Capability-Centric Enterprise Architecture for Business-Technology Alignment and AI Adoption

Bhagyeshkumar Chokhawala[1][0009-0001-5146-832X]

[1]*Comcast Corporation, Philadelphia, Pennsylvania, USA,bhagyeshkumar.chokhawala@comcast.com*

Abstract: This paper presents an integrated overview of four complementary enterprise architecture artifacts: Business Capability Maps, Technical Capability Maps, shared technical implementation models, and business-to-technical traceability matrices. Business Capability Maps capture the stable 'what' of the enterprise, while Technical Capability Maps capture reusable technology-side abilities independent of specific products and projects. Shared technical implementation models define how common capabilities are delivered across teams using shared, federated, or local patterns, improving reuse, governance, and speed. Traceability matrices provide the relationship layer that links business priorities and capabilities to technical capabilities, platforms, applications, and controls for prioritization, gap analysis, impact analysis, and governance. The paper then explains how to adopt artificial intelligence (AI) across these artifacts by modeling AI as business value (AI-enabled sub-capabilities), technical enablement (inference, guardrails, monitoring, auditability), shared platform capability (federated enterprise AI), and DevOps extensions (MLOps and LLMOps). The proposed approach enables organizations to align AI investments with business outcomes, risk controls, and operational resilience.

Keywords: Business capability map; technical capability map; enterprise architecture; traceability matrix; platform engineering; shared services; AI governance; MLOps; LLMOps

Introduction

Enterprises need architecture practices that connect strategy, business priorities, technology enablement, and implementation decisions without relying only on organizational charts, application inventories, or project-specific solution designs. A capability-centric approach addresses this need by organizing planning around stable abilities rather than transient structures or tools.

This paper provides an integrated overview of four artifacts that work together: Business Capability Maps, Technical Capability Maps, shared technical implementation models, and traceability matrices. It then explains how AI can be adopted consistently across all four.

Business Capability Maps

A Business Capability Map is a hierarchical representation of the enduring abilities an organization must have to execute its mission and strategy. It describes what the business does, not how work is performed, who performs it, or which applications currently support it.

Typical capabilities include customer management, order management, billing and invoicing, supplier management, and service assurance. These capabilities remain relatively stable over time, making them effective anchors for modernization, portfolio planning, and strategy alignment.

Business Capability Maps are commonly structured into levels (e.g., Level 1 domains, Level 2 sub-capabilities, and optional Level 3 detail). They are frequently used to support maturity heatmaps, application rationalization, transformation roadmaps, and risk/resilience planning.

Technical Capability Maps

A Technical Capability Map organizes the reusable technology-side abilities required to enable business goals. Examples include identity and access management, API management and integration, workflow orchestration, event streaming, observability, data engineering, CI/CD and DevSecOps, and cloud platform operations.

Unlike application inventories, a Technical Capability Map does not focus on which specific systems currently exist. Unlike solution architectures, it does not describe the design of one implementation. Instead, it captures stable technical abilities that can be implemented in different ways and reused across domains.

A key modeling rule is to name the capability as an ability (e.g., API Management) rather than as a vendor product. This keeps the technical capability model stable as platforms and tools evolve.

Shared Technical Implementation Models

A shared technical implementation model describes how technical capabilities are delivered across the enterprise. It is the bridge between capability maps and platform or reference architecture decisions.

In practice, organizations often use three delivery patterns: shared (a centralized platform service), federated (central standards and common components with domain-specific customization), and local (exception-based implementations under guardrails).

Shared implementation matters because foundational capabilities such as identity, observability, CI/CD pipelines, cloud landing zones, and data platform services are expensive and risky to duplicate. A shared model improves consistency, governance, reuse, and delivery speed.

Traceability Matrices for Business-Technical Alignment

A traceability matrix is a relationship view that maps business-side elements (e.g., capabilities, goals, value streams, requirements) to technical-side elements (e.g., technical capabilities, platforms,

applications, controls). Each intersection indicates whether and to what extent a technical element supports a business element.

Capability maps and traceability matrices are complementary. Capability maps provide the building blocks (inventory/hierarchy), while traceability matrices reveal the dependencies and support relationships across layers.

Traceability matrices support investment prioritization, gap analysis, rationalization, impact analysis, and governance. They are most useful when matrix cells include explicit semantics such as support type (direct/foundational/indirect), criticality, maturity fit, risk, ownership, and notes.

Figure 1 summarizes the integrated capability-centric architecture flow.

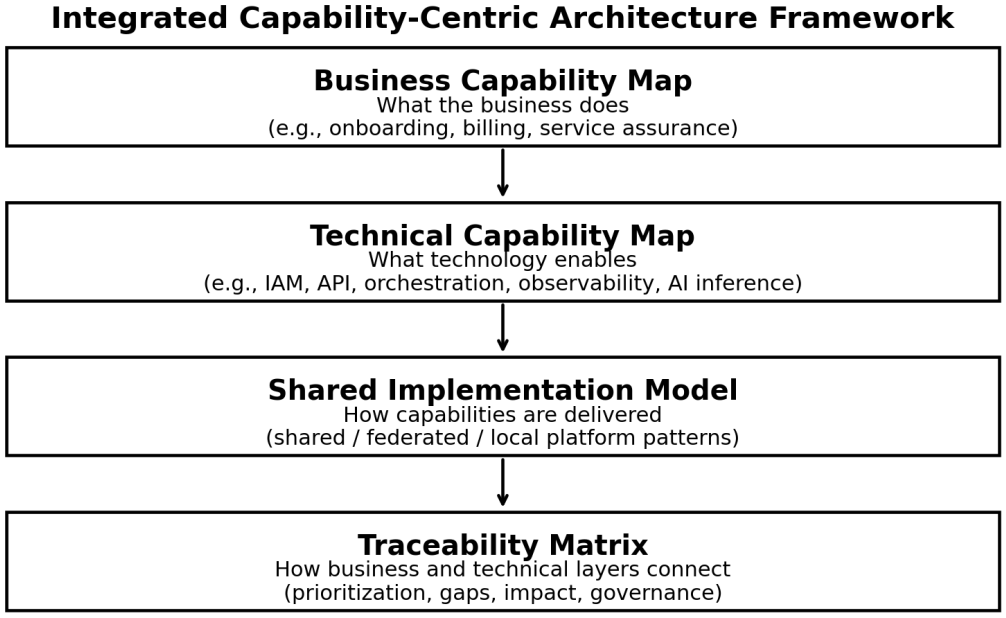

Figure 1. Integrated capability-centric architecture framework linking business, technical, shared implementation, and traceability layers.

Adopting AI Across the Four Artifacts

AI should not be modeled as a disconnected tool initiative. Instead, it should be integrated into the same capability architecture used for other enterprise investments.

First, AI should be represented in Business Capability Maps as AI-enabled sub-capabilities within existing domains (e.g., customer insight and segmentation, order fallout prediction, billing anomaly detection, incident prediction), and optionally as a standalone AI-enabled decisioning and automation domain when AI is strategically differentiating.

Second, AI should be represented in Technical Capability Maps as reusable technical abilities, such as data preparation for AI, model development and experimentation, model serving/inference runtime, prompt orchestration or LLM gateway, vector retrieval services, model monitoring and drift detection, AI guardrails and policy enforcement, human-in-the-loop workflows, and AI audit logging/decision traceability.

Third, AI should be implemented through shared technical implementation models. Shared AI platform capabilities can include approved model access, inference APIs, prompt gateways, experiment

tracking, model registries, AI observability and cost monitoring, safety guardrails, and review/override workflows. A practical operating model combines shared baseline controls, federated domain customization, and local exceptions only when justified.

Fourth, AI should be incorporated into DevOps/engineering enablement as MLOps and LLMOps capabilities, including pipeline automation, model/prompt versioning, release controls, AI test automation, drift monitoring, cost optimization, incident response runbooks, and change governance.

Finally, AI must be made visible in traceability matrices by adding AI-related technical columns (e.g., AI inference/model serving, AI guardrails, AI monitoring/drift, feature/AI data pipelines, human-in-the-loop review, MLOps/LLMOps pipelines) and AI-specific governance overlays (decision criticality, explainability requirement, model risk, oversight requirement, data sensitivity, monitoring status, fallback mode).

Practical Examples, Matrices, and Illustrative Figures

This section provides worked examples and visual artifacts to operationalize the capability-centric framework presented in the paper. The examples illustrate a telecom/connected-operations context and can be adapted to other domains.

Figure 2 provides a visual heatmap representation of an AI-extended business-to-technical traceability matrix.

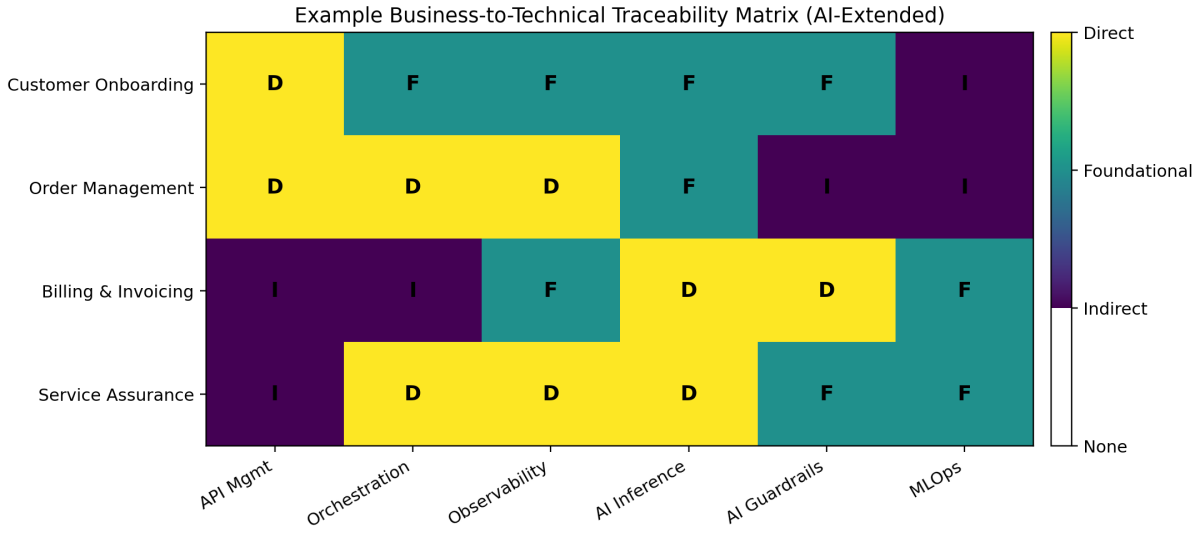

Figure 2. Example AI-extended traceability heatmap showing relative support relationships (Direct, Foundational, Indirect).

Example 1: Business Capability Map Slice (Telecom / Connected Operations)

Table 1 shows a simplified business capability hierarchy (L1-L3) with sample maturity values and AI enablement opportunities. The intent is to demonstrate how business capabilities can be structured and prioritized before selecting specific technical solutions.

L1 Domain	L2 Capability	L3 Capability	Current	Target	AI Opportunity
Customer	Customer Onboarding	Identity Verification and	2	4	Yes (assist + recommend)

		Account Activation			
Order-to-Activate	Order Management	Order Decomposition and Exception Handling	2	5	Yes (predict + prioritize)
Order-to-Activate	Provisioning & Activation	Service Provisioning and Activation Confirmation	2	4	Limited (optimize)
Revenue	Billing & Invoicing	Bill Generation and Invoice Delivery	4	4	Yes (anomaly detection)
Service Assurance	Incident / Trouble Management	Incident Triage and Escalation	3	4	Yes (triage + root cause suggestion)

Table 1. Example business capability hierarchy with maturity targets and AI opportunity indications.

Example 2: Technical Capability and Shared Implementation Model

Table 2 illustrates how technical capabilities can be classified by implementation model (shared, federated, or local), provider team, and standardization level. This helps operationalize platform decisions and reduce duplication.

Technical Capability	Implementation Model	Provider Team	Standardization	Primary Consumers
IAM / SSO / Federation	Shared	Security Platform	Mandatory	All channels and apps
API Management	Shared	Integration Platform	Mandatory for external and core APIs	Digital channels, CRM, OSS/BSS
Workflow / Orchestration	Federated	Platform + Domain Teams	Preferred with reference patterns	Fulfillment, care, operations
Observability (logs, metrics, tracing)	Shared	SRE / Platform	Mandatory baseline	All services and products
AI Inference / Model Serving	Shared	AI Platform	Preferred with approved model access	Service assurance, billing, care
AI Guardrails / Policy Enforcement	Shared	AI Platform + Security	Mandatory for customer-impacting use cases	All AI-enabled domains
Feature / AI Data Pipelines	Federated	Data Platform + Domains	Preferred	Analytics and AI use cases
LLMOps / MLOps Pipelines	Shared	Platform Engineering + AI Platform	Preferred templates	AI product and domain teams

Table 2. Example technical capability mapping to shared implementation models.

Table 3 shows a balanced **shared vs. federated implementation model**. **Shared capabilities** (e.g., IAM, API management, observability, AI guardrails, MLOps pipelines) are centrally provided to ensure consistency, security, and reuse, while **federated capabilities** (e.g., orchestration and AI data pipelines) allow domain teams to adapt implementations using common patterns. Overall,

it demonstrates how enterprises can scale both core platforms and AI capabilities with strong governance and domain flexibility.

Technical Capability	Implementation Model	Provider Team	Standardization	Primary Consumers
IAM / SSO / Federation	Shared	Security Platform	Mandatory	All channels and apps
API Management	Shared	Integration Platform	Mandatory for external and core APIs	Digital channels, CRM, OSS/BSS
Workflow / Orchestration	Federated	Platform + Domain Teams	Preferred with reference patterns	Fulfillment, care, operations
Observability (logs, metrics, tracing)	Shared	SRE / Platform	Mandatory baseline	All services and products
AI Inference / Model Serving	Shared	AI Platform	Preferred with approved model access	Service assurance, billing, care
AI Guardrails / Policy Enforcement	Shared	AI Platform + Security	Mandatory for customer-impacting use cases	All AI-enabled domains
Feature / AI Data Pipelines	Federated	Data Platform + Domains	Preferred	Analytics and AI use cases
LLMOps / MLOps Pipelines	Shared	Platform Engineering + AI Platform	Preferred templates	AI product and domain teams

Table 3. Example Shared Technical Implementation Model (Landscape View)

Example 3: Business-to-Technical Traceability Matrix (AI-Extended)

Table 4 provides a sample traceability matrix linking business capabilities to core technical and AI technical capabilities. Cell semantics use D = Direct, F = Foundational, and I = Indirect support.

Business Capability	API Mgmt	Orch.	Obs.	AI Infer.	AI Guard.	MLOps	Human Review
Customer Onboarding	D	F	F	F	F	I	F
Order Management	D	D	D	F	I	I	I
Billing & Invoicing	I	I	F	D	D	F	F
Service Assurance	I	D	D	D	F	F	D

Table 4. Example traceability matrix linking business capabilities to technical and AI capabilities.

AI Governance Overlay for Traceability Matrices

Table 5 adds AI-specific metadata fields that can be applied to traceability entries to improve governance, auditability, and operational control.

Overlay Field	Example Values	Purpose
Decision Criticality	Advisory; Human-approved; Automated	Defines approval and oversight expectations
Explainability Requirement	Low; Medium; High	Defines documentation and model transparency needs

Model Risk Level	Low; Medium; High	Prioritizes validation, monitoring, and review rigor
Data Sensitivity	Internal; Confidential; Regulated	Controls handling and approved deployment patterns
Monitoring Status	Not implemented; Partial; Production baseline	Tracks operational readiness
Fallback Mode	Manual; Rule-based; Alternate model	Supports resilience and continuity planning

Table 5. AI-specific governance overlays for traceability matrix relationships.

Worked Example Interpretation

In the example matrices, Order Management and Service Assurance show strong dependencies on shared platform capabilities, including API Management, orchestration, observability, and AI-enabled services. This indicates that investments in shared integration, telemetry, and AI platform guardrails can improve multiple business outcomes at once, including reduced order fallout, better incident triage, and improved customer transparency. The matrices also highlight where AI capabilities require governance overlays (e.g., human review and explainability) for higher-risk operational decisions.

Discussion

The integrated use of Business Capability Maps, Technical Capability Maps, shared technical implementation models, and traceability matrices creates a coherent architecture method for business-technology alignment. The framework separates concerns while preserving traceability from business strategy to technical execution and governance.

This separation is especially important for AI adoption because AI spans business design, platform engineering, operational controls, and risk management. Modeling AI across all four artifacts makes dependencies explicit and reduces the risk of fragmented, tool-centric adoption.

Conclusion

Capability-centric enterprise architecture offers a practical foundation for aligning business goals, technical enablement, and implementation governance. Business and technical capability maps define stable 'what' views, shared implementation models define reusable delivery patterns, and traceability matrices define the relationship logic needed for decision-making and auditability.

Adopting AI across these same artifacts allows organizations to scale AI in a business-aligned, governable, and operationally resilient manner rather than as disconnected experiments. Future work may focus on empirical validation of AI-extended traceability models and maturity frameworks for shared AI platform capabilities.

References

1. Amazon Web Services. (2023, September 20). AWS Well-Architected Framework DevOps guidance. AWS Documentation. https://docs.aws.amazon.com/wellarchitected/latest/devops-guidance/devops-guidance.html
2. Amazon Web Services. (n.d.). [OA.STD.1] Organize teams into distinct topology types to optimize the value stream. AWS Documentation. https://docs.aws.amazon.com/wellarchitected/latest/devops-guidance/oa.std.1-organize-teams-into-distinct-topology-types-to-optimize-the-value-stream.html
3. Ardoq. (2023, March 7). Business capabilities vs. technical capabilities. https://www.ardoq.com/blog/technical-capabilities
4. Google Cloud. (n.d.). What is platform engineering? https://cloud.google.com/solutions/platform-engineering
5. Google Cloud. (n.d.). What is an internal developer platform? https://cloud.google.com/discover/what-is-an-internal-developer-platform
6. IBM. (n.d.). DOORS - Linking and traceability. IBM Documentation. https://www.ibm.com/docs/en/engineering-lifecycle-management-suite/doors/9.7.1?topic=requirements-links-traceability
7. Microsoft. (n.d.). What is an Azure landing zone? Microsoft Learn. https://learn.microsoft.com/en-us/azure/cloud-adoption-framework/ready/landing-zone/
8. SAP LeanIX. (n.d.). What is business capability: Definition & mapping. https://www.leanix.net/en/wiki/ea/business-capability
9. SAP LeanIX. (n.d.). How to create a business capability map? https://www.leanix.net/en/wiki/ea/how-to-create-a-business-capability-map
10. Sparx Systems. (n.d.). Business capability analysis. Enterprise Architect User Guide. https://sparxsystems.com/enterprise_architect_user_guide/17.1/guide_books/tech_business_capability_analysis.html
11. The Open Group. (n.d.). The TOGAF Standard. https://www.opengroup.org/togaf