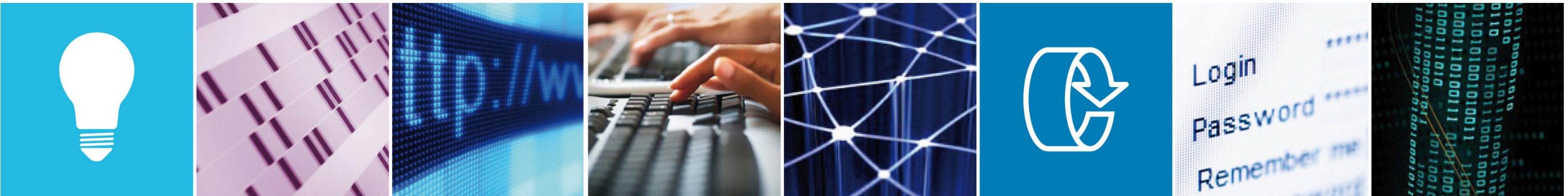
SOA ARCHITECT SUMMIT

Turn your ideas into practical solutions.

Gary Farrow / SOA Anti-Patterns

9th September 2009



Objectives

- To understand specific design anti-patterns that may occur within SOA deliveries
- To identify symptoms and understand root causes of these anti-patterns
- To review refactoring approaches for the identified anti-patterns

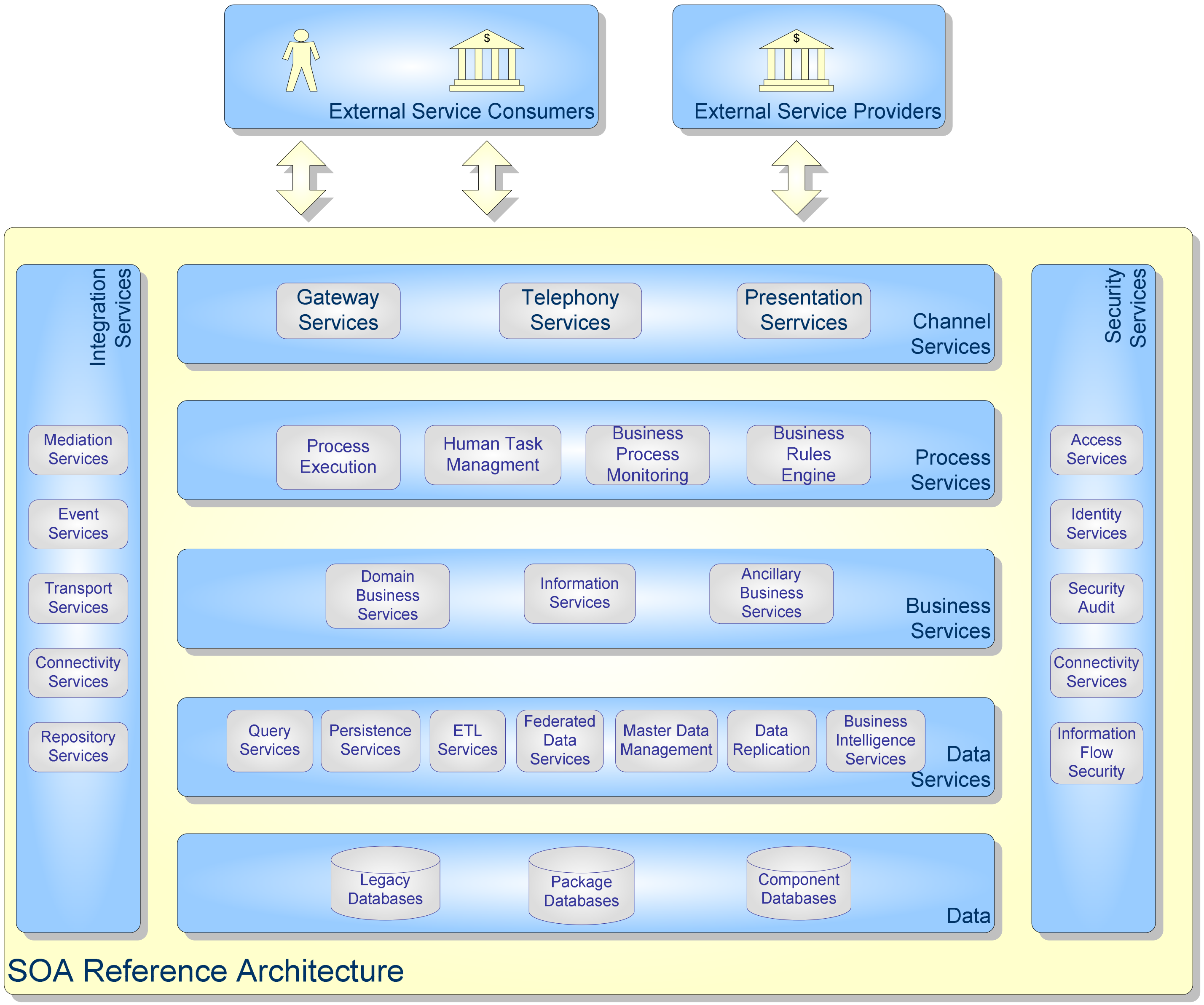
Agenda

- SOA Reference Architecture Model
- Context – Portal Channel Delivery
- Anti-Pattern Descriptions
- Refactoring Approaches

SOA Reference Architecture

A conceptual model for Service Oriented Architecture is now presented

SOA Reference Architecture Model



What

- Conceptual level architecture for enterprise, service oriented solutions
- Provides a framework for architecture services within the IT estate.
- Promotes the architecture itself as a set of underlying services.
- Services are grouped into a hierarchy providing increasing level of detail. In this way views of a solution at varying levels of detail can be provided.

Purpose

- To provide a logical grouping of related architecture services;
- To achieve a clear separation of concerns between architectural building block responsibilities;
- To establish a comprehensive taxonomy of the set of architecture services;

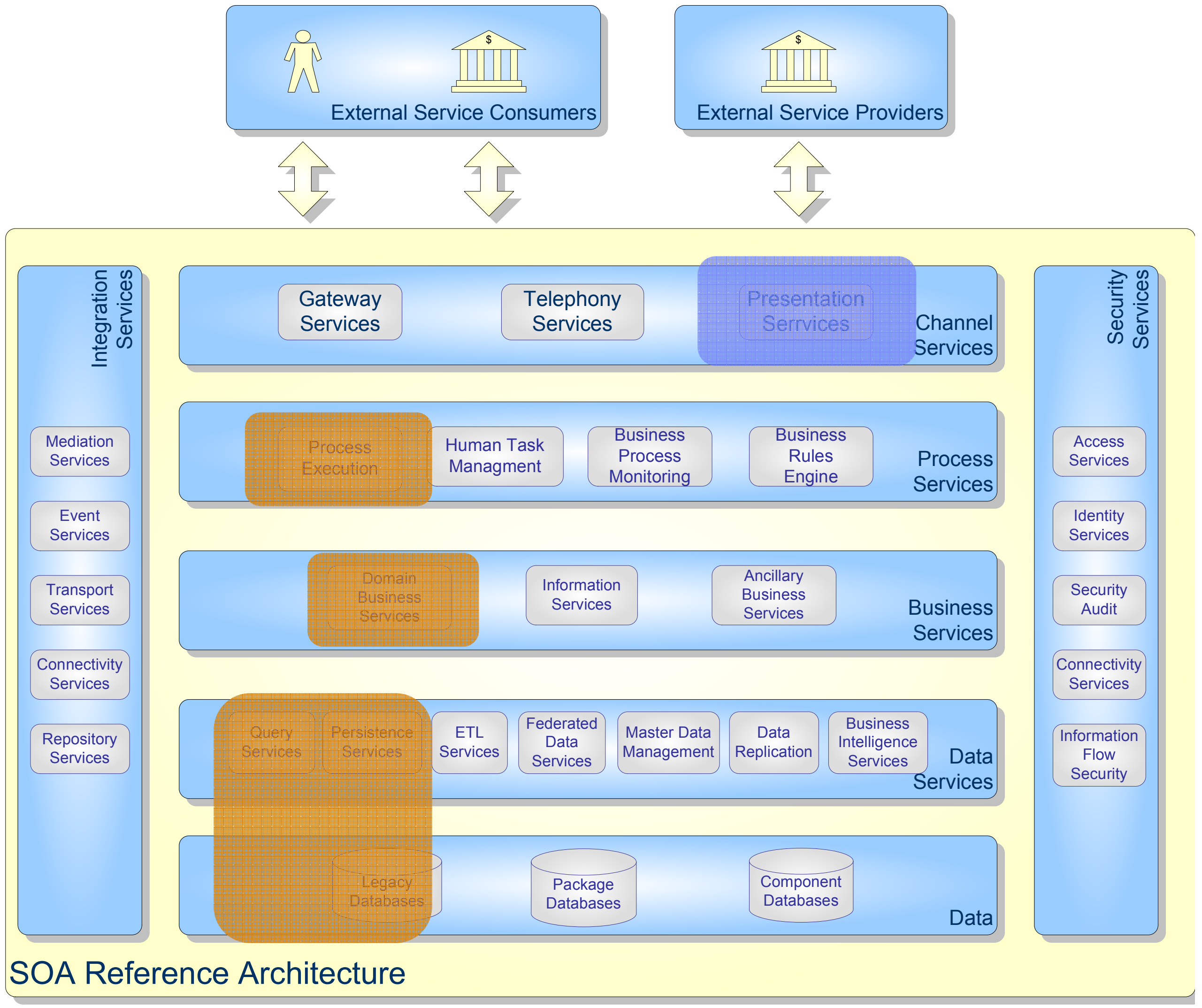
SOA Reference Architecture Uses

- To clarify architecture principles and illustrate the architectural impact of the chosen principles;
- To illustrate enterprise architecture solution patterns for given business scenarios,
  - Patterns can be defined using collaborating architectural building blocks
- As a tool for enterprise architecture roadmap planning
  - illustrating which architectural services are required at a given point in time to support a business programme
- To illustrate mappings of architecture services or building block to specific technology implementations;
- In a multi-supplier delivery environment, to confirm organisational boundaries of responsibilities where different suppliers are responsible for different parts of the end-end delivery.

Context – Portal Delivery

Experiences and problems observed with a Portal delivery are now presented in the form of anti-patterns

Delivery Model: Architectural Layer



- IBM Team responsible for building the presentation services
- Client team responsible for high level design and interface specifications
- Client team responsible for all of
  - Core and Ancillary Business Services
  - Data Services to support these
  - Process Services
  - Integration between Presentation Services and the Process Services

Anti-Pattern Descriptions

Two anti-patterns arising from this delivery model are now described

Anti-Pattern 1: Interface Bloat

Name	**Interface Bloat**
Also Known As	**Data Tsunami**
Most Frequent Scale	**System, Enterprise**
Re-factored Solution Name	**Process**
Root Causes	**Mistrust, Time pressure, inexperience**
Unbalanced Forces	**Management of Functionality, Management of performance**
Anecdotal Evidence	**We don’t know what we need so we’ll have the lot please.**

Interface Bloat – Root Causes

- Client were immature in their design process
  - Concept of a formal design process was relatively new and hence unproven
  - Production of formal specification of interfaces between components had not previously been done;
  - Being unable to achieve a specific focused specification a loose, over-specified interface was adopted.
- Desire to build services that could be reused
  - An over-emphasis on building reusable services in the Process Services layer
  - Interfaces between the Portal application and components in the Process Services layer was too generalized.
- Time pressure on delivery
  - Proper analysis was not undertaken in advance of the detailed technical design and coding.

Consequence

- Too much data was passed between the Presentation Service and the Process Services Reference Architecture layers
- Additional query and data transfer time was required
- Intensive processing was required to parse data in the Presentation Services layer and extract the sub-set of data needed for a specific operation
- Performance issues arose through slower response time due to marshalling and passing the data

Anti-Pattern 2: Reference Architecture Redundancy

Name	**Reference Architecture Redundancy**
Also Known As	**Pass the Parcel**
Most Frequent Scale	**Enterprise**
Re-factored Solution Name	**Role**
Root Causes	**Time pressure, paradigm misuse**
Unbalanced Forces	**Management of Functionality, Management of performance**
Anecdotal Evidence	**Service method name appears in all software layers**

Root Causes

- The SOA paradigm is relatively new for client organizations.
- In such circumstances a SOA Reference Architecture may have been explicitly defined or may be implied but its practical application is not fully understood within the client organization.
- There is a misunderstanding of the responsibilities of each layer in the Reference Architecture resulting in:
  - The same method signatures appearing on different services residing within each of the reference architecture layers;
  - An over generalization of the transfer objects passing between layers;
  - Business logic being pushed into the Data Services components instead of residing within components within the Business Services layer.

Consequences

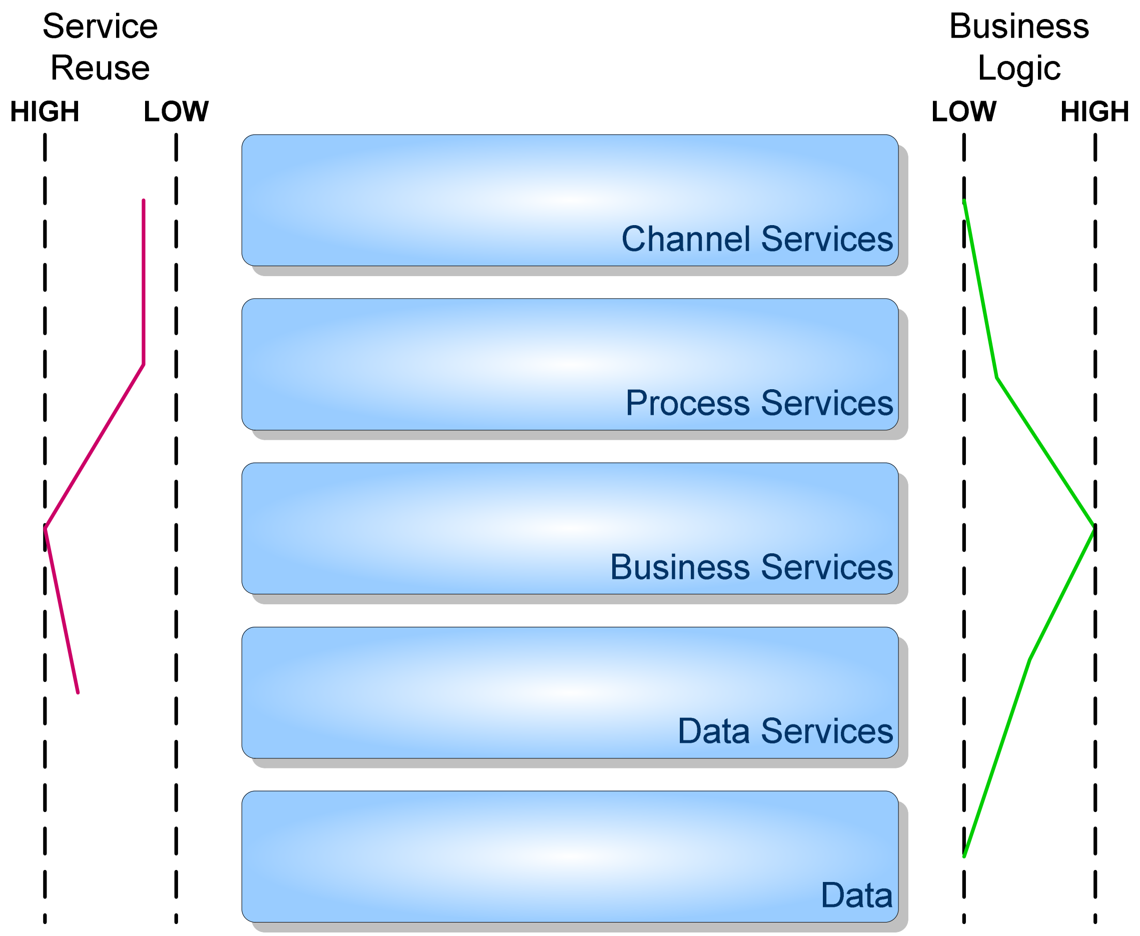
- **Reference Architecture layers are redundant**
  - They perform no added value and merely pass-through data to the next immediate layer in the architecture
- **Breaks ‘Law of Demeter’**
  - Don’t talk to strangers
  - Designed DTO’s to account for Channel Services
- **Reuse of Business Services is not achieved**
- **The advantages of the SOA paradigm are negated**

Refactoring Approaches

Approaches to refactoring the design to eliminate the anti-patterns are now described

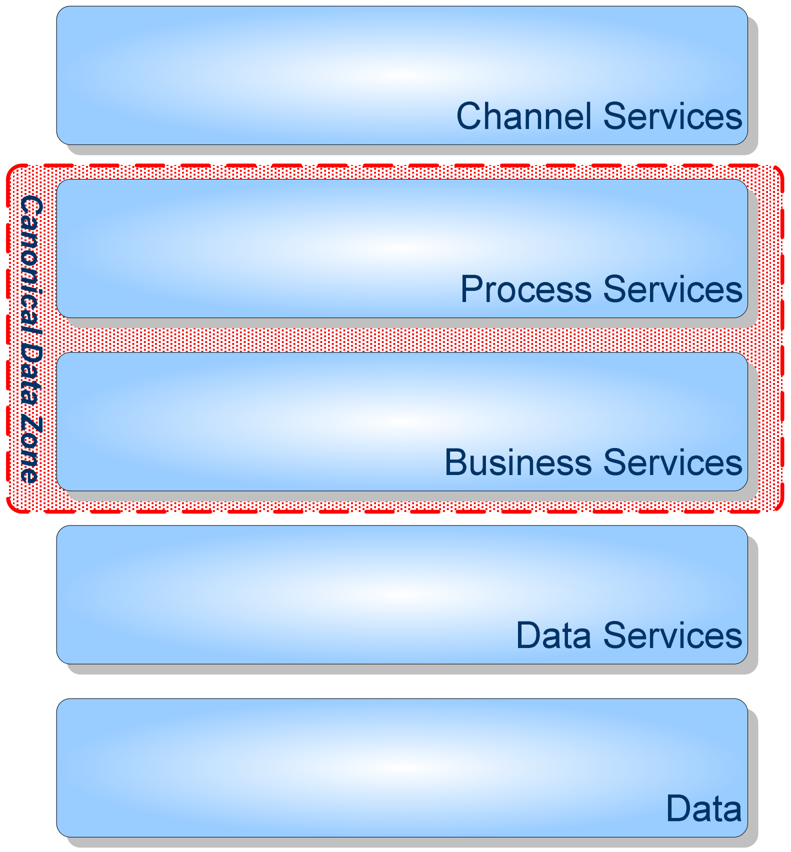
1. *Business Logic Distribution*
2. *The Canonical Data Zone*
3. *Reference Architecture Rules*
4. *Optimize Delivery Model*

Refactoring 1: Business Logic Distribution



- Diagram shows the optimal split of business logic across the defined architectural layers and level of reuse
- Channel Services layer Validation logic
- Process Services layer - Process logic relating to variations in business process
- Business Services layer – the business rules that define the business
- Data services / Data – data relationships and constraints
- Emphasis on reuse should be focussed at the Business Services Layer
  - Same services shared in different process contexts

Refactoring 2: The Canonical Data Zone



- That portion of a SOA Reference Architecture that operates using a common data format
  - Within service operations & data architecture
  - Illustrated in terms of the service layers
- Benefits
  - Support for *channel harmonisation*
  - The goal of a single common business process irrespective of the delivery channel
  - Vastly simplified IT architecture
- Implications
  - Channel services layer optimsed for the channel type (B2B, Portal)
  - Specific transformations undertaken by the Integration Service

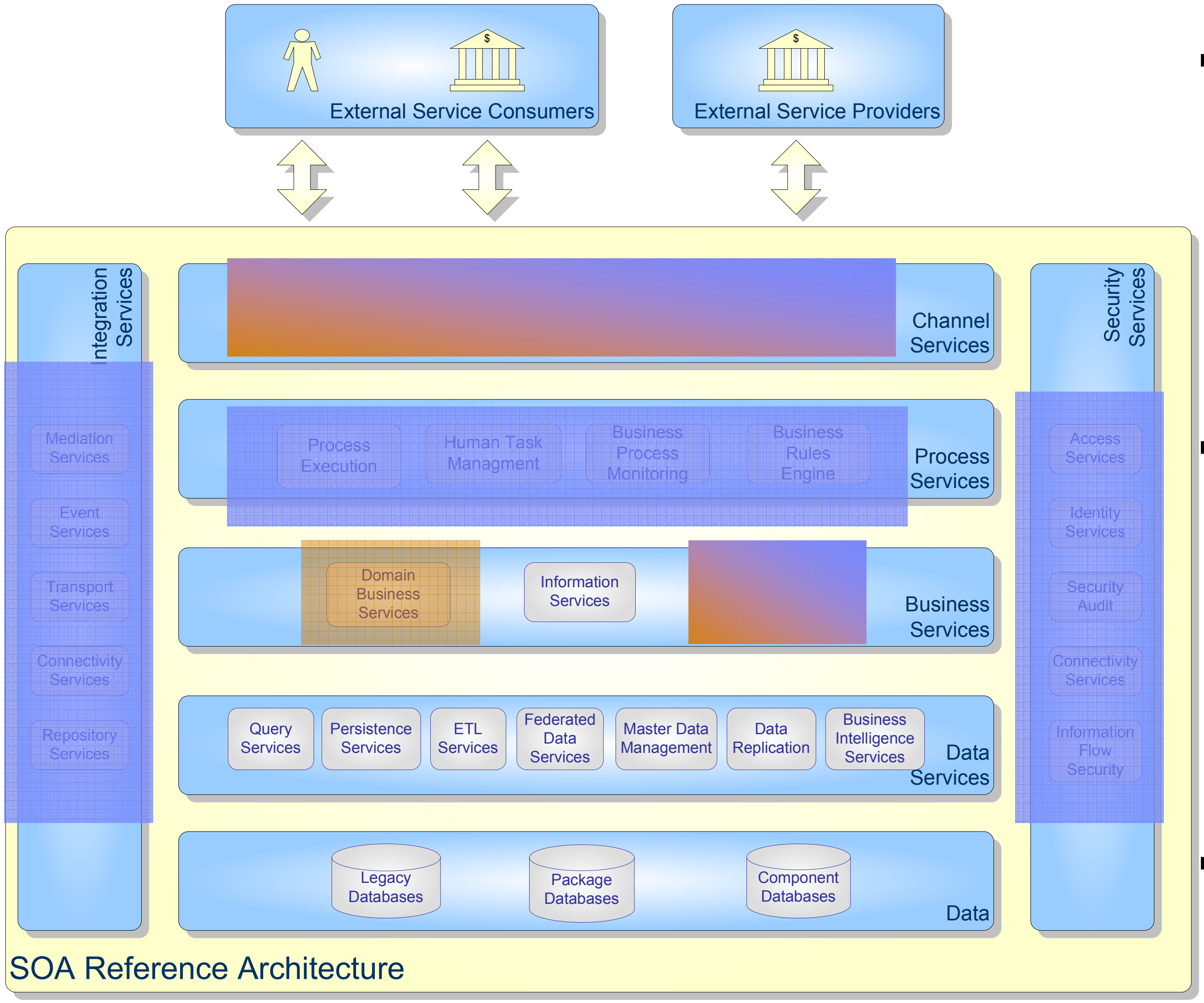
Refactoring 3: Reference Architecture Rules

- Allow flexibility in the way the Reference Architecture is applied
- Embodied in a set of architecture principles that accompany the SOA Reference Architecture.

Example:

- Data service operations that merely retrieve or update data, without the application of business logic should be permitted to be called directly from the Channel Services layer.
- There is no need to go through the Business Services layer in these circumstances.
- This principle should be embodied as part of the Reference Architecture definition.

Refactoring 4: Optimise Delivery Model



- Partition delivery to strengths of client and partner organisations
- System Integrator (IBM) focus
  - Integration Services
  - Process Services
  - Leverages expertise in systems integration
  - Leverages product expertise
- Client focus
  - Business services
  - Leverages their domain expertise
  - Leverages expertise in understanding of the IT systems
- IBM or client focus
  - Ancillary Business Services
  - Not domain specific

Review of Objectives

- **You should now**
  - Understand two specific design anti-patterns that may occur with SOA deliveries
  - Be able to identify symptoms and understand the root causes of the anti-patterns
  - Understand and apply re-factoring approaches for the identified anti-patterns