic. The term has diffused quickly enough that it risks becoming a marketing label detached from any specific technical claim, applied equally to a rule-based workflow automation, a single large language model (LLM) call with function calling enabled, and a genuinely autonomous multi agent system that plans, monitors, and re-plans without human intervention. Construction management (CM) organizations evaluating this tooling, and researchers studying its effects, currently lack a shared, domain-specific vocabulary for distinguishing these architectures from one another.

The word agentic is worth defining before going further. A chatbot, in the sense popularized by the public release of ChatGPT in late 2022, is a single-turn responder: it returns a fluent completion with no persistent memory, no ability to call external tools, and no capacity to check its own output. Generative AI, the broader category a chatbot belongs to, supplied the underlying capability, fluent text generation, but on its own is not agentic in any meaningful sense. An AI agent adds a narrow set of further capabilities to a single model, typically tool use and some form of planning, so it can look something up or take a bounded action rather than only generating text. Agentic AI, the term this paper is concerned with, describes a further step: multiple such agents, or a single agent with a fuller set of these capabilities, coordinating across steps, tools, and sometimes other agents to pursue a goal with limited human intervention (Sapkota et al. 2025). Within construction specifically, a companion framework organizes this same autonomy spectrum, perceptual, cognitive, and embodied agents, against the construction problem each is applied to, finding that the most capable deployments combine more than one agent type (Samsami 2026). This paper sits at the cognitive, coordinated end of that spectrum, tracing the same progression its title describes, from chatbots to coordinated agents, rather than the perceptual or embodied agents that sense or act physically on site.

Outside construction, the general software engineering community has begun to formalize this vocabulary. Gullí (2025) catalogues twenty recurring architectural patterns observed across production agentic systems, from simple prompt chaining to multi agent orchestration, retrieval augmented grounding, and formal safety guardrails, providing a reference structure analogous to the classic software design pattern literature. These patterns are framework agnostic and domain neutral by design; they describe structural roles, such as a routing step, a critique step, or a memory store, rather than any particular construction, healthcare, or finance implementation.

This paper makes four contributions. First, it translates twenty of these patterns into CM vocabulary, pairing each domain neutral definition with an original architecture diagram and a brief illustrative case grounded in transportation construction. Second, it organizes the twenty patterns into four functional groups relevant to CM decision making, (1) Foundational Workflow, (2) Multi Agent and Knowledge, (3) Governance and Reliability, and (4) Optimization and Reasoning, so that practitioners can reason about pattern selection at the category level before drilling into specifics. Third, it discusses failure modes specific to construction's risk profile: contractual liability, life safety consequences, fragmented multi-party data, low tolerance for silent failure, and audit and discovery requirements that do not apply with the same force in most other software domains. Fourth, and central to this paper's contribution, it moves beyond narrative description for one pattern, Retrieval, building a working retrieval pipeline and testing it against a synthetic bridge rehabilitation specification corpus to quantify, rather than only assert, how grounding failures occur and what mitigates them.

The intended audience is CM researchers, digital transformation and innovation leads at AEC organizations, and technical staff evaluating or building agentic tools for project delivery. The paper does not evaluate specific commercial products; it is deliberately architecture focused, so that its framework remains useful as the underlying vendor landscape changes. All case studies presented in the section on the twenty patterns are illustrative scenarios constructed by the author to demonstrate how each pattern's structural role maps onto a recognizable CM workflow. They are not reports of validated field deployments, and they should be read as examples rather than empirical findings; the Method section, presented next, explains this distinction and the reasoning behind it in more detail.

BACKGROUND AND RELATED WORK

AI adoption in CM has accelerated markedly since 2020. A systematic review of 191 peer reviewed articles published between 2020 and 2025 found AI applications concentrated in risk and safety management, decision support, and monitoring and control, while legal analytics, robotics, and cybersecurity applications remain comparatively underexplored, with computer vision and deep learning dominating safety monitoring and defect detection tasks specifically (Razi et al. 2026). A separate review of the CM project management literature similarly documents growing use of machine learning (ML), computer vision (CV), and natural language processing (NLP) across scheduling, cost, and quality functions, while identifying the black box nature of many AI algorithms as a

persistent barrier to accountability in safety-critical decisions (Savaş 2025). Earlier foundational work by Abioye et al. (2021) had already flagged data fragmentation, workforce upskilling, and organizational culture as first-order barriers to AI adoption in construction, barriers that agentic architectures do not resolve automatically and, in some cases, sharpen. Explainability, liability, and the ethical boundaries of delegating engineering judgment to an AI system are a related and still largely unresolved concern specific to civil engineering practice, where a wrong or unexplainable recommendation can carry professional liability that a generic AI governance framework does not fully anticipate (Badhan and Samsami 2026).

Market data suggest the pace of adoption is real even where realized value lags expectations. Industry estimates place the global AI in construction market at roughly $4.86 billion in 2025, with substantial projected growth through the end of the decade (RTS Labs 2026). At the same time, industry sentiment surveys show a more cautious picture specifically for agentic systems. Autodesk's 2025 State of Design and Make report found construction sector sentiment toward AI declined as organizations moved from experimentation to attempted production deployment, a pattern consistent with the classic technology hype cycle, in which the gap between marketed capability and realized value becomes most visible during early production attempts (World Construction Network 2026).

This paper is distinct from existing CM AI review literature in scope. Rather than cataloguing AI techniques by function, such as CV for progress monitoring or NLP for document review, it catalogues architectural patterns, the structural roles that any of those techniques can be assembled into, such as a routing step, a critique loop, or a shared multi agent workspace. This distinction matters because the same underlying model can be deployed as a single shot classifier or as part of a reflective, multi agent, tool using system with materially different reliability and failure characteristics. Existing CM AI literature, including the systematic reviews cited above, has not systematically mapped the general-purpose agentic pattern literature onto construction workflows.

Finally, because several of the patterns discussed below concern autonomy, delegation, and safety constraints, this paper draws on the emerging agent governance literature outside construction, including Kolt's (2025) treatment of legal and organizational structures for governing autonomous AI agents and Kraprayoon et al.'s (2025) field guide to agent governance practices, both of which inform the discussion of guardrails and human in the loop design below.

The twenty patterns catalogued below also have an independent lineage in the broader agentic AI literature that predates and informs Gullí's (2025) synthesis, cited directly below alongside Gullí where relevant. Prompt-level reasoning and tool use trace to chain of thought prompting (Wei et al. 2022; Kojima et al. 2022), interleaved reasoning-and-acting (Yao et al. 2023b), and self-supervised tool use (Schick et al. 2023). Reflection builds on verbal reinforcement learning (Shinn et al. 2023), and deliberate multi-path reasoning draws on tree of thought search (Yao et al. 2023a), self-consistency (Wang et al. 2022), and multiagent debate (Du et al. 2023). Retrieval grounding follows the original RAG formulation (Lewis et al. 2020), and persistent agent memory was demonstrated at scale by Park et al. (2023). Multi-agent orchestration draws on frameworks such as AutoGen (Wu et al. 2023), and guardrail patterns respond to documented prompt-injection vulnerabilities (Greshake et al. 2023). Governance-oriented patterns draw on recent agent-oversight literature (Kolt 2025; Kraprayoon et al. 2025; Raza et al. 2025; Kirchhof et al. 2025; Hughes et al. 2025), situated within a consolidating survey literature on LLM-based autonomous agents (Wang et al. 2024).

METHOD: TRANSLATING GENERAL-PURPOSE PATTERNS TO THE CM DOMAIN

The translation followed four steps. First, pattern definitions were extracted from Gullí (2025) at the level of structural function, the role a pattern plays in an agentic system, rather than at the level of specific code implementation, since implementation detail is framework and vendor specific and outside this paper's scope. Second, each pattern's structural function was mapped onto common CM information workflows: requests for information (RFI), submittal review, change order processing, punch list closeout, daily field reporting, safety observation, cost and schedule updates, and building information modeling (BIM) coordination. These workflow categories were selected because they recur across nearly all commercial and public sector building and transportation construction projects regardless of delivery method, and because they are the workflows most frequently cited as AI application targets in the CM literature reviewed in the Background and Related Work section above.

Third, for each pattern the author constructed an illustrative case study: a short, hypothetical scenario, grounded in domain knowledge of common CM workflows, that demonstrates how the pattern's structural strength addresses a documented pain point in that workflow. These case studies are conceptual illustrations rather than reports of implemented or empirically evaluated systems, and they are presented as such throughout. This framing is intentional and follows directly from the paper's stated contribution: it translates architecture, not deployment

results, and no claim of measured performance, adoption, or outcome is made for any illustrative case. Fourth, failure modes were evaluated against constraints that distinguish construction from the general software domains in which the source patterns were originally documented: contractual liability and potential litigation exposure, life safety consequences, data fragmentation across many independent contracting parties, low organizational tolerance for silent failure given the cost of field rework, and the audit and inspection requirements attached to project records.

Patterns are grouped into four functional categories to support category-level reasoning about pattern selection: (1) Foundational Workflow patterns govern task decomposition and execution, (2) Multi Agent and Knowledge patterns govern coordination and grounding, (3) Governance and Reliability patterns govern accountability and safety, and (4) Optimization and Reasoning patterns govern how compute and reasoning depth are matched to task difficulty. For each pattern, an original diagram was produced to depict its structural role independent of any specific product or vendor implementation. Table 1 summarizes all twenty patterns with their group and a one-line illustrative example for quick reference; full definitions, diagrams, and case study discussion follow in the next section.

TABLE 1 Overview of the Twenty Agentic Design Patterns Translated for CM

#	Group	Pattern	Illustrative CM Example
1	Foundational Workflow Patterns	Prompt Chaining	Submittal review broken into extract, check, draft, and route-for-signature steps
2		Routing	RFI intake classified by discipline and urgency and dispatched to the right specialist agent
3		Parallelization	Photos, sensor feeds, and daily logs processed concurrently, then merged into one progress report
4		Reflection	Change order narrative self-critiqued against contract language before reaching the PM
5		Tool Use	Scheduling agent queries the live schedule instead of guessing critical path impact
6		Planning	Punch list closeout decomposed into per-trade, dependency-aware sub-plans before a walkthrough
7	Multi-Agent and Knowledge Patterns	Multi-Agent Collaboration	Manager, structural, MEP, and report-writer agents jointly produce one clash report
8		Memory Management	Safety agent recalls a site's recurring hazard history during a new walkthrough
9		Learning and Adaptation	RFI-drafting agent's prompts improve as PM edit patterns are fed back over the project
10		Retrieval (RAG)	Specifications question answering grounded in a project's actual manual and latest addenda
11		Inter-Agent Communication	Subcontractor's and GC's scheduling agents exchange structured availability messages
12	Governance and Reliability Patterns	Goal Setting and Monitoring	Bridge erection agent flags schedule drift in days of float consumed, not just behind
13		Exception Handling and Recovery	Permit-status agent retries a timeout and escalates an unrecognized status instead of guessing
14		Human-in-the-Loop	Recordable-injury reports always route to a human safety manager before logging
15		Evaluation and Monitoring	Submittal agent evaluated weekly against a golden set of known-correct adjudications
16		Guardrails and Safety	Field-documentation agent sandboxed against injected instructions in uploaded files
17	Optimization and Reasoning Patterns	Resource-Aware Optimization	A cheap model triages routine RFIs while a larger model is reserved for high-dollar cases
18		Reasoning Techniques	A self-consistency check flags a load path question when reasoning paths disagree
19		Prioritization	Open-issue backlog re-ranked daily by schedule impact, cost, safety relevance, and staleness

#	Group	Pattern	Illustrative CM Example
20		Exploration and Discovery	Years of closed RFIs and change orders are clustered to surface an unnoticed recurring gap

THE TWENTY PATTERNS

The subsections below present the twenty patterns in the four functional groups introduced in the Method section above.

Foundational Workflow Patterns

Foundational workflow patterns form the base layer of agentic system design: the basic mechanisms by which a task is broken apart, routed, executed, and checked, independent of any higher-level coordination or governance concern. These six patterns govern how a single task is decomposed, dispatched, executed, and checked. They are the building blocks that nearly every other pattern in this taxonomy composes with, and they are typically the first patterns a CM organization implements because they map most directly onto existing, well understood workflows.

1. Prompt Chaining

Prompt chaining decomposes a complex task into a sequence of discrete model calls, where the validated output of one step becomes the input to the next, rather than asking a single call to do everything at once (Gullí 2025). The approach echoes a broader lesson from the prompting literature: LLMs perform more reliably on multi-step problems when intermediate reasoning is made explicit and checkable rather than compressed into one pass (Wei et al. 2022). **Figure 1** shows the four-step submittal chain used on the bridge deck rehabilitation contract. On a bridge deck rehabilitation submittal, such a chain extracts spec references, cross-checks them against current addenda, drafts numbered reviewer comments, and routes the draft for engineer sign-off, so a misread section reference is caught before it reaches a formal response. Chains are only as reliable as their weakest, unvalidated link and add latency, so they suit staged review workflows better than time-critical field questions.

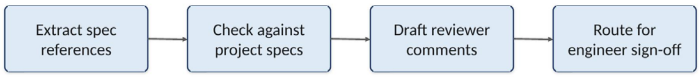

Figure 1. Prompt Chaining. A four-step submittal-review chain: spec references are extracted, checked against current specifications, drafted into reviewer comments, and routed for engineer sign-off, with each step validated before the next begins.

2. Routing

Routing uses a gating step to examine an incoming request and dispatch it to the specialist agent, model, or workflow best suited to handle it, rather than sending everything through one generalist pipeline (Gullí 2025). **Figure 2** shows how an incoming RFI on the interchange project is classified and routed to the right specialist agent. On a design-build interchange project, incoming RFIs can be classified by discipline and urgency and dispatched to a structural, drainage, or fast-path queue accordingly, keeping a schedule-impacting RFI from sitting in a slow queue over a weekend. Because misrouting carries asymmetric cost, low-confidence classifications should default to human triage rather than to whichever agent scored highest.

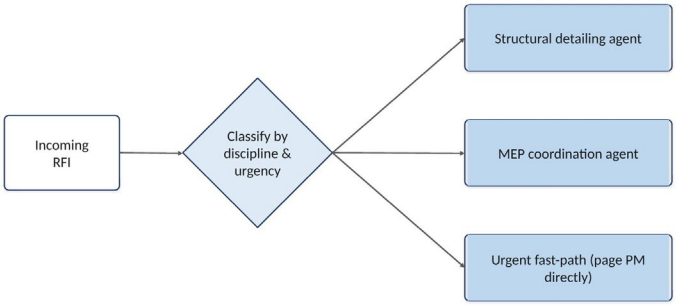

Figure 2. Routing. An incoming RFI is classified by discipline and urgency, then dispatched to a structural agent, an MEP coordination agent, or a fast-path queue that pages the project manager directly.

3. Parallelization

Parallelization splits independent subtasks off for concurrent execution rather than sequential processing, then merges their outputs (Gullí 2025). **Figure 3** shows the three data sources merged each evening into one reconciled progress report for the reconstruction corridor. Daily progress reporting on a reconstruction corridor can run photo-based computer vision, text-log parsing, and equipment telemetry concurrently, merging them into one reconciled report each morning rather than requiring manual reconciliation late in the evening. Merging is the hard part: disagreements between sources need an explicit conflict-resolution rule rather than silent averaging, which can mask discrepancies that matter for pay-application accuracy.

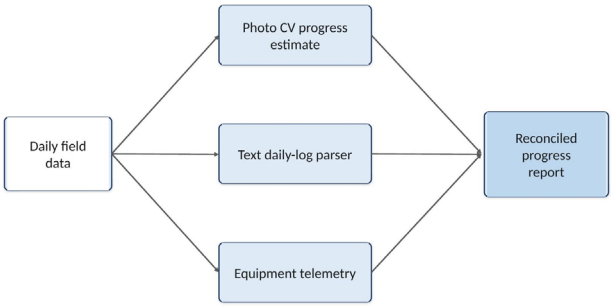

Figure 3. Parallelization. Daily field data is split into three concurrent branches, photo-based computer vision, text log parsing, and equipment telemetry, then merged into one reconciled progress report.

4. Reflection

Reflection follows a generation step with a distinct critique step, in which the same or a different model reviews the output against defined criteria before it is finalized and revises it if it fails, an approach closely related to verbal self-reinforcement frameworks in which an agent critiques its own trial and carries that critique forward as episodic feedback for the next attempt (Shinn et al. 2023; Gullí 2025). **Figure 4** shows the critic step that gates the trunnion-girder change-order narrative before it reaches the project manager. On a movable bridge rehabilitation contract, a critic pass can check a change-order narrative's causation claim, cost reconciliation, and schedule consistency against the contract's changed-conditions clause before it reaches the project manager, catching an unreconciled figure that would otherwise undermine the claim's credibility. Reflection roughly doubles cost and latency per gated output, so it is best reserved for artifacts with real consequence, with a hard cap on revision cycles.

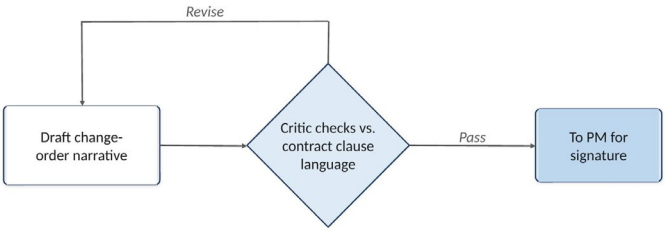

Figure 4. Reflection. A drafted change-order narrative is checked against contract clause language by a critic step; failing drafts loop back for revision, and only a passing draft reaches the project manager for signature.

5. Tool Use

Tool use gives an agent a defined set of external functions or application programming interfaces (APIs) it can call, such as a calculator, a database query, or scheduling software, and lets it decide when and how to invoke them rather than answering everything from generated text alone, an ability demonstrated at scale by self-supervised tool-calling models (Schick et al. 2023) and by frameworks that interleave reasoning with tool invocation so that actions and thoughts inform each other step by step (Yao et al. 2023b; Gullí 2025). **Figure 5** shows the scheduling agent querying the live schedule, cost database, and weather data before answering the widening project's delay question. On a highway widening project, a scheduling agent with tool access to the live schedule, cost database, and weather data can return a grounded delay estimate rather than a fabricated one when asked what a supplier delay does to substantial completion. Tool access needs explicit authorization scoping and a defined fallback for tool failures, rather than letting the agent guess when a call fails.

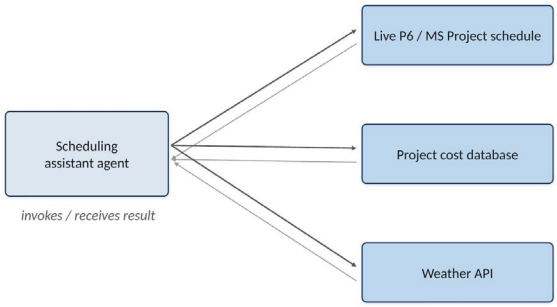

Figure 5. Tool Use. A scheduling assistant agent calls three external tools, the live project schedule, the cost database, and a weather API, rather than generating an unverified estimate.

6. Planning

Planning takes a high-level goal and decomposes it into an ordered set of sub-goals, milestones, and dependencies before execution begins, revising the plan as new information arrives (Gullí 2025). **Figure 6** shows the punch-list goal decomposed into per-trade sub-plans ahead of the lane reopening. On a bridge deck replacement ahead of a lane reopening, a punch-list goal can be decomposed into per-trade sub-plans sequenced against trade availability and inspection windows, automatically re-planning when a delivery delay is reported. Plans built on stale data produce confident-looking schedules that are wrong, so planning agents should expose their dependency assumptions for a human to catch a bad one.

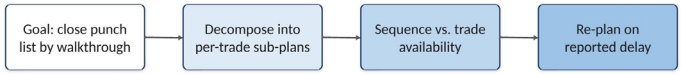

Figure 6. Planning. A high-level punch-list goal is decomposed into per-trade sub-plans, sequenced against trade availability, and automatically re-planned when a trade reports a delay.

Multi-Agent and Knowledge Patterns

Multi-agent and knowledge patterns sit one layer above the foundational patterns: rather than governing how a single task executes, they govern how several agents divide labor and how any one of them stays grounded in accurate information rather than relying on what a language model already happens to know. These five patterns govern how multiple agents coordinate and how a system stays grounded in accurate, current, project specific information rather than generic parametric knowledge, a distinction that matters enormously in construction, where the governing document is always the specific contract rather than typical industry practice.

7. Multi-Agent Collaboration

Multi-agent collaboration assigns bounded pieces of a larger task to multiple specialized agents, often coordinated by a manager or orchestrator agent, rather than asking one generalist agent to attempt everything, an architecture popularized by open-source conversational frameworks that let multiple customizable agents converse with each other and with tools to accomplish a shared task (Wu et al. 2023; Gullí 2025). **Figure 7** shows the manager agent coordinating the structural-clash, MEP-clash, and report-writer agents on the light rail station's federated model. On a light rail station's federated BIM model, a manager agent can coordinate structural-clash, MEP-clash, and report-writer agents, compiling one prioritized clash report rather than requiring manual merging of separate detection passes. Without clear role boundaries, agents duplicate work or contradict each other, so authority should mirror the accountable-party structure already in place, with cross-discipline conflicts escalated rather than auto-resolved.

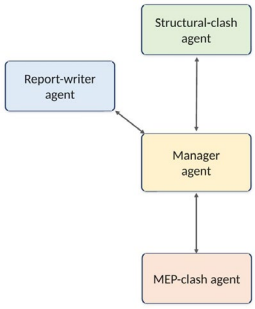

Figure 7. Multi-Agent Collaboration. A manager agent coordinates three bounded specialist agents, structural-clash, MEP-clash, and report-writer, each handling one piece of a BIM coordination task.

8. Memory Management

Memory management addresses what an agent retains across a single task, described as short-term or working memory, across a session, described as episodic memory, and across the life of a project or organization, described as long-term memory, along with how relevant memories are retrieved when needed, an architecture first demonstrated at scale in generative agents that combine a continuously updated observation stream with periodic reflection and retrieval to inform later behavior (Park et al. 2023; Gullí 2025). **Figure 8** shows the site-safety agent drawing on episodic and long-term memory of the recurring taper hazard. On an interstate widening corridor, a site-safety agent can hold short-term, episodic, and long-term memory of a site's recurring hazards, surfacing a pattern like a recurring lane-shift taper citation unprompted during a walkthrough rather than treating each visit as a blank slate. Long-term memory that is not scoped and auditable becomes a liability, since safety records can be subject to discovery in claims and litigation.

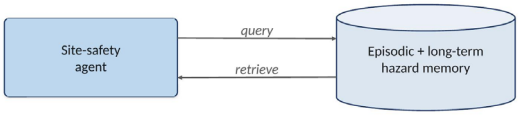

Figure 8. Memory Management. A site-safety agent draws on an episodic and long-term hazard-memory store to recall recurring site-specific issues rather than treating each interaction independently.

9. Learning and Adaptation

Learning and adaptation lets a system improve over time by incorporating feedback, such as corrected outputs, human overrides, or outcome data, into updated prompts, retrieval indexes, fine-tuned policies, or evaluation sets, rather than remaining static after deployment (Gullí 2025). **Figure 9** shows resident-engineer edits to shoring RFIs feeding back into the agent's prompt library over the program. On a multi-structure bridge replacement program, an RFI-drafting agent can track which drafts a resident engineer edits heavily and feed that feedback into its prompt library, improving accuracy on categories like shoring questions over the life of the program. Adaptation loops that learn from a small or biased feedback sample can overfit to individual preference rather than genuinely improving accuracy for the broader team.

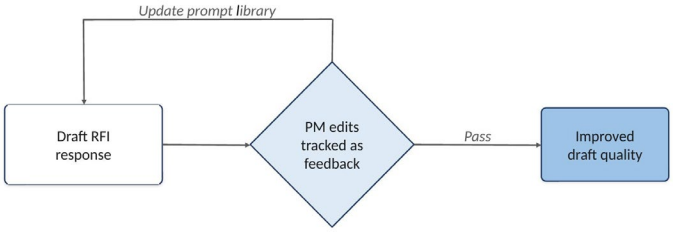

Figure 9. Learning and Adaptation. Project-manager edits to drafted RFI responses feed back into the agent's prompt library, closing a loop that improves draft quality over time rather than leaving it fixed at deployment.

10. Retrieval (RAG)

Retrieval, commonly abbreviated RAG for retrieval-augmented generation, parses, chunks, embeds, and indexes a document corpus, then retrieves and reranks the most relevant passages to ground a generated answer, connecting the model to authoritative, current project documents rather than relying on parametric knowledge alone, following the original formulation that pairs a pretrained generator with a non-parametric, updatable index of source

documents (Lewis et al. 2020; Gullí 2025), with a recent controlled empirical test of exactly this grounding question in a transportation-construction context finding that both retrieval-augmented and full-document context outperform closed-book prompting on regulatory question answering across CTDOT, OSHA, and FHWA source documents (Samsami and Sayed 2026). **Figure 10** shows the specifications agent retrieving the addendum-updated cure-time clause for the station platform pour. On a design-build light rail extension, a specifications agent grounded in the current, addenda-updated project manual can return a revised cure-time requirement rather than a superseded figure that would understate it by several days. Retrieval is only as current as its index; the case study presented later in this paper builds a working retrieval pipeline on a synthetic version of exactly this scenario and measures how often a stale index returns superseded language.

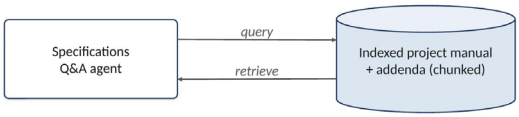

Figure 10. RAG. A specifications question-answering agent queries an indexed, chunked store of the project manual and addenda and retrieves the relevant passage before answering, rather than relying on general knowledge.

11. Inter-Agent Communication

Inter-agent communication relies on formal protocols, including message schemas, agent identifiers, conversation or session identifiers, and message expiry, that let agents from different systems or vendors exchange structured requests and results reliably rather than relying on ad hoc text parsing between systems, an approach conceptually related to open multi-agent conversation frameworks that formalize how separate agent instances exchange messages and hand off tasks (Wu et al. 2023; Gullí 2025). **Figure 11** shows the structured message exchange between the steel erection subcontractor's agent and the general contractor's master-schedule agent. On a design-build bridge replacement, structured messages carrying identifiers and expiry timestamps let a steel erection subcontractor's agent and the general contractor's master-schedule agent exchange availability twice weekly, catching a girder-sequencing conflict well before the normal weekly coordination call. Because cross-organizational agent communication crosses contractual and liability boundaries, a subcontractor's agent should never silently commit the general contractor's schedule of record without human confirmation.

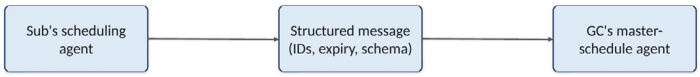

Figure 11. Inter-Agent Communication. A subcontractor's scheduling agent and a general contractor's master-schedule agent exchange a structured message carrying identifiers and an expiry, rather than free-text correspondence.

Governance and Reliability Patterns

Governance and reliability patterns address a different concern altogether: not whether a system can complete a task, but whether its output can be trusted, audited, and safely bounded once it does. These five patterns exist to keep agentic systems accountable, auditable, and safely bounded, properties that matter more in construction than in most software domains given the contractual liability, life safety consequences, and inspection and audit requirements attached to project records.

12. Goal Setting and Monitoring

Goal setting and monitoring expresses an agent's objective as measurable key performance indicators (KPIs) with target thresholds, and continuously monitors actual performance against those targets, flagging drift so that the goal, or the plan to reach it, can be corrected (Gullí 2025). **Figure 12** shows the days-of-float-consumed trend that triggers a correction during segment erection. On a segmental bridge project, an agent can track a quantified days-of-float-consumed figure daily rather than a generic behind-schedule flag, giving the project manager time to add a second launching-gantry shift while the float deficit is still recoverable. Goals set as a single scalar target without constraint context can drive an agent toward locally improving the metric in ways that increase crew fatigue or fall-hazard exposure, so a well-designed monitoring agent should surface the tradeoff, not just the recommendation.

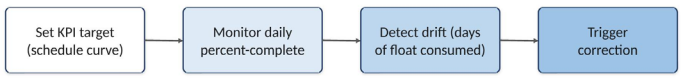

Figure 12. Goal Setting and Monitoring. A schedule KPI is set, monitored daily against the baseline curve, checked for drift in days of float consumed, and used to trigger a correction before the milestone is missed.

13. Exception Handling and Recovery

Exception handling and recovery classifies failures, distinguishing a tool timeout from a genuinely unanswerable question from a low-confidence result, applies an appropriate recovery strategy such as retry with backoff, fallback to a simpler method, or escalation to a human, and does not treat every failure identically (Gullí 2025). **Figure 13** shows how a permit-portal timeout is classified and escalated rather than guessed. On a roadway resurfacing project, an agent checking a permitting portal can retry a timeout with backoff and escalate an unrecognized status code to a human by phone rather than guessing, giving the superintendent two days' notice instead of discovering a permit hold the morning milling was set to begin. The most damaging exception-handling failure in CM is silent degradation, where a confidently wrong status report is worse than a visible error.

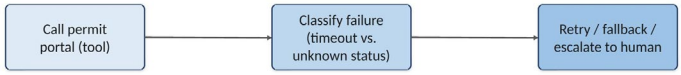

Figure 13. Exception Handling and Recovery. A tool call to a permit portal is classified by failure type, a timeout versus an unrecognized status, before the system retries, falls back, or escalates to a human.

14. Human-in-the-Loop

Human-in-the-loop design creates defined checkpoints where a human reviews, edits, or approves agent output before it takes effect, with the review trigger tied to risk level, confidence score, or contractual requirement rather than applied uniformly to everything, an approach consistent with emerging governance guidance calling for oversight calibrated to an agent's actual level of consequential autonomy rather than applied as a blanket rule (Kraprayoon et al. 2025; Gullí 2025). **Figure 14** shows the risk-based gate that routes tunnel boring incident reports to automatic posting or mandatory safety-manager review. On a tunnel boring project, incident reports involving a recordable injury, a high-severity near-miss, or low model confidence can route to mandatory safety-manager review, while routine housekeeping observations post directly, concentrating limited review time where misclassification would matter most. Two opposite failure modes threaten this pattern, review fatigue from overly broad triggers and missed edge cases from overly narrow ones, so the trigger logic itself needs periodic audit.

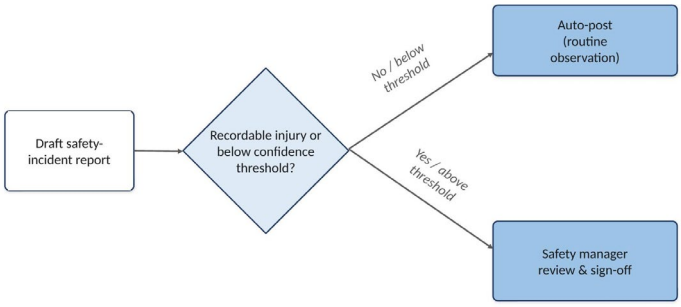

Figure 14. Human-in-the-Loop. A drafted safety-incident report is gated on injury severity and confidence; low-risk reports post automatically while high-risk or low-confidence reports route to a safety manager for sign-off.

15. Evaluation and Monitoring

Evaluation and monitoring measures system performance on an ongoing basis against a curated set of known-correct golden test cases, service-level targets, and drift detection, treating evaluation as a continuous production discipline rather than a one-time pre-deployment check, consistent with calls in the emerging agentic-systems literature to treat agent oversight as an ongoing operational function rather than a static launch gate (Hughes et al. 2025; Gullí 2025). **Figure 15** shows the weekly golden-set check that catches the structural submittal accuracy drop on the interchange program. On an interstate interchange design-build program, a submittal-review agent evaluated weekly against a golden set of known-correct cases can catch a structural accuracy drop, traced to an un-incorporated spec revision, before a reviewer complaint surfaces it. Monitoring only aggregate accuracy can hide a system that performs well on routine cases but fails badly on rare, high-consequence ones, so evaluation should weight by consequence rather than by frequency alone.

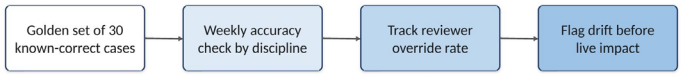

Figure 15. Evaluation and Monitoring. A submittal-review agent is checked weekly against a golden set of known-correct cases and a tracked override rate, so accuracy drift is flagged before it affects a live decision.

16. Guardrails and Safety

Guardrails and safety impose explicit constraints, including content filters, sensitive-data handling rules, prompt-injection defenses, and sandboxed execution, that bound what a system can output or do, independent of whether the underlying model wants to do something unsafe, a necessity underscored by documented demonstrations that retrieval-augmented, tool-using systems can be indirectly manipulated through instructions hidden inside the very documents or data they are asked to process (Greshake et al. 2023; Kolt 2025; Gullí 2025). **Figure 16** shows the sandbox screening step that isolates the embedded instruction found in an uploaded bridge-site photo. On a river-crossing bridge project, an agent ingesting daily reports and photos from a dozen subcontractors can be sandboxed so that an embedded instruction hidden in a photo's caption field is logged as an anomaly rather than acted on. Because construction data flows through many external parties uploading files from a wide range of devices, the injection surface is wide, and filters need domain-aware tuning rather than generic consumer-application defaults.

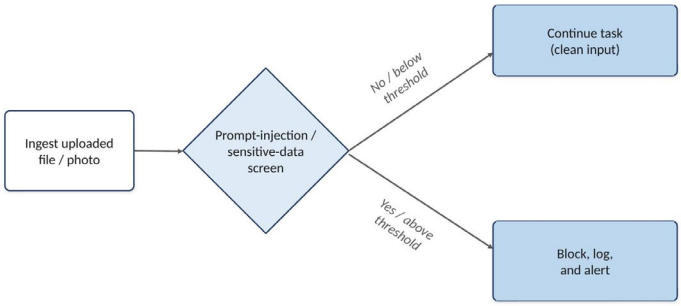

Figure 16. Guardrails and Safety. An uploaded file or photo is screened for prompt-injection or sensitive-data content before the task continues; flagged content is blocked, logged, and escalated rather than acted on.

Optimization and Reasoning Patterns

Optimization and reasoning patterns are concerned with efficiency and depth of thought rather than task structure or oversight: they determine how much computational effort and reasoning depth a given task actually warrants. These four patterns govern how a system allocates compute and reasoning effort, matching cost and depth to the actual difficulty and stakes of the task rather than treating every request identically.

17. Resource-Aware Optimization

Resource-aware optimization routes tasks to the model, tool, or compute tier whose cost and capability match the task's actual complexity, using a small, fast model for routine classification and reserving a larger reasoning model for genuinely hard cases, rather than sending every request through the most expensive available option (Gullí 2025). **Figure 17** shows routine RFIs handled by the small model while complex, high-dollar cases escalate to the larger reasoning model. On a large public infrastructure program, a small, fast model can triage routine, low-ambiguity RFIs cheaply and at high volume, escalating only ambiguous or high-dollar cases to a larger reasoning model, keeping operating cost proportional to task complexity. Cost-based routing that under-invests in the escalation tier defeats its own purpose, since savings should be measured against the cost of a wrong answer on a schedule- or safety-critical item, not against per-token price alone.

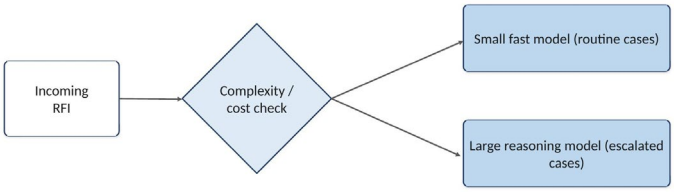

Figure 17. Resource-Aware Optimization. Incoming RFIs are checked for complexity and cost impact, with routine cases handled by a small, fast model and only escalated cases reaching a larger reasoning model.

18. Reasoning Techniques

Reasoning techniques encompass structured strategies such as chain of thought, in which a model works through a problem in explicit intermediate steps (Wei et al. 2022), its zero-shot variant that elicits similar step-by-step reasoning without worked examples (Kojima et al. 2022), tree of thought, in which multiple candidate reasoning paths are explored and evaluated before one is selected (Yao et al. 2023a), self-consistency, in which several independent reasoning paths are sampled and the most consistent final answer is taken (Wang et al. 2022), and debate, in which two or more agent instances argue opposing conclusions over multiple rounds before a judge or consensus mechanism decides (Du et al. 2023), used for problems that benefit from explicit, inspectable reasoning rather than a single fast guess (Gullí 2025). **Figure 18** shows the three reasoning paths generated for the overpass bent-jacket load-path question, flagged for engineer review because they disagree. On a highway overpass seismic retrofit, a self-consistency check generating three independent reasoning paths through an undocumented bent-jacket's load-path analysis can flag disagreement for engineer review rather than presenting one answer with false confidence. These techniques multiply compute cost and are not a substitute for licensed engineering judgment; their role is to surface disagreement, not resolve it.

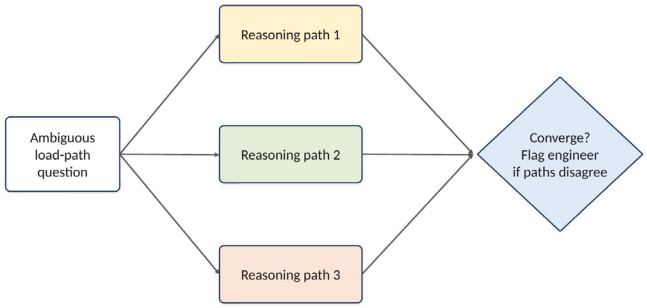

Figure 18. Reasoning Techniques. An ambiguous load-path question is run through three independent reasoning paths; the system flags the case for engineer review if the paths do not converge on the same conclusion.

19. Prioritization

Prioritization scores incoming tasks against multiple weighted factors, such as value, effort, urgency, and risk, and dynamically re-orders them as new information arrives, rather than processing them strictly first in, first out (Gullí 2025). **Figure 19** shows the daily re-ranked backlog that puts the signal-conduit submittal ahead of the higher-cost station-finish item. On a light rail extension backlog spanning RFIs, submittals, and punch-list entries, scoring by schedule impact, cost, safety relevance, and staleness can surface a lower-cost, higher-risk item ahead of a higher-cost, lower-risk one, an ordering a purely dollar-weighted queue would get backward. Scoring weights should be set and periodically reviewed by the project manager rather than treated as a fixed, opaque default; a related empirical comparison of a cognitive prioritization agent against transparent rule-based and opaque statistical baselines on real bridge-inventory data found the agent matched the transparent baseline's accuracy while both outperformed the statistical learner on the policy-relevant outcome, suggesting transparency need not come at an accuracy cost even when it is not free (Samsami and Khajedehi 2026).

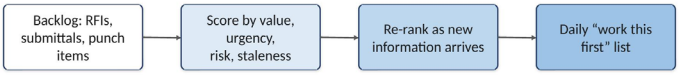

Figure 19. Prioritization. A backlog of RFIs, submittals, and punch items is scored by value, urgency, risk, and staleness, then dynamically re-ranked into a daily work-this-first list as new information arrives.

20. Exploration and Discovery

Exploration and discovery maps an unfamiliar space, clustering similar items and probing for patterns to surface novel or unexpected structure, rather than pursuing one known goal, and is used when the useful question is not yet known and only the raw data is available (Gullí 2025). **Figure 20** shows the recurring coordination cluster surfaced by clustering four years of closed RFIs and change orders. Given four years of a regional contractor's closed RFI and change-order records with no specific question posed, clustering can surface that a disproportionate share of MEP change orders trace back to one recurring coordination gap between two BIM platforms, a pattern no individual project manager had connected across projects. Exploratory clustering can surface spurious correlations in a limited dataset, so discovery output should be framed as a hypothesis for a human analyst to investigate, not a validated finding.

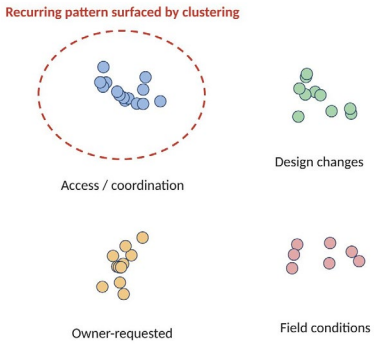

Figure 20. Exploration and Discovery. Four years of closed RFI and change-order records are clustered by root cause; the dashed outline marks a recurring access-and-coordination cluster surfaced without a specific question being posed.

CASE STUDY: EMPIRICAL VALIDATION OF THE RETRIEVAL PATTERN

The illustrative cases in the preceding section demonstrate how each pattern's structural role maps onto a construction management workflow, but they do not, on their own, establish that a pattern actually behaves as claimed. This section addresses that gap for one pattern, Retrieval, by building a small working retrieval pipeline and testing it against a synthetic but realistic bridge rehabilitation specification corpus, quantifying the addenda-grounding failure mode described above rather than only asserting it.

Data

A synthetic specification corpus was constructed for a hypothetical bridge rehabilitation contract, modeled on common bridge rehabilitation scope items. The corpus comprises 16 base specification sections across six divisions (concrete, structural steel, coatings, temporary traffic control, electrical, and drainage) and 6 addenda, each of which revises a single numeric or procedural value in one base section, for example extending a concrete cure period or tightening a section-loss threshold. A test set of 24 questions was authored against this corpus, grouped into three types: 10 stable questions whose governing section no addendum has touched, 6 addendum-affected questions whose correct current answer is the addendum text rather than the superseded base section, and 8 distractor questions phrased to be semantically close to a different, topically similar section, to test whether retrieval selects the correct one among confusable neighbors. All corpus and question content is fabricated for this experiment and does not describe a real structure; this is a stated limitation, discussed below.

Method

Retrieval was implemented using term frequency-inverse document frequency (TF-IDF) vectorization and cosine similarity (Manning, Raghavan, and Schütze 2008), a standard, reproducible baseline retrieval method that isolates the grounding and chunking questions this case study is concerned with from the separate question of embedding-model choice. Two experiments were run. The first compares three indexing conditions on the full 24-question set: a stale index built only from the 16 original base sections, representing a RAG system whose index was never rebuilt after addenda were issued; a current index adding the 6 addenda as separate chunks; and the current index combined with a simple supersession-aware re-ranking rule that promotes an addendum chunk over its corresponding base section when both appear among the top two matches by similarity. The second experiment compares chunking strategy on the current index: clause-aware chunking, one chunk per specification section or addendum item, against naive fixed-width chunking, the same corpus text concatenated and split into 180-character windows with 40-character overlap without regard to clause boundaries. In both experiments the outcome measure is top-1 retrieval accuracy, whether the single highest-scoring chunk for a question matches the current, correct governing chunk.

Results

Table 2 reports the indexing-condition results. The stale index answered every stable question correctly but returned the superseded base section for all 6 addendum-affected questions, a 0 percent accuracy rate that directly reproduces, in a measurable form, the silent-degradation failure mode described in the pattern discussion above. Adding the addenda to the index raised addendum-affected accuracy to 50 percent without any change to the retrieval method itself, and the supersession-aware re-ranking rule raised it further to 100 percent, but at a cost: distractor accuracy fell from 87.5 percent to 62.5 percent, as the re-ranking rule occasionally promoted an addendum

chunk for a question it did not actually govern. No condition changed stable-question accuracy, which remained 100 percent throughout.

TABLE 2 Retrieval Accuracy by Indexing Condition (Top-1 Accuracy, N = 24)

Condition	Overall	Stable	Addendum-affected	Distractor
Stale index (pre-addenda only)	70.8% (17/24)	100% (10/10)	0.0% (0/6)	87.5% (7/8)
Current index (addenda added)	83.3% (20/24)	100% (10/10)	50.0% (3/6)	87.5% (7/8)
Current index + supersession re-rank	87.5% (21/24)	100% (10/10)	100% (6/6)	62.5% (5/8)

Table 3 reports the chunking-strategy results. Clause-aware chunking outperformed fixed-width chunking overall, 83.3 percent versus 79.2 percent, and clause-aware chunking was markedly better on stable questions, 100 percent versus 80 percent, consistent with the concern raised earlier that splitting a clause from its title or context can retrieve a misleading fragment. Counterintuitively, fixed-width chunking outperformed clause-aware chunking on the addendum-affected subset, 66.7 percent versus 50.0 percent: the shorter, more targeted windows produced by fixed-width chunking appear to concentrate term overlap on the specific revised value, giving it a stronger match to a narrowly phrased question than the full addendum chunk, which includes more boilerplate framing language that dilutes the match.

TABLE 3 Retrieval Accuracy by Chunking Strategy (Current Index, No Re-Ranking, N = 24)

Chunking strategy	Overall	Stable	Addendum-affected	Distractor
Clause-aware (one chunk per section/addendum)	83.3% (20/24)	100% (10/10)	50.0% (3/6)	87.5% (7/8)
Fixed-width (180-char windows, 40-char overlap)	79.2% (19/24)	80.0% (8/10)	66.7% (4/6)	87.5% (7/8)

Discussion

Three findings carry a practical implication for CM organizations deploying the Retrieval pattern. First, the stale-index result is not a hypothetical concern; a plain TF-IDF retriever missed every addendum-affected question when its index was not current, and this class of error is silent, a fluent, well-formatted, wrong answer, rather than a visible failure. Second, simply adding current data to the index is necessary but not sufficient: even with addenda indexed, unassisted retrieval still missed half of the addendum-affected questions, because nothing in a standard retrieval score tells the system that one chunk supersedes another; recency and supersession are metadata the retrieval layer has to be told about explicitly, not something it infers from text similarity alone. Third, the chunking result is a genuine tradeoff rather than a clean recommendation: clause-aware chunking is the safer general-purpose default because it preserves the section-level context that disambiguates stable and distractor questions, but the supersession-aware re-ranking evaluated here shows that closing the addendum gap can introduce new errors elsewhere, so any such rule needs its own evaluation rather than being adopted on faith.

This case study has real limitations. The corpus and question set are synthetic, authored by the same person who designed the experiment, and small relative to an actual project's specification manual, so the absolute accuracy figures should not be read as predictive of performance on a real, larger corpus; the directional findings, that stale indexes silently fail on superseded content and that chunking strategy trades off differently across question types, are the more durable takeaway. TF-IDF is a deliberately simple retrieval method chosen for transparency and reproducibility; a production system would likely use dense embedding-based retrieval, which may narrow or widen the gaps reported here in ways this study cannot predict. The evaluation also stops at retrieval; it does not test whether a downstream language model, given the correct retrieved chunk, actually generates a correct grounded answer, a separate evaluation this paper leaves for future work, though a companion study by the author, using three contemporary LLMs against seventy validated inspection questions drawn from real transportation-agency documents, provides an initial answer at the generation stage and reaches a broadly consistent conclusion about the value of grounding over closed-book prompting (Samsami and Sayed 2026). Finally, this case study validates one pattern; the other nineteen patterns in this taxonomy still rest on the illustrative-case treatment described in the Method section above and would each benefit from a comparable empirical test in future work.

PATTERN COMPOSITION IN REAL CM WORKFLOWS

Real CM workflows rarely deploy a single pattern in isolation. They compose several patterns into an end to end pipeline whose overall reliability depends on how the pieces fit together, not just on the quality of any one component. Two illustrative examples of a) submittal review and b) daily field reporting show what this composition looks like in practice.

A realistic submittal review pipeline composes Routing, to classify submittal type and discipline, with Retrieval to ground the review against the project's current specifications and addenda, Tool Use to query the submittal log and common data environment for status and history, Prompt Chaining to extract, compare, and draft comments, Reflection to self-critique the draft against spec language, Human-in-the-Loop for licensed reviewer sign-off, Evaluation and Monitoring to track reviewer override rate over time, and Learning and Adaptation to feed corrections back into the drafting prompts. A tool that implements only the first three of these steps and calls itself agentic is a materially different, and materially less reliable, system for formal contractual output than one that implements the full sequence.

A comparable pipeline for daily field reporting composes Parallelization, to process photos, sensor feeds, and text logs concurrently, with Multi-Agent Collaboration, in which a manager agent reconciles the three specialist outputs, Exception Handling, to flag rather than silently proceed if a data source is missing, Memory Management, to log the reconciled report to episodic project memory, and Goal Setting and Monitoring, to compare the result against the schedule baseline and flag drift.

The practical implication for CM organizations is that pattern selection should follow the workflow's risk and ambiguity profile rather than adopt maximal architectural complexity everywhere. A routine, low stakes classification task, such as tagging site photos by trade, needs only Routing and Tool Use. A formal document with contractual or safety consequence needs the fuller Reflection, Human-in-the-Loop, and Evaluation stack. Treating every workflow identically either under-protects high stakes outputs or imposes unnecessary latency and cost on routine ones.

IMPLEMENTATION CONSIDERATIONS FOR AEC ORGANIZATIONS

Several practical considerations follow from the pattern taxonomy for organizations planning to build or procure agentic AI capability. Patterns that depend on grounding, including Retrieval, Memory Management, and Learning and Adaptation, require structured, current project documentation, so organizations with fragmented document control should sequence data hygiene work ahead of agentic pilots rather than in parallel with them. Guardrail and human in the loop trigger design should map directly onto existing delegation of authority and quality assurance sign-off structures rather than create a parallel approval path staff will route around under schedule pressure.

Vendor claims of agentic AI should be evaluated against which of the twenty patterns are actually implemented and to what depth, given the documented gap between agentic AI marketing and realized production value industry-wide (World Construction Network 2026). Adopting pattern based agentic tooling is itself an organizational change process, complementary to established change-management frameworks already in use elsewhere in CM digital transformation. Organizations are generally better served by piloting one bounded workflow, RFI routing is a common first choice, with explicit success metrics before composing multiple patterns into an end-to-end system.

LIMITATIONS AND FUTURE WORK

Nineteen of this paper's twenty patterns are illustrated only by brief, author-constructed cases rather than validated production deployments, and this limitation is disclosed explicitly rather than implied; the case study above narrows that gap for one pattern, Retrieval, but its own limitations, a synthetic corpus, a simple TF-IDF baseline, and a retrieval-only evaluation stopping short of generation, are stated in the Discussion subsection and are not repeated here. Extending a comparable empirical test to the remaining patterns, particularly the multi-agent, planning, and prioritization patterns that are harder to isolate for a small-scale experiment, is the most direct next step and a natural extension of this work. The underlying pattern taxonomy originates in general purpose software engineering (Gullí 2025); some patterns, particularly Inter-Agent Communication across organizational and contractual boundaries, may require construction specific protocol extensions that do not yet exist as industry standards. Future work should also examine transportation sector specific constraints in more depth, including public agency data sharing rules, state department of transportation approval workflows, and procurement regulations governing autonomous decision making on publicly funded projects, and should test the composition sequences proposed above, and the case study's retrieval pipeline itself, against real project data rather than synthetic or illustrative scenarios.

CONCLUSIONS

As agentic AI tooling enters CM faster than the industry has built shared vocabulary for describing it, a pattern based framework gives practitioners, researchers, and procurement teams a common language for asking what a given agentic tool actually does, where it is likely to succeed, and where its failure modes intersect with the risk, liability, and multi-party coordination realities specific to construction projects. The twenty patterns translated here are not a menu to implement in full on every project; they are a vocabulary for deliberate architecture choices, matched to the ambiguity and consequence of the specific CM workflow at hand. The case study demonstrates that this vocabulary can be tested rather than only argued: a working retrieval pipeline, evaluated against a synthetic but realistic specification corpus, showed a stale index missing every addenda-affected question and a targeted re-ranking fix trading distractor accuracy for addendum accuracy, a genuine tradeoff rather than a clean win. Extending that kind of test to the remaining nineteen patterns is the clearest path from this taxonomy to a body of applied evidence CM organizations can act on.

ACKNOWLEDGMENT OF ARTIFICIAL INTELLIGENCE USE

In accordance with TRB's policy on the use of generative AI in manuscript preparation, the author discloses that Claude (Anthropic) was used as a generative AI writing assistant during preparation of this manuscript. All AI-assisted modifications were reviewed, fact-checked against original sources, and edited by the author prior to submission. No empirical data, results, or findings were generated by AI.

AUTHOR CONTRIBUTIONS

The author confirms sole responsibility for the following: study conception and design, literature review, development of the pattern taxonomy and illustrative case studies, diagram preparation, and manuscript preparation. The author reviewed the results and approved the final version of the manuscript.

DECLARATION OF CONFLICTING INTERESTS

The author declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

FUNDING

The author disclosed no financial support for the research, authorship, and/or publication of this article.

REFERENCES

Abioye, Sofiat O., Lukumon O. Oyedele, Lukman Akanbi, Anuoluwapo Ajayi, Juan Manuel Davila Delgado, Muhammad Bilal, Olugbenga O. Akinade, and Ashraf Ahmed. 2021. “Artificial intelligence in the construction industry: A review of present status, opportunities and future challenges.” Journal of Building Engineering 44: 103299. https://doi.org/10.1016/j.jobe.2021.103299.

Badhan, Sharmin Jahan, and Reihaneh Samsami. 2026. “A construction-specific framework addressing explainability, liability, and ethical boundaries of AI use in civil engineering.” Buildings 16 (14): 2759. https://doi.org/10.3390/buildings16142759.

Du, Yilun, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. 2023. “Improving factuality and reasoning in language models through multiagent debate.” arXiv preprint arXiv:2305.14325.

Greshake, Kai, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. 2023. “Not what you’ve signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection.” In Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security, 79–90. arXiv:2302.12173.

Gullí, Antonio. 2025. Agentic Design Patterns: A Hands-On Guide to Building Intelligent Systems. Springer Nature.

Hughes, Laurie, Yogesh K. Dwivedi, Tegwen Malik, Mazen Shawosh, Mousa Ahmed Albashrawi, Il Jeon, Vincent Dutot, et al. 2025. “AI agents and agentic systems: A multi-expert analysis.” Journal of Computer Information Systems 65 (4): 489–517. https://doi.org/10.1080/08874417.2025.2483832.

Kirchhof, Michael, Gjergji Kasneci, and Enkelejda Kasneci. 2025. “Position: Uncertainty quantification needs reassessment for large language model agents.” arXiv preprint arXiv:2505.22655.

Kojima, Takeshi, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. “Large language models are zero-shot reasoners.” arXiv preprint arXiv:2205.11916.

Kolt, Noam. 2025. “Governing AI agents.” arXiv preprint arXiv:2501.07913.

Kraprayoon, Jam, Zoe Williams, and Rida Fayyaz. 2025. “AI agent governance: A field guide.” arXiv preprint arXiv:2505.21808.

Lewis, Patrick, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, et al. 2020. "Retrieval-augmented generation for knowledge-intensive NLP tasks." arXiv preprint arXiv:2005.11401.

Manning, Christopher D., Prabhakar Raghavan, and Hinrich Schütze. 2008. Introduction to Information Retrieval. Cambridge University Press.

Park, Joon Sung, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2023. "Generative agents: Interactive simulacra of human behavior." arXiv preprint arXiv:2304.03442.

Raza, Shaina, Ranjan Sapkota, Manoj Karkee, and Christos Emmanouilidis. 2025. "TRiSM for agentic AI: A review of trust, risk, and security management in LLM-based agentic multi-agent systems." arXiv preprint arXiv:2506.04133.

Razi, Niloofar, Sharmin Jahan Badhan, and Reihaneh Samsami. 2026. "Artificial intelligence (AI) in construction management (CM): A systematic review of models and methods." Buildings 16 (11): 2225. https://doi.org/10.3390/buildings16112225.

RTS Labs. 2026. "AI agents for construction: Real-world use cases and implementation guide." Accessed July 1, 2026. https://rtslabs.com/ai-agents-for-construction.

Samsami, Reihaneh. 2026. "Artificial intelligence (AI) agents in construction: A conceptual framework across autonomy types and application domains." engrXiv. https://doi.org/10.31224/7439.

Samsami, Reihaneh, and Rouzbeh Khajedehi. 2026. "Cognitive agents for bridge inspection prioritization: An empirical test of predictability and transparency using the National Bridge Inventory (NBI)." engrXiv. https://doi.org/10.31224/7703.

Samsami, Reihaneh, and Khaled Sayed. 2026. "Grounding large language models (LLMs) in agency standards: A controlled comparison of closed-book, retrieval-augmented, and full-document question answering for construction inspection." engrXiv. https://doi.org/10.31224/7545.

Sapkota, Ranjan, Konstantinos I. Roumeliotis, and Manoj Karkee. 2025. "AI agents vs. agentic AI: A conceptual taxonomy, applications and challenges." arXiv preprint arXiv:2505.10468.

Savaş, Sezer. 2025. "Artificial intelligence in construction project management: Trends, challenges and future directions." Journal of Design for Resilience in Architecture and Planning 6 (2): 221–238. https://doi.org/10.47818/DRArch.2025.v6i2165.

Schick, Timo, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. "Toolformer: Language models can teach themselves to use tools." arXiv preprint arXiv:2302.04761.

Shinn, Noah, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. "Reflexion: Language agents with verbal reinforcement learning." arXiv preprint arXiv:2303.11366.

Wang, Lei, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, et al. 2024. "A survey on large language model based autonomous agents." Frontiers of Computer Science 18 (6): 186345.

Wang, Xuezhi, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. "Self-consistency improves chain of thought reasoning in language models." arXiv preprint arXiv:2203.11171.

Wei, Jason, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2022. "Chain-of-thought prompting elicits reasoning in large language models." arXiv preprint arXiv:2201.11903.

World Construction Network. 2026. "Agentic AI in construction: Is the industry ready?" Accessed July 1, 2026. https://www.worldconstructionnetwork.com/sponsored/agentic-ai-in-construction-is-the-industry-ready/.

Wu, Qingyun, Gagan Bansal, Jieyu Zhang, Yiran Wu, Shaokun Zhang, Erkang Zhu, Beibin Li, Li Jiang, Xiaoyun Zhang, and Chi Wang. 2023. "AutoGen: Enabling next-gen LLM applications via multi-agent conversation framework." arXiv preprint arXiv:2308.08155.

Yao, Shunyu, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023a. "Tree of thoughts: Deliberate problem solving with large language models." arXiv preprint arXiv:2305.10601.

Yao, Shunyu, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023b. "ReAct: Synergizing reasoning and acting in language models." arXiv preprint arXiv:2210.03629.