International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(6s) 2026 pp. 609-616
https://doi.org/10.70917/ijcisim-2026-2960
Article

Bridging Deterministic Workflows and Stochastic Reasoning: A Multi-Agent Reference Architecture for Autonomous Telecom E-Commerce Buyflows

Rajasekhar Vetukuri
Independent Researcher, E-Mail- v.rajasekharraju@gmail.com

Abstract: Telecommunications Service Providers (Telecom Operators) needed to accommodate richer, heterogeneous & higher-variance e-commerce buyflows. Deterministic, BPMN-based orchestration continued to be needed for traceability/control of committed transactions. But baked-in contextual modeling wasn't designed to reason about ambiguous/personal shopper intent. Alternatively, LLM agents could bring contextually-driven reasoning but struggled to satisfy reliability/governance/compliance requirements. A Middle Ground emerged in the form of Multi-Agent Reference Architectures (MARA) anchored by a Deterministic Execution Guardrail (DEG). This research decouples intent discovery from commitment to transaction and introduces event-driven state synchronization to limit stale-state failures. It was stress-tested in a simulated environment with telecom-op style bundle mixins, compatibility traps, state drift, etc. MARA was able to deliver ~92% completion, reduce fallout by ~67% compared to a deterministic baseline, and hit 685 ms p99 latency while maintaining deterministic compliance via encoded guardrail checks. This validates guarded agentic orchestration as a viable approach for regulated telecom buyflows but will require further validation in operator-grade conditions.

Keywords: autonomous agents; business support systems; business process management; e-commerce buyflows; multi-agent systems; service orchestration; telecom order management; workflow orchestration.

1. Introduction

The telecommunications sector was transformed during the migration from monolithic, connectivity-centric operating models toward digital service orchestration, in which broadband, mobile, media, IoT, and edge offerings were composed across heterogeneous Business Support Systems (BSS) and Operations Support Systems (OSS). In that setting, deterministic BPMN engines were widely used because auditable execution, transactional traceability, and controllable long-running processes were required On the other hand, recent BPM publications announced it was being reinvented through AI-driven, conversational, and adaptive patterns of execution. This revealed limitation of brittle, flow-chart-centric approaches to orchestration when faced with high-variance demand [1] [2].

Significant advances were announced around LLM-based autonomous agents, tool-using agents, and collaboration between agents. Agent capability surveys described patterns of planning, memory, reflection, coordination, and tool invocation. Weaknesses continued to be reported around reliability, reproducibility, tool choice, and enterprise governance. Workflow-specific studies further showed that large language models were not dependable workflow orchestrators out of the box and typically required additional structure, supervision, or constraints to perform reliably on orchestration tasks [3]–[5].

Those limitations were amplified in telecom e-commerce buyflows. Telecom order management and service-order orchestration required compatibility checking, eligibility validation, inventory synchronization, pricing integrity, and downstream activation across customer-facing and production-facing service layers. Intent-based service orchestration and telecom architecture references emphasized that modern order operations depended on standards-based orchestration across service delivery, inventory, and activation interfaces [6], [7].

A research gap was therefore identified in the absence of a rigorous architecture for reconciling stochastic customer behavior and LLM-based reasoning with the deterministic requirements of telecom billing, provisioning, and compliance control. Although recent work had examined intent-based service orchestration at the network and infrastructure layers, a formal commerce-layer reference architecture integrating stochastic proposal generation with deterministic enterprise enforcement remained insufficiently developed. The problem was thus defined as the minimization of buyflow fallout probability under hard constraints on pricing, policy, inventory, compliance, and latency.

Three principal contributions were made. First, a dual-layer orchestration model was formalized in which stochastic intent reasoning was separated from transaction commitment. Second, the Deterministic Execution Guardrail (DEG) was defined not merely as a validation module, but as a formal admissibility operator that restricted enterprise execution to configurations satisfying pricing, compatibility, inventory, policy, and compliance invariants. Third, an event-driven state synchronization mechanism was incorporated so that the admissibility decision could be evaluated against near-real-time enterprise state. Those contributions were designed to establish a guarded autonomy pattern suitable for regulated telecom buyflows rather than unconstrained end-to-end agentic execution [8].

2. Literature Survey

Research relevant to autonomous telecom buyflows fell into four streams. The first stream concerned deterministic business-process orchestration. Classical BPMN and service-oriented orchestration approaches remained valuable because they provided traceability, repeatability, and auditable control over long-running enterprise transactions. Recent BPM scholarship, however, argued that AI was shifting the field away from purely static process models toward more dynamic, contextual, and conversational modes of execution [1], [2].

The second stream concerned LLM-based autonomous agents and tool-using agent systems. Recent surveys described a common agent architecture composed of planning, memory, tool use, feedback, and multi-step reasoning, and they showed that such systems could improve task flexibility under open-ended input. At the same time, those surveys identified persistent weaknesses in reliability, generalization, tool selection, coordination overhead, and reproducibility [3], [9]–[11].

The third stream concerned the intersection of LLMs and workflow-oriented process execution. Recent work such as WorkflowLLM and business-process-modeling evaluations suggested that general-purpose foundation models were not dependable workflow orchestrators out of the box. Performance gains could be realized through the introduction of workflow structure, schema constraints, intermediate representations, and specialized supervision [4] [5].

The final research stream related responsible AI with enterprise process governance. Some of the most recent BPM-inspired responsible-AI literature had made the case that trustworthy process systems required well-defined control points, audit-logging, and policy-enforcement boundaries. That thesis resonated strongly with telecom buyflows which operated under strict requirements around pricing accuracy, PII/PCI restrictions, inventory validity, and O2A state synchronization. Prior research supported the notion of hybrid autonomy but a telecom-commerce reference architecture had yet to be defined that formalized responsibility-boundaries between stochastic and deterministic processing [8].

3. Methodology

A Design Science Research (DSR) methodology was adopted because the central output was an architectural artifact intended to solve a high-stakes enterprise problem rather than merely explain an observed

phenomenon. In accordance with established DSR guidance, the study was organized into problem identification, objective definition, artifact design, implementation, demonstration, and evaluation [12], [13].

Methodological rigor was preserved through fixed scenario-generation procedures, controlled random seeds, common catalog structures across all baselines, and explicit metric definitions established before experimentation. Those controls strengthened internal validity and reproducibility, while telecom-specific constraints and O2A-style outcomes improved construct validity relative to generic agent benchmarks [12], [13].

3.1 System Model and Problem Formulation

The telecom buyflow was conceptualized as a constrained state-transition system. Let $i \in$ represent a customer-intent instance consisting of natural language utterance and session context. Let $P = \{p1, \ldots, pn\}$ represent the product

catalog. Every product pk has an associated deterministic constraint set Ck that includes rules related to compatibility, pricing, eligibility, inventory, and compliance. Let x ∈ £ represent the enterprise state such as customer profile, credit status, serviceability, and realtime inventory. Let s ∈ s represent a candidate configuration, where a configuration represented both a bundle proposal as well as an execution path.

A binary routing policy α(i, x) ∈ {0, 1} was defined where α(i, x) = 0 iff the request could be considered low-variance and resolved with a deterministic BPMN path, and α(i, x) = 1 iff the request was high-variance and required some degree of agentic reasoning.

3.2 Deterministic Execution Guardrail as an Admissibility Operator

Candidate configurations x were considered executable iff they satisfied the Deterministic Execution Guardrail predicate G(s, x) = 1. We presented the DEG as an admissibility operator over candidate configurations rather than introducing it as middleware validation subroutine.

Formula 1: G(s, x) = Gcompat(s, x) A Gpolicy(s, x) A Gpricing(s, x) A Ginv(s, x) A Gcomp(s, x).

In Eq. 1 we used decomposition to assign modular responsibilities where Gcompat enforced feasibility constraints across the product-cartesian-product, Gpolicy checked business rules and credit guardrails, Gpricing enforced tariff integrity rules, Ginv enforced inventory validity rules, and Gcomp enforced encoded compliance rules. The DEG function blocked transitions out of the reachable state space s that violated execution-safety invariants encoded across these five system dimensions.

We therefore defined the DEG-guarded configuration subspace as sG = { s ∈ s | G(s, x) = 1 }. Because enterprise execution was constrained to sG, we could equivalently describe the DEG as a projection operator ΠG: s → sG that projects the unconstrained stochastic proposal space to an execution-safe subset prior to commit.

Proposition 1 (Execution Safety Under Guarded Commitment): If s ∈ s was generated by the reasoning layer, execution layer commit was only allowed if G(s, x) = 1. It follows that, under strong implementation of the guardrail predicates, the execution layer could not commit a configuration that violated any encoded compatibility, pricing, inventory, policy, or compliance rules. (Proof directly follows from the architectural invariant that all commit transitions must pass through the DEG acceptance function.).

3.3 Utility Optimization

A request-specific utility function was defined to balance intent relevance, business value, latency cost, and operational risk.

Formula 2: U(s | i, x) = λ1 R(s, i) + λ2 M(s) − λ3 L(s) − λ4 Q(s)

where R(s, i) represented intent relevance, M(s) represented business value, L(s) represented latency cost and Q(s) represented operational risk. The optimization problem was then stated over the guarded feasible set sG so that utility maximization remained subordinate to execution safety.

Formula 3: s* = arg max{s ∈ sG} U(s | i, x) subject to L(s) ≤ τmax

The fallout probability was defined as the likelihood that either the DEG rejected the proposal or downstream activation failed after orchestration.

Formula 4: Pf = Pr[G(f(i, x), x) = 0 ∨ A(f(i, x), x) = 0]

where f represented overall routing decisions and A(·) represented downstream success rate. MARA's goal was thus to minimize Pf subject to guardrail and latency constraints. Defining feasibility in terms of G allowed the DEG to shape the allowed response space of the system as opposed to rejecting results posterior.

4. Proposed Architecture

The proposed MARA architecture was organized into four layers to isolate stochastic reasoning from enterprise commit authority and to maintain modularity under telecom-grade constraints. Fig. 1 presents the proposed architecture and highlights the role of the DEG as the formal admissibility boundary between the reasoning layer and enterprise commit paths.

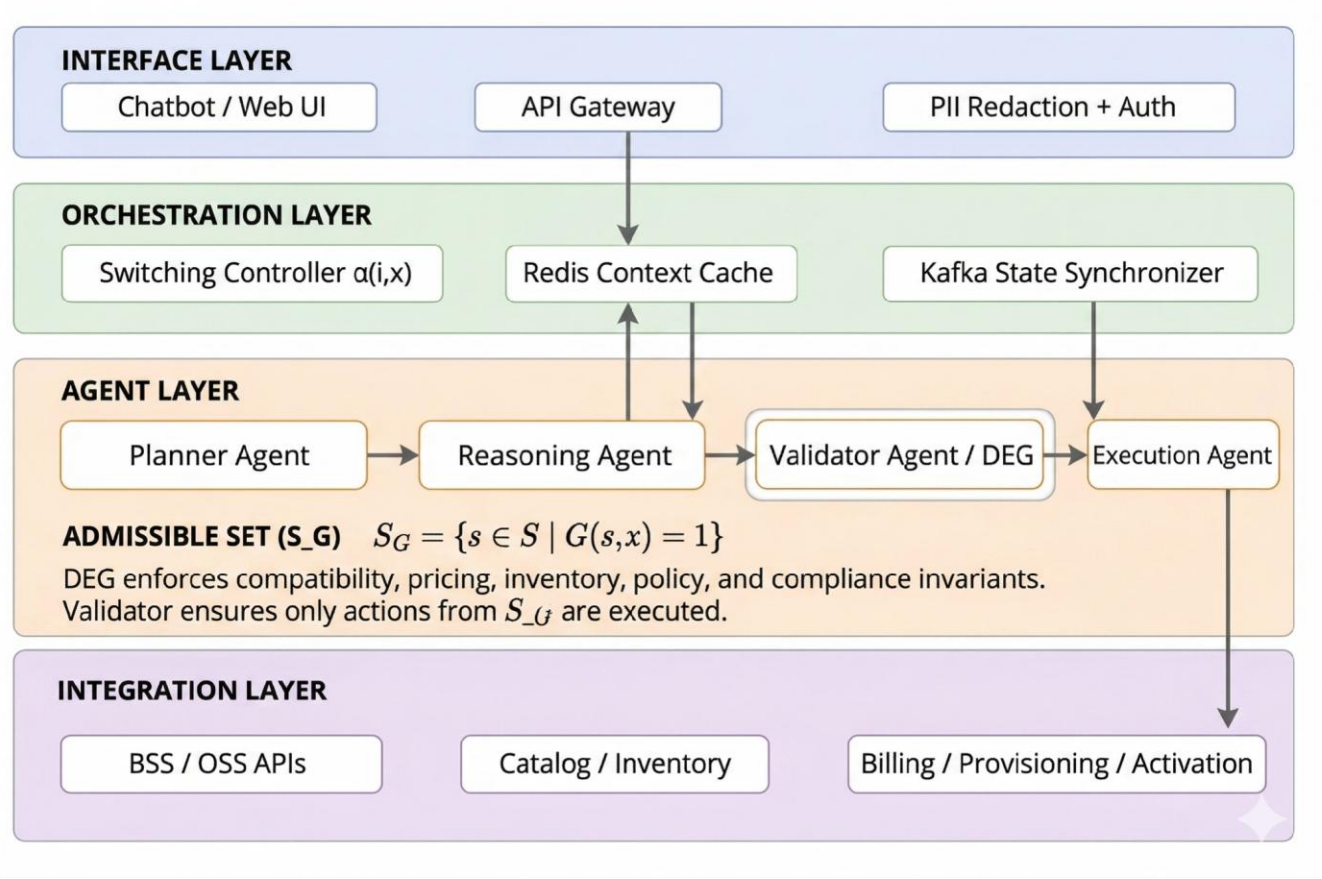

Figure 1: Proposed MARA architecture for guarded telecom buyflows.

4.1 Interface Layer

The interface layer handled multimodal entry through chatbots, web channels, and API gateways. Edge preprocessing was applied for authentication, session binding, and personally identifiable information

redaction before data reached the reasoning layer. That separation reduced exposure of sensitive attributes to stochastic models and aligned the architecture with responsible-AI control principles [8].

4.2 Orchestration Layer

The orchestration layer contained a Switching Controller that implemented the routing policy α. A request-complexity score was computed from ambiguity, breadth of product span, prior path similarity, and exception likelihood. Low-variance requests were routed to deterministic BPMN flows, whereas high-variance requests were routed to the agent layer. A context cache, backed by Redis, retained previously validated configuration motifs. A State Synchronizer subscribed to Kafka events so that the reasoning context could be updated with near-real-time inventory, credit, and serviceability state [6], [7].

4.3 Agent Layer

The agent layer used a hierarchical multi-agent system. A Planner Agent decomposed customer intent into ordered subgoals. A Reasoning Agent produced candidate bundle configurations under the current mirrored state. A Validator Agent implemented the DEG as the formal admissibility operator over candidate configurations. Rather than serving as a post hoc rule checker, the DEG constrained the reachable execution space by enforcing execution-safety invariants across compatibility, pricing, inventory, policy, and compliance dimensions. Only configurations belonging to the admissible set were exposed to the Execution Agent, which alone was authorized to invoke downstream commit APIs after guardrail acceptance. This separation confined stochastic reasoning to proposal generation while reserving state mutation for deterministic control [3], [10].

4.4 Integration Layer

The integration layer translated agent-generated typed JSON payloads into the REST, SOAP, or event contracts required by legacy BSS/OSS systems. In addition, the layer published execution outcomes back to Kafka so that the mirrored state could remain synchronized with downstream enterprise reality.

4.5 Algorithmic Flow

The operational flow of MARA was defined as follows: (1) a request envelope was ingested and preprocessed; (2) authentication, session binding, and PII redaction were applied; (3) the routing score $\alpha(i, x)$ was computed; (4) if $\alpha = 0$, deterministic BPMN execution was selected; (5) if $\alpha = 1$, the Planner Agent decomposed intent into subgoals; (6) the Reasoning Agent generated m candidate bundle configurations; (7) each candidate was validated by the DEG; (8) the highest-utility feasible candidate was selected; (9) the Execution Agent invoked downstream commit APIs; and (10) execution outcomes were published to Kafka for state synchronization.

5. Implementation

The prototype was constructed as a microservice-based system. Java 21 and Spring Boot 3.3 were used for the DEG services, policy adapters, and commit interfaces. Camunda 8 was used for deterministic workflow execution. LangGraph-style orchestration semantics were used for the multi-agent control loop. Apache Kafka 3.6 was used for event streaming and mirrored-state synchronization, while Redis 7 was used for transient context caching and validated-path reuse.

Two model roles were instantiated. GPT-4o-mini was assigned to reasoning-intensive candidate generation, while a local Llama-3-8B class model was assigned to planner-style decomposition in order to reduce cost and latency. Model outputs were normalized into a typed JSON schema. Schema violations, missing attributes, or malformed tool arguments were rejected before the DEG rule engine was entered.

The DEG itself was realized through a rule-based constraint enforcement subsystem instantiated using Drools and containing more than 250 service-compatibility and policy-validation rules. This rule base encoded hardware-plan compatibility, add-on eligibility, pricing restrictions, and inventory admissibility. The implementation choice was incidental to the architecture: the essential role of the DEG was to enforce admissibility over candidate configurations prior to commit, independent of the particular rule-engine technology employed. Parameter tuning was performed through pilot runs over the routing threshold θ, candidate count m, and cache time-to-live tttl. We chose $\theta = 0.62$, $m = 5$, and $tttl = 300$ s for the final system because those parameters minimized p99 latency without harming completion rate on our pilot cluster.

Random cancellations emulated state drift within the reasoning window. The circuit-breaker pattern causes any running engine to hard-reset to its safe-state recommendation when reasoning takes longer than ttl.

6. Experimental Setup

We ran all experiments on a Kubernetes cluster whose worker nodes were equipped with 16 vCPU cores, 64 GB of RAM, and NVIDIA L4-class GPUs to serve ML models. Kafka and Redis each ran on dedicated worker nodes. Ubuntu 22.04 served as OS baseline, and we used OpenJDK 21, Python 3.11 language wrappers for model-serving integration, Spring Boot 3.3, Camunda 8, Redis 7, and Kafka 3.6.

We synthesized a telecom-specific benchmark. No public dataset offers a holistic combination of buyflow intent, bundle compatibility decisioning logic, downstream activation results, and labeled policy violation events together in a single benchmark. Our scenario engine generates 5,000 possible telecom buyflow scenarios including plan renewals, bundle upgrades, add-device, attach-IoT, and 5G-media bundling. We injected 20% invalid scenarios containing unseen compatibility traps, 15% sessions exhibiting inherent intent ambiguity, and 30% sessions with simulated state drift occurring between proposal generation and activation time.

We benchmarked three logical systems. Baseline A was purely deterministic: BPMN-only system with no stochastic guardrails or propose-gen behavior. Baseline B was purely stochastic: guardrail-free agent system. Our MARA system featured a best-of-both-worlds approach wherein stochastic actors propose candidate offers and deterministic guardrails execute orders. During each run of the experiment, we sampled 1,000 scenarios from this pool of 5,000 and repeated the entire experiment 10 times using different random seeds.

Completion rate, fallout ratio, compliance rate, and p99 latency were measured during each simulation condition. Completion rate metrics answer the question: what percentage of sessions successfully confirmed an order? Fallout ratio answers the question: what percentage of sessions fail at DEG validation or downstream activation after a proposal was generated? Compliance rate measures were determined by passing each completed session against the encoded guardrail rules in the DEG. Inferential statistics are reported as mean ± std. across runs, and we report p-values from a two-sided test of difference in completion rate between MARA and BPMN-only conditions across all 10 runs with $\alpha = 0.01$.

7. Results

The MARA hybrid system had both the highest completion rate and lowest fallout ratio across tested systems when evaluated on the synthetic telecom buyflow benchmark. Relative performance is displayed in Table I; results here are intended to represent simulation results, not field-measured operator level performance.

Table I: Comparative System Performance in the Synthetic Telecom Buyflow Benchmark

Architecture	Completion Rate (%)	p99 Latency (ms)	Fallout Ratio	Compliance Rate (%)
BPMN-Only	**78.2 ± 1.4**	**118 ± 10**	**0.12 ± 0.02**	**100**
Pure Agentic	64.5 ± 3.1	2480 ± 380	0.27 ± 0.05	83 ± 4
MARA (Hybrid)	91.8 ± 0.8	685 ± 72	0.04 ± 0.01	100

Note: Values are reported as mean ± standard deviation across 10 seeded runs. Each run evaluated 1,000 scenarios sampled from a 5,000-scenario synthetic telecom buyflow pool. Compliance rate refers to conformance with the encoded DEG policy set in the simulated environment.

The improvement of MARA over BPMN-only was statistically significant ($p < 0.01$). Compared to BPMN-only, MARA decreased observed fallout ratio from 0.12 to 0.04, equivalent to a ≈66.7% reduction in the simulated environment. Compared to the stochastic-only baseline, MARA boosted completion as well as encoded compliance, and maintained p99 latency under 1 second.

Ablation study further broke down impact of major architectural elements. From Table II we can see that removal of DEG decreased success rate by 15.6%, while removal of Kafka-based synchronization hurt success rate by 18.4%.

Table Ii: Ablation Analysis of Success Rate under Controlled Component Removal

Configuration	Success Rate (%)	Δ from Full MARA
Full MARA	91.8 ± 0.8	—
Without DEG (Validator)	76.2 ± 2.1	−15.6%
Without Kafka Synchronization	73.4 ± 1.9	−18.4%

Note: Ablation values are reported relative to the full MARA configuration under the same seeded synthetic workload. The observed reductions reflect the removal of control mechanisms rather than retuning of the remaining pipeline.

The ablation results indicated that deterministic guarded validation and event-driven state alignment were first-order contributors to reliability rather than secondary implementation conveniences.

8. Discussion

Results showed that telecom buyflows required hybrid guarded autonomy, as both rigid deterministic orchestration and unconstrained agentic execution performed worse. The BPMN-only baseline-maintained auditability and enforced encoded compliance but lost out on completion as ambiguous/underspecified requests were rejected at a higher rate. Although the stochastic only-baseline sampled from a larger space of proposals, it suffered dramatically worse latency, poorer compliance with regards to our modeled checks, and experienced more fallout. By contrast, MARA demonstrated how deterministic safety could be preserved while also recovering valid configurations that were missed by static flows [1], [3].

Our ablation study showed that the DEG should not be viewed as merely an implementation convenience or middleware validation utility. The DEG acted as a formal acceptance boundary which translated stochastic proposals to enterprise-safe transactions, and which perturbed the feasible execution space of the system itself. Removing the DEG caused execution-safety invariants to be violated (i.e. decrease in Success %) and led to measurable policy failures and compatibility errors. As such, the DEG should be considered a first-class architectural control primitive, rather than a heuristic or post hoc rule checker. The Kafka-based sync

primitive was shown to fulfill a similar role; stale-state failures were introduced when reasoning occurred against stale/inaccurate inventory or credit context. [8]

TABLE II shows the causal impact of major architectural controls via ablation. TABLE III breaks down the structure of failures witnessed by DEG. Together, we see that the guardrail served as semantic and operational filter along many enterprise risk dimensions.

Table Iii: DEG Rejection Taxonomy in the Simulated Environment

Rejection Category	% of Rejected Candidate Bundles	Operational Meaning
Compatibility Violation	34%	Cross-product mismatch
Pricing Rule Violation	21%	Invalid tariff or promotional composition
Inventory Violation	18%	Out-of-stock or stale-state resource
Policy/Credit Violation	16%	Credit, fraud, or eligibility failure
Compliance Violation	11%	Encoded PII/PCI or execution-path restriction

Note: Percentages sum to 100% over the set of candidate configurations rejected by the DEG within the synthetic evaluation environment. These values characterize rejection composition, not population-level operational frequencies in a live operator network.

The DEG uncovered plurality failure modes across the rejected candidate set rather than a singleton rule failure mode. Rejections due to compatibility violations and pricing violations each constituted plurality classes themselves. Inventory rejections followed by policy and compliance violations rounded out the observed coverage. This distribution bolsters our belief that the DEG functions as a first-class architectural control that limits permissible enterprise execution along multiple vectors concurrently.

685 ms p99 latencies demonstrated guarded agentic reasoning could co-exist with high-throughput retail frontends at simulated load. While we should not read too much into this result as readiness for broad deployment, as the experiments were executed against a synthetic benchmark and specific model stack, it did suggest the latency overhead for sandboxed reasoning and validity checking was orders of magnitude smaller than that of agentic execution without constraint.

There were several caveats. First, the benchmark was synthetic and thus unable to model all real-world undocumented operator-specific catalog semantics or production failure modes. Second, the achieved results were dependent on model-version. Third, encoded compliance ~= DEG policy coverage, not necessarily complete legal /

regulatory accreditation. Fourth, our routing policy was binary, and can likely be enhanced with confidence-aware or multi-stage routing.

9. Conclusion

A Multi-Agent Reference Architecture for autonomous telecom e-commerce buyflows was presented. The architecture formalized a dual-layer orchestration model, a Deterministic Execution Guardrail, and an event-driven synchronization mechanism for near-real-time state alignment. Under controlled simulation conditions, the framework achieved higher completion, lower fallout, and full compliance with the encoded guardrail checks relative to deterministic-only and stochastic-only baselines.

The principal implication was that deterministic enterprise workflows did not need to be replaced by unconstrained AI agents. Instead, selective reasoning, synchronized state, and guarded execution were shown to provide a more credible path toward autonomous telecom buyflows. Future work should therefore focus on production-grade validation with operator catalogs and APIs, richer inferential analysis with confidence intervals and effect sizes, confidence-aware routing, and formal verification of guardrail predicates.

Reference

1. M. Rosemann, J. vom Brocke, A. Van Looy, and F. Santoro, “Business process management in the age of AI – three essential drifts,” Information Systems and e-Business Management, vol. 22, no. 3, pp. 415–429, 2024.
2. T. Kampik et al., “A Vision for Business Process Management in the Age of Generative AI,” KI – Künstliche Intelligenz, 2025.
3. L. Wang et al., “A Survey on Large Language Model based Autonomous Agents,” Frontiers of Computer Science, vol. 18, no. 6, 2024.
4. S. Fan et al., “WorkflowLLM: Enhancing Workflow Orchestration Capability of Large Language Models,” 2024.
5. H. Kourani et al., “Evaluating large language models on business process modeling: framework, benchmark, and self-improvement analysis,” Software and Systems Modeling, 2025.
6. D. Brodimas et al., “Intent-Based Infrastructure and Service Orchestration,” IEEE Open Journal of the Communications Society, 2024.
7. TM Forum, “Open Digital Architecture (ODA),” industry architecture reference, 2025.
8. G. Pisoni et al., “Responsible AI-Based Business Process Management and Improvement,” 2024.
9. S. Du et al., “A Survey on the Optimization of Large Language Model-Based Agents,” ACM Computing Surveys, 2026.
10. W. Xu et al., “LLM-Based Agents for Tool Learning: A Survey,” Data Intelligence, 2025.
11. X. Zhang et al., “A Survey of Multi-AI Agent Collaboration: Theories, Methods, and Open Challenges,” 2025.
12. A. R. Hevner, S. T. March, J. Park, and S. Ram, “Design Science in Information Systems Research,” MIS Quarterly, vol. 28, no. 1, pp. 75–105, 2004.
13. K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, “A Design Science Research Methodology for Information Systems Research,” Journal of Management Information Systems, vol. 24, no. 3, pp. 45–77, 2007.