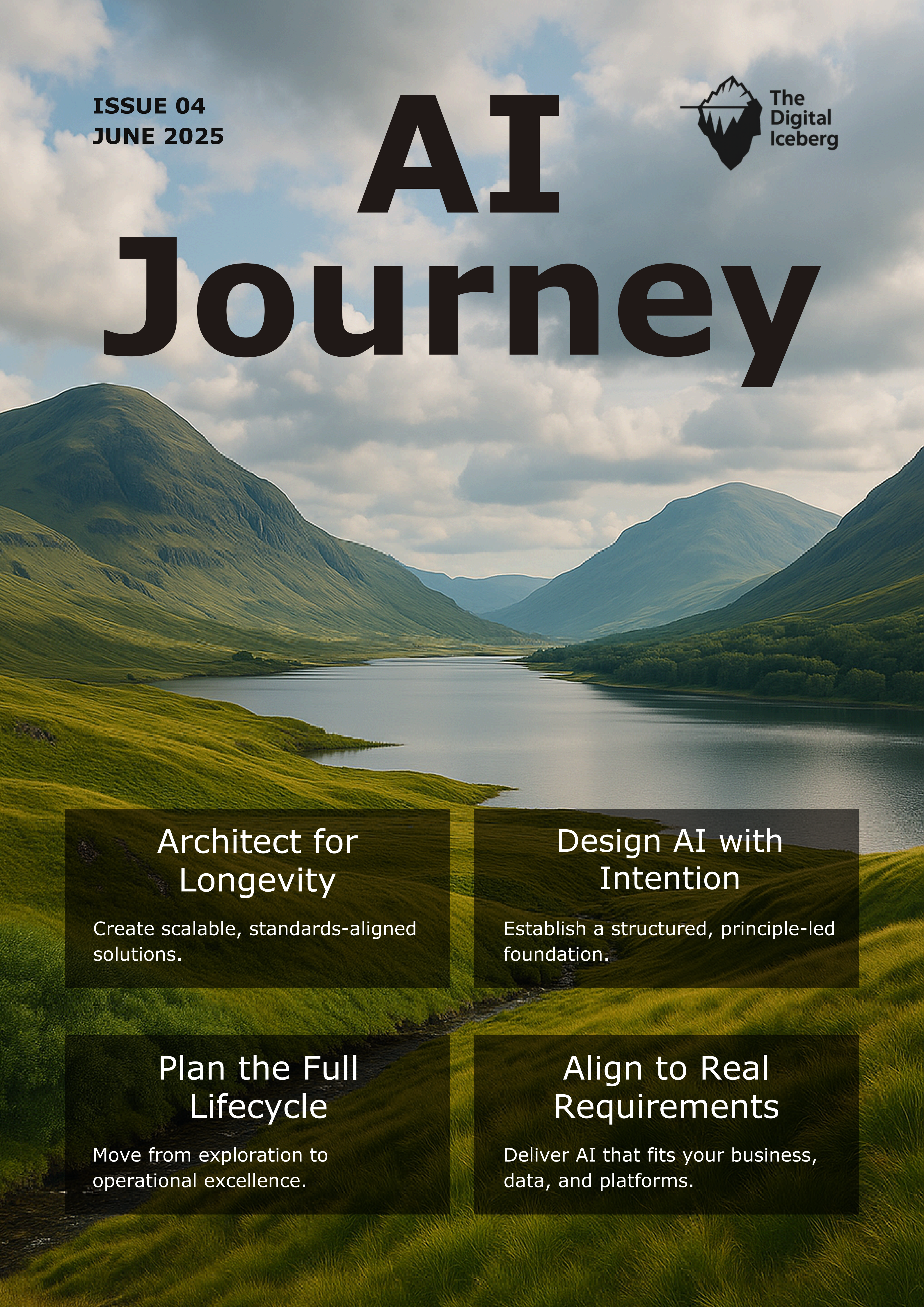

ISSUE 04
JUNE 2025

The Digital Iceberg

AI Journey

Architect for Longevity

Create scalable, standards-aligned solutions.

Design AI with Intention

Establish a structured, principle-led foundation.

Plan the Full Lifecycle

Move from exploration to operational excellence.

Align to Real Requirements

Deliver AI that fits your business, data, and platforms.

Creating the Optimal AI Solution Architecture

Introduction

In the age of enterprise Artificial Intelligence (AI), organisations are seeking structured approaches to design, build, and govern AI solutions at scale. The Open Group Architecture Framework (TOGAF) – specifically its Architecture Development Method (ADM) – provides a step-by-step methodology to align technology projects with business strategy. This e-book explores how each phase of the TOGAF ADM can be applied to AI solution architecture. Each chapter (phase) will explain the phase's purpose and **how it guides the design, implementation, and governance of AI systems** in an enterprise context. We will cover strategic goals of each phase, key artefacts produced, critical decisions, examples of AI projects, and pitfalls to avoid. The guidance is vendor-neutral, with occasional references to common platforms (e.g. Microsoft Azure, AWS, Google Cloud, ServiceNow) as examples of implementation options. Whether you are an enterprise architect, AI engineer, or product manager, this detailed walkthrough will help you leverage TOGAF for successful AI-driven transformations.

Preliminary Phase: Establishing the AI Architecture Foundation

The **Preliminary Phase** sets the groundwork for all architecture work. In an AI solution context, this phase establishes how the organisation will approach AI architecture and ensures the effort aligns with business strategy and governance standards. It is about preparing the enterprise for AI projects by defining the *scope, principles, and governance* needed to safely and effectively deploy AI solutions.

Strategic Goals

This phase focuses on creating an environment where AI initiatives can thrive under proper oversight. Key goals include:

- **Define AI Strategy and Alignment:** Ensure the AI initiative is guided by an overall strategy linked to business objectives (e.g. improving customer experience or operational efficiency). This involves articulating why the organisation is investing in AI and how success will be measured in business terms.

- **Establish Governance and Principles:** Set up an AI governance framework and high-level principles (e.g. ethics, data privacy, transparency) to guide all subsequent architecture work. For example, a principle might be *"AI decisions must be explainable"*, ensuring ethical AI use from the start. Governance may include forming an **AI Governance Board** or Centre of Excellence to oversee AI projects.
- **Scope and Framework Definition:** Determine the scope of architecture effort (which part of the enterprise or which systems the AI solution will impact) and tailor the TOGAF framework as needed. This may involve integrating industry-specific standards or agile methodologies relevant to AI. It's also important to identify how existing enterprise architecture processes (e.g. architecture repository, enterprise continuum) will include AI components.

Key Artefacts & Deliverables

The Preliminary Phase produces foundational documents and structures:

- **Architecture Charter:** A document authorising the AI architecture work, outlining scope, stakeholders, and objectives.
- **Architecture Principles (including AI Principles):** A set of guiding principles covering technology and AI ethics (e.g. fairness, accountability), which will steer design decisions.
- **AI Governance Model:** Definition of roles, such as an AI governance board or steering committee, and processes for oversight. This might detail how compliance, risk, and value of AI will be monitored.
- **Tailored Framework & Method:** Adjustments to the standard ADM process for AI projects. For example, deciding to incorporate agile iterations or specific modelling techniques for AI (such as CRISP-DM for data science) alongside TOGAF.

Decision Points

Several early decisions set the direction for the AI architecture initiative:

- *Selecting Stakeholders:* Identifying who needs to be involved from business units, IT, data science teams, compliance, etc. (e.g. business sponsors, lead AI engineers, data privacy officers). A pitfall is excluding key stakeholders, which can cause misalignment later.
- *Governance Scope:* Deciding how strict or adaptive the governance should be. For AI, balancing innovation with control is crucial. Organisations must decide on processes for ethical review (e.g. will there be an AI ethics review step for new use-cases?).
- *Tooling and Repository:* Choosing tools for architecture modelling and AI development. For example, deciding whether to use an architecture repository (like **Enterprise Continuum** in TOGAF) to store AI solution patterns and models. Also, considering if new AI-supportive tools are needed (such as model registries or data catalogues).

Example

Imagine a financial enterprise embarking on AI for fraud detection. In the Preliminary Phase, the company establishes an AI Strategy that declares "reduce fraudulent transactions by 40% through machine learning." They set principles like *"AI use must comply with data regulations and produce auditable decisions."* An AI Governance Board is appointed, including the Chief Data Officer, Head of Enterprise Architecture, and Compliance Lead, to review AI project proposals. They also decide to incorporate responsible AI guidelines into their architecture method (e.g. mandating bias testing for models as part of the methodology).

Pitfalls to Avoid

- **Skipping Stakeholder Buy-In:** Not securing executive sponsorship and end-user stakeholder input can doom the AI initiative from the start. Ensure business leaders and users understand and support the goals. Early buy-in prevents later resistance.
- **Undefined Success Criteria:** If strategic objectives for AI are too vague (e.g. "we want to use AI because it's trendy"), the project can drift. Define clear, measurable goals in this phase.
- **Inadequate Principles:** Omitting AI-specific principles (ethics, data management) may lead to compliance or reputational issues later. For example, failing to establish a principle on data privacy could result in using

sensitive data inappropriately. Always include guidelines for security and ethics up front.

- **Overlooking Framework Tailoring:** Using TOGAF "out of the box" without adapting to the fast-paced, experimental nature of AI can make the process too rigid. It's a mistake to ignore agile practices – consider integrating agile or DevOps approaches in the ADM if needed, so that innovation is not stifled by bureaucracy.

Phase A: Architecture Vision – Setting the AI Solution Direction

Phase A establishes the **Architecture Vision** – a high-level view of the target solution and its value proposition – and secures initial agreement from stakeholders. In an AI project, this means crystallising what the AI solution will do for the business and envisioning how it will work at a conceptual level. This phase bridges the gap between strategic intent and actionable plan by answering *"What will the AI solution achieve, and what will it look like broadly?"*

Strategic Goals

For an AI solution, the Architecture Vision phase aims to:

- **Define Business Outcomes for AI:** Identify the specific business problems or opportunities the AI will address (e.g. reducing support ticket resolution time, improving supply chain forecasting). The vision must tie directly to business value – for example, "an AI-powered customer service agent to handle 80% of inquiries, improving customer satisfaction by 20%."
- **Scope the Solution and Objectives:** Determine the scope of the AI solution (which domains, products, or processes it will cover) and set clear objectives and success metrics. This often involves defining Key Performance Indicators (KPIs) such as accuracy of AI predictions, time saved, cost reduction, or revenue uplift. For example, *AI success metrics* might include *"predictive model to cut downtime by 30%"* or *"chatbot to achieve 90% first-contact resolution rate"*.
- **Articulate a Vision and High-Level Concept:** Create a compelling vision statement and a high-level model of the solution. In TOGAF, this can be captured in an **Architecture Vision document** which includes a narrative of the future state and possibly a simple diagram (though we will stick to text).

For AI, the vision might describe how AI, data, and users interact: e.g. *"Our AI recommendation engine will proactively suggest optimal products to customers in real time, seamlessly integrated into the e-commerce platform."*

Key Artefacts & Deliverables

- **Architecture Vision Document:** Summarises the above – including business objectives, stakeholder needs, the envisioned solution, scope boundaries, and high-level requirements. It may also outline initial risk considerations (e.g. regulatory or ethical risks of the AI) and the projected impact on business outcomes.
- **High-Level Requirements:** A preliminary set of requirements (business, data, application, technology requirements) that the AI solution must meet. For example, a business requirement could be *"system must handle 1000 queries per minute"*, or a regulatory requirement *"AI decisions must be explainable to auditors."*
- **Stakeholder Analysis & Concerns:** Identification of key stakeholders (executives, end-users, IT, data scientists, etc.) and their concerns and expectations. TOGAF often uses a Stakeholder Map or matrix. In AI projects, stakeholders can include compliance officers concerned with ethics and regulators concerned with bias. Capturing their concerns early ensures the vision addresses them (e.g. including a plan for model transparency to satisfy regulators).

Decision Points

- *Selecting Use Cases:* Often there are many possible AI use cases. In this phase, the team decides which use case(s) to pursue. Criteria include feasibility (do we have the data?), business value, and alignment with strategy. Choosing the right initial use case is critical – e.g. pick a problem with high impact and a good chance of success, rather than a trendy but ill-defined application.
- *Risk Versus Reward:* Evaluate risks like data privacy, model bias, or technical complexity against the expected benefits. Decisions here include whether to proceed given these risks, and what risk mitigations to plan (for example, if data privacy is a risk, maybe anonymise data from the start).
- *Buy vs. Build at High-Level:* Decide if the envisioned solution will use off-the-shelf AI services or custom models. At the vision phase, this is preliminary –

for example, deciding *"We will likely use a cloud AI service (Azure Cognitive Services or AWS AI) for basic image recognition, rather than build our own model from scratch."* Such decisions set the tone for later phases and influence the team composition and budget.

Example

Consider an enterprise envisioning an AI-based IT support assistant. The architecture vision might be: "**AI Service Desk Assistant** that automatically handles common IT support requests (password resets, FAQs) via chat, reducing resolution time by 50%." The team identifies stakeholders: IT support managers (want reduced workload), employees (want quick solutions), the CIO (wants cost efficiency), and compliance (concerned about data security). They define high-level requirements: the AI assistant must integrate with the existing IT service platform (e.g. ServiceNow) and conform to IT security policies. A risk identified is that the chatbot could give incorrect advice, so a mitigation noted in the vision is to always allow human escalation for complex issues. The success metric is set: e.g. *first-level resolution rate of at least 70%* within six months of deployment.

Pitfalls to Avoid

- **Technophilia without Purpose:** A common mistake is setting an AI project in motion without a clear business goal, simply because the technology is exciting. Avoid vague visions like "We will use AI across the company" – instead tie it to specific outcomes (e.g. *"reduce churn by predicting customer behaviour"*). Every AI initiative should be value-driven, not tech-driven.
- **Overambitious Scope:** Trying to "boil the ocean" by including too many features or a very broad scope in the initial vision can set the project up for failure. It's better to have a focused, achievable vision (possibly a pilot or MVP) than an unrealistic grand plan. For example, aiming to implement an AI that solves *all* enterprise problems in one go will likely stall; focus on one domain first.
- **Ignoring Stakeholder Concerns:** If the vision doesn't address what stakeholders care about, it will lack support. For example, if the compliance team's need for auditing AI decisions isn't reflected, they may later block deployment. Engage stakeholders early and reflect their concerns (such as including an explainability component for regulators).
- **Neglecting High-Level Requirements:** Skipping the definition of key requirements (like performance, security needs) at this phase may cause blind

spots. For example, not noting that "response time must be under 2 seconds" could lead to building an AI system that is too slow to be usable. Early recognition of such requirements guides better design downstream.

Phase B: Business Architecture – Aligning AI with Business Capabilities

In **Phase B (Business Architecture)**, the focus is on how the AI solution fits into and transforms the business. This phase documents the current and target *business processes, organisational structures, and capabilities* relevant to the AI initiative. For an AI solution to succeed, it must be woven into the business fabric – this phase ensures the technology design is guided by business needs and that the organisation is prepared to leverage it.

Strategic Goals

Key objectives in the Business Architecture phase for an AI project include:

- **Map AI to Business Capabilities:** Identify which business functions or capabilities will be enhanced or newly enabled by AI. For example, an AI might strengthen the *customer service capability* by enabling 24/7 automated support or create a new capability in *predictive maintenance* for manufacturing equipment. Mapping this out ensures the AI isn't an isolated tech project but directly supports business competencies.
- **Model Business Processes with AI Integration:** Document how workflows will change when AI is introduced. This could involve using process modelling to show "before and after" scenarios. For example, *with* an AI-based fraud detection, the process might be "Transaction → AI fraud check → flag or approve → analyst review if flagged," whereas before, analysts manually reviewed random samples. By modelling processes, architects and stakeholders can see exactly where AI intervenes, who interacts with it, and what steps are eliminated or modified.
- **Assess Organisational Impact:** Determine changes in organisational structure or roles. AI solutions often shift how people work – some tasks are automated, and employees may need to focus on higher-value activities or manage the AI systems. For example, if an AI system takes over initial customer inquiries, the role of customer service reps may evolve to handle only complex cases. Phase B should outline these changes and what new roles

or training might be required (e.g. needing **data curators** or **ML model managers** in the team).

Key Artefacts & Deliverables

- **Baseline and Target Business Process Models:** Diagrams or descriptions of current vs. future workflows that highlight changes due to the AI solution. Using notations like BPMN (Business Process Model and Notation) is common. E.g., a target process model showing an *AI engine* performing a step previously done by a person.
- **Business Capability Assessment:** A list or model of business capabilities impacted. This might be presented as a *business capability map* with certain capabilities marked as being upgraded by AI. Deliverables can include an **AI-enhanced Capability Model** that describes how capabilities (like "Customer Support" or "Analytics") are improved by AI (e.g. faster response, better predictions).
- **Gap Analysis (Business):** A document or matrix comparing current business performance or process metrics to expected post-AI metrics, identifying what needs to change to achieve the vision. For example, identifying that currently it takes 5 steps and 3 days to fulfil a request, but with AI it should take 1 step and immediate response – the gaps might be lack of automation and needed integration with the CRM system.
- **Change Impact Assessment:** Qualitative reports on how roles, skills, or organisational units will be affected. This might note, for example, that the Marketing department will need training on using the new AI-driven customer insights platform, or that a new Data Science team is needed to maintain the AI models.

Decision Points

- *Process Selection:* Decide which business processes to target for AI improvement. Often there are many processes that could benefit; architects must prioritise those with high impact and feasibility. This may involve stakeholder input to pick, say, the top 5 customer pain points or internal inefficiencies where AI could help.
- *Organisational Model:* Determine if new organisational structures are needed. For example, should there be a centralised AI team serving the whole enterprise, or will each business unit handle their own AI (federated model)?

This decision affects how governance and knowledge are shared. Many enterprises initially create a centralised *AI Centre of Excellence* to support various departments.

- *Value Stream Integration:* Ensure AI initiatives integrate with existing value streams or customer journeys. A decision here is how to embed the AI into the day-to-day operations. E.g., if implementing an AI in e-commerce personalisation, decide how it feeds into the customer's buying journey and existing CRM processes. It might involve deciding not to deploy AI in isolation but as part of an end-to-end process that includes human steps.
- *Policy and Regulation Alignment:* At the business level, decide how to handle policies. For example, HR policies might need updating if an AI is used in hiring (to prevent bias), or customer-facing policies might be needed to disclose AI usage. Ensuring legal/regulatory compliance is addressed at the business design stage (e.g. require human oversight in decisions that impact customers' rights).

Example

An insurance company planning to use AI for claims processing will map out its claims process. Current state: customers file claims, human agents review for fraud and assess damages, etc., which may take weeks. Target state: customers file claims via an app, an AI model automatically detects possible fraud (flagging suspicious ones), and estimates damages from photos using computer vision, with human adjusters only handling exceptions. The Business Architecture defines this target workflow, identifies that the *"Claims Handling" capability* is enhanced by AI (faster processing, lower cost). It also highlights organisational impacts: some claim adjusters will be re-skilled to work on AI oversight and customer care for complex cases. The company might decide to create a small *AI Operations team* within the Claims department to maintain the models and manage exceptions. Business metrics like claim processing time and customer satisfaction are identified for improvement. The gap analysis shows current average processing is 10 days; target is 2 days with AI.

Pitfalls to Avoid

- **Ignoring Business Value and Adoption:** A classic mistake is designing an AI solution without fully considering how the business will use it. If the AI doesn't fit into business processes or if users don't trust it, it will languish. Avoid building "solutions looking for a problem." Always start from business needs and ensure the solution will actually be adopted by the business units (e.g. involve end-users in design to ensure usability).

- **Underestimating Change Management:** Automating a process with AI often changes jobs and workflows. If you ignore the human side – training, new job definitions, potential job losses or shifts – the project can face resistance or fail outright. Make sure to plan for training staff to work alongside AI (augmented intelligence) and possibly reallocate roles. Not addressing this in Business Architecture is a pitfall.

- **Overlooking Process Exceptions:** AI may handle standard cases, but there will be exceptions and edge cases. If the business process design doesn't account for how exceptions are routed (e.g. "what if the AI is not confident or the data is missing?"), you could have process breakdowns. Always design fallback paths (like human review steps) for when AI cannot confidently complete a task.

- **Misalignment with Business Policy:** Failing to align the AI solution with business policies or industry regulations at this stage can lead to trouble. For example, if an AI is making credit decisions, the Business Architecture must include required compliance checkpoints (due to fair lending laws, etc.). Skipping this can result in non-compliant processes that later require costly redesign.

Phase C: Information Systems Architecture - Designing Data and Application Architecture for AI

Phase C is concerned with the **Information Systems Architectures**, which in TOGAF includes both **Data Architecture** and **Application Architecture**. In an AI solution, these are critical: AI is highly data-driven and often involves new application components like machine learning models, APIs, and integration points. This phase defines how data will be managed and how the software components will be structured to fulfil the vision.

Data Architecture Considerations for AI

Strategic Goals (Data)

The data architecture for AI ensures the AI solution has the *right data, at the right quality, at the right time*. Goals include:

- **Define Data Requirements & Sources:** Identify what data the AI needs (e.g. historical customer transactions, sensor readings, text from tickets) and where it comes from. In AI, data can be structured (databases), unstructured (documents, images), or streaming. This step catalogues all relevant data sources and data types.
- **Ensure Data Quality and Governance:** AI outcomes are only as good as the data feeding it. Establish data quality metrics and governance frameworks so that data is clean, consistent, and compliant. For example, if building a predictive model, ensure there's a process to handle missing values, outliers, and that sensitive personal data is handled according to laws (GDPR, etc.). An **AI Data Governance Framework** might be created, extending existing data governance to cover model training data, feature stores, and so on.
- **Design Data Flow & Storage Architecture:** Determine how data flows into the AI solution and out to users or systems. This includes choosing storage and processing platforms. Modern AI solutions often leverage specialised data stores: e.g. data lakes for large raw data, data warehouses for aggregated data, or **vector databases** to store AI embeddings for semantic search. The architecture must also support real-time data needs if applicable (for example, streaming data pipelines for live predictions).

Key Artefacts (Data)

- **Data Model & Data Dictionary:** A model of key data entities and their relationships (for AI, this could include feature schemas, training data sets, etc.). Also, documentation of data definitions – e.g., what exactly is "customer churn rate" and how is it calculated.
- **Data Flow Diagrams / Pipeline Designs:** Visual or textual description of how data moves from sources to the AI model and to outputs. For example, a pipeline: CRM Database → ETL process → Feature Store → AI Model → Output to Dashboard. This shows integration points and transformation steps.
- **Data Architecture Diagram:** Illustrating components like databases, data lakes, message queues, streaming processors, and how they connect. In an AI context, this might include components like a *data lake on AWS S3, a Spark*

processing job for feature engineering, a vector database (e.g. Pinecone) for similarity search, etc. These choices should remain vendor-neutral in design, focusing on functionality (with examples of technologies in parentheses).

- **Data Governance Policies:** Documents specifying how data will be governed in the AI solution – covering data ownership, data refresh cycles, quality checkpoints, privacy compliance, and retention policies. For example, a policy might state that *"customer data used for AI training must be anonymised and archived after one year."*

Decision Points (Data)

- *Centralised vs Federated Data:* Decide if the AI solution will pull data into a central repository (like a consolidated data lake) or access data from distributed sources on the fly. Centralising can simplify model training but may raise security concerns; federated approaches (where the model comes to the data) might be needed if data cannot move from certain locations for legal reasons.
- *Storage and Processing Tech:* Choose technologies for storing and processing data. Options include cloud data warehouses (e.g. Snowflake, BigQuery), NoSQL databases, distributed file systems, or specialised stores. If the AI requires knowledge representation, one might consider graph databases or knowledge graphs. The decision must consider scale (how big is the data), speed (real-time vs batch), and existing enterprise standards. For example, if the enterprise already has an Azure data lake, leveraging that might be wise for compatibility.
- *Data Integration Strategy:* Determine how to get data from existing enterprise systems (ERP, CRM, IoT sensors) into the AI pipeline. Will you use batch ETL, real-time streaming via Kafka, APIs, or a combination? Also, decide on data frequency – for a fraud detection AI, maybe real-time streaming is necessary, whereas for a weekly sales forecast model, daily batch updates might suffice.
- *Metadata and Lineage:* Decide how to track data lineage and metadata for the AI datasets. This is crucial for trust and compliance. The team might choose to use a metadata catalogue or lineage tool to record where training data came from and how it has been processed. This aids in debugging the AI and explaining its outputs (traceability from output back to source data).

Example (Data)

A retailer building an AI demand forecasting tool identifies data needs: historical sales per store, promotional calendars, economic indicators, weather data, etc. In Data Architecture, they design a pipeline where data from the sales system (on SAP) and marketing system are regularly extracted to a cloud data lake. They use a data processing engine to clean and combine this data with external weather data via API. They decide to use a **feature store** (a centralised repository for machine learning features) to serve the prepared data to models, ensuring consistency between training and inference. They also implement data governance rules: for example, ensure that any customer data used (maybe for personalised forecasts) is anonymised. The data architecture diagram might show Azure Data Lake Storage for raw data, an Azure Data Factory or Databricks pipeline for ETL, and an Azure Feature Store, but they note these choices as examples, keeping the design abstract (it could similarly be achieved on AWS or GCP). A key decision was to centralise data for model training to one cloud platform for ease of management, while using APIs to fetch up-to-minute data for real-time predictions in production.

Pitfalls to Avoid (Data)

- **Garbage In, Garbage Out:** If you do not invest in data quality and just feed the AI whatever data is available, the AI's results will be poor or misleading. Avoid neglecting data cleansing and verification. For example, a model trained on outdated or biased data can perpetuate errors. Always allocate effort for data preparation – a known fact is data preparation can take 80% of an AI project's time.
- **Ignoring Data Governance and Security:** AI projects often move fast, but ignoring governance (who can access the data and outputs, how to ensure privacy) can lead to breaches or compliance violations. Avoid copying sensitive data to unmanaged environments. Always apply enterprise data security standards (encryption, access controls) to your AI data stores.
- **Overengineering the Data Pipeline:** Conversely, a pitfall is to design an overly complex data architecture for a simple pilot. For example, implementing a full streaming pipeline with micro-batches and seven databases when a simpler solution would do. This can delay the project. It's important to design for the target scale but also consider a **phased approach** – perhaps start with a simpler data setup for a pilot, then evolve it (we'll see this in Phase E/Phase F).
- **Not Planning for Data Evolution:** AI models often need continuous learning and new data. If your data architecture is static (doesn't account for new data

sources or changes in data schema), the solution will become stale. Plan for how new data will be onboarded and how data drift will be detected. For example, if the distribution of incoming data changes (say customer behaviour shifts), there should be a mechanism to flag this so the model can be retrained or adjusted.

Application Architecture Considerations for AI

Strategic Goals (Application)

This aspect of Phase C defines the structure of the software components (applications and services) that will implement the AI capabilities. Goals include:

- **Define AI Solution Components:** Break down the AI solution into logical components or services. Common components for an AI system might be: a *data ingestion service*, a *model training pipeline*, a *model inference service (API)*, a *user interface* or application that consumes the AI, and *integration components* to connect with other systems. Defining these ensures clarity on what needs to be built or configured.
- **Specify Interactions and Integration Points:** Outline how the AI components interact with each other and with existing enterprise applications. This might involve identifying APIs, events, or message flows. For example, how does the customer service chatbot component connect to the backend ticketing system (like ServiceNow or Salesforce)? The architecture should position the AI within the overall application landscape of the enterprise (possibly using application portfolio maps or integration context diagrams).
- **Ensure Alignment with Enterprise Application Standards:** The AI application architecture must comply with general architecture principles: e.g. if the company strategy favours microservices and containerisation, the AI components should follow suit. If the enterprise uses certain middleware or API gateways, the AI services should integrate via those rather than creating isolated silos. Essentially, treat the AI solution as part of the broader application ecosystem, not a standalone skunkworks project.

Key Artefacts (Application)

- **Application Component Model:** A breakdown of new and existing application components involved. This can be presented as a block diagram or catalogue. For an AI solution, components may include things like *"Model Training*

Service," "Model Registry," "Inference API," "Web Dashboard," "Mobile App," "Integration Adapter (to ERP)", etc.

- **Interface Catalogue / Integration Design:** A list of interfaces (APIs, data feeds, message topics) the AI solution will require. For example, *API A:* to get customer data from CRM; *API B:* to push predictions into the workflow system; *Message Queue:* to receive real-time events from IoT sensors, etc. Defining these in Phase C ensures that by the time of implementation, integration requirements are known and no system is left out.
- **Use-Case or Sequence Diagrams:** To illustrate how the system behaves for key scenarios. For example, a sequence diagram might show the steps when a user asks the AI chatbot a question: User → Chatbot UI → AI NLP Service → Knowledge Base → Response returned → (if confidence low, escalate to human agent system). While not strictly required by TOGAF, such diagrams can help communicate how AI and conventional systems will interact.
- **Application Architecture Document:** A narrative description of how the solution's software is structured, including decisions on architecture patterns (e.g. event-driven, client-server, microservices) and any new platforms or frameworks being introduced (like a decision to use a particular AI framework or library, subject to vendor-neutral discussion). It may also list **application architecture principles** observed (like modularity, reuse of existing services, etc).

Decision Points (Application)

- *Build vs. Buy for Components:* Decide for each component whether to custom-build it or use a commercial/off-the-shelf solution. For example, do you build your own model serving API or use a managed service like AWS SageMaker endpoints or Azure ML endpoints? Do you develop a new UI for AI results or integrate into an existing portal (like showing AI insights in the company's existing dashboard tool)? These decisions affect cost, time, and integration complexity.
- *Integration Method:* Determine the best way to integrate AI outputs into business processes. Options include embedding AI into existing applications (like adding AI recommendations inside an ERP interface) versus creating a new standalone application. Often, seamless integration into the tools employees already use (e.g. AI insights appearing in ServiceNow for ITSM, or in a CRM for salespeople) can drive adoption. This decision influences user experience and technical complexity.

- *Architecture Pattern:* Choose an architecture style. Many modern AI solutions follow a microservices approach for flexibility (e.g. separate services for data ingestion, model serving, etc.). However, for a simpler use case, a monolithic application might be quicker initially. Also, decide on synchronous vs asynchronous processing: does the AI respond in real-time (requiring low-latency calls) or offline (batch predictions stored for later)? For example, a fraud detection might need to be synchronous within a transaction (very low latency), whereas a nightly predictive analytics can be batch.
- *Use of AI/ML Platforms:* Determine if the architecture will leverage an existing AI platform. Many enterprises adopt platforms like **Azure Machine Learning, AWS SageMaker, or Google Vertex AI** for model development and deployment. The decision here is whether to incorporate such platforms as components in the architecture (which can expedite development by using managed services). Alternatively, using open-source libraries (TensorFlow, PyTorch) in custom applications is an option. This decision should align with enterprise tech strategy and skills available.

Example (Application)

A healthcare provider is implementing an AI diagnostic assistant. The application architecture identifies components: (1) a front-end web application used by doctors to input patient symptoms or tests, (2) an inference engine which is an API that takes this input and returns diagnostic suggestions, (3) a machine learning model (or ensemble of models) behind the API, (4) a knowledge base application that the model or doctors can reference (medical guidelines), and (5) integration connectors to the electronic health record (EHR) system to pull patient history. The architects decide to use a microservices approach: the inference engine is a containerised service exposing a REST API; the knowledge base is an existing system accessed via API; integration to EHR is done through HL7/FHIR interfaces (standard healthcare APIs). They consider using a cloud ML service for hosting the model (for scalability and compliance), but also ensure the design could switch to another provider if needed (avoiding lock-in). An interface catalogue is created listing: *API to EHR (FHIR endpoints for retrieving patient data), API of AI service (for application to call), Logging service interface (to record recommendations), etc.* They also address how errors are handled: e.g. if the AI service is down or uncertain, the system should gracefully allow the doctor to proceed manually. The result is a clear blueprint of software components and their interactions.

Pitfalls to Avoid (Application)

- **Isolated AI Silo:** A major pitfall is building the AI solution as a standalone application that doesn't integrate well with existing systems or workflows. This can happen if data scientists work separately from enterprise architects. The result might be a great model that no one uses because it's not embedded in the tools where decisions are made. Always integrate AI outputs into the business applications or process flow (e.g. have that recommendation appear in the sales system where the sales rep works).
- **Overcomplicating the System:** Over-engineering can strike here too. Using the latest, trendiest architectural patterns (like a bunch of microservices or excessive use of serverless functions) can introduce complexity without clear benefit. The architecture should be as simple as possible while meeting requirements. For example, don't split into many services if the team and problem could be served by one or two modules. Excessive complexity can increase points of failure and maintenance burden.
- **Neglecting Non-Functional Requirements:** It's easy to focus on the AI's functional ability (e.g. make predictions) but forget non-functional needs. Ensure the application architecture addresses scalability (can it handle more load if AI usage grows?), performance (response times, throughput), reliability (failover strategies if the model service crashes), and security (how is the data and model access secured?). Missing these can derail an AI system in production – e.g. an AI service that times out frequently or cannot handle concurrency will frustrate users.
- **Not Reusing Existing Capabilities:** Sometimes teams build everything from scratch, but often the enterprise already has components to reuse – e.g., an existing data integration platform, message bus, or identity and access management system. Ignoring these and creating redundant functionality is inefficient and may cause conflicts. Always check the enterprise architecture repository for reusable assets (e.g. a common API gateway or UI component library) to incorporate rather than reinvent. This also ensures consistency (like using the company's single sign-on for the AI app's user access, rather than a separate login).

Phase D: Technology Architecture – Architecting the AI Infrastructure and Platforms

Phase D covers the **Technology Architecture**, defining the underlying infrastructure and technology stack needed to support the AI solution. This includes computing hardware, networks, cloud services, middleware, and any technology platforms. In the context of AI, Technology Architecture must address the significant computational needs of AI models, the environments for development and deployment, and ensure alignment with enterprise technology standards (like cloud strategy, security frameworks, etc.).

Strategic Goals

- **Select Appropriate Compute Infrastructure:** AI solutions, especially those involving machine learning or deep learning, may require powerful compute resources (GPUs, TPUs, etc.). A goal is to determine what infrastructure is needed for both development (training models) and production (inference serving). For example, during development you might use scalable cloud instances with GPUs to train a model on big data, while in production you might need an auto-scaling cluster to handle varying load of inference requests.
- **Choose Technology Platforms and Services:** Identify and decide on the technology platforms that will realise the application components. This includes cloud platforms (AWS, Azure, GCP, or on-premise stack), container orchestration (like Kubernetes or serverless functions), databases (SQL/NoSQL for application state, specialised stores for AI like vector DBs), and AI/ML services (managed machine learning services, AI APIs). The aim is to leverage platforms that provide reliability and scalability, while staying vendor-neutral in design (i.e. ensuring portability if needed).
- **Ensure Infrastructure Meets Scalability, Security, and Compliance:** The technology design must support the scaling expectations (e.g. can handle more data or users), meet security requirements (enterprise security architecture compliance), and any regulatory or compliance needs (data residency, encryption standards, etc.). For example, if dealing with sensitive data, maybe the tech arch requires a private cloud or on-premises deployment for that component. Or if using cloud, ensure proper Identity and Access Management (IAM), network isolation, and encryption are planned.

Key Artefacts & Deliverables

- **Technology Architecture Diagram:** A high-level diagram of the infrastructure layout. This could show, for example, a cloud architecture with regions, VPCs/VNets, subnets, and key services (databases, compute clusters, etc.), or an on-premises architecture with physical servers or virtualisation. The diagram ties the application components (from Phase C) to the underlying tech: e.g. "Inference Service runs on Kubernetes cluster; Data resides in cloud storage X; Model training uses GPU cluster Y." It also indicates connections (network flows, APIs over the network).

- **Technology Standards Profile:** A document or list aligning the project with relevant technology standards or guidelines. For example, referencing that the design uses the company's **Technology Reference Model** with extensions for AI – including approved technologies like certain AI frameworks (TensorFlow, PyTorch), containers, monitoring tools, etc. It might reference specific industry standards too (like PCI-DSS if credit card data is involved, or ITIL processes if relevant to operations).
- **Environment and Deployment Model:** Description of how development, testing, and production environments will be structured technologically. For AI, this might include a sandbox environment for data scientists, a separate staging area to test the AI in an isolated way, and the production setup. For example, deliverables could be an **Environment Setup Document** (specifying cloud accounts, CI/CD pipelines, MLOps tools for deployment).
- **Security Architecture & Compliance Checklist:** While security spans phases, here it is important to finalise security mechanisms in the tech stack – documented perhaps as an extension to the architecture. This could include decisions on encryption (data at rest and in transit), key management, access control design, and how the solution aligns with security policies. A checklist or matrix might be delivered to verify all security controls (firewalls, IAM roles, audit logging, etc.) are in place for the chosen tech components.

Decision Points

- *Cloud vs On-Premises vs Hybrid:* One of the biggest decisions – whether to deploy the AI solution on public cloud, private cloud/on-prem, or a hybrid model. Many enterprises choose cloud for AI due to scalability and availability of AI services (like GPU machines on demand, or services like AWS SageMaker, Azure AI, Google Vertex AI). But considerations like data sensitivity or latency might dictate on-prem or edge deployments. The decision might even be per component: e.g. training might occur in cloud, but inference maybe on edge devices (for IoT scenarios) or on-prem for data locality. In 2025, a hybrid approach is common, balancing cloud scale with on-site control.
- *Hardware and Accelerators:* Determine the hardware needed. Do you need GPU instances? If using deep learning, likely yes. If the AI is simpler (like some

business rule AI or small models), CPU might suffice. Also consider specialised AI chips or accelerators (TPUs, FPGAs) if relevant. The cost-performance trade-off is key: GPUs are expensive, so maybe during prototyping a local GPU or a small cloud GPU is fine, but for production at scale, you might invest in a managed service that optimises cost.

- *Tooling for Deployment and Ops:* Decide on the tooling for deployment (DevOps/MLOps). Will you use containerisation (Docker, Kubernetes) for deploying the AI services? Perhaps use a CI/CD pipeline for model updates. The enterprise might already have DevOps pipelines – can those be extended to include model deployment (a practice known as MLOps)? Selecting a consistent approach ensures the AI solution can be managed like any other enterprise app. For example, using Azure DevOps or Jenkins pipelines to deploy AI containers to a cluster, or using specialised tools like MLflow for model management.
- *Third-Party Platforms and Services:* Evaluate and decide on any third-party components. For example, if you need natural language processing, do you use an open-source library or a cloud API like Google NLP? If you need a search engine for knowledge base, do you use Elasticsearch or a cloud search service? Such decisions should consider vendor lock-in, cost, and flexibility. TOGAF encourages creating a **Technology Reference Model (TRM)** – you might extend it to include AI-related services as acceptable standards. The decision could be to standardise on one platform for consistency (e.g. "we will use Azure Cognitive Services for any computer vision tasks, given our MS ecosystem"), or to keep it open.

Example

Consider an AI-powered analytics platform for an enterprise. The tech architecture decisions: They choose a **cloud-first approach** – using AWS as the primary platform – to leverage its AI services and scalable infrastructure. The Technology Architecture diagram shows: an Amazon S3 data lake for storing large datasets, Amazon SageMaker for model training and hosting (managed Jupyter notebooks and training clusters), an AWS EKS (Kubernetes) cluster for deploying custom microservices (the inference API and a web app), and Amazon DynamoDB or Aurora for application state data. They include AWS IAM for access control, and AWS KMS for encryption key management to secure data. They also plan for a dev/test/prod environment separation using different AWS accounts or clusters. To reduce dependency on one cloud, they containerise the core AI services so they *could* be ported to another Kubernetes environment if needed, and they avoid using very proprietary services for critical components (for example, they use an open-source machine learning library within SageMaker so models can be moved elsewhere if necessary). The **Technology Standards** note that all APIs must use TLS encryption, all sensitive data must be encrypted at rest (AWS provides tools for this out-of-box, which aligns with enterprise security standards). They also check compliance: since the data involves EU customers, the architecture will deploy certain data processing in an EU region of the cloud to meet GDPR data residency requirements.

Pitfalls to Avoid

- **Underestimating Infrastructure Needs:** A common mistake is not provisioning enough or the right kind of infrastructure for AI workloads. For example, training a complex deep learning model on a standard CPU server could take weeks where a GPU instance could do it in hours. Conversely, over-provisioning is also a risk – e.g. using expensive GPU servers when not needed. Always align infrastructure with the AI's computational demands (profiling early can help determine this). Failing to do so can either bottleneck the project or waste budget.
- **Neglecting Scalability and Reliability:** If the technology architecture isn't designed for scale from the start, an AI pilot that succeeds might fail when scaling enterprise-wide. For example, deploying a single-instance model server might work for a pilot, but enterprise deployment may require load balancing and auto-scaling. Not planning for that in Phase D can lead to painful refactoring later. Similarly, ensure redundancy – e.g. if an AI service goes down, is there a failover? Not planning these aspects can result in outages or poor performance at critical times.
- **Security as an Afterthought:** It's a serious pitfall to ignore security and compliance requirements at the infrastructure level. Examples include failing to encrypt a database of training data, which could be a huge breach risk, or not restricting network access to the AI services (leaving an API endpoint open to the internet without proper authorisation). Always integrate with the enterprise's security architecture – e.g. use existing identity providers for authenticating to the AI app, put AI services in secure network zones, etc. A related mistake is not accounting for regulatory compliance in the tech choices – e.g. using a cloud region that violates data residency laws. These oversights can be project-killers if found later by compliance audits.
- **Vendor Lock-In Without Strategy:** While using cloud and vendor services is often beneficial, a pitfall is getting locked into a proprietary solution that limits future flexibility or bargaining power. If you rely too heavily on a single vendor's specialised service (say a unique machine-learning API), you might

struggle to move or adapt later. It’s wise to have an exit strategy or use more open solutions for core components if that’s a concern. TOGAF’s emphasis on vendor-neutral architecture is worth remembering: design your architecture such that components can be replaced or migrated if needed (this might be as simple as using containers, or ensuring data is stored in standard formats).

Phase E: Opportunities & Solutions – From Architecture to Implementation Plan

Phase E, **Opportunities & Solutions**, is about figuring out *how to implement* the target architecture. In this phase, the high-level architecture from Phases B, C, D is reviewed against practical considerations: identifying possible solutions (including commercial products or internally developed systems), evaluating implementation options, and forming the initial cut of an implementation roadmap. In the context of AI, this phase often involves deciding on pilot projects, quick wins, or staged rollouts that will achieve the architecture vision.

Strategic Goals

- **Identify Solution Building Blocks:** Determine the specific solution components (or **“work packages”** in TOGAF terms) that will realise the architecture. Some components might be delivered by purchasing or subscribing to a product, others by custom development. For AI, this could mean selecting which AI platforms or tools to use for each part of the design. For example, deciding whether to use a pre-built AI service (like an OCR API for text recognition) versus building a custom model. The goal is to leverage opportunities (existing solutions, reusable components) to expedite development while meeting requirements.
- **Evaluate Implementation Alternatives:** For each major architecture component, consider options and choose the best fit. For example, if your architecture needs a chatbot capability: options might be building one with open-source libraries, using a cloud AI chatbot service, or extending an existing enterprise chatbot tool. Evaluate based on criteria such as cost, time, skills, vendor fit, and how well each option meets the requirements (accuracy, maintainability, etc.). Often a **trade-off analysis** is done here.
- **Prioritise and Sequence Projects:** Often the target architecture can’t be implemented all at once – especially for large AI initiatives. This phase sets a prioritised sequence of projects or phases. The strategic aim is to maximise

value delivery early (e.g. prove the AI's value with a pilot or MVP) while laying groundwork for scaling. For AI projects, a common strategy is to start with a **Pilot or MVP (Minimum Viable Product)** to test the concept and deliver immediate value. Then, plan subsequent increments to refine and expand the solution.

Key Artefacts & Deliverables

- **Preliminary Implementation Roadmap:** A high-level timeline or plan that maps out the sequence of implementation projects. It identifies quick wins vs longer-term projects. For example: *Phase 1: Pilot in Department X (Q1), Phase 2: Extend to Department Y (Q3), Phase 3: Full enterprise rollout (Next Year)*. This roadmap is fleshed out in Phase F, but an initial version is drafted here.
- **Solution Options and Decisions Log:** Documentation of the options considered for key architecture elements and the decisions made. For example, a table listing each major component (e.g. "Model Serving Platform") with columns for Option A, Option B, Option C (with descriptions/pros/cons) and a note on the chosen option. This provides rationale for design choices (maybe captured in an Architecture Decision Record format).
- **Updated Architecture (if necessary):** Sometimes evaluating solutions might cause slight adjustments to the architecture. For example, discovering that a certain product provides additional features could refine some interface definitions, or realising that a requirement can't be met leads to revisiting some design. Phase E can produce an updated target architecture diagram annotated with chosen solutions (this transitions to a more implementation-aware model).
- **Outline of Projects/Work Packages:** A listing of the projects or agile epics that will be undertaken. For an AI solution, this could include projects like "Data Lake Setup Project", "Model Development Project (Pilot)", "AI Integration with CRM Project", etc., each corresponding to pieces of the architecture. Each entry might have an objective and brief scope description.

Decision Points

- *Pilot or Proof of Concept Scope:* Decide what the pilot implementation will cover. Many organisations choose a small scope that demonstrates the AI's capabilities. For example, if the overall goal is an enterprise-wide AI forecasting system, the pilot might focus just on one product line or one region to test the approach. Deciding this scope is crucial – it should be manageable but representative enough to yield learnings.

- *Use of Existing Platforms:* Identify if existing enterprise platforms can be utilised as part of the solution. For example, the company might already have a data science platform or an analytics tool. Or maybe leverage a platform like ServiceNow or Salesforce which has some AI capabilities (like ServiceNow's AI-driven ticket routing or Salesforce Einstein). If those can fulfil part of the requirements, a decision might be to integrate rather than reinvent. This can reduce cost and implementation time.
- *Buy vs Build for AI Components:* Perhaps re-emphasising, but here is where you might formally decide "buy" for some parts. E.g., *"We will use Microsoft Azure Cognitive Services for language translation instead of developing our own model,"* or *"We will purchase an MLOps tool for model monitoring rather than building one."* These decisions often depend on time-to-market needs and internal skill availability.
- *Architecture Compliance of Solutions:* As solutions are chosen, ensure they comply with the earlier defined principles and requirements. A decision might be needed if no solution perfectly fits – e.g. if a vendor product meets 90% of needs but lacks something, do you accept that and adjust requirements, or do you go custom? Trade-offs are decided here, often requiring stakeholder input (e.g. is it acceptable to use a slightly less accurate but ready-made AI model to save time?).

Example

A bank has a target architecture for an AI-based credit scoring system. In Phase E, they identify opportunities to reuse or buy: They discover their core banking software vendor offers an AI module for credit scoring – an opportunity to consider. They evaluate that vs. developing their own model in-house. Option analysis might show the vendor module is quick to deploy but is a black-box (less control), whereas an in-house model is more transparent but would take longer. They decide to pilot using the vendor's AI module in one product line to get started quickly, while simultaneously starting a project to develop proprietary models for long-term differentiation (a two-pronged strategy). They also identify a quick win: using an existing data warehouse and BI tool to incorporate some AI-driven risk analytics, which can be delivered in a few months to show value to stakeholders. The preliminary roadmap is drawn: Q1 pilot with vendor solution, gather results; Q2–Q3 develop internal model; Q4 compare and possibly replace vendor module with internal model if successful. They package these as two projects. They adjust the architecture diagram to highlight the interim use of the vendor module as a placeholder for the scoring component. They also ensure the plan includes user testing and feedback loops after the pilot.

Pitfalls to Avoid

- **Analysis Paralysis:** With many AI tools and platforms available, teams can get stuck endlessly evaluating options or chasing the "next big thing." This can delay progress significantly. Avoid overly prolonged analysis – time-box the evaluation phase. It's better to choose a reasonably good solution and iterate than to search for a perfect solution forever. In AI, technologies evolve quickly, so an 80% solution now might beat a 100% solution chosen too late.
- **Ignoring Quick Wins:** Sometimes architects focus solely on the end-state and miss interim opportunities to deliver value. This can cause stakeholder patience to wane. Avoid a plan that says "we'll spend 2 years building this AI and then value will come." Instead, identify something tangible to deliver early (like an MVP that automates one step or provides some insights). Not doing so is a pitfall because AI projects often benefit from iterative learning and proving value early to secure further investment.
- **Choosing Solutions Incompatible with Enterprise Strategy:** If a solution doesn't fit the broader enterprise direction (e.g. picking a technology that the company is phasing out, or a cloud provider that isn't approved), it can cause trouble later. Always cross-check solution choices with enterprise architects and IT strategy. For example, if the company has standardised on Azure and you bring in a niche AWS service for AI, you might face integration or governance issues. Deviations might be justified in some cases, but they should be conscious decisions with stakeholder buy-in.
- **Budget and Resource Neglect:** It's a mistake to outline solutions and projects without considering if you have the budget and people to do them. In Phase E, make sure to reality-check: do we have skilled data scientists to build that custom model? Do we have the funds to purchase that expensive AI software license? Not aligning the plan with resource realities can lead to failure in execution. Secure at least preliminary budget estimates and commitments if possible, for the planned initiatives.

Phase F: Migration Planning – Roadmap to Enterprise AI Deployment

Phase F, **Migration Planning**, takes the outputs of Phase E and turns them into a detailed implementation plan or roadmap. It is about scheduling and managing the move from the current state to the target state. For AI solution architecture, this phase is crucial to *operationalise* the plan – determining how to roll out AI capabilities across the enterprise, how to manage the transition (especially if going from a pilot to full production), and how to coordinate with other ongoing projects.

Strategic Goals

- **Formulate a Detailed Roadmap:** Develop a timeline with milestones for deploying the AI solution. This includes phases of work, interim architecture states, and how the enterprise will transition from any legacy processes to the new AI-enhanced processes. The goal is to minimise disruption while ensuring the solution is delivered in a logical sequence. For AI, the roadmap might start with limited deployments (e.g. one department or a subset of data) and gradually expand.
- **Allocate Resources and Budget:** Plan the required resources for each step – teams, skills, technology environments, and budget. The aim is to ensure that each project or work package identified has the necessary support. For example, schedule when data engineers will build the data pipeline vs. when data scientists will train models, ensuring those resources are available at the right times. Also align with budget cycles (e.g. if a big expenditure on GPU hardware is needed, plan it in a fiscal quarter where funds are allocated).
- **Mitigate Migration Risks:** Identify risks associated with the transition and prepare mitigation plans. In AI projects, risks might include model accuracy not meeting expectations in broader use, user adoption issues, or technical integration problems at scale. Migration planning sets up fallback plans or parallel runs (e.g. keeping old process running until new AI process proves stable) to reduce risk.

Key Artefacts & Deliverables

- **Implementation and Migration Plan:** A comprehensive document or set of documents detailing what will be done when, by whom. It often includes a Gantt chart or timeline, descriptions of each work package/project, resource assignments, and interdependencies. For an AI rollout, this might detail tasks like *"Develop training data pipeline (Jan–Mar), Complete Model training (Apr), Pilot deployment (May), User training (Jun), Production rollout (Jul)"*, etc., with specific owners.

- **Transition Architecture States:** If the target solution cannot be reached in one step, intermediate architectures are defined. For example, **Interim State 1** might be "AI pilot is live alongside existing system (users can choose old or new process)", **Interim State 2** might be "AI system replaces old system for one business unit" and so on, until the final state. These states are documented to clarify what the architecture looks like at each milestone.
- **Migration Risk Assessment & Mitigation Plans:** A log or section in the plan that lists potential pitfalls during implementation. For each risk, a mitigation or contingency is described. E.g., *Risk:* "Model performance may drop in real-world use," *Mitigation:* "Have a human review process for first 3 months, and plan for a model retraining if accuracy falls below X." Another example: *Risk:* "Integration with legacy system Y might fail," *Mitigation:* "Run legacy and new system in parallel for one cycle to validate outputs, with rollback procedures documented."
- **Communication and Change Management Plan:** Often overlooked in architecture docs but critical for AI projects – a plan for communicating the changes to stakeholders and users, and training them. This could be a separate deliverable or included in the implementation plan. It ensures that as you migrate, the people impacted are kept informed and prepared (e.g. training sessions for staff on the new AI tool scheduled in the timeline).

Decision Points

- *Phasing Strategy:* Decide how many phases or increments the implementation will be broken into. Will it be a **big bang** deployment (all at once) or gradual? Gradual is usually safer for AI. For example, an AI might first be a decision support tool (advising humans) and later move to full automation once trust is built. Deciding these phases is key.
- *Pilot to Production Transition:* If a pilot was done, how will you scale it? The team must decide if the pilot system can be evolved into production or if it needs to be rebuilt more robustly. Sometimes, pilots are done quickly and cut corners (which is fine); then a decision is needed whether to refactor that pilot code or start fresh using lessons learned. This phase plans that transition.
- *Timing and Coordination:* Decide the timing of the rollout in coordination with other business events. For example, do not schedule a disruptive change right before holiday season if it affects e-commerce, or end-of-quarter if it affects finance. Also coordinate with other IT changes (no conflicting system downtimes, etc.). This might involve deciding a deployment window or a sequence like department by department or region by region.

- *Resource Commitments:* If multiple projects are running in parallel (e.g. data pipeline building and model development), decide if you need to hire or reassign staff, or bring in external consultants. Make decisions on training needs for existing staff to be able to work on new tech (maybe data engineers need training on a new streaming platform being introduced). Secure commitment from department managers if their people are needed for subject matter expertise or testing. Essentially, finalise "who does what" and ensure buy-in.

Example

An e-commerce company is implementing an AI personalisation engine. In Migration Planning, they lay out a roadmap: Quarter 1 – set up data pipelines and develop the model (by a small data science team); Quarter 2 – pilot the personalisation on a small percentage of users on the website (maybe 5% of traffic gets AI-driven recommendations, while others see the old system); Quarter 3 – ramp up to 50% if pilot metrics are good, and integrate the AI with email marketing as well; Quarter 4 – full deployment to all users and all channels. They define intermediate states: at first, the AI runs shadow mode (making recommendations that are not actually shown, just to compare performance). Then an interim state where AI recommendations are shown but users can also see "others also bought" (old system) as backup. Final state: AI fully replaces old recommender. They identify tasks such as "Train marketing team on AI-driven campaign planning – July" as part of change management. They also plan for scaling: e.g. in Q3, need to optimise the model serving infrastructure to handle more traffic (so they allocate time for performance tuning). Risks they list: "Model might not scale to full traffic" – mitigation: invest in load testing and have additional servers ready. Another risk: "Customer reaction – some might find recommendations odd" – mitigation: include a feedback mechanism on the site for customers to thumbs-up/down recommendations, and monitor it. They coordinate with the Web Platform team to ensure no other major feature rollouts collide with the AI launch. By the end of Phase F, they have a clear schedule and everyone (IT, business, marketing) knows the plan.

Pitfalls to Avoid

- **Unrealistic Timelines:** Perhaps the most frequent issue – being overly optimistic about how quickly things can be done. Developing and deploying AI can involve research uncertainties (maybe the model needs more iterations) and integration challenges (data issues, refactoring). If the plan is too aggressive, deadlines will be missed and confidence lost. Avoid pressure to promise an AI solution "next month" if data isn't ready or the team is new to AI. Instead, include buffer for experimentation and unforeseen issues. It's better to under-promise and over-deliver.

- **Lack of Contingency Plans:** If you don't plan what to do if something goes wrong (and in AI projects something often does, e.g. model doesn't meet accuracy target initially), the project can get derailed. Not having a fallback (like continuing with the old process a bit longer or simplifying the model) means you might face a crisis close to rollout. Always include contingency in migration planning – e.g. parallel run plans, staged rollouts that can be paused, etc. Avoid a plan that has no "Plan B".
- **Ignoring Operational Aspects:** Sometimes the focus is just on building and launching the AI, but not on who will run it thereafter. Phase F should ensure operational readiness – e.g. decide who will monitor the AI, who retrains models when needed, what the support process is if the AI system has issues. If you ignore setting up these operational roles and processes, post-deployment can be chaotic (no one to respond to incidents, or models go stale because no retraining scheduled). Include establishing an **MLOps process** or similar in the plan.
- **Poor Communication of the Plan:** If the migration plan isn't communicated to all stakeholders (especially those outside the core project team), you risk confusion and lack of coordination. For example, if end-users aren't told when the new AI feature will show up, they might be caught off guard or not use it. Or if IT ops isn't aware of an upcoming deployment, they might not be prepared for it. Always disseminate the plan appropriately – perhaps through governance forums or direct stakeholder meetings – and update everyone if timelines change. A pitfall is treating the plan as an internal secret; transparency helps in enterprise environments to align everyone.

Phase G: Implementation Governance – Ensuring the AI Solution Meets Architecture and Standards

Phase G is **Implementation Governance**, where the focus shifts to overseeing the actual development and deployment to ensure it aligns with the architecture and meets quality standards. In simpler terms, as the AI solution is built and rolled out, this phase is about *governing* the process: making sure the project sticks to the plan, the design principles, and addresses any issues of compliance or performance. It is where the architecture team plays a watchdog and guidance role during implementation.

Strategic Goals

- **Ensure Conformance to Architecture:** As development teams and vendors implement the AI solution, governance ensures they follow the agreed architecture design. The goal is to prevent scope creep or "rogue" changes that violate the architecture. For example, if the architecture mandates using a certain data pipeline and someone tries to bypass it, governance will catch that. This keeps the solution on-track to what was planned and avoids unwarranted divergences that could cause integration problems.
- **Monitor Quality and Performance Metrics:** Set and enforce standards for the AI solution's quality – both in terms of software quality (robustness, security) and AI-specific metrics (accuracy, fairness, response time). The governance team might require periodic reports or tests. For example, they might establish that *"the model must maintain at least 80% accuracy on the test dataset"* or *"API latency should be under 1 second for 95% of requests"*. Tracking these ensures the solution will meet the business expectations set earlier.
- **Oversee Compliance and Risk Management:** Governance ensures that as the AI is implemented, it adheres to regulations, ethical guidelines, and internal policies. For example, verifying that data used in development was properly anonymised, or that the AI's decisions can be audited. It also involves checking that security controls are implemented as designed (penetration testing, code reviews for security, etc.). Essentially, governance acts as the gatekeeper so that nothing goes live that would put the enterprise at risk.

Key Activities and Artefacts

- **Architecture Compliance Reviews:** Formal checkpoints where the implementation (design documents, code, configurations) is reviewed against the architecture. This could be at multiple stages (design sign-off, pre-deployment review, etc.). Findings are documented in **Compliance Assessment Reports** highlighting any deviations. For example, if the review finds that a team used an unapproved database or the data quality checks are not as specified, those are noted and must be addressed.
- **Testing & Validation Reports:** Governance includes verifying that the solution works as intended. Apart from normal software testing, for AI there is

model validation. Deliverables here might be a **Model Validation Report** (did the model meet accuracy and bias criteria in testing?) and a **Performance Test Report** (did the system meet throughput/latency under expected load?). If using external AI services, ensure SLAs (Service Level Agreements) are met.

- **Risk and Issue Logs:** Throughout implementation, issues or risks should be tracked. The governance function keeps an eye on these and makes sure they are resolved. For example, if during development it's discovered that data volume is larger than anticipated and the system is running slow, that risk is logged, and an action (like optimise code or add hardware) is taken. Regular status reports might be part of governance.
- **Go/No-Go Decision Records:** Before final go-live, governance often facilitates a go/no-go decision process. This is a checkpoint where stakeholders review whether all conditions are met (testing passed, users trained, support in place) before approving to launch. The decision and any conditions (like "go live but monitor X closely" or "no-go until Y is fixed") are recorded.

Decision Points

- *Handling Changes and Deviations:* During implementation, change requests might arise – perhaps a new requirement or a necessary design change. Governance must decide whether to allow a change, and ensure any change goes through proper impact analysis (maybe even cycling back to earlier ADM phases if significant). For example, if mid-project the team suggests using a different AI algorithm that wasn't initially planned, governance evaluates if it still meets requirements and principles. If it's a good change, incorporate it formally; if not, veto it or defer to a later phase.
- *Readiness for Deployment:* Decide if/when the solution is ready to move from development to production. This is typically based on test results and meeting exit criteria. Governance might have a checklist: all critical bugs resolved, performance metrics green, user acceptance done, compliance sign-off obtained. Only then give the green light.
- *Exception Handling:* Determine how to handle any exceptions to architecture compliance. In some cases, a project might have to deviate due to practical reasons (maybe a component was not feasible, so a workaround is used). Governance can grant exceptions, but with conditions. For example, allow a temporary solution that doesn't fully comply, with the decision that it will be revisited in the next iteration or a later project. Document such exceptions clearly (so they aren't forgotten).

- *Operational Handoff:* Decide when and how to hand over the system to operational teams (like IT operations or a product team). This includes confirming that monitoring, support processes, and personnel are in place. Governance might set criteria: e.g. "we will hand off after one month of stable operations post-launch" or similar. This ensures long-term ownership is established.

Example

A telecom company implementing an AI network monitoring system is in the build phase. Implementation governance kicks in: The architecture board schedules a design review and finds that the developers want to use a different data storage for logs than specified (they chose a local storage for quick fix). Governance reminds them of the requirement to use the central logging system for compliance and has them adjust the implementation. They also set up a weekly checkpoint meeting to track progress and issues. One issue arises: the model's false-positive rate in detecting network anomalies is higher than expected, threatening the business goal of reducing alert fatigue. The governance team works with the data scientists to address this (perhaps by retraining or adjusting the threshold), and they decide that go-live will be conditional on bringing that metric within an acceptable range. As go-live nears, a compliance officer in the governance process checks that the system's audit logs are properly recording every alert decision (for future audit). Before launch, the team conducts performance testing simulating peak network events; governance reviews the report which shows response times are within limits. They hold a go/no-go meeting with stakeholders and decide to proceed, with a note that if the false positives issue resurfaces, they have a rollback plan. Throughout, all these decisions and findings are documented. After deployment, Implementation Governance might require a post-implementation review after a few weeks to ensure the system truly meets the goals and any remaining issues are handed to maintenance.

Pitfalls to Avoid

- **Overbearing Governance (Too Rigid):** Governance is important, but being too rigid or bureaucratic can stifle the project. For example, if every minor change requires a lengthy approval and paperwork, the team might lose agility and the project can slow to a crawl. Avoid micro-managing the implementation; instead, focus on critical compliance points. It's a pitfall to enforce the letter of the architecture without considering practical realities – some flexibility is needed as long as the core objectives and principles are upheld. Remember, TOGAF 10 encourages agile iteration within governance.

- **Insufficient Monitoring:** On the flip side, a hands-off approach can lead to drift. A pitfall is failing to catch deviations early. For example, if no one checks whether the data being used to train the AI was properly cleansed of sensitive info, you might discover a privacy issue after deployment – too late. Or if performance isn't tested, you might only find out in production that the system lags. To avoid this, governance should actively require and review test results, and verify requirements are being met *during* development, not just at the end.
- **Ignoring AI Ethical Considerations:** Traditional IT governance might not check for things like bias or fairness, but for AI these are crucial. A pitfall would be to green-light an AI solution that technically works but is making biased decisions (e.g. disproportionately flagging a certain group in fraud detection). Implementation governance for AI should include ethical AI checks – e.g. require bias testing results or ensure an ethics committee reviewed the use case if high-impact. Not doing so risks reputational damage and regulatory backlash.
- **Poor Documentation of Decisions:** If governance activities and decisions are not documented, knowledge can be lost and accountability weakened. For example, if an exception was allowed but not recorded, later teams may not know it's a temporary fix. Or if go-live criteria were informally waived, people might have different understandings of what was agreed. Always document the outcomes of governance reviews, including any accepted risks or technical debt, so that the enterprise has a record. Lack of this can lead to confusion later about why certain choices were made.

Phase H: Architecture Change Management – Evolving the AI Architecture Over Time

Phase H, **Architecture Change Management**, is about ensuring that the architecture remains useful and intact as changes occur over time. After the AI solution is implemented, the environment will continue to evolve: business needs shift, technology advances, new data becomes available, or issues are discovered. This phase establishes a process to manage updates or new iterations of the architecture in a controlled manner. For AI solutions, this is especially important because AI models and technologies can change rapidly (think of how quickly new algorithms or regulations appear).

Strategic Goals

- **Monitor Performance and Business Impact:** Continuously track how the AI solution is performing in production and what business value it's delivering. If the AI is not meeting the expected KPIs or if external conditions change (e.g. user behaviour changes), that might trigger a need to adjust the architecture. For example, monitor model accuracy, system usage statistics, feedback from users, and achievement of the original business objectives (like reduction in processing time or increase in customer satisfaction).
- **Identify and Assess Change Triggers:** Establish mechanisms to identify when a change to the AI or its architecture is needed. Triggers can be *business-driven* (new goals, mergers, entering new markets), *technology-driven* (new AI techniques, better tools, obsolete components), or *external* (new regulations like data privacy laws, changes in data availability). The goal is to proactively adapt the architecture so the AI solution remains relevant and compliant. For example, if a new, more accurate machine learning model type emerges (say the latest generation of GPT in 2025), you might assess if the architecture should incorporate it for better performance.
- **Govern the Evolution:** Ensure any changes to the architecture go through proper governance (much like Phase G, but on an ongoing basis). This prevents ad-hoc modifications that could break the integrity of the solution or conflict with enterprise architecture. The goal is a controlled evolution: improvements and fixes are made with awareness of all stakeholders and alignment to overall strategy.

Process and Activities

- **Regular Architecture Review Cycles:** Set up periodic reviews of the AI architecture – for example, quarterly or biannually – involving key stakeholders (business owners, architects, AI team, operations). In these reviews, discuss what's working, what's not, and any new opportunities or requirements. This might produce a backlog of potential enhancements or changes.
- **Change Request Mechanism:** Establish a formal way for changes to be proposed and evaluated. In TOGAF, this is often through Architecture Change Requests. For an AI solution, a change request could be like "Add capability for multilingual support in the chatbot" or "Incorporate new data source X to improve model accuracy." Each request is analysed for impacts, costs,

benefits. Minor changes might be approved for immediate action; major ones might trigger a new ADM cycle (going through phases A-H again for a next version of the architecture).

- **Continuous Improvement / MLOps Integration:** In AI specifically, integrate with **MLOps** (Machine Learning Operations) practices where the model training and updating is continuous. For example, an AI might require retraining with new data every month. Phase H would ensure that such model updates are considered architectural changes if they affect other components or performance. Activities could include scheduled retraining, model performance validation after each retrain, etc. The architecture management should ensure the pipeline for updates is in place and functioning.

- **Retirement and Sunset Planning:** Over time, parts of the solution might become obsolete or need replacement. Phase H includes deciding when to retire or replace components. For example, if you initially used a vendor's AI service but later developed your own, planning the deprecation of the old service is a managed change. Or if an AI feature is no longer needed due to business strategy change, properly turning it off and removing its components (and updating documentation) is done under change management.

Artefacts & Deliverables

- **Change Log / Repository:** A record of all changes proposed and their status. This might tie into the enterprise architecture repository. It helps maintain a history – for example, it might log that in January 2025 a change was implemented to use a new version of the machine learning model with details on why and how.

- **Updated Architecture Documentation:** When changes are approved and implemented, the architecture models (from phases B, C, D) must be updated to reflect the new reality. This ensures the documentation stays current (common failure is documentation lags behind – this process fights that by making updating a formal step). For example, if a new data source is added in the data architecture or a new microservice is introduced in application architecture, those diagrams and catalogues are revised.

- **Post-Implementation Evaluation Reports:** After significant changes, there might be a practice of evaluating if the change delivered expected results. For an AI model update, a report might show "after switching to Model v2, prediction accuracy improved 5% and user satisfaction increased accordingly." This closes the feedback loop and provides justification (or lessons if it didn't work as expected).

- **Communication of Changes:** Not exactly a document, but ensuring all relevant parties know about changes. This could be part of a broader architecture governance newsletter or meetings. For example, notifying the support team that "our recommendation engine now also considers inventory data since last week's update, which should reduce stock-outs by 15%." Communicating helps with transparency and usage.

Decision Points

- *Minor vs. Major Change:* Determine the level of change. Minor changes (no significant impact on business or other systems) might be handled within the maintenance team's authority. Major changes (like pivoting to a new AI platform or adding a major new functionality) might need re-initiation of the ADM cycle. For example, a decision to expand an AI solution enterprise-wide (when initially it was just one division) might be considered a new architecture project, effectively looping back to Phase A for that new scope.
- *Timing of Changes:* Decide when to implement changes. Some might be urgent (e.g. a regulation change with a fixed compliance date). Others can be bundled into a planned update cycle (maybe the company does major system updates every quarter or aligns with product release cycles). Timing might also consider AI-specific factors – e.g. don't retrain the model right before Black Friday if it's an e-commerce AI; do it well before or after critical business periods.
- *Acceptance Criteria for Changes:* For each change, decide how to measure success. For example, if adding multilingual support, criteria could be "the system can handle inputs in Spanish and French with equal accuracy as English as measured by test datasets." If a new model version is deployed, criteria might be "no degradation in key metrics and at least X% improvement in desired metric." Having these decided ensures you know if the change is successful or if you need to roll back.
- *Continuous vs. On-Demand Evolution:* Decide whether architecture evolution will be handled continuously or at set intervals. Some organisations adopt a continuous improvement mindset (especially with agile and MLOps, small improvements roll out frequently). Others might prefer a more waterfall-like approach with versioned releases of the AI system. For AI, a hybrid is common: models may update frequently, but major architectural shifts are less frequent. This decision shapes the operating model for the team – e.g. will they be in an agile sprint mode indefinitely (continuously integrating changes) or do periodic big updates?

Example

After deploying an AI-powered loan approval system, a bank enters Phase H. They monitor it and find that while it works well generally, the model's accuracy in predicting defaults drops as market conditions change (e.g. in a recession scenario). This is a trigger – they realise they need to retrain the model with recent data more frequently. They put in place a change: every quarter, retrain the model with latest data, and adjust the model features if needed (this is documented as an architectural enhancement to include a periodic retraining process). Additionally, new regulations come out requiring explanations for automated decisions. The bank raises a change request to add an *explainability module* to the AI system (maybe using an AI interpretability tool to provide reason codes for each approval/denial). This is a significant change that goes through architecture review – impacting data (need to store explanation data), application (new component or service for explanations), and potentially business process (informing loan officers of the AI's reasons). They treat it as a minor ADM cycle: revisit requirements (Phase A: regulatory requirement for explainability), adjust architecture in Phase B/C (maybe update business process that loan officers must review AI explanations, update application architecture to include an explanation engine). They implement it and update all documentation. They also schedule an annual architecture review meeting to consider any new AI techniques or business goals for loans – e.g. next year they might consider using AI for a different stage of the loan process and that would be planned here and possibly spun out as a new project. Throughout, they maintain a log of changes: when model versions changed, when explainability was added, etc., so anyone looking at the architecture repository can understand the system's evolution.

Pitfalls to Avoid

- **Architecture Drift (Uncontrolled Changes):** The biggest risk is that after initial implementation, people start making changes to the AI system in an ad-hoc way (especially technical teams tweaking things to improve performance or adding features at business requests) *without* updating the architecture or informing stakeholders. This leads to "architecture drift" – the implemented system diverges from the documented architecture and possibly from the intended structured design. To avoid this, enforce the change management process: even small changes should be logged and reviewed at the appropriate level. If drift occurs, the architecture might become outdated and lose credibility. A sure sign is when documentation no longer matches reality, making future maintenance difficult.

- **Stagnation (Ignoring Needed Changes):** The opposite pitfall is being too rigid and not updating the architecture when you should. If the team treats the architecture as frozen and doesn't adapt to new requirements or lessons learned, the AI solution may become less effective or even fail. For example, if user feedback overwhelmingly suggests a needed feature but architecture change process is slow or resistant, the system may not deliver optimal value. Or if a new AI model could vastly improve outcomes but architects say "No, we stick to the old design," that's missed opportunity. Balance governance with agility – be open to change when justified.

- **Lack of Retrospective Learning:** Failing to review what went well or poorly in the initial ADM cycle is a mistake. Phase H is a great time to do a *post-mortem or retrospective* on the project: did our assumptions hold? What could we improve in the next iteration? If this isn't done, the organisation might repeat mistakes. For example, maybe it turned out the data took far longer to prepare than anticipated – note that and next time involve data engineering earlier. Continuous learning from the process ensures future AI initiatives are more effective. TOGAF 10's modular structure encourages learning and adapting the method itself as needed for different scenarios.
- **Resource Disengagement:** After go-live, sometimes the key talent (data scientists, architects) are pulled to other projects, leaving a skeleton crew to handle maintenance. If the change management phase doesn't have proper resources, the system may degrade. An AI model left retrained or without expert oversight can become stale or buggy. Avoid completely disbanding the team post-launch; ensure there's an ongoing team or at least assigned responsibility for monitoring and updating the AI solution. Lack of this often results in great pilot projects that wither in the long run. It might be formalised as an *AI product team* or assigning it to an operations unit, but make sure someone "owns" the continued evolution.

Requirements Management – Continuous Requirements Monitoring in the AI Lifecycle

TOGAF's ADM includes Requirements Management as a central, ongoing process (often depicted at the centre of the ADM cycle). It is not a one-time phase but an activity that persists throughout all phases. Given its importance, we dedicate this chapter to how continuous requirements management applies in an AI solution context. In AI projects, requirements can evolve rapidly – initial goals may change as you discover new insights, and new requirements (especially around model performance, data needs, and compliance) often emerge mid-project. A robust approach to managing these requirements ensures that the architecture stays aligned with business needs and that any change in requirements is properly handled through the ADM cycle.

Role and Goals

Requirements Management ensures that:

- **All Requirements Are Captured and Addressed:** From business objectives to technical constraints, every requirement should be documented and linked to the architecture and implementation. For AI, requirements might include functional needs (what decisions the AI should make, what data it must use) and quality attributes (accuracy, response time, usability, security). For example, a requirement could be *"The fraud detection AI must identify at least 90% of fraudulent transactions with less than 5% false positives."* This would influence design choices and testing criteria.
- **Traceability:** One should be able to trace each requirement to the architecture artefacts and ultimately to solution components that fulfil it. In practice, one might maintain a **requirements traceability matrix** mapping requirements to specific phases or artefacts (e.g., which phase/step of the ADM addresses it, which component implements it). In AI solutions, traceability might involve linking a requirement like "explainable AI" to the design of an explanation module in Phase C and to a testing step in Phase G verifying it.
- **Manage Changes to Requirements:** As business or technical requirements change, the process ensures these changes are evaluated and, if accepted, propagated through the relevant ADM phases. For example, if mid-project a new requirement emerges *"the AI chatbot must also support Spanish language"*, Requirements Management will capture this and trigger potentially updates in several areas: data (need Spanish data), application (multilingual NLP support), etc. The ADM may cycle through adjustments in architecture vision or design to accommodate this.

Process in Practice

- **Requirements Gathering:** Initially (Preliminary and Phase A), gather high-level requirements from stakeholders. Throughout subsequent phases, gather more detailed requirements specific to those domains (business rules in Phase B, data volume/quality requirements in Phase C, integration requirements in Phase C/D, etc.). For AI, it's common to discover new requirements during prototyping (like realising you need a certain type of data or a certain inference speed for user satisfaction). Maintain a living requirements document or backlog.

- **Requirements Repository:** Store requirements in a central repository (could be a simple document, spreadsheet, or a specialised tool) that team members can access. Each requirement could have attributes: description, source (who requested it), priority (Must / Should / Could), status (proposed / approved / implemented), and trace links. In enterprise contexts, sometimes requirements are managed in tools that integrate with architecture repositories (ensuring architects can see them easily).
- **Continuous Alignment Checks:** At every phase, part of the work is to check that the architecture outputs align with the requirements. For example, after Phase B, verify that business architecture addresses all business requirements (if a requirement was "reduce manual effort by 50%," check that the target process flows and capabilities reflect that). After Phase C/D, ensure all technical requirements (like "must integrate with CRM system" or "data must not leave EU") are reflected in the designs. If a gap is found (a requirement not met), either update the design or revisit the requirement (maybe it was unrealistic or out of scope – in which case formally adjust or defer it).
- **Change Management Integration:** When a new requirement arises or an existing one changes (common in AI projects as data characteristics or business expectations shift), evaluate the impact. Small changes might be handled within the current phase activities; large changes might require going back to an earlier phase. For example, a new regulatory requirement could mean going back to Phase A to adjust vision/scope and then revising Phase B (business process to include compliance steps) and Phase C (maybe add encryption or consent management in data architecture). The requirement management process coordinates this so that nothing is lost in the shuffle.

Example

During the deployment of an AI-driven marketing recommendation system, suppose marketing comes up with a new requirement: "The AI should also consider real-time inventory levels so we don't recommend products that are out of stock." This requirement was not in the original scope. The Requirements Management process kicks in: it logs this requirement, assesses it – it likely affects data architecture (need a feed of inventory data), application logic (the recommendation algorithm must factor stock), and perhaps technology (integration with inventory system in real-time). The change is significant; it triggers an iteration of the ADM: architects revisit

Phase A to ensure the vision now includes reducing out-of-stock recommendations, update Phase B by adding this requirement as a business goal, adjust Phase C by designing a data flow from the inventory database and altering the model design, etc. Once the design is updated, implementation will proceed to include it. The important point is, the requirement was captured and systematically handled, not just informally passed to developers (which could have led to a quick hack that breaks architecture). After implementation, the Requirements Management would update the status of that requirement to "implemented in version 2" and possibly note the results (e.g., this change led to a 10% increase in recommendation conversion rate, showing a clear business impact).

Pitfalls to Avoid

- **Requirements Creep without Control:** AI projects often start with enthusiasm and then more and more ideas flood in ("Can it also do X? What about adding Y?"). Unmanaged scope creep can jeopardise the project timeline and focus. The pitfall is saying yes to everything without assessing impact. Requirements Management is there to enforce discipline: changes should go through impact analysis and formal approval (likely via governance board or project steering committee). Avoid the scenario where developers keep getting new tasks from business sponsors directly and the architecture becomes a patchwork of last-minute additions.
- **Overlooking Implicit Requirements:** Not all requirements are stated explicitly by stakeholders. Some are implicit or assumed (like compliance requirements, usability needs, etc.). A failing would be to ignore these until late. For example, an implicit requirement might be that the AI system's interface must adhere to the company's UX standards, or that it should handle peak loads. If these aren't captured, they might be missed in design. Engage stakeholders to draw out such needs and include them formally. Additionally, consider non-functional requirements as first-class citizens – performance, security, maintainability should be in your requirements list, not afterthoughts.
- **Poor Prioritisation:** Not all requirements are equal. If the team treats every requirement as top priority, they might get overwhelmed or lose sight of what's critical. Particularly with AI, some features might be "nice-to-have" vs. "must-have." If time is running out, it's essential to know which requirements can be postponed. A pitfall is failing to prioritise, leading to either trying to do everything (and doing none well) or dropping something important. Use a method (like MoSCoW prioritisation: Must, Should, Could, Won't) or value vs. effort matrix to focus on the most impactful requirements first.

- **Lack of Traceability and Testing against Requirements:** If you don't map requirements through design to implementation and testing, some can fall through cracks. For example, if a requirement was "system must log every user query for audit," and nobody traced that to a design element, it might be missed in implementation and discovered later during an audit – too late. The pitfall is not having a process to verify each requirement was indeed implemented and tested. Good practice is to write test cases or acceptance criteria for each requirement. In the audit log example, a test at Phase G would check that querying the system generates an entry in the log. Requirements management isn't just a spreadsheet; it's ensuring reality matches those needed features and constraints.

Conclusion

Requirements Management is the glue that holds the ADM phases together, ensuring that what is delivered meets the real needs of the enterprise. In AI solution architecture, this continuous thread is vital – it guards against the solution drifting away from business value and ensures agility in responding to new insights or changes. By diligently managing requirements, architects and product managers can maintain clarity on *"why"* behind every part of the AI solution, thus delivering a system that truly aligns with strategic goals and adapts to the enterprise's evolving landscape.

References

The Open Group. (2022). *TOGAF® Standard, 10th Edition*. Link: https://www.opengroup.org/togaf

The Digital Iceberg

www.thedigitaliceberg.com