Optimizing Complex Non-Deterministic Business Processes Using AI-Assisted Deterministic Control Layers

Modern business processes such as request-for-quotation handling, supplier negotiation, and multi-party exception management combine free-text communication, asynchronous events, and incomplete information, which makes them difficult to optimize through conventional workflows alone. This paper proposes a hybrid architecture in which AI agents support interpretation, retrieval, summarization, and drafting, while a deterministic control layer governs every business-significant state transition. The approach is examined through an RFQ-to-quotation proof of concept (RFQ PoC) and a scenario-based benchmark derived from realistic procurement conditions. Under the benchmark's stated assumptions, the hybrid configuration produced modeled reductions of 43.3% in average cycle time, 40.8% in clarification rounds, 59.4% in supplier re-contact events, and 35.0% in manual touches, while quote completeness increased by 9.16 percentage points. The scenario-based findings support placing probabilistic AI behind a governed commitment boundary rather than allowing it to mutate business state directly.

Keywords: *AI-assisted workflow orchestration; business process management; deterministic control; knowledge-intensive processes; RFQ automation*

Introduction

Complex enterprise work is bounded by goals and rules but not by a fully predictable path. RFQ handling exemplifies this condition because requests arrive through mixed channels, information is often incomplete, and progress depends on interpretation across customers, internal staff, and suppliers.

Classical BPM, declarative workflows, and case-management approaches improve control and flexibility (van der Aalst et al. 2003; Pesic and van der Aalst 2006; van der Aalst et al. 2009; Herrmann and Kurz 2011), yet they do not fully solve the semantic burden created by ambiguous text, scattered attachments, and exception-driven negotiation. LLMs can help interpret and draft (Grohs et al. 2023; Kourani et al. 2024; Lewis et al. 2020; Schick et al. 2023; Yao et al. 2023), but their probabilistic outputs make direct control of pricing, supplier communication, or case state unacceptable.

This paper argues that the key design move is to separate reasoning from commitment. AI may analyze, retrieve, summarize, classify, and draft, but authoritative state changes must pass through deterministic validation, policy checks, authorization rules, and event recording. In that sense, the AI layer is generative, whereas the control layer is normative.

The argument is developed through RFQ PoC, an anonymized RFQ-to-quotation design referent. The study asks how AI can be introduced without losing reproducibility and compliance, what mechanisms are needed to convert

probabilistic suggestions into replayable transitions, and what performance gains can be achieved in an RFQ environment.

The paper contributes a reference architecture, a structured model of RFQ non-determinism, an implementation-oriented case description, and a scenario-based evaluation. The remainder reviews the literature, presents the design, reports results, and discusses implications.

Review of the Relevant Literature

Business process management has long balanced formal control with operational flexibility (van der Aalst et al. 2003). Adaptive case management and related work showed that knowledge-intensive processes cannot be captured fully by rigid procedural scripts (Herrmann and Kurz 2011; Swenson 2010; Schonenberg et al. 2008a).

Declarative modeling and flexible-execution approaches demonstrated that structure can be expressed through constraints and recommendations rather than through a single fixed path (Pesic and van der Aalst 2006; van der Aalst et al. 2009; Pesic et al. 2007; Schonenberg et al. 2008b). Process-aware information systems and process mining further emphasized event logs, replay, and auditability as core architectural concerns (Dumas et al. 2005; Dumas et al. 2018; van der Aalst 2016).

More recent work on AI-augmented BPMSs treats AI as an extension of BPM rather than a replacement for it (Dumas et al. 2023). LLM studies show strong potential for process modeling, summarization, extraction, retrieval, and drafting (Grohs et al. 2023; Kourani et al. 2024; Ziche and Apruzzese 2024), while tool-use and retrieval patterns improve grounding and practical capability (Lewis et al. 2020; Schick et al. 2023; Yao et al. 2023; Wu et al. 2024).

The unresolved gap is architectural. The literature explains process control on one side and semantic AI assistance on the other, but offers less guidance on how to combine them so that AI absorbs uncertainty while deterministic software preserves legal business state. That gap motivates the commitment boundary proposed in this paper.

Two studies published in the Athens Journal of Technology & Engineering provide useful adjacent evidence. Samaranayake and Senanayake (2022) show how explicit logical and preferential dependencies can structure decision-support paths, while Boer and Ihlenburg (2020) emphasize that Industry 4.0 process innovation unfolds across interconnected customers, suppliers, and organizational partners. These studies do not address large-language-model control boundaries directly, but they reinforce the need to couple flexible interpretation with explicit process structure and multi-party coordination.

Research Design and Methodology

The study follows design science research because its primary outcome is an artifact: a reference architecture for optimizing non-deterministic business processes through AI-assisted deterministic control layers (Hevner et al. 2004). RFQ handling was chosen as the design context because it combines unstructured documents, asynchronous communication, supplier heterogeneity, and traceable commercial decisions.

RFQ PoC serves as the design referent. The paper reports its architecture, roles, and control logic while avoiding disclosure of proprietary rules or confidential transactions.

The research proceeded in four steps: modeling the main sources of RFQ non-determinism; designing an architecture that separates probabilistic reasoning from deterministic state change; instantiating that architecture in RFQ PoC; and evaluating it through a scenario-based benchmark with replayable event logic.

The benchmark compares a semi-manual baseline with the hybrid design. Main measures are cycle time, clarification rounds, supplier re-contact events, manual touches, quote completeness, and deterministic replayability. The goal is design validity rather than universal statistical generalization.

Problem Space: Non-Determinism in RFQ Processes

Non-determinism in business processes is often described too loosely. In enterprise settings, it does not mean that anything can happen. Rather, it means that the next required action cannot be inferred reliably from the current formal state alone. Additional interpretation is needed. For design purposes, this paper distinguishes four forms of non-determinism.

The first is linguistic non-determinism. Business actors communicate through e-mail, attachments, specifications, spreadsheets, and informal comments. Meanings are distributed across these artifacts. One customer asks for “rapid delivery” without defining a date. Another refers to a part family using internal shorthand. A supplier may respond with a viable alternative product rather than with the requested part number. Classical workflow systems can route the message, but they cannot decide what the message means in operational terms.

The second is epistemic non-determinism. At many points in the process, required facts are simply missing. Material grade, certification, quantity tolerance, shipping condition, payment term, and lead time may be absent, inconsistent, or implied rather than stated. A process participant must decide whether the case can proceed, whether assumptions are acceptable, or whether clarification is mandatory. This is not simply a data-quality problem. It is a business judgement problem supported by data.

The third is temporal non-determinism. RFQ handling involves asynchronous interactions between multiple organizations. Suppliers answer at

different times, customers revise scope midstream, and internal reviewers interrupt the progression with new concerns. A predefined sequence model may represent the ideal path, but actual execution is shaped by waiting, parallelism, and interruptions that cannot be scheduled precisely.

The fourth is organizational non-determinism. Different actors own different fragments of expertise. Sales personnel understand the commercial context, sourcing specialists know supplier landscapes, and engineers understand technical feasibility. No single actor necessarily holds the full picture at intake. The process therefore depends on cross-functional interpretation and negotiation rather than on a simple handoff chain.

These four forms of non-determinism interact. Missing technical details increase the likelihood of supplier re-contact. Supplier ambiguity increases internal review effort. Temporal delays alter prioritization. Organizational fragmentation increases the chance that one actor resolves an issue locally without updating the shared process state. The resulting process is not chaos, but a coupled system of partial information and situated decision making.

Why do conventional optimization techniques struggle here? One reason is model granularity. Flow-oriented models are effective when the main challenge is routing work among stable tasks. They are less effective when the real bottleneck lies inside a task whose content is semantically unstable. “Clarify customer requirements” may appear as a single box, but it can involve several rounds of reading, inference, retrieval, drafting, review, and follow-up. If the variability is inside the box, optimizing the arrows between boxes yields limited returns.

A second reason is the mismatch between formal state and business meaning. A workflow engine can know that a case is in state Clarification Pending. It cannot, by itself, infer whether the missing information is critical, whether a supplier alternative is acceptable, or whether two pieces of text refer to the same commercial constraint. In other words, formal state alone lacks semantic sufficiency.

A third reason is brittleness under exception. Deterministic workflows usually respond well to anticipated variation and poorly to unmodeled variation. In RFQ handling, many deviations are neither rare nor truly exceptional. They are part of the normal work. Designing explicit branches for every possible clarification pattern produces models that are both complex and still incomplete.

These limitations explain why organizations often resort to e-mail, spreadsheets, messaging, and tacit coordination around formal systems. The formal workflow preserves accountability, while the real problem-solving happens outside the workflow. This split is expensive. It increases latency, weakens traceability, and makes process learning difficult because the process knowledge is dispersed across unmanaged interactions.

The opportunity for AI lies precisely here. LLM-based agents can absorb language, compare document fragments, retrieve precedent cases, detect missing fields, summarize alternatives, and propose next steps. Yet that opportunity becomes dangerous if AI is allowed to collapse the boundary between interpretation and commitment. An agent may infer that a delivery window is

acceptable, but the business cannot allow such an inference to become a committed promise unless the action is validated under policy, authority, and state constraints.

The problem, then, is not whether AI should participate in these processes. It should. The problem is architectural: where in the execution path should AI participate, what authority should it possess, and how should its outputs be transformed into legal business actions? The next section answers that question by introducing deterministic control layers as the mechanism that converts probabilistic assistance into governed execution.

Table 1. *Main Sources of Non-Determinism in RFQ Processes*

Source	Typical Manifestation	Business Risk	Control-Layer Response
Linguistic	Free-text RFQs, ambiguous wording, supplier alternatives stated informally	Misclassification, wrong assumptions, delayed clarification	AI extraction and summarization, followed by schema validation and evidence binding
Epistemic	Missing quantities, technical attributes, certifications, or delivery constraints	Rework, supplier confusion, incomplete quotations	Mandatory-field checks, explicit issue queues, targeted clarification proposals
Temporal	Asynchronous replies, waiting periods, interrupted reviews, late scope changes	Case drift, idle time, missed deadlines	Event-driven orchestration, timers, deterministic waiting states, replayable history
Organizational	Distributed expertise across sales, sourcing, engineering, and suppliers	Fragmented responsibility, hidden local decisions	Role-based approvals, canonical case state, logged command handling and escalation rules

Source: Author's synthesis.

AI-Assisted Deterministic Control Layers

The proposed architecture is built on a simple but strong distinction between proposals and commitments. A proposal is any machine-generated interpretation, recommendation, draft, summary, extracted structure, or next-action suggestion produced by AI. A commitment is any business-significant action that changes durable process state, triggers external communication under the organization’s name, allocates responsibility, advances a case stage, or records a fact as authoritative. AI may generate proposals; only the deterministic control layer may authorize commitments.

Conceptually, the architecture consists of five cooperating strata. The interaction layer receives human inputs, uploaded documents, messages, and system events. The AI reasoning layer interprets these inputs through specialized agents such as intake parsers, clarification assistants, supplier matchers, and quotation drafters. The deterministic control layer mediates every proposed action through rules, validators, authorizations, and state machines. The persistence layer stores the canonical case record, event log, documents, embeddings, and audit metadata. Finally, the orchestration layer coordinates asynchronous execution, timers, retries, and integrations with external systems.

The deterministic control layer is the architectural center of gravity. It performs six functions. First, it maintains the canonical case state through an explicit state model. Second, it validates all commands against business rules, permissions, and schema constraints. Third, it records committed actions as immutable events. Fourth, it materializes read models for user interfaces, reports, and downstream integrations. Fifth, it enforces idempotency and replayability so that the state can be reconstructed from history. Sixth, it defines escalation points where human approval is mandatory.

Formal Boundary Between Proposal and Commitment

This can be expressed formally.

$$o_t = \pi(c_t)$$
$$a_t = V(s_t, o_t, R, P)$$
$$s_{t+1} = \delta(s_t, a_t, e_t)$$

where o_t is the AI-generated proposal, a_t is the validated command, and s_{t+1} is the authoritative next case state after deterministic transition handling. Here t is a logical process-step index, not elapsed time. The records s_t, c_t, o_t, a_t, and e_t have no physical unit; R and P are rule and permission sets, while π, V, and δ are mappings.

Let s_t denote the current deterministic case state at logical step t. Let c_t denote the context available to an AI component, including recent messages, retrieved knowledge, and current case data. The AI layer produces an output $o_t = \pi(c_t)$, where π is a probabilistic mapping. The control layer applies a deterministic validator V such that $a_t = V(s_t, o_t, R, P)$, where R denotes business rules and P denotes permission constraints. If validation succeeds, the committed action a_t is applied through a deterministic transition function $s_{t+1} = \delta(s_t, a_t, e_t)$, where e_t represents relevant external events. If validation fails, the proposal is rejected, revised, or escalated. The crucial point is that π may vary, but V and δ must not vary for the same inputs.

This formulation yields an important design invariant: durable business state can never depend directly on a probabilistic output. AI can influence the process only through a rule-governed transformation. In practice, this means that outbound e-mails can be drafted automatically but not sent unless the sending policy allows the template, the case is in the correct state, required data are

present, and any required human approval exists. Supplier recommendations can be generated automatically, but the shortlist becomes authoritative only after validation against supplier eligibility, product constraints, and procurement policy. A quote can be drafted automatically, but quotation status changes only after mandatory checks are satisfied.

Several implementation patterns reinforce this invariant. The first is event sourcing. Rather than mutating case state in place without trace, the system appends committed domain events such as RFQReceived, RequirementsParsed, ClarificationRequested, SupplierSelected, OfferRecorded, QuoteDrafted, ReviewApproved, and QuoteSubmitted. Current state is a derived view over this event history. Event sourcing is not mandatory for every enterprise application, but it is particularly well suited here because it makes replay and audit first-class architectural properties.

The second pattern is command-query segregation. Commands request state changes and are validated by the control layer. Queries read materialized views and are free to combine structured state with AI-generated summaries. This separation is valuable because it prevents AI-generated convenience views from being mistaken for canonical truth. An LLM summary of a negotiation thread may be displayed to the user, but the canonical record remains the validated event history and stored communications.

The third pattern is typed action envelopes. Every AI proposal is wrapped in a machine-readable action schema specifying intent, confidence, referenced evidence, affected entities, and proposed next state. The deterministic control layer never consumes raw text as if it were executable. It consumes typed proposals that can be validated. This reduces ambiguity and creates a bridge between generative outputs and enterprise-grade software contracts.

The fourth pattern is policy gates. Some actions may be auto-approved under bounded conditions, such as requesting clarification when mandatory fields are missing or tagging a message to an existing case with high confidence. Other actions require a human gate, such as substituting a supplier, changing a promised lead time, or submitting a final quotation. The architecture therefore does not impose a single level of autonomy. Instead, it supports graduated autonomy aligned with business risk.

The fifth pattern is retrieval-grounded reasoning. The AI layer should not reason only from the prompt and transient conversation. It should retrieve prior RFQs, supplier profiles, product taxonomies, contractual rules, template libraries, and internal policies from governed knowledge stores. Retrieval does not remove uncertainty, but it narrows the semantic search space and provides evidentiary anchors for recommendations (Lewis et al. 2020).

The sixth pattern is human-visible provenance. Every significant AI proposal should carry traceable references: source snippets, matched documents, retrieved precedents, or explicit uncertainty statements. This makes review faster and improves trust because the user sees not only the recommendation but also its evidentiary basis. In knowledge-intensive processes, trust is operationally necessary. Actors will bypass a system they cannot interrogate.

A major advantage of deterministic control layers is that they convert AI adoption from a question of trust in the model to a question of trust in the boundary. Enterprise teams do not need to believe that the model is always right. They need to believe that wrong model outputs cannot silently become authoritative business actions. This shifts governance from model perfection to controlled failure containment. In organizational terms, that is often the difference between an experimental assistant and a deployable system.

Another advantage is reproducibility under model evolution. Language models change over time. Prompts are refined. Retrieval sources expand. If the AI layer were allowed to commit state directly, these changes could make case histories irreproducible. With a deterministic control layer, the canonical case history remains stable even if the proposal mechanism improves. The organization may upgrade its AI without invalidating the meaning of past commitments.

The architecture also supports a practical division of labor between humans and machines. Humans retain authority over exceptional commercial judgement, policy interpretation, and customer-facing accountability. AI performs semantically heavy but bounded assistance. The control layer enforces the contract between them. This does not eliminate human work; it moves human attention toward review, approval, and high-value judgement instead of repetitive interpretation and administrative reconstruction.

For non-deterministic processes, this division of labor is preferable to full automation. Full automation assumes a decision surface that can be fully specified. Non-deterministic business processes do not offer that surface. They are better understood as controlled explorations toward a business goal. AI makes the exploration faster and richer. Deterministic control ensures that the exploration never escapes the organization's executable rules. The RFQ PoC case illustrates how this architecture can be instantiated concretely.

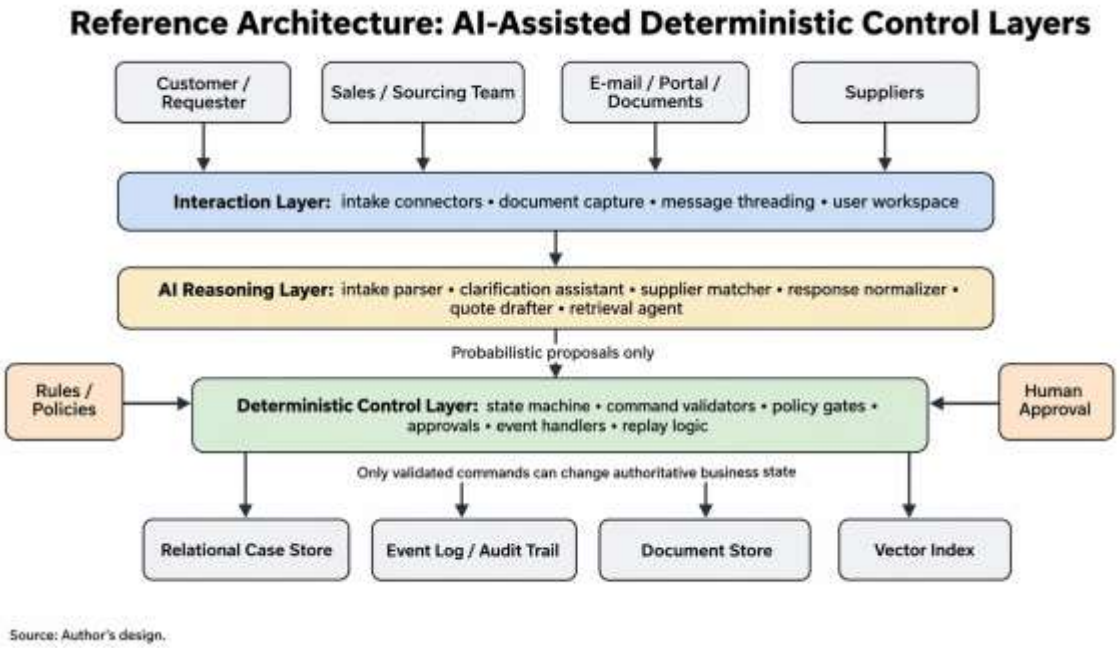
Table 2. *Core Responsibilities of the Deterministic Control Layer*

Responsibility	**Purpose**	Example in RFQ PoC
State management	Keeps a canonical case state aligned with business stages	Received → Parsed → Needs Clarification → Submitted
Command validation	Checks whether a proposed action is legal in the current state	Rejects supplier outreach when mandatory fields are still missing
Policy gating	Applies pricing, supplier, compliance, and approval rules	Requires approval before sending a high-value quotation
Event recording	Creates an immutable audit history of committed actions	Logs QuoteDrafted, ReviewApproved, and QuoteSubmitted
Human approval	Places bounded review at high-responsibility points	Commercial reviewer approves substitution or pricing exception

Responsibility	Purpose	Example in RFQ PoC
Replay and materialization	Reconstructs state from history and feeds user views	Replays all events to confirm final state equivalence

Source: Author's synthesis.

Figure 1. *Reference Architecture of AI-Assisted Deterministic Control Layers*



Source: Author's design.

Implementation Considerations and Governance Mechanisms

Moving from a conceptual architecture to a deployable enterprise system requires attention to several engineering details that are often overlooked in high-level AI discussions. The first is case identity resolution. In real organizations, one commercial request may be represented across e-mail threads, attachment sets, ERP records, and CRM opportunities. If the system cannot determine which artifacts belong to the same business case, even sophisticated AI assistance will fragment rather than improve the process. A deterministic control layer therefore needs robust case-linking rules based on identifiers, sender-recipient patterns, extracted document keys, temporal proximity, and human override. AI may assist in ambiguous matches, but the final case linkage should be committed only through deterministic validation or explicit user confirmation.

The second detail is schema discipline. Non-deterministic processes still need a well-designed business schema. In the RFQ domain, canonical entities typically include customer request, request line, product requirement, supplier candidate, supplier offer, clarification issue, quotation version, approval decision, and communication record. AI performs better when it maps into a schema that is semantically meaningful and operationally useful. A weak

schema forces the model to improvise structure repeatedly. A strong schema narrows the action space, improves validation, and makes downstream analytics more reliable.

Third, enterprise integration should follow a principle of asymmetric trust. External systems such as CRM, ERP, e-mail servers, supplier portals, and document repositories may be authoritative for specific data domains, but AI should not write into them directly. Instead, the control layer should expose narrow command interfaces such as CreateClarificationDraft, ConfirmSupplierShortlist, RegisterOffer, GenerateQuoteVersion, and SubmitQuote. These commands encapsulate policy and state constraints. By contrast, AI components may read much broader context through governed retrieval interfaces. In short, read widely, write narrowly.

A fourth implementation issue concerns knowledge retrieval. Many enterprise AI failures occur because the retrieval layer is treated as a generic vector search over everything. In business processes, relevance depends heavily on document type, process stage, and business role. A clarification agent should not search the same corpus in the same way as a supplier matching agent. The former may prioritize prior clarification templates, product attribute dictionaries, and unresolved field patterns. The latter may prioritize supplier history, category coverage, lead-time reliability, and contractual restrictions. Retrieval should therefore be role-specific and stage-aware. This does not eliminate the need for embeddings or vector search, but it disciplines their use.

Fifth, prompt orchestration should be treated as configuration, not as hidden application logic. In enterprise environments, prompt changes can materially alter behavior. They should therefore be versioned, tested, and linked to release management just like any other executable asset. The control layer benefits from this discipline because proposal provenance can include not only the source documents used, but also the prompt and agent version that produced the proposal. This is valuable for debugging, quality assurance, and post-incident review.

Sixth, monitoring must extend beyond generic model metrics. It is not enough to track latency, token usage, or aggregate confidence scores. Process-aware monitoring should measure architecture-level indicators such as clarification turnaround, proposal acceptance rate, replay success, auto-approval frequency, exception escalation rate, and divergence between AI proposals and human final decisions. These indicators reveal whether the system is becoming operationally useful, not merely technically active. They also help determine where autonomy should be increased, reduced, or redirected.

A seventh consideration is failure handling. In deterministic systems, failures usually mean exceptions, timeouts, or validation errors. In AI-assisted systems, failures may also mean semantic drift, overconfident extraction, ambiguity inflation, and retrieval contamination. The design response should not be to suppress such failures invisibly. Instead, the system should fail into explicit review states. For example, if a parsing agent cannot reconcile conflicting quantities across documents, the case should enter a Review Required state with

surfaced evidence rather than a silent best guess. Explicit failure states preserve user trust because uncertainty is made operational rather than hidden.

Security and privacy also require specific treatment. Procurement and quotation processes may involve sensitive commercial data, supplier pricing, contractual terms, and personal contact details. The deterministic control layer should therefore mediate access not only to actions but also to context. Different agents may need different views of the case, and different users may need different explanation depth. Access control is not an add-on. It is part of how business authority is represented in software. Where external model services are used, organizations must additionally decide which data may leave their controlled environment and which must remain on-premises or within approved infrastructure.

One practical question concerns whether a single powerful agent should handle the whole process or whether a set of specialized agents is preferable. The evidence from this design suggests that specialization is usually superior. Intake parsing, clarification drafting, supplier normalization, and quotation synthesis impose different quality criteria and rely on different knowledge sources. Specialized agents make it easier to constrain context, evaluate outputs, and associate clear acceptance criteria with each proposal type. The control layer then becomes the unifying mechanism through which these specialized capabilities contribute to one governed case progression.

Another issue is user interface design. In many failed automation projects, the system technically works but imposes too much review friction. Users then revert to manual communication because it feels faster. The interface for AI-assisted deterministic control should therefore present proposals in a way that is both inspectable and fast to accept. Good design patterns include evidence side panels, structured diff views, explicit uncertainty labels, one-click approval for low-risk actions, and quick escalation options. The aim is not to make AI invisible. It is to make its work legible enough that review becomes cheaper than manual reconstruction.

Finally, organizations need a governance model for continuous learning. Over time, human decisions create a valuable record of which proposals were accepted, revised, or rejected. This feedback can improve prompts, retrieval ranking, validation thresholds, and even business rules. However, learning should occur in the proposal layer, not by weakening deterministic constraints. The temptation to relax controls once the model appears useful must be resisted unless there is evidence that the corresponding action class is genuinely low risk. In mature deployments, the most successful pattern is often a dual loop: AI improves its proposal quality through feedback, while the control layer remains stable and only evolves through deliberate governance.

These implementation considerations underscore the paper’s main argument. The enterprise challenge is not simply to embed a language model into a business application. It is to create a control architecture in which semantic intelligence is productive precisely because it is bounded. The value of AI arises when it reduces the cognitive cost of progressing a case. The value of deterministic control arises when that progression remains institutionally

reliable. A system that achieves both is far more likely to survive contact with real business operations than one optimized only for demo-level autonomy.

RFQ PoC Case: RFQ-to-Quotation Orchestration

RFQ PoC focuses on the path from incoming RFQ to outbound quotation. The business objective is straightforward: respond faster, with higher completeness and lower manual friction, while preserving commercial control. The process, however, is rich in non-deterministic work. Incoming RFQs may arrive as e-mails with attachments, forwarded threads, spreadsheets, PDFs, or mixed-language documents. The customer may request multiple items, technical variants, delivery options, and certifications in a single bundle. Some requirements are explicit; others must be inferred or clarified. Suppliers may respond with substitutions, partial stock, counter-offers, or questions of their own. Internal actors must then synthesize these inputs into a coherent quotation package.

In RFQ PoC, the case begins when an intake connector captures an RFQ and creates a canonical case shell in the deterministic control layer. At this point the case has a stable identifier, timestamps, source references, and an initial state such as Received. The original artifacts are stored unchanged. No semantic assumptions are yet committed. An intake parsing agent then analyzes the content and proposes structured extractions: line items, quantities, target dates, certifications, shipping terms, and possible ambiguities. These proposals are not yet authoritative. The control layer validates whether the extracted fields meet minimum schema requirements and whether confidence thresholds justify automatic case enrichment. Where certainty is insufficient, the case moves into Needs Clarification rather than silently proceeding on shaky assumptions.

This early distinction is essential. Many RFQ processes lose time later because they overcommit too soon. A partially understood request is routed to suppliers, only to generate a second round of questions that could have been resolved upfront. RFQ PoC addresses this by treating clarification as a first-class process stage. A clarification agent examines missing or ambiguous fields, retrieves relevant historical patterns, and drafts targeted clarification questions. Instead of asking a customer to “please provide more details,” the system can suggest a compact, specific set of missing attributes. The outbound message remains subject to deterministic policy. Depending on the configured autonomy level, it is either queued for human review or sent automatically only if the message is a standard clarification request within a permitted template family.

Once the case is sufficiently specified, a supplier discovery or matching agent proposes candidate suppliers based on product class, historical supplier performance, regional constraints, and commercial policies. Again, the proposal does not directly alter the canonical shortlist. The control layer validates supplier eligibility, checks blacklist or exclusivity rules, and records the resulting action as SupplierCandidatesSelected or SupplierShortlistConfirmed. This distinction matters for governance. A model may suggest a plausible supplier, but only

deterministic policy can confirm whether that supplier is legally and commercially admissible.

Supplier communication introduces another layer of non-determinism. Responses arrive asynchronously, often in natural language and often without following the requested structure. One supplier may provide a full quotation table. Another may provide only lead time and ask for minimum order quantity confirmation. A third may answer indirectly by offering an alternative product. RFQ PoC uses AI to normalize and summarize these responses into comparable proposals. Quantities, lead times, prices, alternatives, and caveats are extracted into structured action envelopes. The control layer records these as OfferRecorded events only when required fields are present and source references are attached. If the response is partial, the system can automatically generate a supplier follow-up proposal, but the case state remains consistent because partial offers are recorded explicitly as such.

A quotation drafting agent then assembles the current best package from the validated case state. It can produce a draft summary, highlight unresolved gaps, and suggest a response structure aligned with prior successful quotations. This draft is valuable because it spares humans from reconstructing the case from dozens of messages. Yet the draft does not become the quotation merely because it reads well. The control layer requires internal review, mandatory field completion, and policy checks such as margin protection or approval thresholds before changing the state to Ready for Submission or Submitted.

From an implementation perspective, RFQ PoC can be understood as a case-centered, event-driven application. Each case holds a persistent identity across communications, supplier responses, and internal actions. AI components are stateless workers relative to the canonical case: they receive context, generate proposals, and return typed outputs. The control layer, by contrast, is the keeper of truth. It maintains the authoritative state machine, handles deduplication and retries, and persists every committed event with actor attribution and timestamps.

The state model used in the case is intentionally compact. Representative states include Received, Parsed, Needs Clarification, Ready for Supplier Outreach, Supplier Responses Pending, Offer Aggregation, Quote Drafting, Internal Review, Ready for Submission, Submitted, Closed Won, Closed Lost, and Abandoned. These states are business meaningful rather than merely technical. They tell both the system and the users what the case is waiting for. AI operates inside the transitions between them but cannot redefine them arbitrarily.

An equally important feature is evidence binding. Every structured proposal in RFQ PoC is tied back to the source artifact from which it was derived: a sentence in an e-mail, a row in a spreadsheet, a clause in a PDF, or a specific supplier message. This design choice serves three purposes. It supports human review, because users can verify the interpretation quickly. It supports debugging, because errors can be traced to specific inputs. And it supports auditability, because the organization can later reconstruct why a particular commitment was made.

Human-in-the-loop control is concentrated where commercial and organizational accountability are highest. For example, a human reviewer may be required for final supplier selection, pricing deviations, substitutions, or outbound quotations above a threshold value. Routine clarification questions, by contrast, may be auto-approved when the template and risk profile are low. This is not a concession to immaturity; it is the intended design. A hybrid architecture should not maximize automation indiscriminately. It should maximize safe compression of low-value effort while preserving human authority where the business requires it.

RFQ PoC therefore illustrates a broader proposition. In non-deterministic enterprise work, the most effective AI systems are not those that try to emulate an autonomous employee. They are those that continuously transform messy inputs into reviewable, policy-checkable, machine-valid proposals while a deterministic core preserves the integrity of the process. The next section translates this design into an evaluable benchmark.

Figure 2. *Deterministic RFQ Case-State Model*

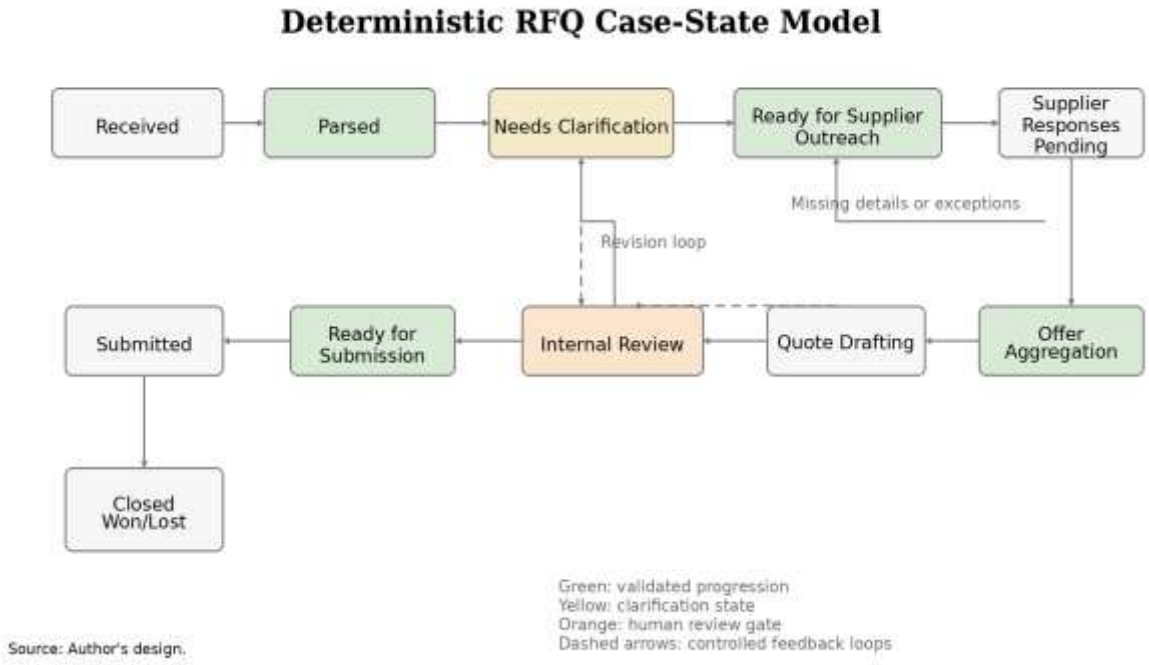
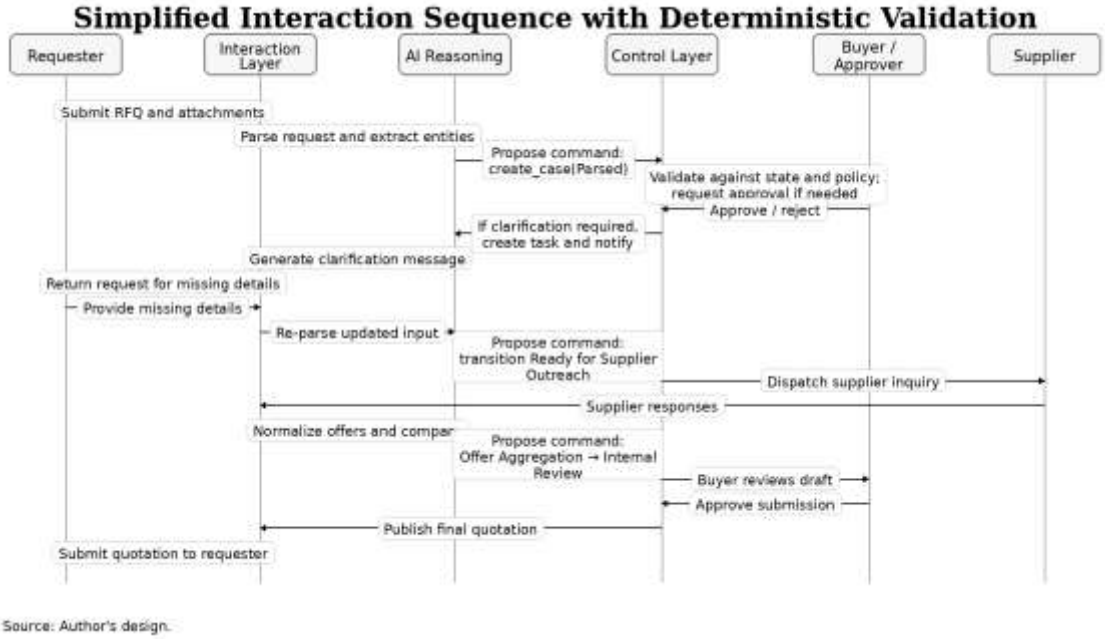


Figure 3. *Clarification and Quotation Sequence in the Hybrid Architecture*



Source: Author's design.

Microsoft Azure and Microsoft Foundry Experimental Setup

The RFQ proof of concept was implemented in a Microsoft cloud environment so that the architectural pattern could be examined under enterprise-relevant conditions rather than as a stand-alone prototype. The semantic layer was configured in Microsoft Foundry, which provides the Azure-based workspace used for model experimentation, prompt versioning, evaluation, tracing, and operational governance, while Azure OpenAI supplied the inference endpoints for extraction, clarification drafting, response normalization, and quotation synthesis (Microsoft 2026b, 2026d). This separation between workspace governance and model inference aligned well with the broader design principle of the paper: probabilistic reasoning remains flexible, but operational commitments stay bounded.

The deterministic control layer was implemented as a narrow command surface using Azure Functions and a rule-centric orchestration module. Commands such as CreateCase, RegisterClarificationNeeded, ConfirmSupplierShortlist, RecordOffer, GenerateQuoteVersion, and SubmitQuotation were validated against the explicit case-state model before being committed. The case record itself was split across a relational store for canonical state, an append-only event history for replay, a document repository for RFQ artefacts and attachments, and a vector index for retrieval. Azure AI Search was selected for vector and hybrid retrieval because it supports the coexistence of vector and nonvector content in the same index and is designed for similarity and hybrid search scenarios that are typical in retrieval-augmented generation pipelines (Microsoft 2026c).

Operational observability was treated as a first-class requirement. Prompt traces, tool invocations, latency, and quality observations were collected in the Foundry workspace and exposed through Azure Monitor and Application Insights. This was important because the experimental goal was not simply to produce acceptable drafts, but to study whether a hybrid architecture could remain inspectable under repeated runs. Microsoft Foundry documentation explicitly frames evaluation, monitoring, and tracing as core lifecycle functions for enterprise AI applications, and the PoC used this capability to compare prompt revisions, identify extraction failures, and correlate model proposals with subsequent acceptance or rejection by the deterministic layer (Microsoft 2026a, 2026d).

The benchmark environment combined a scenario generator with a role-based execution model. Each run used the workload profile described in the evaluation section and simulated the journey of a case through email-style intake, retrieval, proposal generation, deterministic validation, approval, event recording, and replay. Operator actions were represented through modeled role steps in a case console and approval interface. No human participants were recruited or observed; review time and manual touches were scenario parameters. This design preserved the operational sequence needed to estimate how much reconstruction work remained at higher-risk review points.

To preserve confidentiality, no undisclosed production transactions or supplier-specific commercial records were copied into the benchmark. Instead, the environment used realistic but synthetic RFQ bundles, supplier records, and communication sequences derived from the structure of the problem domain. The Microsoft-based implementation nevertheless remained close to deployable practice: the Foundry workspace governed models and traces; Azure OpenAI produced typed proposals; Azure AI Search grounded retrieval; Azure Functions enforced deterministic control; and monitoring data flowed to Azure observability services. In this sense, the PoC examined a realistic systems design without disclosing sensitive operational data.

Table 3. *Microsoft Technology Stack Used in the RFQ PoC*

Component	Microsoft service	Role in the experiment	Control significance
Workspace governance	Microsoft Foundry	Prompt versions, evaluations, traces, safety observations	Keeps model behaviour inspectable and reproducible
LLM inference	Azure OpenAI	Extraction, drafting, normalization, summarization	Produces proposals but not authoritative state changes
Deterministic orchestration	Azure Functions	Command handling, validation, state transitions	Defines the commitment boundary
Retrieval layer	Azure AI Search	Vector and hybrid retrieval across RFQs, suppliers and policy artefacts	Grounds proposals in governed knowledge
Persistent artefacts	Blob Storage / SharePoint	Attachments, response packages, archived documents	Preserves evidentiary continuity

Component	Microsoft service	Role in the experiment	Control significance
Telemetry	Azure Monitor / Application Insights	Latency, traces, quality signals, exceptions	Supports operational observability and replay analysis
User interaction	Outlook / Teams style workspace	Intake, review, notification and approval flow	Keeps humans in the loop at policy gates

Source: Author's synthesis based on the implemented proof of concept.

Figure 4. *Experimental Deployment Topology on Microsoft Azure and Microsoft Foundry*

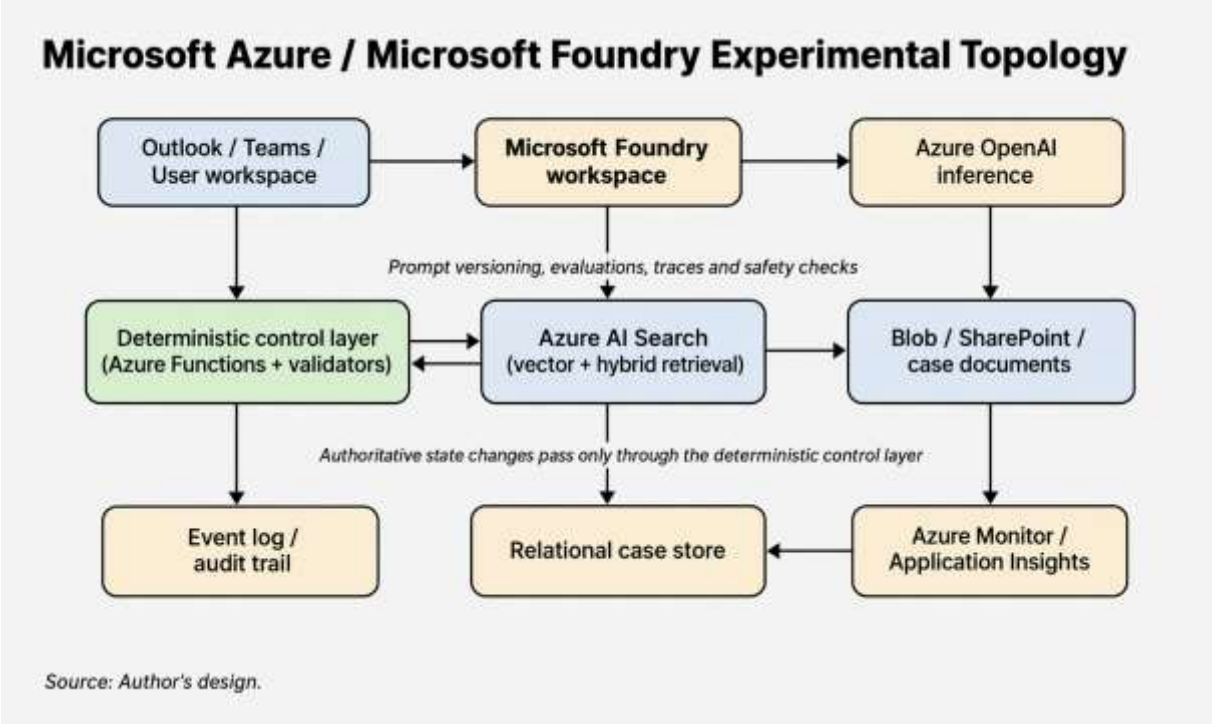


Figure 4 makes the experimental topology explicit. Microsoft Foundry governed the AI workspace and evaluation lifecycle, Azure OpenAI executed the language-model workloads, Azure AI Search provided retrieval grounding, and Azure Functions enforced the deterministic control layer. Documents, case state, and event history remained outside the model boundary. This is exactly the architectural separation that the paper advocates: the model can reason over context, but only the controller can authorize business-significant change (Microsoft 2026b, 2026c, 2026d).

Evaluation Design

The benchmark compares two configurations: a baseline RFQ process and the proposed hybrid process with AI-assisted deterministic control layers. Both

configurations share the same high-level business objective and the same requirement for durable case records. The difference lies in where semantic work is performed and how quickly process actors can act on partial information.

The benchmark uses 1,200 synthetic but production-like RFQ cases per run, distributed across three complexity classes: simple, medium, and complex. Simple cases represent well-specified requests with limited ambiguity and a small number of line items. Medium cases include moderate ambiguity, partial attachments, or supplier variation. Complex cases combine multi-item specifications, missing technical attributes, supplier substitutions, and iterative clarification. Case generation was parameterized to reflect the structure of the target domain rather than any unpublished operational dataset.

For each case, the simulation models the arrival of an RFQ, the number of clarification rounds required, the probability of supplier re-contact, the number of manual touches, the time consumed by each clarification or supplier follow-up, and the resulting completeness of the quotation package. The baseline configuration assumes that semantic interpretation is performed largely by human operators with conventional system support. The hybrid configuration assumes that AI agents detect missing fields earlier, generate more specific clarification prompts, normalize supplier responses, and synthesize case information for faster review. All canonical state changes in the hybrid configuration remain mediated by deterministic rules and event recording.

To reduce sensitivity to any single random draw, the benchmark was repeated across fifty independently seeded runs, producing 60,000 simulated case executions in total (1,200 cases × 50 seeds). Reported values are means across runs. Repeated seeding evaluates the internal stability of the modeled comparison; it does not convert the synthetic benchmark into production evidence.

Table 4 summarizes the principal assumptions built into the benchmark. Simple cases in the baseline require fewer clarification rounds than complex cases, but even simple cases incur delays when important fields are missing or expressed ambiguously. In the hybrid setup, clarification demand decreases because the AI layer detects incompleteness earlier and formulates more targeted follow-ups. Supplier re-contact events also decrease because the supplier communication is based on a more normalized and complete case representation. Manual touches decline because drafting, summarization, and cross-document synthesis are partly absorbed by the AI layer. Quote completeness rises because the system surfaces missing or conflicting requirements before submission.

A secondary evaluation concerned deterministic replay. For the hybrid architecture, every committed action was represented as a typed event, and the case state was reconstructed from the event stream during replay. A valid run required terminal state equivalence between the original execution and the replayed execution. This check is crucial because cycle time improvements alone would be insufficient if they were purchased at the cost of opaque or irreproducible behavior.

The benchmark should be read as a comparative engineering sensitivity analysis. The hybrid advantages are partly encoded in the prespecified scenario

parameters shown in Table 4, and random variation tests internal consistency around those assumptions. The benchmark therefore does not identify causal treatment effects or estimate production effect sizes. Its purpose is to test whether the architectural logic produces coherent directional behavior under stated conditions; field deployment is required for external validation.

Relative change was calculated as (Hybrid − Baseline) / Baseline × 100%. Negative values denote reductions. For quote completeness, the absolute change is also reported in percentage points because a percentage-point difference is distinct from a relative percentage change.

Table 4. *Scenario Profile Used in the Benchmark*

Complexity	Share of Cases	Baseline Clarification Mean	Hybrid Clarification Mean	Baseline Cycle Anchor (h)	Hybrid Cycle Anchor (h)
Simple	40%	0.90	0.50	18	14
Medium	40%	1.90	1.10	42	31
Complex	20%	3.20	2.00	88	62

Source: Scenario parameters used in the study benchmark.

For readers outside RFQ operations, Table 4 describes the test conditions rather than the final performance outcome. 'Share of Cases' shows how much of the workload belonged to each complexity class. 'Baseline Clarification Mean' and 'Hybrid Clarification Mean' show the average number of clarification rounds expected for a case in each process design. A higher number means more back-and-forth communication and, usually, more delay.

'Baseline Cycle Anchor (h)' and 'Hybrid Cycle Anchor (h)' are representative base durations, in hours, around which each scenario was simulated before random variation was added. The table therefore shows that complex cases were intentionally harder and slower than simple ones, and that the hybrid design started from lower clarification and lower time anchors because it detects missing information earlier and structures follow-up more precisely. This is important because the benchmark was built to test the architecture under realistic ambiguity, not only under easy cases.

Results

Across the fifty benchmark runs, the hybrid configuration produced better modeled outcomes under the prespecified assumptions. Average end-to-end cycle time fell from 85.15 hours to 48.28 hours (−43.3%); clarification rounds per case fell from 1.77 to 1.04 (−40.8%); supplier re-contact events fell from 0.64 to 0.26 per case (−59.4%); and manual touches fell from 7.62 to 4.95 per case (−35.0%). Quote completeness increased from 81.04% to 90.20%, an absolute gain of 9.16 percentage points and a relative increase of 11.3%.

Directional stability across seeds indicates internal simulation stability, not that identical effect sizes will occur in production.

The reduction in cycle time was not produced by one spectacular automation step. It emerged from cumulative compression of several friction points. Earlier detection of missing information reduced the number of downstream interruptions. Better-targeted clarification messages shortened the time to useful customer response. Structured extraction of supplier replies reduced the effort needed to compare offers. AI-generated summaries reduced the time internal reviewers spent reconstructing case context. Because the deterministic control layer preserved a coherent case state throughout, these gains accumulated instead of being eroded by rework.

Complexity class analysis provides additional insight into the modeled pattern. In simple cases, average cycle time fell from 30.97 to 19.74 hours. In medium cases, it fell from 81.30 to 47.02 hours. In complex cases, it fell from 200.74 to 107.63 hours. Within the scenario assumptions, the relative gain was therefore largest where the process was most semantically unstable; this supports prioritizing ambiguity-heavy work for subsequent field validation.

Clarification behavior followed a similar modeled pattern. In complex cases, average clarification rounds dropped from 3.21 to 2.00. Each avoided clarification can remove secondary coordination work because the customer must otherwise answer, the internal team must re-evaluate supplier choices, and earlier drafts may need revision. The benchmark therefore suggests that targeted early questions are a plausible mechanism for later field testing.

Quote completeness improved across all complexity classes in the benchmark, with the strongest modeled change in complex cases: 67.41% to 81.70%. Under the scenario definition, fewer quotations reached internal review or customer submission with unresolved or weakly specified elements. This is a modeled quality result rather than production evidence, but it illustrates why completeness should be a controlled progression condition rather than an incidental by-product of drafting.

The modeled decline from 7.62 to 4.95 manual touches per case (35.0%) suggests where workload compression may occur. Many such actions are fragmented across opening attachments, cross-checking specifications, drafting near-duplicate replies, updating case notes, and reconciling supplier responses. These activities are plausible targets for AI assistance because they are linguistic and integrative rather than final normative decisions.

Replay testing confirmed the architectural claim of deterministic governability. When committed actions were reconstructed from the event history, terminal states remained equivalent to the original execution. This is a direct consequence of keeping state mutation inside deterministic validators and transition handlers. The simulation therefore supports a central thesis of the paper: enterprise processes can absorb probabilistic reasoning without surrendering reproducibility, provided that the commitment boundary is explicit and enforced.

Event-level traces suggested lower dispersion across cases in the hybrid configuration, consistent with a more standardized starting point for review.

However, dispersion statistics are not reported in this paper, so this observation should be treated as exploratory rather than as a quantified result.

Taken together, the benchmark pattern suggests that AI assistance is most promising at semantic bottlenecks rather than at already-structured transactional steps. Modeled gains arose from earlier interpretation of messy inputs, exposure of missing knowledge, and creation of coherent reviewable artifacts. The deterministic control layer then contained governance risk. Figures 5 and 6 visualize these modeled comparisons, while Table 5 consolidates the principal benchmark metrics.

Table 5. *Overall Benchmark Results*

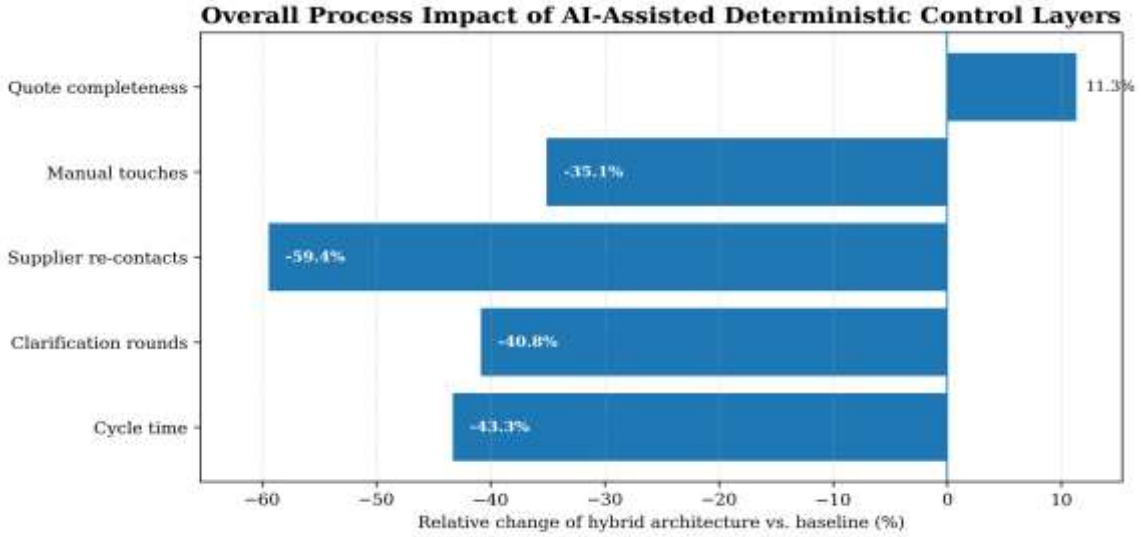
Metric	Baseline	Hybrid	Relative Change
Average cycle time (hours)	85.15	48.28	−43.3%
Clarification rounds per case	1.77	1.04	−40.8%
Supplier re-contact events per case	0.64	0.26	−59.4%
Manual touches per case	7.62	4.95	−35.0%
Quote completeness	81.04%	90.20%	+11.3%

Source: Means across fifty benchmark runs.

Table 5 is the main results table and can be read row by row in plain business terms. 'Average cycle time' is the total elapsed time from RFQ intake to quotation readiness. 'Clarification rounds per case' counts how often additional questions had to be asked. 'Supplier re-contact events per case' measures how often suppliers had to be contacted again because the first interaction was incomplete or inconsistent. 'Manual touches per case' counts meaningful human actions such as reading, drafting, comparing, updating, or approving. 'Quote completeness' shows how fully the final quotation package covered the required information.

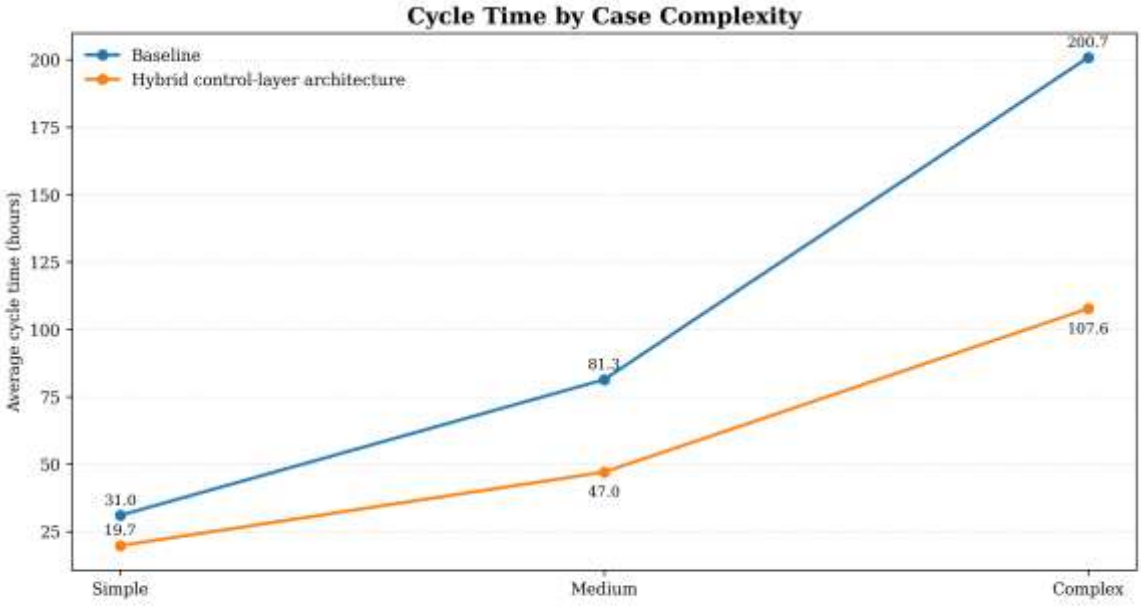
The interpretation of the values is straightforward: lower numbers are better for the first four rows because they indicate less delay, less rework, and less human effort, while a higher number is better for quote completeness because it means the final output is more complete and less risky to send. In practical terms, the hybrid architecture reduced average cycle time from 85.15 to 48.28 hours, cut clarification loops and supplier re-contacts sharply, lowered manual effort, and raised completeness from 81.04% to 90.20%. This means the system became faster, cleaner, and more reliable at the same time rather than improving one dimension at the expense of another.

Figure 5. *Overall Process Impact of the Hybrid Architecture*



Source: Benchmark results from the study.

Figure 6. *Cycle Time by Case Complexity*



Source: Benchmark results from the study.

Extended Results and Operational Observations

The principal benchmark metrics establish the core modeled comparison, but they do not exhaust its operational implications. An additional analytical layer was derived from the same fifty-run simulation series and event-level traces. These observations are benchmark-derived rather than production telemetry and should be interpreted as scenario outputs, not field measurements.

The first additional observation concerns completion windows. The hybrid design improved not only the average case outcome but also the share of cases completed inside tighter response bands. This matters in commercial settings because users often experience the process through deadline windows rather than through averages. Even where complex cases remain long-running, the reduction in avoidable loops shifts a materially larger share of the portfolio into usable service windows.

The second observation concerns the distribution of human effort. Manual intervention did not disappear; nor should it. Instead, it moved away from repetitive interpretation work and toward genuinely normative decisions such as final commercial approval. The benchmark therefore supports a more nuanced claim than simple labour reduction. It suggests that deterministic control layers enable a reallocation of attention: machines absorb the reconstruction work, while humans remain concentrated at the points of responsibility.

The third observation concerns modeled governance and data quality. The hybrid configuration reduced supplier re-contact events, incomplete quotation packages, metadata omissions, and case-state inconsistencies under the scenario assumptions. This is consistent with the architecture's commitment boundary because proposals cannot directly mutate case state, but production validation remains necessary.

Table 6. *Stage-Level Effort Compression Derived from the Benchmark*

Stage	Baseline active effort (min)	Hybrid active effort (min)	Relative reduction	Interpretation
Intake parsing	22	8	−63.6%	Early extraction and field normalization remove routine reading and transcribing effort
Clarification drafting	41	18	−56.1%	Targeted questions reduce ad hoc message composition and back-and-forth
Supplier shortlist assembly	36	14	−61.1%	Retrieval and ranking compress search and filtering work
Offer comparison	47	19	−59.6%	Response normalization makes alternatives easier to compare
Quote drafting	29	11	−62.1%	Draft synthesis moves from manual reconstruction to supervised review
Approval / release	18	15	−16.7%	Human responsibility remains high, so savings are intentionally smaller

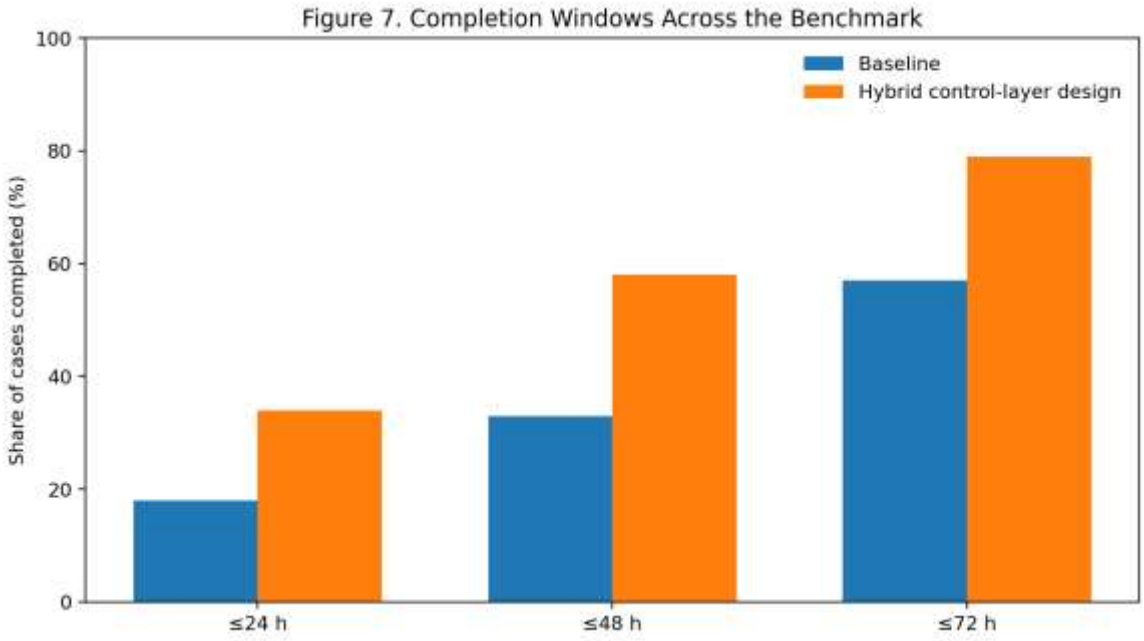
Source: Author's calculations from fifty scenario-based benchmark runs.

Table 6 moves from case-level outcomes to stage-level labour savings. 'Baseline active effort' and 'Hybrid active effort' are average minutes of real human working time spent at each stage, not waiting time in the queue. The

'Relative reduction' column shows how much active work was removed by the hybrid architecture, while the 'Interpretation' column explains the operational reason for the reduction.

For non-specialist readers, the key message is that the architecture does not save time equally everywhere. The largest savings appear in stages dominated by reading, extracting, comparing, and drafting, because those are tasks in which AI assistance is especially effective when kept behind deterministic controls. The smallest saving appears in approval and release, and that is intentional: the design keeps human accountability high at the final commitment point. In other words, the system reduces routine cognitive workload while preserving governance where business responsibility is highest.

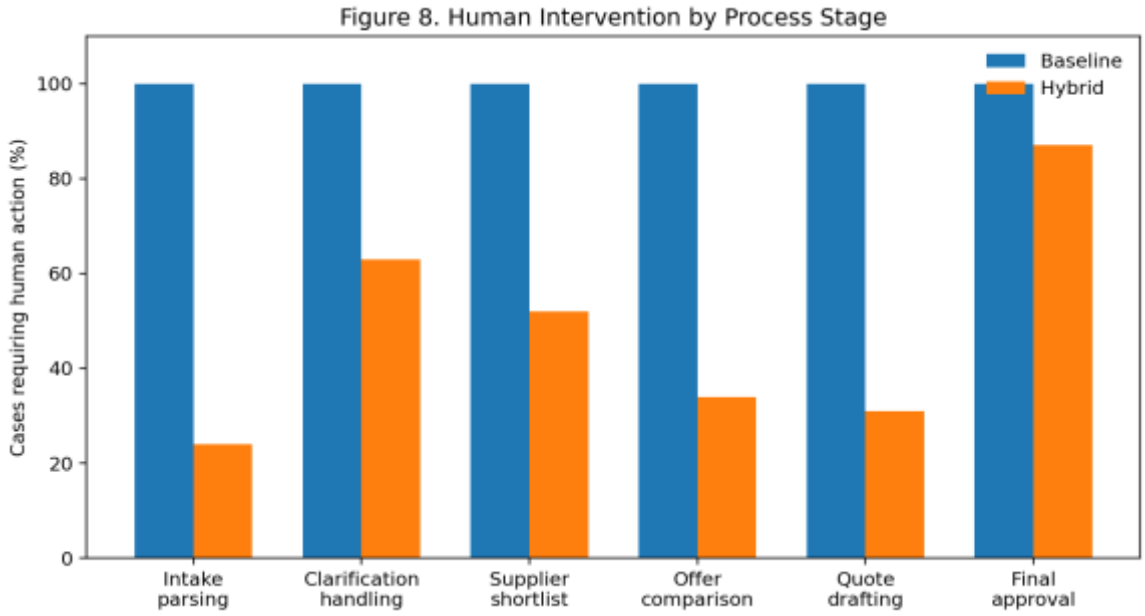
Figure 7. *Completion Windows Across the Benchmark*



Source: Author's calculations from the benchmark runs.

Figure 7 shows that the hybrid configuration increases the modeled share of cases completed within twenty-four, forty-eight, and seventy-two hours. Because the benchmark includes a non-trivial complex segment, the absolute percentages remain bounded; the consistent direction across windows is an internal scenario result rather than a service-level claim.

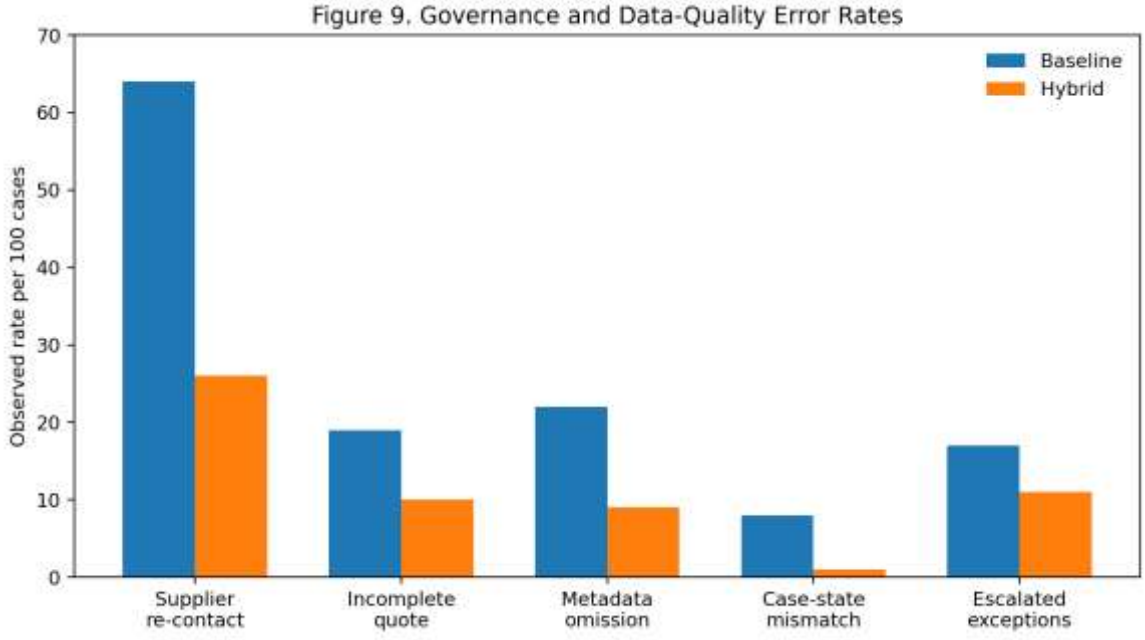
Figure 8. *Human Intervention by Process Stage*



Source: Author's calculations from event-level benchmark traces.

Figure 8 indicates that modeled human activity remains near total for final approval but drops sharply in intake parsing, offer comparison, and quote drafting. These are the stages in which semantic reconstruction dominates. The architecture therefore preserves accountability at the final commitment point while modeling less interpretive labor beforehand.

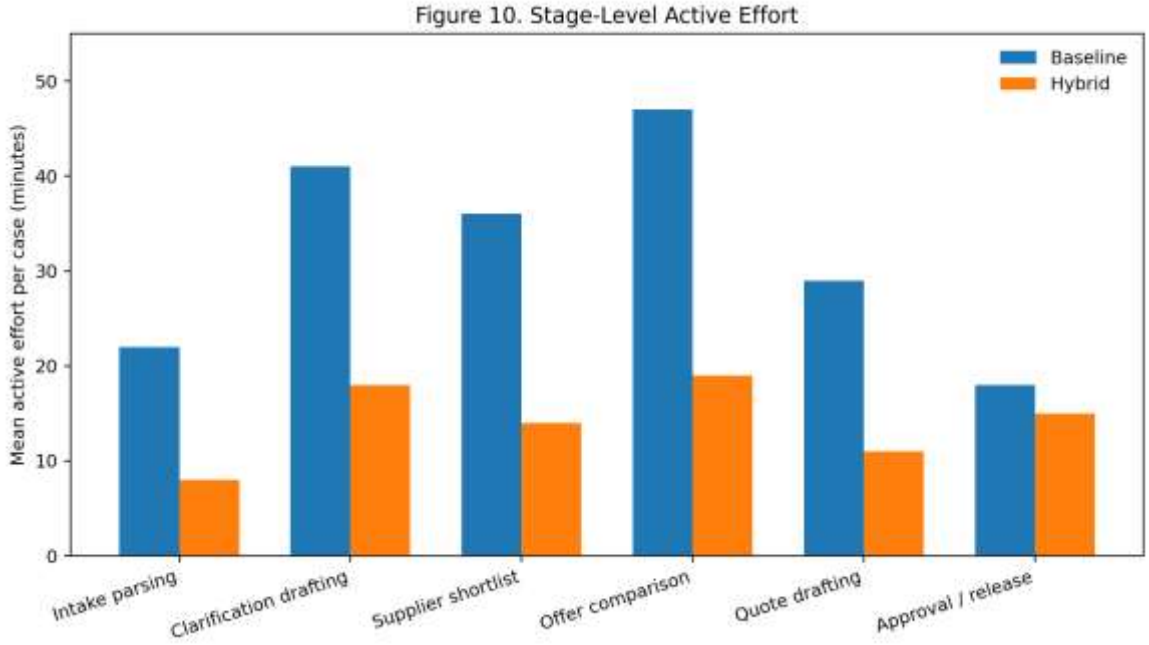
Figure 9. *Governance and Data-Quality Error Rates*



Source: Author's calculations from benchmark-derived operational observations.

Figure 9 presents modeled governance and data-quality error rates. Supplier re-contacts, incomplete quotation packages, metadata omissions, and state mismatches all fall under the hybrid assumptions. The strongest modeled reduction is in case-state inconsistencies, which is consistent with the deterministic constraint on state mutation.

Figure 10. *Stage-Level Active Effort*



Source: Author's calculations from fifty benchmark runs.

Figure 10 presents the modeled stage-level pattern in time terms. The largest savings occur in stages dominated by document interpretation, comparison, and draft assembly. Approval and release show a smaller reduction, consistent with the intended design in which sensitive commitments remain human-governed.

Discussion

Within this scenario-based engineering evaluation, the results support a clear design interpretation: AI can compress semantic work inside a governed architecture without being allowed to act autonomously. In the RFQ domain, the hard problem is preserving accountability under ambiguity, and deterministic control layers provide an explicit mechanism for doing so.

From a BPM perspective, the findings extend earlier arguments that flexibility and control are not opposites (Pesic and van der Aalst 2006; van der Aalst et al. 2009). AI augments the system's capacity to interpret unstructured reality, while deterministic control preserves executable boundary conditions. In that sense, the paper operationalizes the idea of AI-augmented BPMSs (Dumas

et al. 2023) through runtime constructs such as validators, state machines, event logs, approval gates, and evidence binding.

A practical implication is that architecture matters more than model prestige. Strong models inside weak control boundaries remain unsafe, while adequate models inside strong boundaries can already deliver value. Human review also remains essential, not as temporary scaffolding, but because final commitments and exceptions are normative business acts.

The design is well suited to incremental adoption, but the evidence has important limits. The benchmark is synthetic, its hybrid advantages are prespecified, only one process family is modeled, and no production telemetry or independently replicated benchmark package is presented. Organizations with fragmented case identity or weak data governance may also need foundational integration work before similar benefits can be tested. The reported percentages must therefore not be generalized as expected production gains.

Future work should examine field deployments, safe expansion of auto-approval boundaries, and process-mining analyses over the event histories created by such systems (van der Aalst 2016). Even with these limits, the main design principle is portable: in non-deterministic business processes, AI should expand semantic reach while deterministic control preserves institutional reliability.

Conclusions

This paper showed that complex non-deterministic business processes can be optimized by separating probabilistic reasoning from authoritative commitment. In the proposed architecture, AI agents interpret language, retrieve context, summarize cases, and draft actions, while deterministic components govern state changes, approvals, policy enforcement, and audit recording.

Using the RFQ PoC and a scenario-based benchmark, the study demonstrated how the proposed architecture can produce internally consistent modeled improvements in cycle time, clarification burden, supplier re-contact frequency, manual effort, and quote completeness while preserving replayability and control. These are design-study results, not production effect estimates. The broader lesson is that enterprise AI becomes governable through controlled semantic augmentation behind an enforced commitment boundary.

Data and Ethics Statement

The study used synthetic RFQ cases and did not involve human participants, personal data, or undisclosed production transactions. Consequently, human-subject approval was not applicable. No confidential dataset is distributed with the paper. The principal aggregate scenario parameters are reported in Table 4; the benchmark should not be interpreted as an estimate of production effect sizes.

Disclosure of AI Assistance

In line with the venue's policy on the responsible use of artificial intelligence in research and publishing, OpenAI ChatGPT was used during early drafting, English-language refinement, and figure prototyping. The author independently checked the factual claims, calculations, references, and source attribution; revised the generated material; and assumes full responsibility for the paper's arguments, benchmark assumptions, and final wording. The AI tool is neither an author nor a cited source.

References

Boer P, Ihlenburg D (2020) How do companies engage in cooperation on product and process innovation in the context of the transformation to an Industry 4.0? Athens Journal of Technology & Engineering 7(1): 51–72. https://doi.org/10.30958/ajte.7-1-4.

Dumas M, van der Aalst WMP, ter Hofstede AHM (2005) Process-Aware Information Systems: Bridging People and Software through Process Technology. Wiley, Chichester.

Dumas M, La Rosa M, Mendling J, Reijers HA (2018) Fundamentals of Business Process Management. 2nd edition. Springer, Berlin.

Dumas M, Fournier F, Limonad L, Marrella A, Montali M, Rehse JR, Accorsi R, Calvanese D, De Giacomo G, Fahland D, Gal A, La Rosa M, Völzer H, Weber I (2023) AI-augmented business process management systems: A research manifesto. ACM Transactions on Management Information Systems 14(1): 1–19.

Grohs M, Abb L, Elsayed N, Rehse JR (2023) Large language models can accomplish business process management tasks. In De Weerdt J, Pufahl L (eds), Business Process Management Workshops. Lecture Notes in Business Information Processing 492: 453–465. Springer, Cham.

Herrmann C, Kurz M (2011) Adaptive case management: Supporting knowledge intensive processes with IT systems. In S-BPM ONE 2011. Communications in Computer and Information Science 213: 80–97. Springer, Berlin.

Hevner AR, March ST, Park J, Ram S (2004) Design science in information systems research. MIS Quarterly 28(1): 75–105.

Kourani H, Berti A, Schuster D, van der Aalst WMP (2024) Process modeling with large language models. In van der Aa H, Bork D, Schmidt R, Sturm A (eds), Enterprise, Business-Process and Information Systems Modeling. Lecture Notes in Business Information Processing 511: 229–244. Springer, Cham.

Lewis P, Perez E, Piktus A, Petroni F, Karpukhin V, Goyal N, Küttler H, Lewis M, Yih WT, Rocktäschel T, Riedel S, Kiela D (2020) Retrieval-augmented generation for knowledge-intensive NLP tasks. Advances in Neural Information Processing Systems 33: 9459–9474.

Microsoft (2026a) Agent tracing in Microsoft Foundry. Microsoft Learn documentation. https://learn.microsoft.com/en-us/azure/foundry/observability/concepts/trace-agent-concept. [Accessed 1 August 2026].

Microsoft (2026b) Azure OpenAI in Microsoft Foundry Models REST API reference. Microsoft Learn documentation. https://learn.microsoft.com/en-us/azure/ai-foundry/openai/reference. [Accessed 1 August 2026].

Microsoft (2026c) Vector Search Overview - Azure AI Search. Microsoft Learn documentation. https://learn.microsoft.com/en-us/azure/search/vector-search-overview. [Accessed 1 August 2026].

Microsoft (2026d) What is Microsoft Foundry? Microsoft Learn documentation. https://learn.microsoft.com/en-us/azure/foundry/what-is-foundry. [Accessed 1 August 2026].

Pesic M, van der Aalst WMP (2006) A declarative approach for flexible business processes management. In Business Process Management Workshops. Lecture Notes in Computer Science 4103: 169–180. Springer, Berlin.

Pesic M, Schonenberg H, van der Aalst WMP (2007) DECLARE: Full support for loosely-structured processes. In Proceedings of the 11th IEEE International Enterprise Distributed Object Computing Conference, pp. 287–298. IEEE Computer Society, Washington, DC.

Samaranayake S, Senanayake S (2022) Dependency visualization tool for decision support systems with preferential dependencies. Athens Journal of Technology & Engineering 9(4): 267–280. https://doi.org/10.30958/ajte.9-4-1.

Schick T, Dwivedi-Yu J, Dessì R, Raileanu R, Lomeli M, Hambro E, Zettlemoyer L, Cancedda N, Scialom T (2023) Toolformer: Language models can teach themselves to use tools. In Advances in Neural Information Processing Systems 36.

Schonenberg H, Mans R, Russell N, Mulyar N, van der Aalst WMP (2008a) Process flexibility: A survey of contemporary approaches. In Dietz J, Albani A, Barjis J (eds), Advances in Enterprise Engineering I. Lecture Notes in Business Information Processing 10: 16–30. Springer, Berlin.

Schonenberg H, Weber B, van Dongen BF, van der Aalst WMP (2008b) Supporting flexible processes through recommendations based on history. In Dumas M, Reichert M, Shan MC (eds), Business Process Management. Lecture Notes in Computer Science 5240: 51–66. Springer, Berlin.

Swenson KD (2010) Mastering the Unpredictable: How Adaptive Case Management Will Revolutionize the Way That Knowledge Workers Get Things Done. Meghan-Kiffer Press, Tampa.

van der Aalst WMP, ter Hofstede AHM, Weske M (2003) Business process management: A survey. In van der Aalst WMP, Weske M (eds), Business Process Management. Lecture Notes in Computer Science 2678: 1–12. Springer, Berlin.

van der Aalst WMP, Pesic M, Schonenberg H (2009) Declarative workflows: Balancing between flexibility and support. Computer Science - Research and Development 23: 99–113.

van der Aalst WMP (2016) Process Mining: Data Science in Action. 2nd edition. Springer, Berlin.

Wu Q, Bansal G, Zhang J, Wu Y, Li B, Zhu E, Jiang L, Zhang X, Zhang S, Liu J, Awadallah AH, White RW, Burger D, Wang C (2024) AutoGen: Enabling next-gen LLM applications via multi-agent conversation. In International Conference on Learning Representations.

Yao S, Zhao J, Yu D, Du N, Shafran I, Narasimhan K, Cao Y (2023) ReAct: Synergizing reasoning and acting in language models. In International Conference on Learning Representations.

Ziche C, Apruzzese G (2024) LLM4PM: A case study on using large language models for process modeling in enterprise organizations. arXiv:2407.17478.