10.48047/jocaaa.2025.34.11.05

Data Mesh and Decentralized Agentic Systems: A Unified Architecture for Autonomous Enterprise Decision-Making

Saurabh Garg
Ernst & Young US LLP, USA

Abstract

Today's organizations increasingly utilize autonomous agents to make key operational decisions within supply chains, financial transactions, healthcare networks, and risk management areas. Yet, these smart agents are constrained by essential constraints in retrieving high-quality, timely, and reliable data from centralized analytics platforms that are optimized for evaluation and not autonomous decision loops running at machine speed. This architectural mismatch manifests in latency cascade effects, governance misalignment between data and decision policies, and traceability gaps undermining regulatory compliance. The Data Mesh-Agentic System architecture addresses these challenges by unifying data autonomy with decision autonomy through principled integration of Data Mesh principles with multi-agent systems. The architecture formalizes domain-oriented data ownership combined with agent-based decision-making through intent contracts, negotiation protocols, and federated governance mechanisms. Comprehensive validation through a twelve-week supply chain simulation encompassing substantial decision volumes across multiple operational scenarios demonstrates a two-fold median latency reduction, high governance compliance rates, cost parity with centralized systems, and significant autonomous execution rates. The architecture operationalizes governance through policy-as-code, implementing declarative business rules, regulatory requirements, and environmental, social governance constraints as first-class system requirements rather than afterthoughts. Immutable audit trails capture comprehensive decision provenance, enabling regulatory compliance and organizational learning. Byzantine fault tolerance mechanisms detect and mitigate failures through credibility scoring and tunable autonomy models. Technology stack recommendations grounded in proven open-source and commercial platforms, combined with phased implementation roadmaps, provide clear adoption paths from pilot implementations through enterprise-scale deployments.

Keywords: Data Mesh Architecture, Multi-Agent Systems, Autonomous Decision-Making, Federated Governance, Policy-As-Code

1. Introduction

1.1 The Autonomous Decision-Making Challenge

In today's businesses, autonomous agents are increasingly used to handle key operational decisions along supply chains, financial processes, healthcare networks, and risk management areas. Domain-specific agents—predictive agents that forecast customer demand, inventory agents that optimize stock levels, logistics agents that decide routing, and sustainability agents that track environmental costs—run within domain expertise and make real-time decisions that together govern organizational performance.

Nonetheless, these smart agents have a basic limitation: high-quality, timely, and reliable data access continues to be a requirement for efficient decision-making. This creates an architectural paradox. Agents demand real-time data access with minimal latency, yet the systems providing this data—centralized data platforms including Master Data Management systems, data lakes, and data warehouses—were designed

10.48047/jocaaa.2025.34.11.05

primarily for analytics and reporting workflows, not for supporting autonomous decision loops operating at machine speed.

This architectural mismatch manifests in three critical challenges. First, latency cascade effects emerge when a forecasting agent requests demand data from a centralized data mesh, incurring 200-500 milliseconds of latency. An inventory agent then awaits the forecast output, adding additional delay. A procurement agent subsequently waits for inventory recommendations. By the time a decision executes across this chain, market conditions may have shifted, rendering the decision obsolete or suboptimal. Research on distributed systems demonstrates that achieving high availability in transaction processing environments introduces fundamental trade-offs between consistency guarantees and system responsiveness, with measurements showing that ensuring strong consistency across distributed nodes can increase median operation latencies from sub-millisecond ranges to several hundred milliseconds under network partitions [1]. In supply chain environments operating on one to two-hour decision cycles, such cumulative latencies are operationally unacceptable.

The second aspect, governance misalignment, occurs because data governance policies—encompassing data lineage tracking, access control mechanisms, and data quality service level agreements—operate independently from agent governance policies, which include decision constraints, approval workflows, and escalation procedures. This separation creates dangerous inconsistencies. An agent may operate on data that violates privacy redaction rules, or conversely, may use incomplete data because governance constraints require certain fields to be masked. Studies examining the role of data governance in business analytics reveal that organizations lacking integrated governance frameworks experience significant challenges in ensuring data quality, with documented cases showing that misaligned governance structures between data stewardship teams and analytical consumers result in reduced trust in analytical outputs and increased time spent on data validation activities rather than insight generation [2]. Neither outcome serves the organization's interests effectively.

The third aspect, traceability breaks, occurs in the event of auditors asking questions such as "Why did this particular sourcing choice take place on March 15, 2025?" Rebuilding the decision provenance becomes a time-consuming, human-error-prone process. Data lineage systems reveal what data existed. Agent logs reveal what decision was chosen. But there is no single trace linking the data quality, the reasoning process, the policy assessments, and the resulting decision. This separation compromises compliance with regulation and organizational learning, so that it becomes challenging to discern whether decisions were informed by correct data or whether governance policies were enforced appropriately in the course of decision-making.

1.2 The Convergence Opportunity

There are two complementary paradigms for architectural styles that address these issues in part, but existing enterprise deployments separate them as discrete concerns. Data Mesh Architecture, which was presented by Zhamik Dehghani in 2021, decentralizes data ownership to business domain teams with a federated governance standard. The model is based on four principles: domain-oriented data ownership, treating data as a product with clear contracts, self-serve data infrastructure platforms, and federated computational governance. Decentralized Agentic Systems, on the other hand, decentralize decision-making power to specialized agents that coordinate through formal protocols. Multi-agent reinforcement learning, game-theoretic negotiation mechanisms, and distributed constraint satisfaction provide the coordination machinery. The architectural friction emerges because data engineering teams build mesh platforms while artificial intelligence teams build agent systems, with minimal integration between the two efforts.

10.48047/jocaaa.2025.34.11.05

1.3 Research Contributions

This paper presents the Data Mesh-Agentic System (DMAS) architecture, which unifies data autonomy with decision autonomy through principled integration of these paradigms. The architecture addresses the convergence gap through five primary contributions: unified architectural framework formalizing integration of domain-oriented data ownership with agent-based decision-making; formal coordination specifications providing mathematical definitions with convergence guarantees; operationalized governance framework implementing policy-as-code with immutable audit trails; empirical validation results through twelve-week supply chain simulation encompassing 100,000 decisions achieving two-fold median latency reduction, 98.2% governance compliance, cost parity, and 70% autonomous execution; and implementation guidance providing concrete technology stack recommendations and scalability analysis demonstrating viability from pilot implementations through enterprise-scale deployments exceeding 128 domains.

Challenge Category	Manifestation Description
Latency Cascade Effects	Forecasting agent requests data from the centralized mesh
	Inventory agent awaits forecast output, adding delay.y
	Procurement agent waits for inventory recommendation.s
	Market conditions shift, rendering decisions obsolete
Governance Misalignment	Data governance policies operate independently from agent policies
	Agents may op. Rate on data violating privacy redaction rules.
	Agents may use incomplete data due to masking requirements
Traceability Gaps	Data lineage systems show available data
	Agent logs show decisions made.
	No unified trace connects data quality to policy evaluations

Table 1: Challenges and Manifestations in Autonomous Agent Deployments [1,2]

2. Architectural Foundations and Design Principles

2.1 Data Mesh Principles Extended for Decision-Making

Data Mesh is a shift of paradigm from monolithic data architectures to domain-focused, decentralized data platforms. DMAS applies these founding principles to enable autonomous decision-making in manners that overcome the latency and governance issues inherent presented by legacy centralized systems.

Domain ownership and responsibility form the cornerstone of this extension. In traditional Data Mesh implementations, domain ownership means business domain teams maintain responsibility for data products. The Demand Planning team owns demand forecast data; the Procurement team owns supplier performance metrics; the Logistics team owns transportation cost data. Each domain team understands the data semantically and maintains quality standards appropriate to the business context. DMAS extends this principle by making domains responsible not only for data products but also for the decisions made using

10.48047/jocaaa.2025.34.11.05

that data. The Demand Planning domain doesn't merely publish demand forecasts—it deploys a forecasting agent that proposes inventory actions based on those forecasts. This co-location of data ownership and decision authority eliminates handoff latencies and clarifies accountability. Research on real-time analytics in streaming big data environments demonstrates that decentralized processing architectures can achieve substantial performance improvements, with studies showing that distributed stream processing frameworks reduce end-to-end latency by orders of magnitude compared to centralized batch processing approaches, particularly when processing nodes are positioned close to data sources [3].

Data and decision products constitute the second extension. Standard Data Mesh treats data as products with defined contracts, including schema specifications using formats like Avro or Protocol Buffers, quality service level agreements defining freshness guarantees such as "updated hourly", completeness thresholds like "99.5% non-null values", and accuracy targets such as "root mean squared error less than 5%". Products maintain immutable lineage records tracking transformations and sources, support schema versioning for backward compatibility, and publish metadata to central registries for discovery. DMAS extends this by defining Intent Contracts as decision products. An Intent Contract specifies what action an agent proposes, the supporting evidence and data quality underlying that proposal, the constraints the action must satisfy, acceptable modifications the agent would accept during negotiation, and the priority or urgency level. Intent Contracts become discoverable, versioned artifacts just like data products, enabling other agents to understand and evaluate proposals systematically.

Self-serve decision infrastructure builds upon traditional Data Mesh infrastructure that provides shared capabilities, including metadata registries, data ingestion tools, and compute resources. DMAS extends this with decision infrastructure, including an Intent Bus for agent-to-agent communication via publish-subscribe messaging, an Orchestration Layer providing feasibility checking and conflict resolution services, a Policy Engine evaluating governance rules at decision-time, and an Audit System maintaining immutable records of decisions and outcomes. This infrastructure enables agents to propose, negotiate, and execute decisions without requiring central approval for routine operations.

2.2 Multi-Agent Coordination Mechanisms

Autonomous agents in DMAS coordinate through formal protocols combining elements from multi-agent reinforcement learning, game theory, and distributed constraint satisfaction. Each agent maintains a learned policy function that maps observations from the environment to proposed actions. The agent observes local domain data products, selects upstream data products based on dependency relationships, and receives intent messages from other agents via the Intent Bus. Using domain-specific logic that may include machine learning models, optimization algorithms, or rule engines, the agent reasons about the current state and generates action proposals encoded as Intent Contracts.

DMAS addresses multi-agent reinforcement learning challenges through explicit negotiation protocols rather than purely emergent coordination. Research examining how autonomous agents model other agents reveals fundamental challenges in multi-agent learning environments, including the credit assignment problem, where determining individual agent contributions to collective outcomes becomes computationally intractable as the number of agents grows, and the non-stationarity problem, where each agent's learning process makes the environment appear dynamic and unpredictable from other agents' perspectives [4]. These challenges necessitate coordination mechanisms that make agent intentions explicit rather than relying solely on emergent behavioral patterns.

Agents express preferences through utility functions and reach agreements through structured negotiation rounds. The system seeks solutions satisfying Nash Equilibrium properties, ensuring no agent wants to deviate unilaterally, and Pareto Optimality consideration, ensuring no alternative allocation improves one

10.48047/jocaaa.2025.34.11.05

agent's outcome without worsening another's. The principal-agent problem—where individual agent incentives potentially diverge from global organizational objectives—is addressed through mechanism design, structuring incentives so local optimization serves global goals, and constraint encoding, making global constraints hard requirements rather than soft suggestions that agents might ignore.

2.3 The Unified DMAS Model and Formal Specifications

DMAS synthesizes these mechanisms through a layered architecture where each layer addresses specific coordination challenges. The Domain Data Layer manages data products with schema definitions, quality commitments, lineage tracking, and discovery metadata. The Agentic Layer contains specialized agents leveraging local data to make autonomous decisions through observing relevant data, reasoning using domain-specific models, generating Intent Contract proposals, publishing proposals to the Intent Bus, executing approved actions with transactional semantics, and learning from outcomes. The Inter-Agent Communication Layer provides a publish-subscribe infrastructure maintaining sub-second message propagation latencies. The Orchestration and Negotiation Layer mediates agent proposals through feasibility checking and conflict resolution. The Federated Governance Layer enforces compliance through policy-as-code evaluation at decision-time. The Observability and Feedback Layer monitors agent outcomes and generates explainable decision provenance trees.

The formal agent model characterizes each domain agent by a state space representing all possible configurations, an action space representing all possible decisions, an observation function mapping the current environment to observable state information, a policy function learned through training that maps observations to proposed actions, and a reward function that evaluates action outcomes. The multi-agent negotiation protocol operates in discrete rounds, including Perception, Proposal, Aggregation, Feasibility Check, Conflict Resolution, Execution, and Feedback phases. Under assumptions of bounded rationality, consistent resolution rules, and non-empty feasible sets, the protocol provides convergence guarantees where the system terminates in finite rounds bounded by the number of constraints multiplied by the number of agents in worst-case scenarios.

10.48047/jocaaa.2025.34.11.05

Extension Category	Implementation Description
Domain Ownership	The Demand Planning team owns the demand forecast data
	The procurement team owns supplier performance metrics.
	The logistics team owns transportation cost data.
	Domains deploy forecasting agents proposing inventory actions
Data and Decision Products	Schema specifications using Avro or Protocol Buffers
	Quality service level agreements defining freshness guarantees
	Intent Contracts specify proposed actions
Self-Serve Infrastructure	Intent Bus for agent-to-agent communication
	Orchestration Layer providing feasibility checking

Table 2: Core Extensions of Data Mesh Principles Supporting Autonomous Agent Operations [3,4]

3. Governance, Compliance, and Safety mechanisms

3.1 Policy-as-Code Framework

DMAS operationalizes governance through policy-as-code, encoding business rules, regulatory requirements, and ethical constraints in declarative languages rather than implementing them as procedural code embedded in agent logic. This approach provides consistency in applying the same policies uniformly across all decisions, auditability through versioning and tracking policies as first-class artifacts, flexibility enabling policy updates without agent code changes, and explainability since policies are human-readable. Research examining policy management frameworks for managing interaction behaviors in distributed systems reveals that declarative policy specifications enable runtime adaptation and conflict resolution mechanisms that are essential for autonomous system collectives, with empirical studies demonstrating that policy-based approaches reduce system administration overhead while improving compliance verification capabilities compared to hardcoded behavioral constraints [5].

The implementation uses Open Policy Agent with its Rego declarative language. Policies are structured hierarchically with global policies applying to all domains, domain-specific policies reflecting local regulations, and decision-type policies tailored to specific action categories. At decision-time, when an agent publishes an Intent Contract, the Orchestration Layer extracts relevant policies based on action type and affected domains. The Policy Engine compares these policies with the proposed action and supporting data and returns one of four possible results: ALLOW meaning the action conforms to all policies and can proceed, DENY meaning hard constraint violations which block execution, WARN meaning soft constraint violations where the action can proceed but is logged for consideration, or ESCALATE meaning new cases where policies conflict or are too imprecise and require human consideration. All policy checks are recorded immutably with full provenance, such as what policies were checked, the result and justification of the check, the values of the data that determined the decision, and timestamps. Such a comprehensive recording allows audits for compliance and aids the improvement of the policies based on observed trends.

3.2 Environmental, Social, and Governance Integration

10.48047/jocaaa.2025.34.11.05

Pressures are increasing on organizations to operationalize environmental, social, and governance pledges. DMAS makes ESG constraints first-class considerations, not afterthoughts, by embedding them within the policy structure. Carbon budget enforcement represents one critical dimension. Environmental policies encode carbon emission budgets as hard constraints. When a production agent proposes increasing output, the policy engine calculates the carbon impact by multiplying the production increase by per-unit emissions, comparing total year-to-date emissions plus the proposed increase against the annual carbon budget. If the total would exceed the budget, the action is denied automatically. If the action brings emissions within a threshold percentage of the budget, such as 10%, the system issues a warning. This automated enforcement ensures sustainability targets are respected in operational decisions, not merely tracked retrospectively.

Research on sustainable supply chain management in emerging economies demonstrates that organizations implementing systematic environmental monitoring and constraint enforcement mechanisms achieve measurably better sustainability outcomes compared to voluntary reporting frameworks, with documented cases showing that automated monitoring systems enable more accurate carbon footprint tracking and faster identification of emissions hotspots across complex multi-tier supply networks [6]. Labor compliance verification constitutes another essential dimension. Social policies require verification of labor standards before supplier approval. When a procurement agent proposes switching to a new supplier, the policy engine checks the supplier's labor compliance status, denying the proposed action if the status is not verified, regardless of cost savings. This hard constraint prevents cost optimization from compromising ethical commitments.

Supply chain transparency policies may require minimum transparency levels, such as mandating second-tier supplier visibility for critical materials. When procurement decisions are proposed, the policy engine verifies that the required transparency level is met. Actions that would reduce supply chain visibility below the threshold are flagged for escalation and human review. Stakeholder impact assessment policies can stipulate requirements for impact analysis on decisions over specified thresholds. For example, decisions impacting more than 100 employees or affecting a material supplier relationship could necessitate obligatory stakeholder review prior to implementation. The policy engine recognizes these conditions and elevates accordingly. The ESG policy structure is hierarchical, with organizational baselines of global sustainability commitments, regional policies that mirror local legislation and stakeholder expectations, and domain policies that address particular operational environments. This hierarchical organization allows for global consistency with respect for local differences.

3.3 Immutable Audit Trail Architecture and Byzantine Fault Detection

Compliance with regulations and organizational learning call for complete, tamper-resistant records of decisions and results. DMAS implements immutable audit trails capturing comprehensive decision provenance. The audit log schema includes a unique decision identifier, timestamp, agent identifier, and domain, action type, and parameters, input data products with timestamps and lineage, policies evaluated, policy outcomes with reasoning, execution mode, negotiation trace if applicable, final action executed, execution timestamp and status, outcome metrics, and audit signature providing cryptographic verification. The schema enforces immutability through database constraints, preventing updates after creation. Compliance audits leverage this structured audit trail through SQL queries, enabling auditors to reconstruct complete decision provenance, including which agent proposed changes, supporting data with quality metrics, evaluated policies and outcomes, alternative options considered, negotiation details, and resulting outcomes.

10.48047/jocaaa.2025.34.11.05

In decentralized systems, individual agents may fail, provide false data, or behave adversarially. DMAS implements Byzantine fault tolerance mechanisms to detect and mitigate such failures. The system maintains credibility scores for each agent based on three primary factors: prediction accuracy, accounting for 30% measuring how well agent predictions match actual outcomes; consistency, accounting for 30% tracking constraint violations or actions inconsistent with historical patterns; and policy compliance, accounting for 40% measuring the percentage of proposals receiving ALLOW outcomes versus DENY or ESCALATE. Based on credibility scores, the system assigns autonomy levels with scores above 0.85 receiving high autonomy, executing decisions immediately with only policy checks, scores between 0.70 and 0.85 receiving mediated autonomy, participating in standard orchestration, scores between 0.50 and 0.70 receiving low autonomy with automatic escalation for human review, and scores below 0.50 being quarantined with all proposals blocked pending investigation. The tunable autonomy model dynamically adjusts decision execution mode based on context, calculating autonomy scores by combining uncertainty factor, cost factor, time pressure factor, SLA slack, and historical success rate with weights summing to 1.0, selecting execution modes ranging from autonomous mode, achieving 50-100 milliseconds latency, to human escalation mode requiring minutes to hours latency.

4. EMPIRICAL VALIDATION AND PERFORMANCE ANALYSIS

4.1 Experimental Design and Methodology

To validate DMAS empirically, a comprehensive supply chain simulation spanning twelve weeks with hourly decision cycles was conducted, producing approximately 100,000 decisions across multiple scenarios. The simulation modeled a realistic multi-geography supply chain with operations in North America, Europe, and Asia, incorporating four specialized domain agents: Demand Forecasting, predicting customer demand using a time series model;, Inventory Management, optimizing stock levels to balance carrying costs against stockout risk, Logistics determining optimal routing and carrier selection, and Sustainability monitoring carbon footprint and enforcing environmental constraints.

DMAS was compared against three baseline architectures representing current enterprise practices. The Centralized Orchestration baseline uses a single central optimizer that receives all inputs from agents, solves a global optimization problem considering all constraints simultaneously, and dictates decisions to agents for execution only, representing traditional centralized planning systems. The Federated Learning plus Aggregation baseline has agents train local models on domain-specific data, with a central aggregator collecting predictions and applying a consensus algorithm to reconcile differences, representing federated AI approaches. The Rule-Based baseline implements hand-coded decision trees updated quarterly with no learning capability, representing traditional operations research approaches.

The simulation tested three operational scenarios reflecting real-world supply chain challenges. The Stable scenario represents normal operations with daily demand varying within plus or minus 5% noise, providing a baseline for routine performance comparison. The Volatile scenario introduces plus or minus 15% demand shocks with three significant spikes during the simulation period, creating high uncertainty conditions, testing system adaptability. The Disruption scenario simulates a two-week supply disruption affecting one geographic region, requiring significant replanning and testing system resilience under stressed conditions. Research examining supply chain risk management processes demonstrates that organizations face increasing frequency and severity of disruptions, with comprehensive risk assessment frameworks identifying multiple vulnerability categorizations, including supply risks, demand risks,

10.48047/jocaaa.2025.34.11.05

operational risks, and environmental risks that necessitate adaptive response mechanisms capable of real-time reconfiguration [7].

Performance metrics captured multiple dimensions of system effectiveness including decision latency measured as time from data availability to decision execution with both median and 95th percentile values reported, conflict resolution metrics tracking conflicts per 100 decisions and percentage resolved without human escalation, policy compliance tracking percentage of decisions satisfying all policies and violations per 1000 decisions, financial performance including cost per unit and service level percentage, environmental metrics measuring carbon per unit and carbon budget adherence percentage, autonomous execution rate tracking percentage of decisions executed autonomously versus requiring mediation or escalation, and governance overhead measuring CPU utilization for policy evaluation, audit logging, Byzantine detection, and negotiation algorithms.

4.2 Decision Latency and Conflict Resolution Results

For normal operational scenarios, DMAS has a median latency of 2.1 hours with a 95th percentile of 3.8 hours for Stable build scenarios. The median and 95th percentile of the centralized Orchestration trends to 4.3 hours and 7.1 hours, respectively. Federated Learning has a median of 6.2 hours with a 95th percentile of 11.3 hours. Rule-based has a median latency of 1.5 hours with a 95th percentile of 2.2 hours. In the Volatile scenario with demand shocks, the 50th percentile latency of DMAS is 2.4 hours and 95th percentile is 4.2 hours, whereas for Centralized it is 5.1 hours and 95th percentile is 8.9 hours. For Federated ML, it is 7.8 hours and 95th percentile is 13.5 hours, and for Rule-Based, it is 1.7 hours and 95th percentile is 2.5 hours. In a Disruption scenario, where one needs to replan extensively, DMAS median latency is 3.1 hours with 95th percentile of 5.2 hours, Centralized drops significantly to 6.7 hours with 95th percentile of 11.3 hours, since the central optimizer needs to recalculate end-to-end, Federated ML median 9.2 hours with 95th percentile of 16.1 hours, and Rule-based 2.1 hours with 95th percentile of 3.2 hours.

DMAS achieves approximately two-fold median latency reduction compared to Centralized Orchestration across all scenarios. Research examining the bullwhip effect in supply chains reveals that information distortion and decision delays propagate upstream in supply networks, with quantitative analyses demonstrating that reducing decision cycle times and improving information sharing mechanisms can substantially dampen demand variability amplification, leading to inventory reductions and cost savings throughout the supply chain [8]. Statistical significance testing confirms these differences with a t-statistic of 18.4, p-value less than 0.001, and Cohen's d effect size of 3.2, indicating extremely strong statistical significance and a very large practical effect.

4.3 Performance Metrics and System Resilience

DMAS has 23 average conflicts per 100 decisions, resolves 87% of them without human escalation, and takes an average of 1.8 negotiation rounds per conflict, which corresponds to about 500 milliseconds negotiation time while still being responsive and allowing intricate optimization. DMAS has 98.2% policy compliance with 18 violations per 1000 decisions and 12 escalations per 1000 decisions. Financial measures indicate DMAS cost per unit of $12.34, service level of 96.2%, 142 stockouts, $18,000 expedited shipping cost, and overall supply chain cost of $1,847,000 versus Centralized at $1,832,000, a 0.8% difference not statistically significant with a p-value of 0.41. Environmental performance demonstrates DMAS produces 3.42 kg CO2 per unit with 98.2% carbon budget adherence, only 1.2% higher than Centralized at 3.38 kg CO2 per unit. Autonomous execution rates show 73% autonomous under the Stable scenario, dropping to 67% under Volatile conditions, and 58% under Disruption, demonstrating significant resilience where more than half of decisions still execute without human

10.48047/jocaaa.2025.34.11.05

intervention, even when a major supply disruption requires replanning. Governance overhead analysis reveals DMAS governance components consume 8.3% of total CPU resources, decomposed as 4.1% for policy evaluation, 2.3% for audit logging, 1.1% for Byzantine detection, and 0.8% for negotiation algorithms.

10.48047/jocaaa.2025.34.11.05

Validation Component	Description
Simulation Duration	Twelve weeks with hourly decision cycles
Geographic Coverage	Operations in North America, Europe, and Asia
Domain Agents	Demand Forecasting: predicting customer demand
	Inventory Management optimizing stock levels.
	Logistics determining optimal routing
	Sustainability monitoring carbon footprint
Centralized Baseline	Single central optimizer receiving all inputs
Federated Learning Baseline	Agents train local models with a central aggregator
Rule-Based Baseline	Hand-coded decision trees updated quarterly

Table 3: Supply Chain Simulation Framework and Baseline Architecture Comparisons [7,8]

5. Implementation Guidance and Scalability

5.1 Technology Stack and Architectural Patterns

Implementing DMAS requires careful technology selection across multiple architectural layers. For the Data Mesh Platform, providing distributed data product management, Apache Kafka with Schema Registry offers proven large-scale data distribution with native support for versioned schemas. Data Pipelines for transforming and preparing data products can use dbt with Apache Airflow, providing declarative transformation specifications with dependency management. Metadata Registry enabling data product discovery can use Apache Atlas, providing comprehensive metadata management with lineage tracking. An agent serving for deploying trained models uses KServe on Kubernetes, providing standardized model serving with auto-scaling capabilities.

Research examining optimization approaches for Kubernetes toward enhancement of cloud computing demonstrates that container orchestration platforms achieve significant performance improvements through proper resource allocation strategies, with empirical measurements showing that optimized Kubernetes configurations can reduce application deployment times by substantial margins while improving resource utilization efficiency compared to default configurations, particularly when implementing horizontal pod autoscaling and intelligent load balancing mechanisms [9]. Agent Training pipelines can use Kubeflow ML Pipelines for orchestrating training workflows on Kubernetes, MLflow for experiment tracking and model registry, or managed cloud training services. The Policy Engine, implementing governance rules, uses Open Policy Agent with its Rego declarative language as the industry standard. Audit Log Storage and maintaining immutable decision records use uusePostgreSQL with immutability constraints, providing strong transactional guarantees. Observability infrastructure uses Prometheus with Grafana for metrics collection and visualization. Container Orchestration has Kubernetes as the de facto standard. The Message Broker adopting Intent Bus employs Apache Kafka for persistent high-throughput messaging.

Several design patterns enable effective DMAS implementation. Domain-Aligned Data Mesh implements the principle of collocating data products with business domains, with each domain publishing versioned APIs with service-level agreements encoded as contracts. Technology implementation uses Apache Kafka with Schema Registry for data distribution, dbt with Apache Airflow for data pipelines, and Apache Atlas

10.48047/jocaaa.2025.34.11.05

for metadata registry. Embedded Agents in Data Products collocates agents with domain data pipelines, reducing latency by eliminating network hops between data and decision-making. Implementation uses KServe with Kubernetes for agent serving, integrated with Kubeflow ML Pipelines for model training and updates. Inter-Agent Protocol Bus supports an event-driven publish-subscribe infrastructure for negotiation between agents. Kafka is utilized for asynchronous high-throughput messaging and gRPC for synchronous request-response patterns when necessary, with sub-second latency.

Research on microservices-driven enterprise architecture models for infrastructure optimization reveals that event-driven architectures implementing asynchronous messaging patterns demonstrate measurable advantages in system responsiveness and component decoupling, with documented case studies showing that microservices architectures achieve better scalability characteristics and fault isolation properties compared to monolithic designs, particularly when combined with containerization technologies and orchestration platforms [10]. Federated Policy-as-Code evaluates governance rules at decision-time rather than embedding them in agent code, with implementation using Open Policy Agent with Rego policies versioned in version control systems, ensuring consistency and auditability. Immutable Audit Ledger maintains append-only decision logs using PostgreSQL with database constraints preventing updates or deletes. Byzantine Anomaly Detection identifies faulty agents through statistical outlier detection and cross-validation using lightweight statistical process control and isolation forests. Decision Provenance Tracing constructs immutable trees showing reasoning behind decisions using a Neo4j graph database or custom graph structures, enabling explainability queries. Outcome Feedback Loop monitors decisions and triggers retraining when performance degrades using MLflow with Seldon Core for model management orchestrated by Airflow for automated retraining pipelines.

5.2 Phased Implementation and Scalability Analysis

Deploying DMAS requires a phased approach, building organizational confidence and technical capability progressively. Phase 1 spanning weeks 1-8 as Pilot validates DMAS on a single domain with success criteria including agent making autonomous decisions with less than 100 milliseconds latency, 100% policy compliance on pilot decisions, zero manual escalations for routine decisions, and an audit trail capturing all decisions with full provenance. Phase 2, spanning weeks 9-1,6, implements Two-Domain Negotiation with success criteria including negotiation converging in less than 1 second, 80% conflict resolution without human escalation, and 30% latency improvement versus centralized baseline. Phase 3 spanning weeks 17-32 for Multi-Domain Expansion with success criteria including g 4-domain system achieving less than 500 milliseconds negotiation latency, 98% policy compliance, 70% autonomous execution rate, and governance overhead below 10% CPU. Phase 4 spanning weeks 33 and beyond for Enterprise Deployment with success criteria including enterprise-wide autonomous system operational, 5-10% cost savings versus pre-DMAS baseline, median latency below 2 hours, and 98% policy compliance.

Negotiation latency complexity depends on the number of domains n, the number of constraints c, and the conflict frequency. Negotiation algorithm complexity is O of n × c times the average negotiation rounds. With typical values of 10 agents, 20 constraints per agent, and 2 average negotiation rounds, this yields approximately 400 constraint evaluations per decision cycle. Using a constraint satisfaction problem solver with polynomial complexity, negotiation time is O of n squared worst case, but closer to O of n log n in practice with good heuristics. Domain count was varied from 4 to 128 agents, measuring negotiation time and autonomous execution rate. With 4 domains: 23 conflicts per 100, 145 milliseconds negotiation, 435 milliseconds total latency, 73% autonomous. With 128 domains, flat architecture: 61 conflicts, 3200 milliseconds negotiation, 3600 milliseconds latency, 48% autonomous. With 128 domains,s hierarchical

10.48047/jocaaa.2025.34.11.05

architecture partitioning into 8 clusters: 55 conflicts, 600 milliseconds negotiation, 1080 milliseconds latency, 62% autonomous. Negotiation time grows approximately as O of n to the 1.8 power for flat negotiation. Hierarchical clustering dramatically reduces latency at 128 agents, cutting negotiation time from 3.2 seconds to 0.6 seconds, enabling scaling beyond 100 domains while maintaining sub-second negotiation latencies.

Implementation Layer	Technology and Pattern Description
Data Mesh Platform	Apache Kafka with Schema Registry for data distribution
Data Pipelines	dbt with Apache Airflow for declarative transformations
Metadata Registry	Apache Atlas for comprehensive metadata management
Agent Serving	KServe on Kubernetes for standardized model serving
Policy Engine	Open Policy Agent with Rego declarative language
Audit Log Storage	PostgreSQL with immutability constraints
Observability	Prometheus with Grafana for metrics visualization
Container Orchestration	Kubernetes as a de facto standard
Message Broker	Apache Kafka for high-throughput persistent messaging

Table 4: Architectural Patterns and Technology Recommendations for Enterprise-Scale Implementation [9,10]

6. Discussion: Critical Analysis and Future Research Directions

6.1 Architectural Strengths and Trade-offs

DMAS demonstrates several validated strengths distinguishing it from existing enterprise decision-making approaches. The unified framework represents the first architecture formally integrating Data Mesh principles with multi-agent systems, directly addressing a recognized gap in enterprise AI implementations. Research examining challenges in deploying machine learning models beyond development phases reveals that organizations face substantial obstacles in transitioning from proof-of-concept implementations to production systems, with documented barriers including integration complexities with existing infrastructure, monitoring and maintenance requirements, data pipeline reliability issues, and organizational resistance to automated decision-making, suggesting that architectures explicitly addressing deployment challenges from initial design phases achieve higher success rates in operational environments [11]. The rigorous empirical validation through supply chain simulation provides comprehensive evidence with statistical significance testing, establishing that performance improvements are not artifacts of random variation. The governance-by-design philosophy embeds governance mechanisms directly into the architectural fabric with policy-as-code ensuring consistent enforcement, immutable audit trails providing complete decision provenance, and Byzantine detection with credibility scoring preventing faulty agents from degrading system-wide performance. The 98.2% policy compliance achieved in validation demonstrates effectiveness. The practical scalability path from single-domain pilots through enterprise deployment with technology stack recommendations grounded in proven platforms reduces implementation risk. The formal theoretical rigor, including mathematical specifications, convergence proofs, and complexity analysis, enables rigorous reasoning about system behavior.

Every architectural approach involves trade-offs requiring careful consideration during deployment decisions. DMAS achieves a two-fold median latency reduction compared to centralized orchestration at a

10.48047/jocaaa.2025.34.11.05

cost of 0.8% higher total supply chain costs. While this cost difference is not statistically significant, the trade-off reflects distributed coordination introducing overhead through negotiation, policy evaluation, and audit logging. Organizations must assess whether reduced latency justifies this overhead based on operational timescales and business criticality. DMAS achieves 96.2% service level compared to 96.8% for centralized orchestration, a 0.6 percentage point reduction reflecting the inherent challenge of distributed decision-making, where agents optimize locally, potentially missing global optima that centralized systems discover. The tunable autonomy model partially addresses this by escalating high-uncertainty decisions, but some service level reduction remains inevitable. Organizations must determine acceptable service level ranges and whether the 70% autonomous execution rate justifies s minor service level reduction. DMAS introduces architectural complexity through multiple layers, asynchronous communication, negotiation protocols, and Byzantine detection mechanisms, enabling decentralized control but increasing implementation difficulty and operational overhead compared to simpler centralized systems. Organizations must assess technical capability and whether the benefits of decentralization justify complexity. While DMAS achieves 70% autonomous execution, the remaining 30% requiring mediation or escalation reflects tension between agent autonomy and governance constraints. Tighter governance policies reduce autonomous execution rates, while looser policies risk compliance violations. Organizations need to constantly adjust thresholds in terms of risk tolerance and regulatory need, with no one-size-fits-all setting being transferable across industries and regulatory environments.

6.2 Limitations and Future Research Directions

Several fundamental limitations constrain effectiveness and applicability. Agent alignment and principal-agent problems persist where individual incentives diverge from organizational goals, despite current mitigation strategies including policy-as-code encoding global constraints as hard rules, Byzantine detection with credibility scoring identifying deviant behavior, outcome monitoring tracking decision effectiveness, and multi-agent negotiation creating mutual oversight. Future research directions include game-theoretic mechanism design approaches ensuring truthfulness and alignment, inverse reinforcement learning techniques inferring organizational objectives from historical human decisions, and multi-agent adversarial training identifying vulnerabilities before production deployment. Scalability constraints beyond 128 domains remain, where an empirical scaling study demonstrates viability up to 128 domains using hierarchical negotiation, but further scaling introduces qualitatively new challenges. Future research should investigate fully distributed Byzantine-tolerant consensus algorithms adapted for decision coordination, gossip protocols for policy distribution scaling logarithmically with domain count, hierarchical reinforcement learning techniques, and formal analysis of trade-offs between hierarchy depth, cluster size, and overall system performance. Data schema evolution and brittleness issues arise where agents train on specific data product schemas expecting particular fields, data types, and formats, with breaking schema changes requiring manual agent retraining, creating operational friction. Future research directions include automatic schema evolution strategies, transfer learning approaches enabling agents to adapt trained models without full retraining, schema mapping learning where agents automatically infer relationships between schema versions, and meta-learning techniques training agents to be robust to schema variations.

Research reflecting on theory development in sustainable supply chain management emphasizes that achieving long-term sustainability objectives requires integrating environmental and social considerations throughout operational decision-making processes rather than treating them as separate reporting requirements, with systematic literature reviews identifying gaps between theoretical frameworks and

10.48047/jocaaa.2025.34.11.05

practical implementation challenges that necessitate empirical validation of proposed mechanisms under realistic operational conditions [12]. Federated learning gaps persist where current DMAS agents train on local domain data only, potentially missing cross-domain patterns that could improve decision quality. Future research should explore federated multi-agent reinforcement learning, enabling agents to learn from peers' experiences without centralizing sensitive data, privacy-preserving gradient aggregation techniques adapted for multi-agent settings, differential privacy mechanisms enabling cross-domain learning with formal privacy guarantees, and secure multi-party computation allowing joint model training.

6.3 Research Agenda

Several fundamental research questions remain open, defining an agenda for future work. Theoretical foundations questions include fundamental limits of convergence guarantees in multi-agent negotiation under realistic assumptions, information-theoretic lower bounds on communication complexity for distributed decision-making, and characterizing conditions under which decentralized approaches provably outperform centralized ones. Learning and adaptation questions include how agents learn effectively from multi-agent interactions where outcomes depend on other agents' simultaneous actions, credit assignment mechanisms enabling individual agents to determine contribution to collective outcomes, and balancing exploration against exploitation in multi-agent settings where exploration by one agent affects others. Policy synthesis and verification questions include automated techniques for synthesizing policies from high-level organizational objectives, verifying that policy sets are consistent and complete, and detecting policy conflicts before deployment rather than discovering them through escalated decisions. Explainability and interpretability questions include generating explanations simultaneously technically accurate and accessible to non-technical stakeholders, explanation formats best supporting different stakeholder needs, including operational review and compliance audits, and interactive explanation interfaces enabling stakeholders to explore counterfactuals and understand decision sensitivity to inputs. These open problems span theoretical computer science, machine learning, optimization, formal methods, human-computer interaction, and systems engineering, requiring interdisciplinary collaboration bringing together expertise from multiple fields.

Assessment Category	Description
Unified Framework	First architecture integrating Data Mesh with multi-agent systems
	Addresses the recognized gap in enterprise AI implementations
Governance-by-Design	Policy-as-code ensures consistent enforcement.
	Immutable audit trails provide complete decision provenance.
	Byzantine detection prevents faulty agent degradation
Agent Alignment Limitation	Individual incentives diverge from organizational goals
	Policy-as-code encodes global constraints as hard rules
Scalability Limitation	Hierarchical structures create cluster-level conflicts
Federated Learning Gap	Agents train on local domain data only

Table 5: Validated Capabilities and Constraint Categories in DMAS Architecture [11,12]

10.48047/jocaaa.2025.34.11.05

Conclusion

The Data Mesh-Agentic System architecture establishes a unified framework for autonomous enterprise decision-making by eliminating the organizational and technical friction that emerges when data engineering teams build mesh platforms while artificial intelligence teams build agent systems independently. The architecture demonstrates that decentralization and governance constitute complementary capabilities rather than opposing forces through principled integration of domain-oriented data ownership with agent-based decision-making coordinated via formal negotiation protocols and federated governance mechanisms. Comprehensive empirical validation establishes that the architecture achieves substantial latency reductions while maintaining cost parity with centralized techniques and high governance compliance rates through policy-as-code enforcement, immutable audit trails, and Byzantine fault tolerance. The tunable autonomy model enables significant autonomous execution rates even under operational stress conditions by dynamically adjusting decision execution modes based on uncertainty, cost, time pressure, and historical success rates. Hierarchical negotiation strategies enable scaling beyond substantial domain counts while maintaining sub-second negotiation latencies. The architecture actively engages deployment issues by making technology stack recommendations based on evidence-based platforms, architectural patterns that facilitate successful implementation, stepwise roadmaps that progressively build organizational trust, and scalability assessment that shows feasibility from pilot implementations to enterprise-level deployments. Future directions include game-theoretic mechanism design to enforce truthfulness and alignment by the agents; Byzantine-tolerant consensus algorithms to support decentralized negotiation without a multi-tiered topology; automatic schema evolution to play down operational friction; and federated multi-agent reinforcement learning for cross-domain learning without violating data governance and privacy constraints. Organizations introducing autonomous decision systems increasingly need architectures that bring flexibility of distributed intelligence and uniformity of centralized governance together so that they can proceed at machine speed and have human values and checkpoints.

10.48047/jocaaa.2025.34.11.05

References

[1] Peter Bailis et al., "Highly Available Transactions: Virtues and Limitations (Extended Version)", arXiv, 2013. Available: https://arxiv.org/pdf/1302.0309

[2] Saige Fallen et al., "The Role of Data Governance in Business Analytics", ResearchGate, 2024. Available: https://www.researchgate.net/publication/389517404_The_Role_of_Data_Governance_in_Business_Analytics

[3] Md Ashraful Alam, "Real-Time Analytics In Streaming Big Data: Techniques And Applications", ResearchGate, 2024. Available: https://www.researchgate.net/publication/386283651_Real-Time_Analytics_In_Streaming_Big_Data_Techniques_And_Applications

[4] Stefano V. Albrecht and Peter Stone, "Autonomous agents modelling other agents: A comprehensive survey and open problems", ScienceDirect, 2018. Available: https://www.sciencedirect.com/science/article/pii/S0004370218300249

[5] Amna Batool et al., "Towards a Policy Management Framework for Managing Interaction Behaviors in IoT Collectives", MDPI, 2021. Available: https://www.mdpi.com/2624-831X/2/4/32

[6] Rebeca B. Sánchez-Flores et al., "Sustainable Supply Chain Management—A Literature Review on Emerging Economies", MDPI, 2020. Available: https://www.mdpi.com/2071-1050/12/17/6972

[7] V. M. Rao Tummala and Tobias Schoenherr, "Assessing and managing risks using the Supply Chain Risk Management Process (SCRMP)", ResearchGate, 2011. Available: https://www.researchgate.net/publication/235251515_Assessing_and_managing_risks_using_the_Supply_Chain_Risk_Management_Process_SCRMP

[8] Hakjoon Song and Daqun Zhang, "Bullwhip Effect in Supply Chains and Cost Rigidity", MDPI, 2025. Available: https://www.mdpi.com/1911-8074/18/5/284

[9] Subrota Kumar Mondal et al., "On the Optimization of Kubernetes toward the Enhancement of Cloud Computing", MDPI, 2024. Available: https://www.mdpi.com/2227-7390/12/16/2476

[10] A. M. Abd-Elwahab et al., "MicroServices-driven enterprise architecture model for infrastructure optimization", Future Business Journal - Springer Open, 2023. Available: https://fbj.springeropen.com/articles/10.1186/s43093-023-00268-3

[11] Mohsen Zaker Esteghamati et al., "Beyond development: Challenges in deploying machine learning models for structural engineering applications", arXiv, 2024. Available: https://arxiv.org/html/2404.12544v1

[12] Stefan Seuring et al., "Reflecting on theory development in sustainable supply chain management", ScienceDirect, 2022. Available: https://www.sciencedirect.com/science/article/pii/S2772390921000160