International Journal of AI, Big Data, Computational and Management Studies
Noble Scholar Research Group | Volume 7, Issue 2, PP 364-369, 2026
ISSN: 3050-9416 | https://doi.org/10.63282/3050-9416.IJAIBDCMS-V7I2P146

Original Article

A Reference AI Architecture for the Modern Data Organization: Mapping Seven Branches to a Ten-Layer Implementation Stack

Smeet H Patel
Director, Enterprise BI & AI/ML Engineering, USA.

Received On: 24/04/2026 ***Revised On: 23/05/2026*** ***Accepted On: 30/05/2026*** ***Published On: 06/06/2026***

Abstract: *Enterprise data organizations are pursuing AI augmentation, but practitioner literature offers either single-branch agent systems (a data-science agent, a stewardship agent, a project-management agent) or unbounded surveys of agentic capabilities. Neither is implementable on day one. This paper proposes a complete task-to-architecture-to-tooling mapping for a Director of Data Engineering, or technology executive in charge of an enterprise data function. It identifies the seven canonical branches of a mature data organization (BI Engineering, AI/ML Engineering, Analytics, Data Science, Data Stewardship, Project Management Office, and Product Ownership), enumerates the three highest-volume operational tasks per branch (twenty-one tasks total), and proposes a ten-layer reference AI architecture that binds each layer to representative open-source and commercial tools (Cube, AtScale, Snowflake, Databricks, Collibra, Atlan, LangGraph, Pinecone, Monte Carlo, and others). A worked trace shows how a real cross-branch request moves through the architecture, and a phased adoption roadmap converts the framework into a 0-180 day implementation plan that the reader can take to a steering committee on Monday morning. The framework is LLM-agnostic, vendor-substitutable at the capability class, and protocol-aligned with Model Context Protocol and Agent2Agent. Reproducibility and ethical guardrails appear as standalone sections.*

Keywords: *AI-Augmented Data Organization, Multi-Agent Systems, Semantic Layer, Reference Architecture, MCP, A2A, Langgraph, Cube, Snowflake, Enterprise Data Engineering, Mlops, Phased Adoption.*

1. Introduction

A Director of Data, or any technology executive responsible for an enterprise data function, faces a stark gap between AI augmentation rhetoric and implementable architecture. Practitioner literature splits into two unsatisfying camps. The first publishes single-branch agent systems aimed at one slice of the organization: a data-science agent, a stewardship agent, a PMO agent. Each is internally sophisticated but addresses only one branch. The second publishes unbounded "AI for data" surveys cataloging dozens of capabilities without committing to which tools, which integrations, and which sequence make a deployable system. An executive who needs to scope a phased AI implementation in the next sixty days finds neither sufficient.

This paper closes that gap with a single artifact. It enumerates the seven canonical branches of a mature enterprise data organization, identifies the three highest-volume operational tasks per branch (twenty-one tasks total), specifies a ten-layer reference AI architecture, names representative tools per layer with explicit interchangeability at the class level, and maps every task to its supporting architecture layers. The paper then traces one cross-branch request end-to-end, and converts the framework into a phased adoption roadmap covering days 0 through 180. The intended reader is a Director or VP of Data Engineering. The intended outcome is a one-page implementation plan derivable from this paper alone.

The contributions of this work are a published top-3 task enumeration covering all seven data-team branches; a ten-layer reference architecture with named tool classes; a 21-task-to-architecture mapping matrix; a worked example with code-level traceability; a phased adoption roadmap; and reproducibility and ethics guidance applicable across deployment contexts. The remainder of this paper presents related work (Section 2), the seven-branch task taxonomy (Section 3), the reference architecture (Section 4), the task-to-architecture mapping (Section 5), the worked example (Section 6), the phased adoption roadmap (Section 7), discussion and limitations (Section 8), reproducibility (Section 9), ethical considerations (Section 10), and conclusion (Section 11).

2. Related Work

Multi-agent LLM systems for knowledge work matured rapidly between 2024 and 2026. AutoGen [1], LangGraph [2], and CrewAI [3] established role-based agent decomposition as a working paradigm. MetaGPT [4] framed software development as a multi-agent workflow with fixed roles. These efforts demonstrated that role decomposition works in software domains, but none addresses the heterogeneous task distribution of an enterprise data team. Branch-specific data-

team agent work exists in volume. Data Agent [5] proposes a holistic architecture covering data science, analytics, and database administration; the 2025 LLM-based data science agent survey catalogs more than thirty domain-specific systems [6]. Vendor coverage of agentic data stewardship has emerged in industry literature [7]. Each addresses one branch.

Two open protocols are reshaping integration. The Model Context Protocol [8], donated to the Linux Foundation in late 2025 and extended with OAuth 2.1 in early 2026, standardizes agent-to-tool communication. The Agent2Agent protocol [9] standardizes inter-agent messaging. Both are integration primitives, not architectures, and the framework presented here adopts them as such.

The semantic layer is independently mature. Cube provides an open-source headless semantic layer, AtScale provides an enterprise universal semantic layer [10], and dbt ships metric definitions adjacent to dbt models. Vector memory is dominated by Pinecone, Weaviate, and pgvector. Catalog and lineage are dominated by Collibra, Atlan, and Alation. Each is well-known in isolation. None has been published in a unified data-team AI reference architecture.

This paper extends prior work along three dimensions. It enumerates top-3 operational tasks for all seven data-team branches under one roof, rather than per branch in isolation. It binds each layer of a ten-layer reference architecture to a named tool class. And it maps tasks to layers in a single matrix that practitioners can adopt directly. The novel synthesis is the combination, not the individual layers, which are well-established. Table 1 positions this contribution against representative prior work.

Table 1: Positioning of this Work against Representative Prior Work

Prior Work	Branch Coverage	Top-3 Tasks	Architecture	Named Tools	Implementable
MetaGPT [4]	Software only	-	-	partial	partial
Data Agent [5]	DS + Anl + DBA	-	partial	partial	partial
DS-Agent survey [6]	Data Science only	-	-	partial	-
Agentic Stewardship [7]	Stewardship only	-	-	partial	partial
This work	All seven branches	21 tasks	10 layers	Each layer	Yes

3. The Seven Branches and Top-3 Operational Tasks

A mature enterprise data organization comprises seven functional branches. Each has a distinct deliverable, a distinct judgment pattern, and a distinct risk profile. The top-3 operational tasks per branch (Table 2) represent the work that consumes the bulk of weekly capacity. The remainder of this paper specifies an architecture against which all twenty-one tasks can be incrementally augmented.

Table 2: Twenty-One Top-3 Operational Tasks across the Seven Branches of the Enterprise Data Organization

Branch	Top-3 Operational Tasks
BI Engineering	T1: ETL/ELT pipeline development and maintenance T2: Dimensional model and semantic-layer authoring T3: Production reporting model deployment and refresh
AI/ML Engineering	T4: Feature store population and feature pipelines T5: Model deployment and real-time serving T6: Model monitoring, drift detection, and retraining
Analytics	T7: Ad-hoc analysis and SQL authoring T8: Dashboard development and maintenance T9: Business-question answering for non-technical users
Data Science	T10: Predictive model development T11: Causal inference and experimentation T12: Feature exploration and prototype evaluation
Data Stewardship	T13: Data quality monitoring and remediation T14: Catalog and lineage maintenance T15: Privacy and policy enforcement (PII, regulatory)
PMO	T16: Intake triage and request decomposition T17: Status reporting and dependency tracking T18: Capacity planning and cross-team prioritization
Product Ownership	T19: Requirements elicitation and user-story authoring T20: Backlog grooming and sprint planning T21: Acceptance testing and UAT coordination

4. Reference AI Architecture

The reference architecture (Figure 1) comprises ten layers. Each layer is named by capability class; specific tools cited are representative implementations and are interchangeable with equivalents. Layers L1 through L9 are sequenced in request flow; L10 cross-cuts.

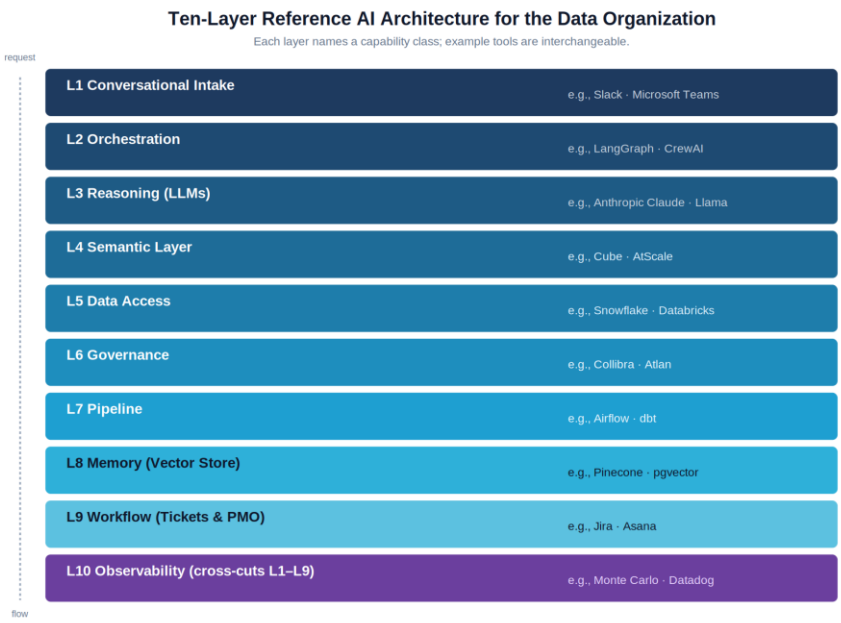

Fig 1: Ten-Layer Reference AI Architecture for the Modern Data Organization

- L1 Conversational Intake: The user-facing surface where requests enter. Slack and Microsoft Teams cover dominant enterprise chat surfaces. Intake normalizes inbound requests, attaches requester identity, and emits a structured task envelope to L2.
- L2 Orchestration: A multi-agent orchestrator that decomposes the request into subtasks, selects relevant specialists, and routes execution. The orchestrator is itself an LLM-driven agent, typically built with LangGraph or CrewAI.
- L3 Reasoning: The LLM family the system relies on for natural-language reasoning, code generation, and judgment. Multi-LLM deployment is recommended: a frontier model for high-stakes reasoning and a cost-efficient model for high-volume routine inference. The architecture is provider-agnostic.
- L4 Semantic Layer: The single source of truth for metric definitions, dimensional models, and access policy. Cube and AtScale are the canonical implementations. Without a semantic layer, AI agents generating SQL produce divergent metric calculations across tools, which is the most expensive failure mode in an AI-augmented data team.
- L5 Data Access: The warehouse, lakehouse, and reverse-ETL surfaces over which the system operates. Snowflake and Databricks are dominant. Read access is the default for AI agents; write access is gated behind explicit human approval (Section 7).
- L6 Governance: Data catalog, lineage, classification, and policy enforcement. Collibra and Atlan are representative. The governance layer answers two questions for every AI-generated query: is the requester authorized to see this data, and is the requested usage compliant with retention and privacy policy.
- L7 Pipeline: Workflow orchestration for batch and streaming data movement. Airflow and dbt cover most production patterns. AI-generated DAG modifications and authoring suggestions are reviewed at this layer before merge.
- L8 Memory: A vector store providing organizational memory across requests. Pinecone, Weaviate, and pgvector are the representative options. Memory is partitioned by access tier (Confidential, Restricted, Public-Internal) and retrieval is gated by requester scope.
- L9 Workflow: Ticketing, project tracking, and acceptance management. Jira and Asana are representative. The workflow layer is the PMO and Product Ownership branches' primary surface.
- L10 Observability: End-to-end monitoring including data quality, pipeline health, model drift, and AI-agent behavior. Monte Carlo and Datadog cover the dominant patterns. Every AI-agent action is traceable through L10 to its originating L1 request.

Four design principles govern the architecture. Layer separation: each layer has a single responsibility and a stable interface. Open-protocol integration: MCP for tool access and A2A for inter-agent messaging. Policy enforcement at the layer of authority: the semantic layer owns metric definitions; the governance layer owns access policy; neither is delegated to LLM prompts. Observability by default: every cross-layer call emits a trace event to L10.

5. Task-to-Architecture Mapping

Table 3 maps each of the twenty-one tasks to the architecture layers required for AI-augmented execution. Layers shown are the minimum viable set; production deployments incorporate L10 for every task and L10 is therefore omitted from the table for readability.

Table 3: The 21-Task-To-10-Layer Mapping Matrix. L10 (Observability) Applies to All Rows and is Omitted for Readability

Task	L1	L2	L3	L4	L5	L6	L7	L8	L9
T1 ETL/ELT pipelines	✓	✓	✓		✓	✓	✓		
T2 Dim model + semantic	✓	✓	✓	✓	✓	✓			
T3 Reporting deploy	✓	✓	✓	✓	✓		✓		
T4 Feature store	✓	✓	✓		✓	✓	✓		
T5 Model deploy/serve	✓	✓	✓		✓		✓		
T6 Model monitor/drift	✓	✓	✓		✓				
T7 Ad-hoc SQL	✓	✓	✓	✓	✓	✓		✓	
T8 Dashboards	✓	✓	✓	✓	✓	✓		✓	✓
T9 Business Q&A	✓	✓	✓	✓	✓	✓		✓	
T10 Predictive model	✓	✓	✓		✓	✓	✓	✓	
T11 Causal inference	✓	✓	✓	✓	✓	✓		✓	
T12 Feature exploration	✓	✓	✓		✓	✓		✓	
T13 Data quality	✓	✓	✓		✓	✓	✓		
T14 Catalog/lineage	✓	✓	✓	✓	✓	✓	✓	✓	✓
T15 Privacy/policy	✓	✓	✓	✓	✓	✓		✓	✓
T16 PMO intake	✓	✓	✓					✓	✓
T17 PMO status	✓	✓	✓					✓	✓
T18 Capacity planning	✓	✓	✓					✓	✓
T19 PO requirements	✓	✓	✓					✓	✓
T20 Backlog grooming	✓	✓	✓					✓	✓
T21 UAT coordination	✓	✓	✓		✓			✓	✓

6. Worked Example

Consider a representative cross-branch request. An executive types in Slack: "Show me weekly customer churn for the last six months by region, with a refresh schedule." The trace below shows how the request moves through the architecture. Latencies and exact code paths are illustrative; production deployments vary.

Table 4: End-To-End Trace of One Cross-Branch Request through the Ten-Layer Architecture

Step	Layer	Action
1	L1 Intake	Slack receives the message; a webhook normalizes it into an envelope with request_id, user, and channel; the envelope is submitted to the orchestrator.
2	L2 + L3	LangGraph decomposes the request into three subtasks (metric resolution, data retrieval, refresh schedule), invoking the LLM for parsing and routing.
3	L4 Semantic	Cube resolves "customer churn" to its governed metric definition. The LLM does not invent the formula; it queries the semantic layer (code below).
4	L5 + L6	The query executes against Snowflake; Atlan-defined masking policies hide PII based on the requester's role; lineage is recorded.
5	L7 Pipeline	Airflow generates a draft DAG for the weekly refresh. The DAG diff is surfaced for human review before commit (write action; gated).
6	L8 + L9	Pinecone retrieves three similar past requests to inform decomposition; on success, the new pattern is written back. Jira opens a tracking ticket.
7	L10 Observability	Monte Carlo and OpenTelemetry emit a trace event for each cross-layer call. The full request is auditable end-to-end through the request_id.

The most distinctive moment is the L4 semantic resolution: rather than asking the LLM to derive the churn formula from prose, the metric specialist queries Cube for the governed definition. This single discipline prevents the most expensive class of failure in an AI-augmented data team, where two agents return numerically different answers to the same business question.

```
# L4: Cube semantic resolution (representative)
import requests

cube_q = {"query": {
   "measures":   ["Churn.weekly_rate"],
   "dimensions": ["Customer.region"],
   "timeDimensions": [{
      "dimension":   "Churn.week",
```

```
      "granularity": "week",
      "dateRange":   "last 6 months"}]}}

resp = requests.post(
   f"{CUBE_URL}/cubejs-api/v1/load",
   headers={"Authorization": CUBE_TOKEN},
   json=cube_q)
data = resp.json()["data"]
```

7. Phased Adoption Roadmap

The framework converts directly into a phased implementation plan. Table 5 specifies a 0-180 day roadmap that an executive can take to a steering committee. The phases are sequential because they encode capability dependencies: layers L1, L2, L3, and L8 form the minimum viable spine; L4 and L6 are non-negotiable before any read-only AI agent is exposed to a real user; write capabilities at L5, L7, and L9 are intentionally last because they carry the most operational and regulatory risk.

Table 5: A 0-180 Day Phased Adoption Roadmap for the Ten-Layer Architecture

Phase	Theme	Scope and Goal
Days 0-30	Foundation	Stand up L1 (Slack bot), L2 (LangGraph), L3 (single LLM provider with function-calling), L8 (pgvector). Enable a single read-only specialist for one analytics task (T7 or T9). Goal: prove end-to-end traceability of one request through the architecture.
Days 31-90	Semantic and governance	Stand up L4 (Cube on top of existing warehouse) and L6 (Atlan or OpenMetadata). Expose three read-only specialists: T7, T9, T14. Begin populating L8 with successful request patterns. Goal: zero divergent metric calculations across agents and BI tools.
Days 91-150	Pipeline read and ML monitoring	Add L5 read access for T1, T4, T13; add L10 (Monte Carlo or OSS equivalent) for data quality and L7 read for pipeline awareness. Begin T6 monitoring with no autonomous retraining. Goal: stewardship and ML/Eng tasks under observability.
Days 151-180	Gated write actions	Enable Checkpoint-tier write actions for L5 (DDL/DML), L7 (DAG modification), L9 (ticket creation). Onboard PMO and PO branches (T16-T21). Begin tier-1 production support. Goal: full seven-branch coverage with explicit human approval on writes.

Two adoption anti-patterns are worth naming. The first is starting with autonomous writes - granting AI agents production DDL or pipeline-modification rights before observability and governance are mature. This is the fastest path to an audit finding. The second is skipping the semantic layer because "the LLM can write SQL." The LLM can write SQL; it cannot guarantee that two agents return the same answer to the same business question without a governed source. Skipping L4 buys two months of speed and twelve months of metric reconciliation work.

8. Discussion and Limitations

The framework's value comes from layer-symmetric design: cross-cutting concerns (auth, audit, retry, rate limiting) are handled at one layer of authority and inherited elsewhere. This is the difference between an implementable architecture and a slide. Vendor selection at each layer is a procurement decision; structural change is required only when capability classes themselves change.

Limitations of this work are: the seven-branch taxonomy reflects a canonical mid-to-large data organization, and ML-research labs or pure consulting shops may consolidate or omit branches; the 21-task list captures top-3 operational work and excludes long-tail tasks that vary by organization; the architecture assumes a unified identity provider and a coherent governance model, and organizations lacking either should remediate before adopting; latency budgets and capacity planning are deployment-dependent and outside the scope of this paper; and the framework does not prescribe LLM evaluation methodology, which remains an active area of practitioner work.

9. Reproducibility

The framework is reproducible from open-source components alone. A minimum viable deployment uses Slack Bolt for L1; LangGraph for L2; any frontier or open-weight LLM with function-calling for L3; Cube Community Edition for L4; any cloud warehouse (Snowflake free tier or DuckDB for prototyping) for L5; OpenMetadata for L6; Apache Airflow or Dagster for L7; pgvector for L8; Jira free tier for L9; and OpenTelemetry plus Grafana for L10. This stack is replicable on a single laptop for development and on commodity cloud for production.

10. Ethical Considerations

Three guardrails apply across all twenty-one tasks. First, write-capable layers (L5 DDL/DML, L7 DAG modification, L9 ticket creation) are gated behind explicit human approval; autonomous execution is reserved for read-only operations. Second, governance and memory enforce access scope; AI agents do not inherit privileges beyond the requesting user, and memory partitioning prevents cross-tenant or cross-classification leakage. Third, the framework rejects productivity rhetoric framing AI augmentation as workforce reduction. The documented intent of any deployment should be capacity reallocation toward higher-judgment work, not headcount substitution. Continuous calibration of AI-agent

reliability is monitored as a first-class signal; silent regressions are framework violations with on-call escalation.

11. Conclusion

This paper introduces a complete task-to-architecture-to-tooling mapping for AI-augmenting an enterprise data organization. It contributes a published top-3 operational task enumeration covering all seven data-team branches (twenty-one tasks total), a ten-layer reference architecture with named tool classes, a 21-task-to-architecture mapping matrix, a worked example with code-level traceability, a phased adoption roadmap, and reproducibility and ethics guidance. The framework is implementable on commodity cloud with open-source components, vendor-substitutable at the capability class, and protocol-aligned with MCP and A2A. A Director of Data, or any technology executive responsible for an enterprise data function, can derive a one-page execution plan from this paper alone. The framework introduces a unified reference architecture for AI augmentation across the full data-team surface, and extends naturally to adjacent knowledge-work organizations - legal, finance, and research operations share analogous branch structures.

References

1. Q. Wu, G. Bansal, J. Zhang, Y. Wu, B. Li, E. Zhu, L. Jiang, X. Zhang, S. Zhang, J. Liu, A. H. Awadallah, R. W. White, D. Burger, and C. Wang, "AutoGen: Enabling next-gen LLM applications via multi-agent conversation," in Proc. Conf. Language Modeling (COLM), 2024.
2. LangChain, "LangGraph: Stateful multi-actor applications with LLMs," LangChain documentation, 2025. [Online]. Available: https://langchain-ai.github.io/langgraph/
3. CrewAI, "CrewAI: Framework for orchestrating role-playing autonomous agents," CrewAI documentation, 2025. [Online]. Available: https://docs.crewai.com/
4. S. Hong, X. Zheng, J. Chen, Y. Cheng, C. Zhang, Z. Wang, S. K. S. Yau, Z. Lin, L. Zhou, C. Ran, et al., "MetaGPT: Meta programming for a multi-agent collaborative framework," in Proc. Int. Conf. Learning Representations (ICLR), 2024.
5. G. Li, X. Zhou, and Z. Sun, "Data Agent: A holistic architecture for orchestrating data and AI ecosystems," in Proc. IEEE Int. Conf. Data Eng. (ICDE), 2025.
6. M. Sahu, S. Guo, P. Trirat, and S. J. Hwang, "LLM-based data science agents: A survey of capabilities, challenges, and future directions," arXiv preprint arXiv:2510.04023, Oct. 2025.
7. M. Villar, "Digital data steward: Leveraging agentic AI for data quality, metadata, master data management, and retention," CDO Magazine, Aug. 2025. [Online]. Available: https://www.cdomagazine.tech/
8. Anthropic, "Model Context Protocol specification," Linux Foundation / Agentic AI Foundation, 2024-2026. [Online]. Available: https://modelcontextprotocol.io/
9. Google Cloud, "Agent2Agent (A2A) Protocol specification," Google Developers Blog, Apr. 2025; merged with IBM ACP Sept. 2025. [Online]. Available: https://developers.googleblog.com/
10. Cube Dev, "Cube: Headless BI and semantic layer documentation," 2025. [Online]. Available: https://cube.dev/docs