Case Studies
Application Data Architecture

Exploring real-world challenges in Banking and Retail through the lens of a practical Application-Data Enterprise Architecture framework.

Introduction

Every organization grapples with complexity when it comes to managing applications, processes, and technology landscapes. The framework at the heart of this case study offers a structured way to catalog applications, actors, servers, locations, databases, and processes. It defines clear relationships such as 'Actor Owns Application', 'Application Is Hosted At Server', and 'Process Uses Application', enabling enterprises to visualize dependencies, measure business fit, and make strategic decisions.

Framework Summary: Application Portfolio Data & Integration

This framework provides a structured approach for managing an organization's **application portfolio**, its **technical underpinnings**, the initiation of **Data Architecture,** and the **business context** in which applications operate. It ensures visibility of applications, their ownership, infrastructure, business alignment, and integration points.

Core Components

Application

- Maintains a comprehensive inventory of applications with key properties such as *criticality, lifecycle recommendation, cost, end-of-life, business fit,* and *technical fit*.
- Enables organizations to assess portfolio health and make rationalization or modernization decisions.

Actors (Technology Custodians)

- Captures technical ownership and accountability through attributes like email and phone.
- Linked to applications via the *ActorOwnsApplication* relationship.

Case Study: Data Dependency EA Framework

Infrastructure Mapping

- **Servers** (vendor, end-of-life, IP) and **Locations** (datacenters, physical sites).
- Relationships (*Application Is Hosted At Server*, *Server Is Child Of Location*, *Application Is Hosted At Location*) ensure end-to-end visibility of hosting dependencies.

Projects and Change Impacts

- Projects are modeled with start and end dates.
- *Application Impacted By Project* highlights how initiatives affect the application landscape.

Business Alignment

- **Business Processes** and **Departments** provide a functional view.
- Relationships:
  - *Department Owns Application* shows operational dependency.
  - *Process Uses Application* demonstrates how applications support execution.
  - Hierarchies captured by *Process Is Child Of Process* and *Department Is Child Of Department*.

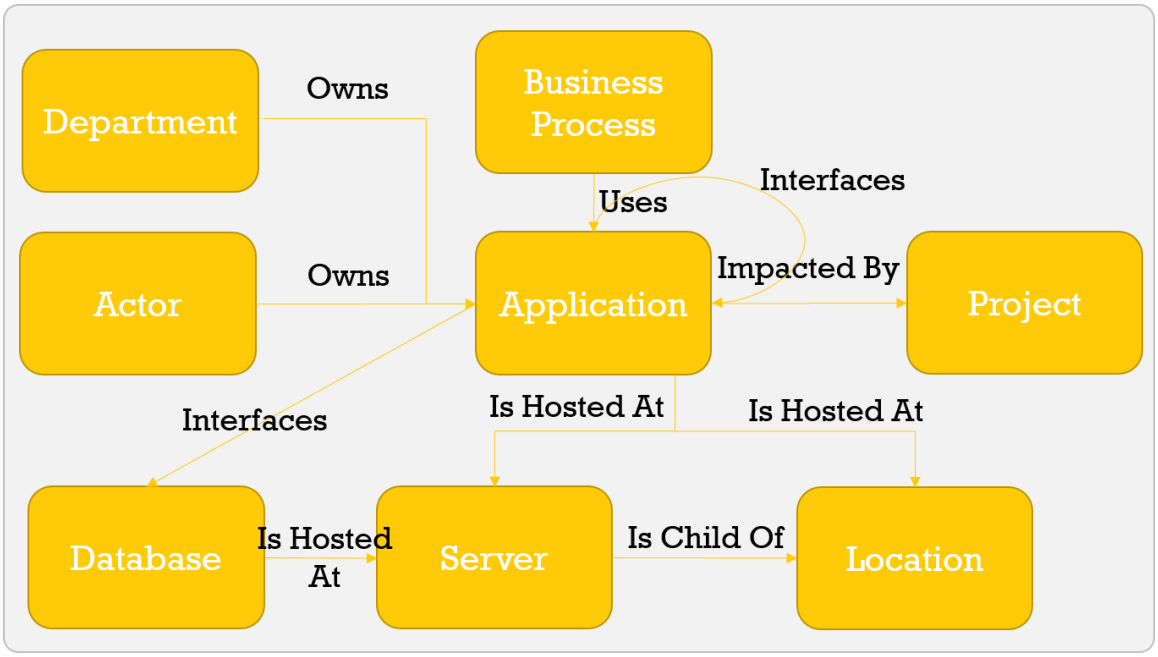
Data Layer (Databases)

- Database catalog extends the framework to include databases. It can be developed further by schemas, tables, and stored procedures.
- Links applications to their underlying data stores.

Integration Mapping

- **Application-to-Application Interfaces** and **Application-to-Database Interfaces** define dependencies.
- Attributes like *protocol* and *port* provide technical details for interface governance and standardization.

Case Study: Data Dependency EA Framework



Scenario 1: Sarah in Banking

The Problem Faced

Sarah is a senior IT manager at a mid-sized bank. Her department is struggling with a tangled web of applications. Many are critical for compliance and daily operations, but no one has a complete view of ownership, dependencies, or lifecycle status. Projects are delayed because teams can't quickly answer simple questions: Which servers host the core applications? Which applications are nearing the end of life? Which departments depend most heavily on outdated systems? How the data are used by the applications?

How the Framework Helps

The framework provided Sarah with a structured catalog of Applications, each tagged with criticality, business fit, technical fit, and end-of-life dates. By linking Actors to Applications via the 'ActorOwnsApplication' relationship, ownership became transparent. She could also trace how Business Processes relied on these applications and how Departments were impacted. By modeling 'Application Is Hosted At Server' and 'Server Is Child Of Location', Sarah gained visibility into the physical hosting environment.

Deliverables Produced

- An Application Portfolio Catalog showing lifecycle recommendations and costs
- A Dependency Map illustrating how business processes use applications
- Ownership Matrix linking actors to applications
- Technology Roadmap highlighting end-of-life systems and upgrade paths

Business Value Created

The bank achieved faster project planning, reduced compliance risks, and clearer accountability. Instead of chasing answers across departments, leadership could now rely on a single, consistent view of the application and process landscape. This visibility saved time and significantly reduced the risk of outages caused by unsupported systems.

Scenario 2: Tomson in Retail

The Problem Faced

Tomson is the Head of IT at a rapidly expanding retail chain. As the company opened new stores and launched an e-commerce platform, its technology environment grew chaotic. Applications had been integrated quickly, often without documentation. Customer-facing processes relied on systems that were interfacing through custom-built connectors no one fully understood. Server costs were rising, yet the business could not explain which applications truly justified the expense.

How the Framework Helps

By applying the framework, Tomson created a unified view of Applications and their interfaces. The 'Application Interfaces Application' relationship exposed how critical systems exchanged data, while 'Application Interfaces Database' clarified backend dependencies. Cataloging Servers and linking them to Applications revealed redundant infrastructure. Meanwhile, mapping Departments to Applications showed which parts of the business were most reliant on costly systems.

Deliverables Produced

- An Integration Map of application-to-application and application-to-database relationships
- A Server Hosting Inventory highlighting underutilized hardware
- Department-to-Application dependency analysis
- Cost vs. Business Fit analysis for prioritization

Business Value Created

The retail chain reduced IT spending by consolidating redundant servers and decommissioning low-value applications. Customer-facing processes became more resilient, as the integration map helped preempt failures during sales peaks. By aligning IT with business priorities, Tomson ensured technology investments supported growth rather than slowed it.

Conclusion

Through Sarah's banking story and Tomson's retail journey, this case study illustrates the power of an Enterprise Architecture framework. By cataloging assets, clarifying relationships, and highlighting lifecycle and ownership details, the framework transforms complexity into clarity. Whether in highly regulated industries like banking or fast-moving sectors like retail, organizations can use this approach to reduce risks, cut costs, and drive business value.