Converging Agentic AI, Data Mesh, and Lightweight Language Models for Resilient System-of-Systems Automation

Ahmed Raza[1], Hira Mahmood[2], Bilal Tariq[3, *]

[1] Department of Computer Science, University of Gujrat, Gujrat, Pakistan, 50700
[2] Department of Software Engineering, University of Engineering and Technology, Taxila, Pakistan, 47050
[3] Department of Information Systems, Institute of Business Administration, Karachi, Pakistan, 75270
*Email: bilal.tariq@iba.edu.pk (Corresponding Author)

Abstract

This article develops a system-of-systems framework for resilient enterprise automation by integrating three fast-developing technological streams: agentic artificial intelligence, domain-oriented data mesh governance, and lightweight language models. Contemporary language-model agents are increasingly able to plan tasks, select tools, call functions, and synthesize multi-source information, but their deployment in complex enterprises remains constrained by data silos, legacy systems, cross-domain compliance requirements, ambiguous instructions, privacy risks, and cascading failure modes. Drawing on the architectural ideas of multi-context agent coordination and mesh-oriented governance, the article proposes a lifecycle framework in which each enterprise domain exposes approved capabilities as governed functions while retaining authority over local data, policies, and risk controls. Lightweight domain-specific models are positioned as the operational substrate for low-latency, privacy-preserving, and cost-sensitive automation, while larger models remain useful for controlled synthesis and exception handling. The paper contributes a six-stage lifecycle model, a domain-capability governance map, a comparative scenario analysis of four automation architectures, and a risk matrix for secure multi-domain agent deployment. The analysis suggests that the strongest architecture is not a single general-purpose agent connected to all enterprise systems, but a federated mesh of domain-specialized agents supported by local data ownership, policy-aware routing, audit trails, fallback controls, and model portfolios. The article concludes with an implementation roadmap and research agenda for benchmarking, security evaluation, human oversight, and responsible governance in future system-of-systems automation.

Keywords: Agentic AI; Data mesh; Lightweight language models; System-of-systems automation; Enterprise architecture; Federated governance; AI safety; Multi-agent systems

Article History:
Received: October 08, 2024
Revised: December 15, 2024
Accepted: February 12, 2025
Available Online: March 30, 2025

I. INTRODUCTION

Enterprise automation is moving from task-specific software toward networks of autonomous or semi-autonomous agents that can interpret requests, retrieve operational context, call tools, and coordinate action across organizational boundaries. The core opportunity is not simply that a language model can generate a response. The deeper change is that language models can become orchestration interfaces for business processes, software services, data products, and physical systems. This shift is visible in tool-using language models, retrieval-augmented systems, program-aided reasoning, and agent simulations that combine planning, memory, and external action (Vaswani et al., 2017; Brown et al., 2020; Devlin et al., 2019; Lewis et al., 2020; Yao et al., 2023; Schick et al., 2023; Park et al., 2023; Shinn et al., 2023; Karpas et al., 2022; Madaan et al., 2023; Gao et al., 2023; Liu et al., 2023). Yet the same flexibility that makes agentic AI attractive also makes it risky: a single general-purpose agent connected to multiple contexts can misread local business rules, trigger wrong functions, expose sensitive data, or propagate an error across dependent systems.

The uploaded source article develops a mesh-oriented multi-context architecture for agentic AI. Its central insight is that complex enterprise ecosystems should not be treated as a single centralized context, because each domain has its own data assets, rules, compliance boundaries, technical legacy, and failure modes. Instead, the source article proposes domain nodes, federated governance, local function calls, privacy-aware sharing, dynamic node management, and small domain-specific models as core design ideas. This paper rewrites that core contribution into a broader future-technologies article: it asks how agentic AI, data mesh thinking, and lightweight language models can converge into resilient system-of-systems

automation. The argument is that safe automation requires three simultaneous shifts: agents must be domain-specialized rather than universally permissive, data access must be organized as governed products rather than centralized extraction, and language models must be deployable at the domain edge rather than only through large centralized foundation-model services.

The system-of-systems perspective is essential because enterprise automation is rarely a single software workflow. A hospital, city authority, bank, manufacturer, logistics firm, or public agency consists of multiple semi-independent systems that must coordinate without surrendering full control. Classical multi-agent and autonomic-computing research already recognized that complex software environments require local autonomy, monitoring, repair, and negotiated coordination rather than central command alone (Wooldridge and Jennings, 1995; Jennings, 2001; Kephart and Chess, 2003; Huebscher and McCann, 2008; Maier, 1998; Boardman and Sauser, 2006; Gorod et al., 2008; Sage and Cuppan, 2001). What is new is the presence of language-model agents that can interpret natural language, synthesize instructions, and call external functions. This creates a new design problem: how can such agents be made powerful enough to coordinate across domains while being constrained enough to avoid systemic harm?

We answer this question by proposing a lifecycle framework for resilient system-of-systems automation. The framework organizes domain-specialized agents into a mesh of governed capabilities, where each node exposes only approved functions, summarizes or transforms sensitive information before sharing, and maintains local accountability for data quality and policy enforcement. Lightweight language models support this approach by making domain-local deployment economically and technically plausible, while compression, distillation, quantization, and parameter-efficient adaptation reduce the dependence on large centralized models (Hu et al., 2022; Dettmers et al., 2023; Hinton et al., 2015; Han et al., 2016; Frantar et al., 2023; Lin et al., 2024; Sanh et al., 2019; Jiao et al., 2020; Lan et al., 2020). The paper contributes an architectural framework, a scenario-based comparative analysis, a governance model, and an implementation roadmap for organizations seeking to combine agentic AI with decentralized enterprise data architectures.

The uploaded article is especially useful because it highlights a problem that conventional agent frameworks often understate: enterprise context is not merely additional information, but a boundary condition for legitimate action. A model may know how to summarize a database, but it may not know whether the database is an audit record, a production asset, a medical record, or a customer-facing service. The same natural-language instruction can therefore imply different acceptable actions across domains. This paper treats domain context as a first-class architectural variable rather than as a prompt-engineering detail.

A second motivation is the growing gap between enterprise ambition and enterprise readiness. Many organizations want autonomous decision support, yet their data remains fragmented across customer relationship management systems, enterprise resource planning platforms, industrial control systems, ticketing systems, spreadsheets, and local databases. Even when a general-purpose language model can be connected to these sources, the resulting automation may be brittle if no domain owner has encoded what the model may do. Data mesh logic helps resolve this gap by making ownership, quality, and governance explicit before automation is scaled.

II. CONCEPTUAL BACKGROUND

The first pillar of the argument is agentic AI. In this paper, agentic AI refers to a system that can interpret a goal, decompose it into tasks, select tools or other agents, call functions, examine intermediate results, and produce a final action or recommendation. The design is related to long-standing work on intelligent agents and autonomous software systems, but contemporary language models expand the interface through which agents can reason about unstructured requests and dynamic contexts (Wooldridge and Jennings, 1995; Jennings, 2001; Kephart and Chess, 2003; Huebscher and McCann, 2008; Vaswani et al., 2017; Brown et al., 2020; Yao et al., 2023; Schick et al., 2023). Enterprise agents are different from consumer chatbots because the cost of wrong action is higher: an ambiguous inventory request may affect production, a mistaken compliance summary may trigger legal exposure, and a badly constrained data operation may damage operational records.

The second pillar is data mesh and domain-oriented data governance. Data governance literature stresses that high-value data cannot be separated from ownership, quality standards, access policies, and context of use (Abraham et al., 2019; Alhassan et al., 2016; Khatri and Brown, 2010; Otto, 2011; Davenport et al., 2012; Chen et al., 2012; Gregor and Hevner, 2013; Benlian et al., 2018; Tiwana, 2015). In a mesh-oriented enterprise, each domain treats its data and capabilities as products. A finance domain, for example, may expose invoice validation, risk scoring, or payment status functions, but it need not export raw ledgers to a universal AI hub. A manufacturing domain may expose machine-health summaries, defect classifications, and capacity estimates, while retaining local authority over process data. This structure reduces the need for

raw-data pooling, improves accountability, and allows local experts to encode domain-specific constraints.

The third pillar is the lightweight language model. Large foundation models are powerful general-purpose engines, but they are not always the safest or most economical way to automate domain work. They require substantial computational infrastructure, raise data residency concerns, and may lack specialized knowledge of local processes. Model compression and adaptation methods make it possible to create smaller agents that are cheaper, faster, and more controllable (Hu et al., 2022; Dettmers et al., 2023; Hinton et al., 2015; Han et al., 2016; Frantar et al., 2023; Lin et al., 2024; Sanh et al., 2019; Jiao et al., 2020; Lan et al., 2020). A domain-specific language model does not need encyclopedic knowledge; it needs reliable knowledge of its local schema, allowed function calls, exception rules, and escalation procedures. The advantage is not only efficiency but governability.

The fourth pillar is secure collaboration. In regulated and competitive environments, collaboration is often blocked not by lack of value but by fear of data leakage, compliance violation, and loss of control. Federated learning, differential privacy, secure aggregation, secure multiparty computation, homomorphic encryption, and secret-sharing methods provide a toolkit for collaborative intelligence without full raw-data exchange (Kairouz et al., 2021; Li et al., 2020; Yang et al., 2019; Lim et al., 2020; Nguyen et al., 2021; Mothukuri et al., 2021; Dwork, 2006; Abadi et al., 2016; Shokri and Shmatikov, 2015; Gentry, 2009; Shamir, 1979; Yao, 1982; Lindell and Pinkas, 2009; Nikolaenko et al., 2013; Bonawitz et al., 2017). These mechanisms are not substitutes for organizational governance, but they make governance technically enforceable. They allow an agent network to answer cross-domain questions through aggregated, masked, or policy-approved outputs rather than unrestricted database access.

The fifth pillar is enterprise technology integration. Microservice architecture, cloud-native migration, and service-oriented patterns show how complex organizations can decompose monolithic systems into interoperable modules (Dragoni et al., 2017; Di Francesco et al., 2019; Soldani et al., 2018; Jamshidi et al., 2018; Taibi et al., 2017; Balalaie et al., 2016). System-of-systems automation extends this principle from software services to agentic decision spaces. The issue is not simply exposing APIs; it is ensuring that APIs are called under the correct context, with the correct permissions, and with appropriate audit trails. In this sense, agentic automation is not a replacement for enterprise architecture. It is a new orchestration layer that must respect the architecture below it.

The literature also warns that automation must remain accountable and interpretable. Human-AI interaction guidelines, algorithmic auditing, fairness research, and explainable AI all show that technical performance alone does not produce trustworthy deployment (Amershi et al., 2019; Raji et al., 2020; Jobin et al., 2019; Floridi and Cowls, 2019; Mittelstadt, 2019; Ribeiro et al., 2016; Lundberg and Lee, 2017; Doshi-Velez and Kim, 2017; Arrieta et al., 2020; Binns, 2018; Selbst et al., 2019). A domain-specialized agent may be small and local, but it can still generate harmful outcomes if its reasoning is opaque or its outputs cannot be challenged. For that reason, the proposed framework treats audit, explanation, fallback, and human override as design features rather than optional afterthoughts.

A useful distinction can be drawn between context, capability, and control. Context refers to the data, documents, events, and histories that allow an agent to understand a request. Capability refers to the functions, tools, or models that allow the agent to act. Control refers to the policies, approvals, and monitoring mechanisms that determine whether the action is legitimate. Many current systems emphasize context and capability while treating control as an administrative afterthought. This paper argues that control must be embedded directly into the agentic architecture.

The boundary between local and shared intelligence is also important. A central model can be helpful when a decision requires broad synthesis across many areas, but most enterprise requests are narrower. They ask for local interpretation: whether a purchase order is unusual, whether a service ticket should be escalated, whether a production anomaly requires maintenance, or whether a compliance exception is permitted. These tasks often require deep domain familiarity rather than broad world knowledge. The more domain-bounded the task, the stronger the case for local models, local retrieval, and local governance.

The proposed framework therefore treats the enterprise not as one knowledge base, but as a federation of accountable knowledge environments. This framing is consistent with data governance, platform ecosystems, microservices, and system-of-systems research, all of which show that scale is achieved not by ignoring local differences but by standardizing interfaces among them. In agentic AI, the interface is not only an API signature. It is also a policy contract: a statement of what the agent is allowed to infer, disclose, recommend, and execute.

III. LIFECYCLE FRAMEWORK FOR CONVERGENT AUTOMATION

The proposed lifecycle framework is organized around six stages: sensing, contextualization, planning, execution, audit,

and learning. Sensing gathers local state from enterprise systems, logs, files, devices, or user requests. Contextualization maps that state to domain rules, data classifications, and capability descriptions. Planning determines whether the task should be executed locally, delegated to another domain, or escalated to a human reviewer. Execution invokes approved functions. Audit records the decision path and compares the action against policy. Learning improves local prompts, policies, small models, and function descriptions over time. This lifecycle transforms agentic AI from an open-ended reasoning engine into a governed operational workflow.

A lifecycle perspective is necessary because most enterprise automation failures do not happen at a single point. A request can be badly specified, a model can misinterpret the request, a policy can be missing, a function can be too broad, a result can be aggregated incorrectly, or a human reviewer can receive an explanation too vague to evaluate. Resilience therefore requires controls at every stage. Industry 4.0 and industrial information integration research shows that intelligent automation depends on connecting cyber-physical systems, AI models, IoT infrastructure, analytics, and business processes into coherent operating models (Lee et al., 2015; Xu et al., 2018; Xu et al., 2014; Ben-Daya et al., 2019; Wamba et al., 2015; Lu, 2017; Lu, 2017; Lu, 2019; Zhang and Lu, 2021; Lu and Xu, 2019). The proposed framework adapts this insight to language-model-based system-of-systems automation.

The lifecycle is designed to be recursive. A cross-domain workflow may pass through the lifecycle in several nodes before a final response is produced. For example, a supply-chain agent may receive a request about delayed shipments, sense local logistics status, contextualize the delay against supplier contracts, delegate a finance question to the finance node, and request a production-capacity estimate from the operations node. Each node performs its own policy check before returning a result. The final response is therefore assembled from governed local contributions rather than from unrestricted data pooling.

This recursive design creates a different failure profile. In a centralized architecture, errors tend to be hidden until the final decision is made. In a mesh architecture, each node can fail visibly at its boundary: it can refuse a request, return a summary instead of raw data, demand human approval, or degrade to a limited response. These controlled failures are not inefficiencies. They are resilience mechanisms. A safe automation system should sometimes say that it cannot act, cannot share, or requires review.

A further lifecycle requirement is semantic versioning of capabilities. Functions change, policies change, and domain terminology evolves. If a purchasing node changes the meaning of a supplier-risk category or a security node changes its incident-severity threshold, other agents should not continue to treat old descriptions as current. Each exposed capability should therefore include version information, policy metadata, owner identity, and expiration or review dates. This turns the agent network into a living system rather than a static collection of tools.

TABLE I. LIFECYCLE LAYERS FOR DOMAIN-SPECIALIZED SYSTEM-OF-SYSTEMS AUTOMATION

Lifecycle Layer	Primary Question	Main Design Element	Failure Mode Controlled
Sensing	What local state is relevant?	Domain data product; event stream	Missing context and stale data
Contextualization	Which local rules apply?	Policy map; schema; boundary	Wrong business meaning
Planning	Should the task be local, delegated, or escalated?	Planner; capability discovery	Wrong routing and overreach
Execution	Which approved function should be called?	Typed function; threshold	Unsafe action
Audit and learning	Can the decision be reviewed and improved?	Logs; explanation; fallback	Opacity and drift

Table I summarizes the main lifecycle layers, the enterprise question each layer answers, and the failure mode that must be controlled. The table deliberately separates data ownership from action execution. A domain may own the data needed for a decision but still delegate part of the execution to another domain, provided that the shared information is policy-approved and the receiving node has sufficient capability. This separation is the central advantage of combining data mesh principles with agentic AI.

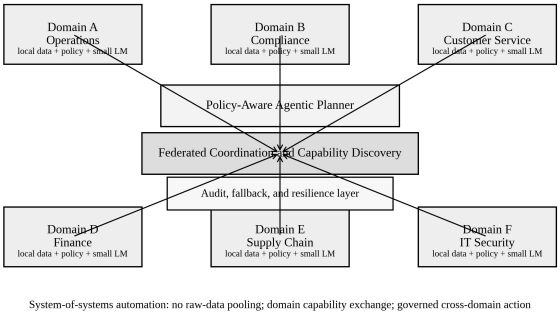

Figure 1. Mesh-oriented convergence architecture for resilient system-of-systems automation.

Figure 1 illustrates the architecture. Each domain node contains local data, rules, functions, and a compact model. A federated coordination layer discovers capabilities and routes tasks without requiring all data to be transferred to a central repository. A policy-aware planner determines whether a request is handled locally or sent to another node, while audit and fallback mechanisms prevent uncontrolled propagation. This design is particularly suitable for organizations that have legacy systems, air-gapped or partially connected environments, and compliance boundaries that prevent unrestricted data sharing.

The framework differs from a single-agent hub in three ways. First, it assumes that local domains know their data better than a central agent does. Second, it treats cross-domain automation as negotiated cooperation rather than as universal access. Third, it treats a model's size as a design variable. The goal is not always to use the largest possible model; it is to place the smallest sufficiently capable model at the right point in the architecture. This allows enterprises to distribute intelligence without distributing all sensitive data.

IV. DOMAIN SPECIALIZATION AND MULTI-CONTEXT DECISION LOGIC

Domain specialization is the mechanism by which agentic AI becomes safe enough for enterprise use. A general agent may understand a user request linguistically but still fail to understand which domain rules apply. A domain-specialized node, by contrast, packages the data schema, process vocabulary, compliance constraints, exception policies, and allowed actions for a specific area. This packaging is not merely descriptive. It defines what the agent is authorized to do. For example, a finance agent may answer questions about invoice status, detect unusual payment patterns, and summarize budget variance, but it may not modify vendor bank details without multi-step approval. A healthcare agent may summarize appointment availability or aggregate resource utilization but may not disclose patient-identifiable data across organizational boundaries.

Multi-context decision logic follows a simple principle: agents should execute locally when the capability and authority exist, delegate when another domain is better suited, and escalate when the request exceeds policy. This principle is compatible with multi-agent system theory and with the practical lessons of microservice architecture, where bounded contexts and explicit interfaces help reduce unintended coupling (Wooldridge and Jennings, 1995; Jennings, 2001; Dragoni et al., 2017; Di Francesco et al., 2019; Soldani et al., 2018; Jamshidi et al., 2018; Taibi et al., 2017; Balalaie et al., 2016; Kairouz et al., 2021). However, language-model agents require an additional layer: prompt and tool constraints must be aligned with data governance policies. An API endpoint is not safe merely because it is available; it is safe only when its calling conditions are known and enforced.

Domain specialization does not mean domain isolation. A mesh system is valuable precisely because it enables cooperation across boundaries. The difference is that cooperation is mediated through declared capabilities rather than through undisciplined access. A customer-service agent may ask the logistics node for delivery-status evidence, but it receives only the level of detail permitted for customer communication. A finance agent may ask the compliance node whether a transaction requires review, but it does not gain access to the entire compliance case repository unless policy permits it.

The decision logic can be formalized through three questions. First, is the required capability available locally? Second, is the request permitted under local policy? Third, if the task is delegated, what is the minimum information that must be shared? This minimum-sharing principle is essential for privacy and security. It shifts agent design from a maximum-context assumption to a sufficient-context assumption. The best agent is not the one that sees everything, but the one that receives enough governed context to complete a task safely.

Organizations may also use domain specialization to manage model heterogeneity. Some nodes may use deterministic rule engines, others may use compact language models, and others may call a larger model through a controlled gateway. The coordination layer should not assume that every node has the same model. It should require only that each node can describe its capability, confidence, data-sharing policy, and output format. This model-agnostic coordination is important because AI technology will continue to evolve faster than enterprise architecture can be replaced.

TABLE II. DOMAIN-CAPABILITY MAP FOR GOVERNED AGENTIC AUTOMATION

Domain	Approved capabilities	Boundary and model strategy
Finance	Invoice check; budget variance	Ledger protected; small LM; approve payment changes
Supply chain	Delay prediction; supplier summary	Contract protected; adapter LM; supplier penalty escalated
Customer service	Complaint triage; response draft	Personal data protected; compact LM; legal claims escalated
IT security	Incident triage; log summary	Credentials protected; anomaly model; privileged action escalated
Operations	Capacity estimate; work-order summary	Telemetry protected; edge model; safety control escalated

Table II provides a domain-capability map for five common enterprise domains. The examples are illustrative rather than exhaustive. Their purpose is to show that the same agentic architecture can support different domains without forcing them into a uniform data model. The finance domain can operate with audit-heavy rules; the supply-chain domain can emphasize exception routing; the customer-service domain can emphasize summarization and escalation; the security domain can emphasize anomaly triage; and the operations domain can emphasize real-time constraints. A resilient architecture must preserve these differences rather than hide them.

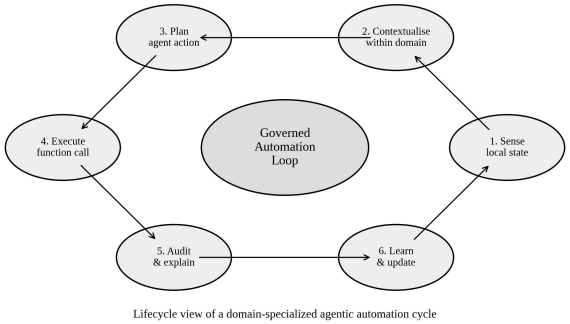

Figure 2. Lifecycle loop for governed domain-specialized agentic automation.

Figure 2 reframes the decision logic as a lifecycle loop. The loop begins with local sensing because the most reliable interpretation of a request depends on domain state. It then moves to contextualization and planning, where the node decides whether the request is local, delegated, or escalated. Execution occurs only after policy checks, and audit captures not only the final output but also the reasoning path, the tool call, and the data-sharing decision. Learning is bounded: the node can improve prompts, rules, and model adapters, but governance changes require explicit approval.

The loop also clarifies why the proposed convergence is more than a technical stack. Agentic AI supplies planning and tool use; data mesh supplies ownership and governance; lightweight language models supply local intelligence and cost control. If any of these elements is missing, the automation pattern becomes fragile. Agentic AI without data governance may overreach. Data mesh without agentic orchestration may remain a passive information architecture. Lightweight models without domain rules may be efficient but unreliable. The convergence creates a more balanced automation paradigm.

V. SCENARIO-BASED DATA ANALYSIS

Because public benchmarks for cross-domain agentic automation are still immature, this paper uses a scenario-based analytical evaluation rather than claiming production-scale empirical validation. Four architectures are compared: a central hub, a tool-only language-model system, a mesh of governed agents, and a mesh of governed agents supported by

lightweight domain-specific models. Six dimensions are scored on a 0-1 scale: reliability, fault isolation, privacy preservation, latency fitness, integration cost (inverted so that higher is better), and governance clarity. Scores are derived from architectural assumptions grounded in the literature on federated learning, microservices, data governance, system-of-systems design, and AI accountability (Maier, 1998; Gorod et al., 2008; Abraham et al., 2019; Alhassan et al., 2016; Dragoni et al., 2017; Di Francesco et al., 2019; Kairouz et al., 2021; Li et al., 2020; Amershi et al., 2019; Raji et al., 2020; Mittelstadt, 2019; Ribeiro et al., 2016).

The comparison is intentionally conservative. The central hub scores reasonably on initial coordination because it is simple to control, but it scores poorly on fault isolation and privacy because shared access creates broad exposure. The tool-only language-model system improves flexibility but remains weak on governance clarity if local policies are not encoded. The mesh-agent architecture improves fault isolation and privacy by distributing authority. The mesh plus lightweight-model architecture gains additional latency and cost advantages because many tasks can be executed near the domain data without calling an expensive or remote model for every request.

The composite index reported in Table III is computed as the unweighted mean of the six dimensions. An unweighted score is used because the appropriate priority weights differ by sector. In finance and healthcare, privacy and auditability may outweigh latency. In manufacturing, latency and fault containment may dominate. In public administration, governance clarity and accountability may be decisive. A practical deployment should therefore adapt the scorecard to its local risk profile. The unweighted result is a neutral baseline rather than a universal ranking.

A sensitivity interpretation can nevertheless be made. If privacy is weighted twice as heavily as the other dimensions, the central hub becomes even less attractive because data exposure is structurally high. If latency is weighted twice as heavily, the mesh plus small-model architecture improves further because local execution reduces remote calls. If integration simplicity is weighted heavily, the tool-only architecture may appear competitive for pilots, but its low governance clarity becomes a liability when workflows expand. This illustrates why early pilot success does not always predict enterprise-scale success.

The analysis also highlights a useful paradox. The architecture with the most distributed authority may be easier to govern at scale than the architecture with the most centralized control. Central control appears simpler because all routing occurs in one place, but it forces one unit to understand every domain's rules. Distributed governance requires more coordination, but each domain governs what it understands. This is the central logic of data mesh applied to agentic AI: local ownership improves global reliability when interfaces are standardized.

Another result concerns model cost. A large model may produce high-quality reasoning across broad topics, but a domain node often needs predictable and repetitive operations. For such tasks, the cost of a large model can be unnecessary, and its generality may even increase risk. A compact model trained or adapted to local documentation can be easier to test, easier to monitor, and easier to update. In high-volume operations, the cost savings from compact local models may be substantial, especially when requests are routine and well structured.

TABLE III. SCENARIO-BASED ARCHITECTURE SCORECARD

Architecture	Main strength	Main weakness	Composite score
Central hub	Simple initial coordination	Low privacy and weak fault isolation	0.49
Tool-only LLM	Flexible tool access	Unclear governance and domain overreach	0.55
Mesh agents	Strong privacy and fault containment	Higher initial design effort	0.78
Mesh + small LMs	Best latency, privacy, and governance balance	Requires model portfolio management	0.84

Table III reports the scenario scores and an average composite index. The highest average score belongs to the mesh plus lightweight-language-model architecture. This does not mean that large centralized models are obsolete. They remain useful for complex reasoning, broad synthesis, and tasks that require general knowledge. The analysis instead suggests that resilient enterprise automation should use a tiered model portfolio: small local models for routine domain tasks, larger models for cross-domain synthesis under control, and human review for irreversible or high-risk actions.

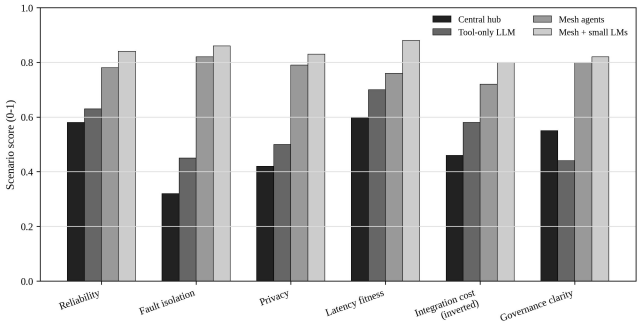

Figure 3. Comparative scenario scores for four enterprise automation architectures.

Figure 3 visualizes the score differences. The greatest gains from the mesh architectures appear in fault isolation, privacy, and governance clarity. These dimensions are often underweighted in early AI adoption because they are less visible than speed or output quality. Yet they are decisive in system-of-systems settings. A minor failure in a local agent can be tolerated if it remains local; the same failure can become catastrophic if the agent has broad access to multiple domains. This is why fault containment should be treated as a core design metric rather than as a post-deployment security concern.

The scenario analysis also shows that integration cost should be interpreted carefully. A central hub may appear cheaper at first because it requires fewer organizational interfaces. Over time, however, each exception, compliance requirement, and legacy-system mismatch must be resolved inside the hub. A mesh architecture has higher initial design complexity, but it distributes responsibility to domains that already understand their systems. This may reduce long-term maintenance cost and improve adaptability, especially in organizations whose structures change frequently.

Latency is similarly nuanced. Large centralized models can handle complex prompts but may introduce delays, network dependencies, and data-transfer concerns. Lightweight domain models can respond quickly when tasks are well-defined. Techniques such as LoRA, QLoRA, distillation, quantization, TinyBERT-style compression, and ALBERT-style parameter sharing make this deployment pattern increasingly practical (Hu et al., 2022; Dettmers et al., 2023; Hinton et al., 2015; Han et al., 2016; Frantar et al., 2023; Lin et al., 2024; Sanh et al., 2019; Jiao et al., 2020; Lan et al., 2020). The more an organization can convert routine domain operations into well-specified functions, the more value it can obtain from small models and local execution.

VI. SECURITY, PRIVACY, AND GOVERNANCE CONTROLS

Secure multi-context automation requires a governance stack with at least five layers. The first is identity and role management: each user, model, agent, and function must have an identity. The second is capability governance: each domain must describe what functions it exposes, who can call them, and what data they may return. The third is privacy control: raw data sharing should be replaced where possible by summaries, secure aggregation, differential privacy, or encrypted computation (Dwork, 2006; Abadi et al., 2016; Shokri and Shmatikov, 2015; Gentry, 2009; Shamir, 1979; Yao, 1982; Lindell and Pinkas, 2009; Nikolaenko et al., 2013; Bonawitz et al., 2017). The fourth is operational resilience: fallback policies, circuit breakers, and alternative routes should prevent a local failure from cascading. The fifth is accountability: every agent decision should be logged in a way that supports review and learning.

Policy design should also distinguish between reversible and irreversible actions. A domain agent that retrieves a report may require standard logging. A domain agent that deletes a file, changes a price, modifies a patient record, approves a transfer, or sends a legal message should require stronger safeguards. Safeguards may include human confirmation, two-agent verification, predefined rollback, simulation before execution, or time-delay mechanisms. The purpose is not to block automation but to match the autonomy level to the action's risk.

One of the most difficult governance problems is ambiguous intent. Natural-language requests often contain verbs such as clean, fix, update, reconcile, adjust, or optimize. These verbs are harmless in conversation but dangerous when attached to tools that can modify enterprise records. The proposed governance stack therefore requires action classification before execution. Requests should be classified as read-only, advisory, reversible, conditionally executable, or irreversible. Each class should trigger a different approval pathway and audit depth.

A second security issue is the possibility of indirect prompt injection. A domain agent may retrieve documents, emails, logs, or web pages that contain malicious instructions designed to manipulate the agent. Because the agent is connected to tools, such injections may produce operational harm. The architecture should therefore separate retrieved content from

system instructions, sanitize external inputs, and require tool calls to be validated by policy rather than by model confidence alone. Security should be enforced outside the model, not only requested inside the prompt.

The privacy layer should also recognize different forms of disclosure. Raw records are obvious risks, but aggregated outputs can also reveal sensitive information if the group is small or if multiple queries are combined. A mesh agent should therefore maintain a disclosure budget or risk score. Differential privacy, secure aggregation, and summary-only responses are possible controls, but organizational rules are still needed to decide when a privacy-preserving computation is acceptable and when no disclosure should be made.

TABLE IV. GOVERNANCE CONTROLS FOR SECURE MULTI-CONTEXT AGENTIC SYSTEMS

Control layer	Mechanism	Metric
Identity and role	Model identity, user identity, signed function call	Unauthorized-call rate
Capability boundary	Domain-owned function registry and typed schema	Out-of-scope action rate
Privacy filter	Summary sharing, differential privacy, secure aggregation	Data-exposure score
Fault containment	Circuit breaker, fallback path, graceful degradation	Propagation depth
Audit and explanation	Action log, rationale, human review queue	Audit completeness
Model maintenance	Adapter versioning, drift monitoring, retraining schedule	Drift response time

Table IV translates these principles into practical controls. The table emphasizes that governance is not a single policy document. It is a technical and organizational system composed of rules, interfaces, logs, thresholds, and escalation paths. A policy that cannot be enforced at the function-call layer is weak. Conversely, technical restrictions that do not map to real organizational responsibility may be too rigid or misaligned with practice.

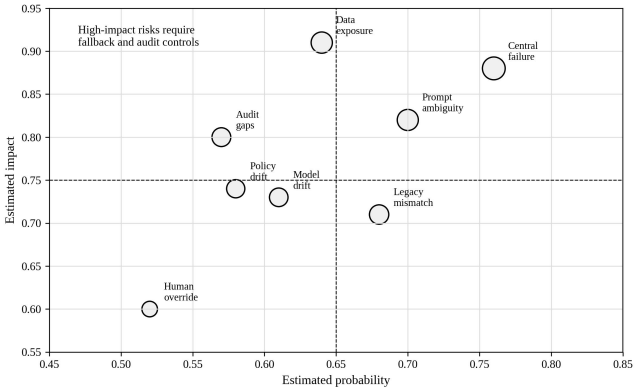

Figure 4. Probability-impact matrix for multi-context agentic automation risks.

Figure 4 maps eight common risk factors by estimated probability and impact. Central failure, data exposure, and prompt ambiguity occupy the high-risk area, while legacy mismatch and model drift sit close behind. The matrix supports the central argument of this paper: a resilient architecture must combine fault isolation, privacy-by-design, and domain-context control. It is not sufficient to make agents more capable; they must also be embedded in architectures that make unsafe actions less likely and less damaging.

Another implication is that governance has to be dynamic. Enterprise environments change: new regulations appear, suppliers change, products evolve, customer needs shift, and data systems are updated. A static policy layer will become obsolete. The mesh structure allows domains to update their own policies and capabilities while still participating in a shared coordination protocol. This feature is especially important for public service, healthcare, finance, and industrial operations, where local regulations and operational constraints vary across sites.

VII. IMPLEMENTATION ROADMAP

The proposed architecture should be implemented gradually. The first phase is domain inventory. Organizations identify domains, data products, critical functions, sensitive datasets, legacy interfaces, and responsible owners. This phase is often more difficult than expected because many enterprises lack a current map of who owns which data and which processes depend on it. Data governance studies show that ownership, stewardship, and decision rights are prerequisites for effective information systems integration (Abraham et al., 2019; Alhassan et al., 2016; Khatri and Brown, 2010; Otto, 2011;

Davenport et al., 2012; Chen et al., 2012; Gregor and Hevner, 2013; Benlian et al., 2018). Without this inventory, an agentic system may automate confusion rather than create intelligence.

The second phase is capability wrapping. Legacy systems are not replaced; their approved operations are exposed as callable functions with strict input and output schemas. Microservice migration research suggests that incremental decomposition is more practical than big-bang modernization (Dragoni et al., 2017; Di Francesco et al., 2019; Soldani et al., 2018; Jamshidi et al., 2018; Taibi et al., 2017; Balalaie et al., 2016). In the agentic setting, wrapping must include semantic descriptions: what the function does, when it should be used, what data it needs, what it returns, what risks it creates, and who owns it.

The third phase is local model deployment. Each domain selects a suitable model strategy: a rule-based tool for deterministic tasks, a compact language model for domain text reasoning, a fine-tuned adapter for local vocabulary, or a larger model for complex synthesis. Parameter-efficient adaptation and quantization allow organizations to tune models without training from scratch (Hu et al., 2022; Dettmers et al., 2023; Frantar et al., 2023; Lin et al., 2024). The domain should evaluate the model not only on answer accuracy but on policy compliance, function-call correctness, latency, and recovery from ambiguous requests.

The fourth phase is federated coordination. Domains publish capabilities to a mesh coordination layer. The coordinator does not become the owner of all data. Its role is to route requests, enforce inter-domain protocol standards, and manage discovery. When a request requires multiple domains, the coordinator orchestrates sub-tasks and aggregates results under policy. If raw data cannot be shared, the result is summarized or anonymized. Federated learning and privacy-preserving computation techniques can be introduced when cross-domain model improvement is needed without centralized training data (Kairouz et al., 2021; Li et al., 2020; Yang et al., 2019; Lim et al., 2020; Nguyen et al., 2021; Mothukuri et al., 2021; Bonawitz et al., 2017).

The fifth phase is monitoring and organizational learning. Every high-risk action should produce an audit record that includes request, policy status, agent plan, function calls, returned data type, human approvals, fallback decisions, and final outcome. These records should feed both technical improvement and governance review. Explainable AI and human-AI interaction research stress that users need not only outputs but also understandable reasons, challenge mechanisms, and clear accountability pathways (Amershi et al., 2019; Raji et al., 2020; Ribeiro et al., 2016; Lundberg and Lee, 2017; Doshi-Velez and Kim, 2017; Arrieta et al., 2020; Binns, 2018; Selbst et al., 2019). The goal is to create an automation system that improves safely rather than one that hides its failures until they become systemic.

A readiness assessment should precede implementation. The assessment should ask whether the domain has named owners, documented processes, stable data definitions, known compliance requirements, and measurable outcomes. Domains that lack these characteristics should not be automated first. A well-scoped support process or reporting workflow is a better starting point than a mission-critical financial approval or safety-sensitive industrial control process. The goal of early deployment is not maximum autonomy, but reliable learning about how the architecture behaves.

Organizations should also create an agent review board that combines technical, operational, legal, and domain expertise. This board should not approve every individual action. Instead, it should approve capability classes, escalation thresholds, and evaluation criteria. The board should review audit samples, incident reports, drift indicators, and user feedback. Such a structure helps prevent a common failure in AI projects: technical teams optimize model performance while governance teams lack visibility into how the system is actually used.

Implementation should include a decommissioning policy. Agents, tools, prompts, adapters, and policies may become outdated. A domain that keeps adding capabilities without retiring obsolete ones creates a growing attack surface and an increasingly confusing decision space. Each capability should therefore have an owner, a review cycle, and a retirement condition. This operational discipline is as important as initial design because system-of-systems automation will become more complex over time.

VIII. MANAGERIAL AND TECHNICAL IMPLICATIONS

For managers, the main implication is that agentic AI should be treated as infrastructure, not as a single application. Buying a general-purpose assistant and connecting it to every enterprise system is unlikely to produce reliable long-term value. Organizations should begin by identifying high-value domains where data ownership is clear, actions are reversible, and success can be measured. Early success should be used to refine governance patterns before expansion to riskier domains. This staged approach matches the evolutionary implementation logic suggested by the uploaded source article, while extending it into a lifecycle model for system-of-systems automation.

For system architects, the main implication is that model selection and architecture selection are inseparable. A powerful remote model may be appropriate for cross-domain reasoning, but a small local model may be safer and cheaper for routine operations. Edge deployment also reduces latency and helps preserve data sovereignty. Industry 4.0 research highlights the importance of cyber-physical integration, IoT security, blockchain-enabled trust, and industrial information integration for intelligent operations (Lee et al., 2015; Xu et al., 2018; Xu et al., 2014; Ben-Daya et al., 2019; Wamba et al., 2015; Lu, 2017; Lu, 2017; Lu, 2019; Zhang and Lu, 2021; Lu and Xu, 2019). Agentic AI adds a new orchestration layer to these existing digital infrastructures, but it cannot ignore their constraints.

For policymakers and governance leaders, the implication is that agentic systems require auditable protocols. Traditional IT controls focus on user permissions and system logs. Agentic AI requires additional controls: model identity, prompt provenance, tool-use justification, data-sharing justification, action reversibility, and escalation thresholds. Algorithmic accountability research shows that internal audits and continuous monitoring are essential for responsible AI deployment (Raji et al., 2020; Jobin et al., 2019; Floridi and Cowls, 2019; Mittelstadt, 2019; Binns, 2018; Selbst et al., 2019). A domain-specialized mesh architecture makes these controls easier because each node can be audited within its own operational context.

For researchers, the proposed convergence opens several questions. How should capability descriptions be standardized across domains? How can domain-specific models negotiate when their local policies conflict? What is the best way to compare central, federated, and mesh-agent architectures empirically? How should organizations measure resilience, not only accuracy? How can privacy-preserving learning be applied to agentic logs without revealing sensitive workflows? These questions require collaboration between AI researchers, information systems scholars, software architects, operations researchers, and governance experts.

A practical implication for procurement is that organizations should avoid vendor lock-in at the orchestration layer. If capability descriptions, logs, policies, and function schemas are tied to a single vendor, the enterprise may lose bargaining power and portability. Mesh architecture works best when domains can change models or tools without breaking the larger coordination protocol. Open schemas, standardized capability registries, and model-agnostic logging therefore have strategic value beyond technical convenience.

For technical teams, evaluation should move beyond offline prompt tests. Agents should be evaluated in workflow simulations that include missing data, conflicting domain responses, policy refusals, malicious instructions, stale tool descriptions, and human-review delays. These tests are closer to the real conditions under which enterprise automation fails. A model that answers test prompts accurately but calls the wrong function under ambiguity is not enterprise-ready. Conversely, a model that refuses unsafe requests may be more valuable than a more fluent but overconfident alternative.

IX. LIMITATIONS AND FUTURE RESEARCH

This paper has three limitations. First, the evaluation is scenario-based rather than based on a production deployment. This is appropriate for an architectural article because cross-domain agentic benchmarks remain immature, but future work should test the framework in real enterprise settings. Second, the framework assumes that domains can specify their capabilities and policies clearly. In practice, some organizations have weak process documentation or contested ownership. Third, the article focuses on enterprise and public-service automation; consumer-facing multi-agent ecosystems may require different trust, consent, and liability models.

Future research should develop benchmark suites for multi-context enterprise decision systems. Such benchmarks should include local tasks, cross-domain tasks, privacy-constrained tasks, ambiguous prompts, malicious requests, and failure-recovery scenarios. Accuracy alone will not be enough. Evaluation should include latency, policy compliance, data exposure, audit completeness, fault propagation, human acceptance, and rollback success. Existing agent benchmarks are useful for general tool use, but system-of-systems automation requires domain rules and governance stress tests (Park et al., 2023; Shinn et al., 2023; Karpas et al., 2022; Madaan et al., 2023; Gao et al., 2023; Liu et al., 2023; Amershi et al., 2019; Raji et al., 2020).

Another research direction is the economics of lightweight model deployment. While compression and fine-tuning methods are advancing quickly, organizations need decision models that map task type, risk level, data sensitivity, and required reasoning depth to model choice. The best solution may combine local small models, retrieval over domain documentation, a larger central model for synthesis, and human approval for irreversible decisions. Research should also examine when model specialization increases safety and when it creates fragmentation or inconsistency across domains.

A final direction concerns regulatory and ethical design. System-of-systems automation often crosses jurisdictional and

organizational boundaries. It may involve personal data, commercial secrets, safety-critical systems, or public-service decisions. A mesh architecture can support data sovereignty, but sovereignty alone does not guarantee fairness, transparency, or accountability. Future work should integrate technical controls with governance theories, fairness metrics, and organizational responsibilities so that autonomous systems do not simply distribute risk without assigning responsibility.

Future work should also examine the social organization of domain ownership. A data mesh assumes that domains can act as responsible owners, but many organizations have overlapping responsibilities and unclear authority. Agentic automation may expose these governance ambiguities rather than solve them. Empirical studies should therefore examine not only technical performance but also organizational negotiations over who owns a capability, who approves a policy, and who is accountable for cross-domain outcomes.

Another important topic is cross-domain conflict resolution. If a finance node recommends delaying a payment for risk review while a supply-chain node recommends immediate payment to avoid disruption, a higher-level decision mechanism is needed. Such conflicts cannot be solved by model reasoning alone because they involve organizational priorities. Future architectures should therefore include explicit conflict-resolution policies, not just task-routing mechanisms.

X. CONCLUSION

Agentic AI, data mesh, and lightweight language models are often discussed separately, but their convergence is likely to shape the next generation of enterprise automation. Agentic AI supplies the ability to interpret goals and coordinate tools. Data mesh supplies the organizational logic of domain ownership, data products, and federated governance. Lightweight models supply the technical feasibility of local, low-latency, domain-specialized intelligence. When these elements are combined, organizations can move toward resilient system-of-systems automation that preserves local control while enabling cross-domain action.

The paper proposed a lifecycle framework that organizes this convergence through sensing, contextualization, planning, execution, audit, and learning. It presented an architectural model, a domain-capability map, a scenario-based score analysis, a risk matrix, and an implementation roadmap. The main lesson is that automation should not be centralized merely because language models make broad orchestration possible. In complex enterprises, resilience comes from bounded autonomy: each domain remains responsible for its own data, policies, and actions, while a shared protocol enables cooperation. This combination offers a practical path toward secure, adaptive, and governable automation in complex system-of-systems environments.

Reference

1. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. arXiv. https://doi.org/10.48550/arXiv.1706.03762

2. Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. arXiv. https://doi.org/10.48550/arXiv.2005.14165

3. Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. Proceedings of NAACL-HLT, 4171-4186. https://doi.org/10.18653/v1/N19-1423

4. Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Kuttler, H., Lewis, M., Yih, W.-T., Rocktaschel, T., Riedel, S., & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. arXiv. https://doi.org/10.48550/arXiv.2005.11401

5. Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., & Cao, Y. (2023). ReAct: Synergizing reasoning and acting in language models. arXiv. https://doi.org/10.48550/arXiv.2210.03629

6. 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. arXiv. https://doi.org/10.48550/arXiv.2302.04761

7. Park, J. S., O'Brien, J., Cai, C. J., Morris, M. R., Liang, P., & Bernstein, M. S. (2023). Generative agents: Interactive simulacra of human behavior. Proceedings of UIST 2023. https://doi.org/10.1145/3586183.3606763

8. Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K., & Yao, S. (2023). Reflexion: Language agents with verbal reinforcement learning. arXiv. https://doi.org/10.48550/arXiv.2303.11366

9. Karpas, E., Abend, O., Belinkov, Y., Lenz, B., Lieber, O., Ratner, N., Shoham, Y., Bata, H., Levine, Y., Leyton-Brown, K., Muhlgay, D., Roit, P., Shashua, A., & Shoham, Y. (2022). MRKL systems: A modular, neuro-symbolic architecture that combines large language models, external knowledge sources and discrete reasoning. arXiv. https://doi.org/10.48550/arXiv.2205.00445

10. Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegreffe, S., Alon, U., Dziri, N., Prabhumoye, S., Yang, Y., Gupta, S., Majumder, B. P., Hermann, K., Welleck, S., Yazdanbakhsh, A., & Clark, P. (2023). Self-Refine: Iterative refinement with self-feedback. arXiv. https://doi.org/10.48550/arXiv.2303.17651

11. Gao, L., Madaan, A., Zhou, S., Alon, U., Liu, P., Yang, Y., Callan, J., & Neubig, G. (2023). PAL: Program-aided language models. arXiv. https://doi.org/10.48550/arXiv.2211.10435

12. Liu, X., Yu, H., Zhang, H., Xu, Y., Lei, X., Lai, H., Gu, Y., Ding, H., Men, K., Yang, K., Zhang, S., Deng, X., Zeng, A., Du, Z., Zhang, C., Shen, S., Zhang, T., Su, Y., Sun, H., & Tang, J. (2023). AgentBench: Evaluating LLMs as agents. arXiv. https://doi.org/10.48550/arXiv.2308.03688

13. Wooldridge, M., & Jennings, N. R. (1995). Intelligent agents: Theory and practice. The Knowledge Engineering Review, 10(2), 115-152. https://doi.org/10.1017/S0269888900008122

14. Jennings, N. R. (2001). An agent-based approach for building complex software systems. Communications of the ACM, 44(4), 35-41. https://doi.org/10.1145/367211.367250

15. Kephart, J. O., & Chess, D. M. (2003). The vision of autonomic computing. Computer, 36(1), 41-50. https://doi.org/10.1109/MC.2003.1160055

16. Huebscher, M. C., & McCann, J. A. (2008). A survey of autonomic computing-degrees, models, and applications. ACM Computing Surveys, 40(3), 1-28. https://doi.org/10.1145/1380584.1380585

17. Maier, M. W. (1998). Architecting principles for systems-of-systems. Systems Engineering, 1(4), 267-284. https://doi.org/10.1002/(SICI)1099-1743(199804/06)1:4<267::AID-SYS3>3.0.CO;2-D

18. Boardman, J., & Sauser, B. (2006). System of systems-the meaning of of. IEEE International Conference on System of Systems Engineering, 118-123. https://doi.org/10.1109/SYSOSE.2006.1652284

19. Gorod, A., Sauser, B., & Boardman, J. (2008). System-of-systems engineering management: A review of modern history and a path forward. IEEE Systems Journal, 2(4), 484-499. https://doi.org/10.1109/JSYST.2008.2007163

20. Sage, A. P., & Cuppan, C. D. (2001). On the systems engineering and management of systems of systems and federations of systems. Information Knowledge Systems Management, 2(4), 325-345. https://doi.org/10.3233/IKS-2001-2404

21. Abraham, R., Schneider, J., & vom Brocke, J. (2019). Data governance: A conceptual framework, structured review, and research agenda. International Journal of Information Management, 49, 424-438. https://doi.org/10.1016/j.ijinfomgt.2019.07.008

22. Alhassan, I., Sammon, D., & Daly, M. (2016). Data governance activities: A comparison between scientific and practice-oriented literature. Journal of Enterprise Information Management, 29(2), 300-316. https://doi.org/10.1108/JEIM-01-2015-0007

23. Khatri, V., & Brown, C. V. (2010). Designing data governance. Communications of the ACM, 53(1), 148-152. https://doi.org/10.1145/1629175.1629210

24. Otto, B. (2011). Organizing data governance: Findings from the telecommunications industry and consequences for large service providers. Communications of the Association for Information Systems, 29, 45-66. https://doi.org/10.17705/1CAIS.02903

25. Davenport, T. H., Barth, P., & Bean, R. (2012). How big data is different. MIT Sloan Management Review, 54(1), 43-46. https://doi.org/10.7551/mitpress/10217.003.0006

26. Chen, H., Chiang, R. H. L., & Storey, V. C. (2012). Business intelligence and analytics: From big data to big impact. MIS Quarterly, 36(4), 1165-1188. https://doi.org/10.2307/41703503

27. Gregor, S., & Hevner, A. R. (2013). Positioning and presenting design science research for maximum impact. MIS Quarterly, 37(2), 337-355. https://doi.org/10.25300/MISQ/2013/37.2.01

28. Benlian, A., Kettinger, W. J., Sunyaev, A., & Winkler, T. J. (2018). Special section: The transformative value of cloud computing. Journal of Management Information Systems, 35(3), 719-739. https://doi.org/10.1080/07421222.2018.1481638

29. Tiwana, A. (2015). Evolutionary competition in platform ecosystems. Information Systems Research, 26(2), 266-281. https://doi.org/10.1287/isre.2015.0573

30. Dragoni, N., Bucchiarone, A., Giallorenzo, S., Lafuente, A. L., Mazzara, M., Montesi, F., Mustafin, R., & Safina, L. (2017). Microservices: Yesterday, today, and tomorrow. In Present and Ulterior Software Engineering (pp. 195-216). https://doi.org/10.1007/978-3-319-67425-4_12

31. Di Francesco, P., Lago, P., & Malavolta, I. (2019). Architecting with microservices: A systematic mapping study. Journal of Systems and Software, 150, 77-97. https://doi.org/10.1016/j.jss.2018.09.001

32. Soldani, J., Tamburri, D. A., & Van Den Heuvel, W.-J. (2018). The pains and gains of microservices: A systematic grey literature review. Journal of Systems and Software, 146, 215-232. https://doi.org/10.1016/j.jss.2018.09.082

33. Jamshidi, P., Pahl, C., Chinenyeze, S., & Liu, X. (2018). Cloud migration patterns: A multi-cloud service architecture perspective. Service-Oriented Computing and Applications, 12, 141-158. https://doi.org/10.1007/s11761-018-0231-4

34. Taibi, D., Lenarduzzi, V., & Pahl, C. (2017). Processes, motivations, and issues for migrating to microservices architectures: An empirical investigation. IEEE International Conference on Software Architecture Workshops, 22-29. https://doi.org/10.1109/ICSAW.2017.11

35. Balalaie, A., Heydarnoori, A., & Jamshidi, P. (2016). Microservices architecture enables DevOps: Migration to a cloud-native architecture. IEEE Software, 33(3), 42-52. https://doi.org/10.1109/MS.2016.64

36. Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., et al. (2021). Advances and open problems in federated learning. Foundations and Trends in Machine Learning, 14(1-2), 1-210. https://doi.org/10.1561/2200000083

37. Li, T., Sahu, A. K., Talwalkar, A., & Smith, V. (2020). Federated learning: Challenges, methods, and future directions. IEEE Signal

Processing Magazine, 37(3), 50-60. https://doi.org/10.1109/MSP.2020.2975749

38. Yang, Q., Liu, Y., Chen, T., & Tong, Y. (2019). Federated machine learning: Concept and applications. ACM Transactions on Intelligent Systems and Technology, 10(2), 1-19. https://doi.org/10.1145/3298981
39. Lim, W. Y. B., Luong, N. C., Hoang, D. T., Jiao, Y., Liang, Y.-C., Yang, Q., Niyato, D., & Miao, C. (2020). Federated learning in mobile edge networks: A comprehensive survey. IEEE Communications Surveys & Tutorials, 22(3), 2031-2063. https://doi.org/10.1109/COMST.2020.2986024
40. Nguyen, D. C., Ding, M., Pathirana, P. N., Seneviratne, A., Li, J., & Poor, H. V. (2021). Federated learning for Internet of Things: A comprehensive survey. IEEE Internet of Things Journal, 8(16), 12816-12843. https://doi.org/10.1109/JIOT.2021.3075439
41. Mothukuri, V., Khare, P., Parizi, R. M., Pouriyeh, S., Dehghantanha, A., & Srivastava, G. (2021). Federated-learning-based anomaly detection for IoT security attacks. IEEE Internet of Things Journal, 9(4), 2545-2554. https://doi.org/10.1109/JIOT.2021.3077803
42. Dwork, C. (2006). Differential privacy. In Automata, Languages and Programming (pp. 1-12). https://doi.org/10.1007/11787006_1
43. Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., & Zhang, L. (2016). Deep learning with differential privacy. Proceedings of the ACM CCS, 308-318. https://doi.org/10.1145/2976749.2978318
44. Shokri, R., & Shmatikov, V. (2015). Privacy-preserving deep learning. Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, 1310-1321. https://doi.org/10.1145/2810103.2813687
45. Gentry, C. (2009). Fully homomorphic encryption using ideal lattices. Proceedings of the 41st Annual ACM Symposium on Theory of Computing, 169-178. https://doi.org/10.1145/1536414.1536440
46. Shamir, A. (1979). How to share a secret. Communications of the ACM, 22(11), 612-613. https://doi.org/10.1145/359168.359176
47. Yao, A. C. (1982). Protocols for secure computations. 23rd Annual Symposium on Foundations of Computer Science, 160-164. https://doi.org/10.1109/SFCS.1982.38
48. Lindell, Y., & Pinkas, B. (2009). Secure multiparty computation for privacy-preserving data mining. Journal of Privacy and Confidentiality, 1(1), 59-98. https://doi.org/10.29012/jpc.v1i1.566
49. Nikolaenko, V., Ioannidis, S., Weinsberg, U., Joye, M., Boneh, D., & Taft, N. (2013). Privacy-preserving ridge regression on hundreds of millions of records. IEEE Symposium on Security and Privacy, 334-348. https://doi.org/10.1109/SP.2013.30
50. Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H. B., Patel, S., Ramage, D., Segal, A., & Seth, K. (2017). Practical secure aggregation for privacy-preserving machine learning. Proceedings of the ACM CCS, 1175-1191. https://doi.org/10.1145/3133956.3133982
51. Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., & Chen, W. (2022). LoRA: Low-rank adaptation of large language models. arXiv. https://doi.org/10.48550/arXiv.2106.09685
52. Dettmers, T., Pagnoni, A., Holtzman, A., & Zettlemoyer, L. (2023). QLoRA: Efficient finetuning of quantized LLMs. arXiv. https://doi.org/10.48550/arXiv.2305.14314
53. Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. arXiv. https://doi.org/10.48550/arXiv.1503.02531
54. Han, S., Mao, H., & Dally, W. J. (2016). Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. arXiv. https://doi.org/10.48550/arXiv.1510.00149
55. Frantar, E., Ashkboos, S., Hoefler, T., & Alistarh, D. (2023). GPTQ: Accurate post-training quantization for generative pre-trained transformers. arXiv. https://doi.org/10.48550/arXiv.2210.17323
56. Lin, J., Tang, J., Tang, H., Yang, S., Dang, X., Han, S., & Wang, S. (2024). AWQ: Activation-aware weight quantization for LLM compression and acceleration. arXiv. https://doi.org/10.48550/arXiv.2306.00978
57. Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter. arXiv. https://doi.org/10.48550/arXiv.1910.01108
58. Jiao, X., Yin, Y., Shang, L., Jiang, X., Chen, X., Li, L., Wang, F., & Liu, Q. (2020). TinyBERT: Distilling BERT for natural language understanding. arXiv. https://doi.org/10.48550/arXiv.1909.10351
59. Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., & Soricut, R. (2020). ALBERT: A lite BERT for self-supervised learning of language representations. arXiv. https://doi.org/10.48550/arXiv.1909.11942
60. Amershi, S., Weld, D., Vorvoreanu, M., Fourney, A., Nushi, B., Collisson, P., Suh, J., Iqbal, S., Bennett, P. N., Inkpen, K., et al. (2019). Guidelines for human-AI interaction. Proceedings of CHI 2019, 1-13. https://doi.org/10.1145/3290605.3300233
61. Raji, I. D., Smart, A., White, R. N., Mitchell, M., Gebru, T., Hutchinson, B., Smith-Loud, J., Theron, D., & Barnes, P. (2020). Closing the AI accountability gap: Defining an end-to-end framework for internal algorithmic auditing. Proceedings of FAT* 2020, 33-44. https://doi.org/10.1145/3351095.3372873
62. Jobin, A., Ienca, M., & Vayena, E. (2019). The global landscape of AI ethics guidelines. Nature Machine Intelligence, 1, 389-399. https://doi.org/10.1038/s42256-019-0088-2
63. Floridi, L., & Cowls, J. (2019). A unified framework of five principles for AI in society. Harvard Data Science Review, 1(1). https://doi.org/10.1162/99608f92.8cd550d1
64. Mittelstadt, B. (2019). Principles alone cannot guarantee ethical AI. Nature Machine Intelligence, 1, 501-507. https://doi.org/10.1038/s42256-019-0114-4

65. Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). Why should I trust you? Explaining the predictions of any classifier. Proceedings of KDD 2016, 1135-1144. https://doi.org/10.1145/2939672.2939778

66. Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. arXiv. https://doi.org/10.48550/arXiv.1705.07874

67. Doshi-Velez, F., & Kim, B. (2017). Towards a rigorous science of interpretable machine learning. arXiv. https://doi.org/10.48550/arXiv.1702.08608

68. Arrieta, A. B., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., Garcia, S., Gil-Lopez, S., Molina, D., Benjamins, R., Chatila, R., & Herrera, F. (2020). Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges. Information Fusion, 58, 82-115. https://doi.org/10.1016/j.inffus.2019.12.012

69. Binns, R. (2018). Fairness in machine learning: Lessons from political philosophy. Proceedings of FAT* 2018, 149-159. https://doi.org/10.1145/3287560.3287582

70. Selbst, A. D., Boyd, D., Friedler, S. A., Venkatasubramanian, S., & Vertesi, J. (2019). Fairness and abstraction in sociotechnical systems. Proceedings of FAT* 2019, 59-68. https://doi.org/10.1145/3287560.3287598

71. Lee, J., Bagheri, B., & Kao, H.-A. (2015). A cyber-physical systems architecture for Industry 4.0-based manufacturing systems. Manufacturing Letters, 3, 18-23. https://doi.org/10.1016/j.mfglet.2014.12.001

72. Xu, L. D., Xu, E. L., & Li, L. (2018). Industry 4.0: State of the art and future trends. International Journal of Production Research, 56(8), 2941-2962. https://doi.org/10.1080/00207543.2018.1444806

73. Xu, L. D., He, W., & Li, S. (2014). Internet of Things in industries: A survey. IEEE Transactions on Industrial Informatics, 10(4), 2233-2243. https://doi.org/10.1109/TII.2014.2300753

74. Ben-Daya, M., Hassini, E., & Bahroun, Z. (2019). Internet of Things and supply chain management: A literature review. International Journal of Production Research, 57(15-16), 4719-4742. https://doi.org/10.1080/00207543.2017.1402140

75. Wamba, S. F., Gunasekaran, A., Akter, S., Ren, S. J.-F., Dubey, R., & Childe, S. J. (2015). Big data analytics and firm performance: Effects of dynamic capabilities. Journal of Business Research, 70, 356-365. https://doi.org/10.1016/j.jbusres.2016.08.009

76. Lu, Y. (2017). Industry 4.0: A survey on technologies, applications and open research issues. Journal of Industrial Information Integration, 6, 1-10. https://doi.org/10.1016/j.jii.2017.04.005

77. Lu, Y. (2017). Cyber physical system (CPS)-based Industry 4.0: A survey. Journal of Industrial Integration and Management, 2(3), 1750014. https://doi.org/10.1142/S2424862217500142

78. Lu, Y. (2019). Artificial intelligence: A survey on evolution, models, applications and future trends. Journal of Management Analytics, 6(1), 1-29. https://doi.org/10.1080/23270012.2019.1570365

79. Zhang, C., & Lu, Y. (2021). Study on artificial intelligence: The state of the art and future prospects. Journal of Industrial Information Integration, 23, 100224. https://doi.org/10.1016/j.jii.2021.100224

80. Lu, Y., & Xu, L. D. (2019). Internet of Things (IoT) cybersecurity research: A review of current research topics. IEEE Internet of Things Journal, 6(2), 2103-2115. https://doi.org/10.1109/JIOT.2018.2869847