Business Process Digitalization and Cloud Computing

4. Architecture fundamentals

Andrea Morichetta, Phd

Computer Science Division

October 10, 2018

Table of contents

What is an Architecture?

Software architecture is a description of a software system in terms of its major **components**, their **relationships**, and the **information** that passes among them.

In essence, architecture is a plan for building systems that meet well-defined **requirements**.

A fundamental purpose of software architecture is to help **manage the complexity** of software systems and the **changes** in the business, organizational, and technical environments.

Architectural style

An **architectural style** is a family of architectures related by common principles and attributes. It contains a well-defined **set of patterns** that constitute a common way for enterprise solution components to interact with one another.

For example, an n-tier architectural style is designed to meet the following requirements:

- distribution
- scalability
- interface flexibility
- device independence
- business service reuse
- application integration

Architectural Principles and Practice

Separation of Concerns:
Reduce the complexity of the architecture simply separating elements. Changes in one part of the system does not adversely affect other parts. An example of this principle is the **separation of interface from implementation**.

Architectural views:
describe the **inclusion or exclusion of specific details** and the presentation of information to different stakeholders. Typical **software views** are logical, deployment, process, and network. Typical **enterprise architectural views** (concerns) are business, information, application, technology, and implementation.

Accommodation of changes:
Flexible architecture are more prone to satisfy **future application requirements** (e.g. using trends).

Architectural Principles and Practice

Abstraction

is the key method for **decoupling**, accommodating change and **separating concerns**

Pattern

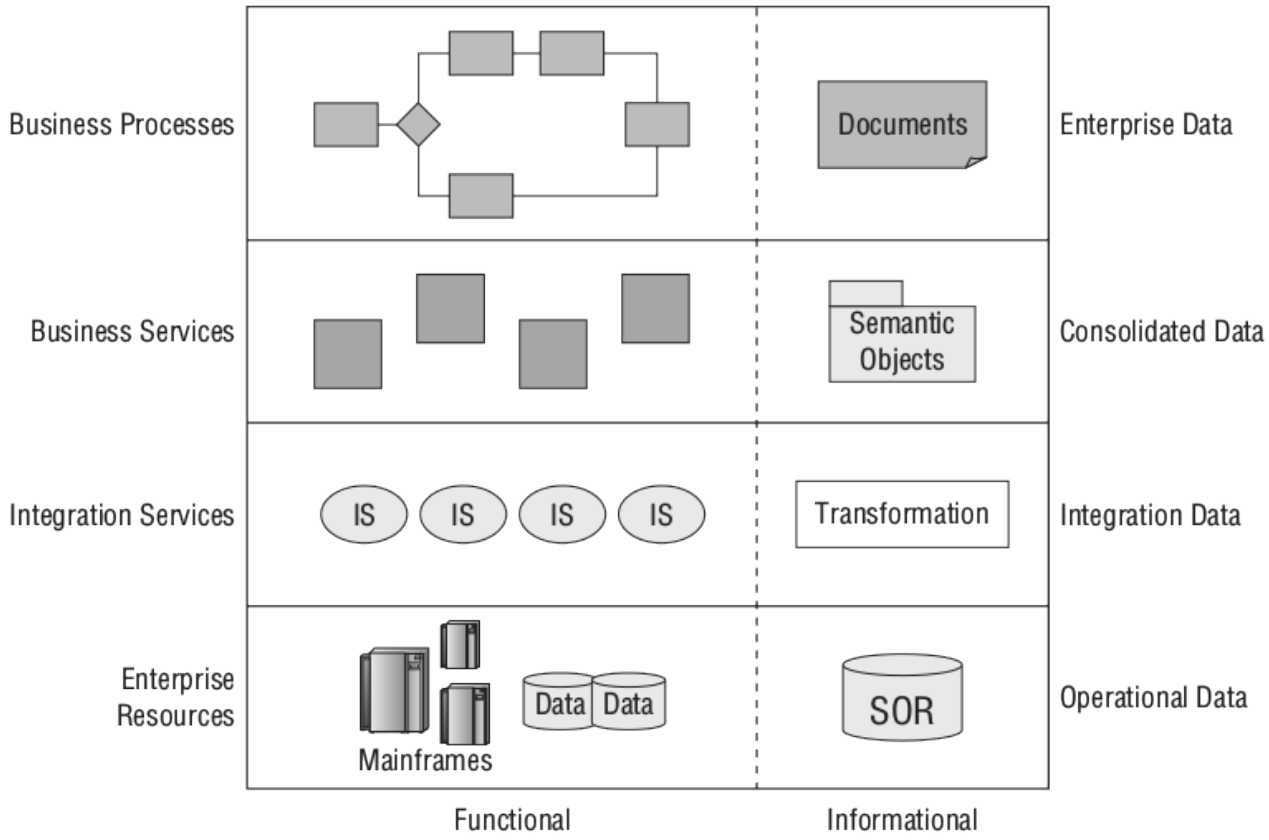
is a template for a **solution to a specific set of requirements**.
"Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use the same solution a million times over, without ever doing it the same way twice."

Communication

is the mechanism for people to communicate and **reach a common understanding**.

Service Oriented Architecture

The real value of SOA comes when **reusable services are combined to create agile, flexible, business processes**. The architecture of SOA is responsible for **creating the environment necessary to create and use composable services**



Architectural element of SOA

Enterprise Resources

Consists of **existing applications**, legacy, and COTS systems, including CRM and ERP packaged applications, and older object-oriented implementations.
The execution of an operation will typically cause one or more **persistent data records** to be read, written, or modified in a System of Record (SOR)
Data at this layer resides in **existing applications** or **databases**.

Integration services

Provide integration between and access to existing applications.
Involves the **transformation of data** and functions from whats desired at the **business service level** to what is actually possible in the **existing systems**.

Architectural element of SOA

Business services

provide high-level business functionality throughout the Enterprise.
Provides a **service interface abstraction** and integration of the layer below, breaking the direct dependence between processes and existing systems
Describe the information that should be **passed** and **shared** between services.

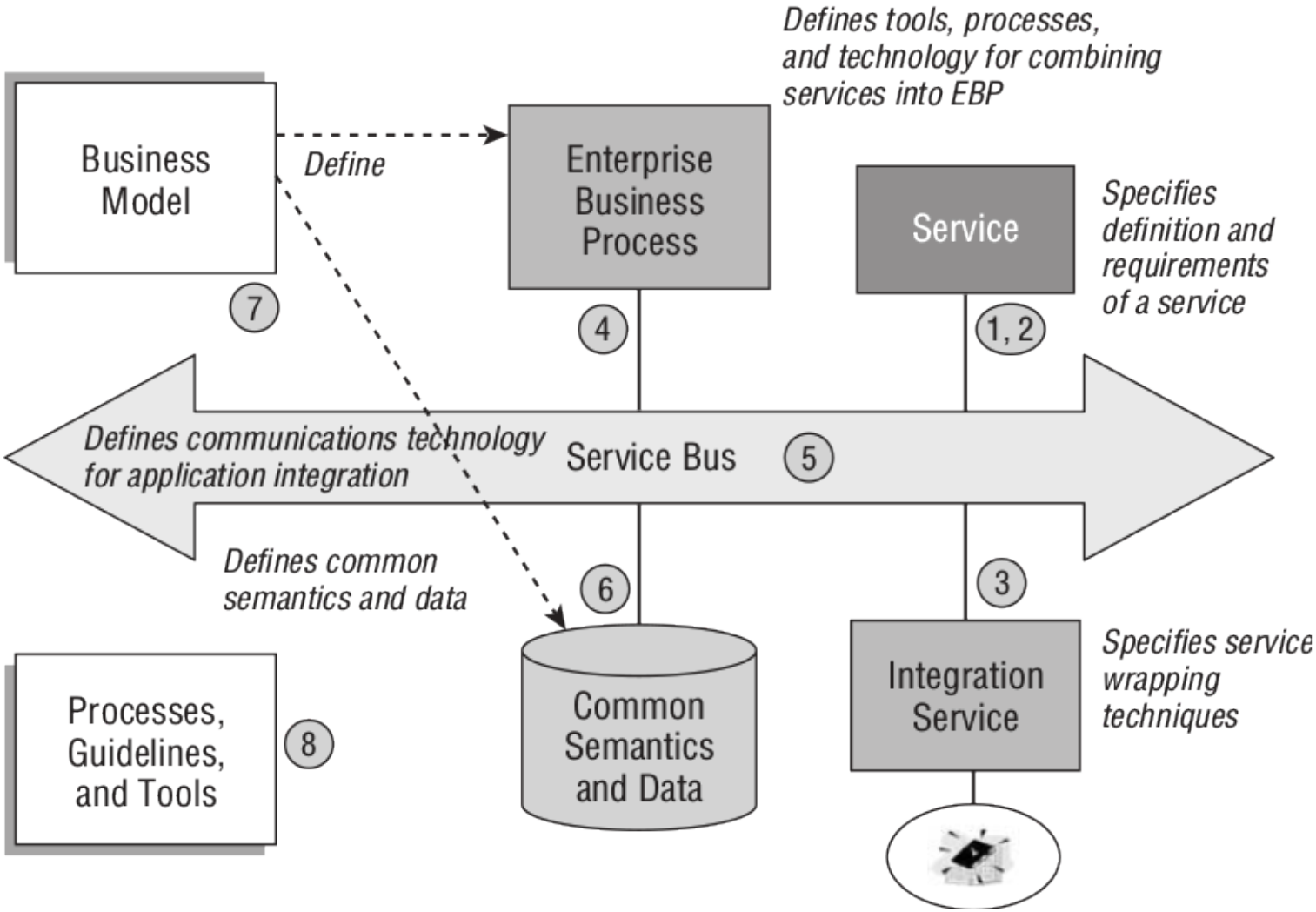
Business Process

Consists of a series of operations that are executed in an ordered sequence according to a set of **business rules**
Business processes provide long-running sets of actions or activities described using BPMN and executed by BPMS.

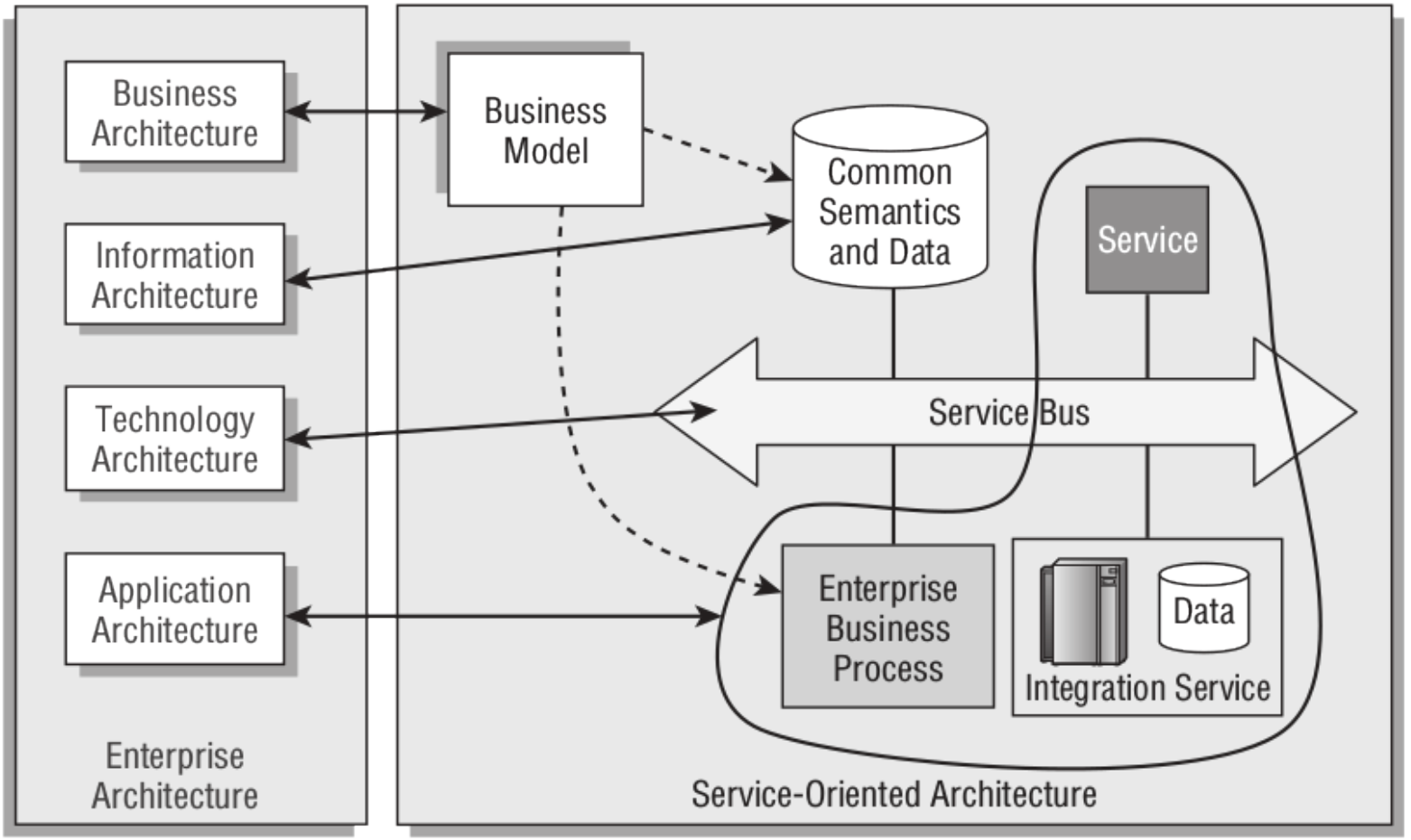
How services should be built and used?

- Granularity The appropriate **size** of the service.
- Type or style of interface Guidelines for interface design.
- Configuration mechanisms Standard mechanisms for configuring services should be defined.
- Other artifacts support the service with design models and specifications, **documentation**, test plans, and so on.
- Associated information Additional information that should be part of a service to support run-time and design-time inspection, such as the **version, author, date, keywords**, and so on.
- Dependency management and other patterns Specific **design patterns** that should be followed to keep services independent and reusable.

Perispective of SOA



Enterprise Architecture and SOA



Business-Driven SOA

Web Services provide a convenient **technology** for the infrastructure of services.

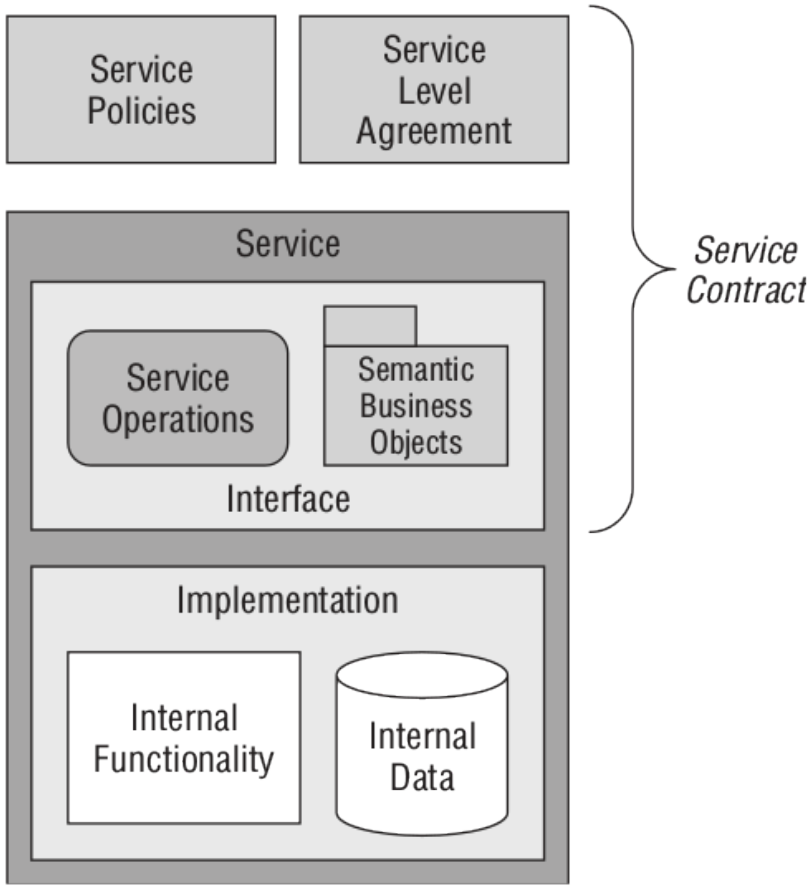
Business services **designed to exchange business documents** written in Extensible Markup Language (XML).

Business Process Management (BPM) provides a perfect solution and complement to the implementation of services and SOA

Business Process Execution Language (BPEL) is designed to work explicitly with Web Services and provide **coordination and integration** of Web Services into higher-level business services.
The "LEGO" analogy is often used to describe the service-oriented enterprise.

What is a web services?

Discrete unit of business functionality that is made available through a service contract.

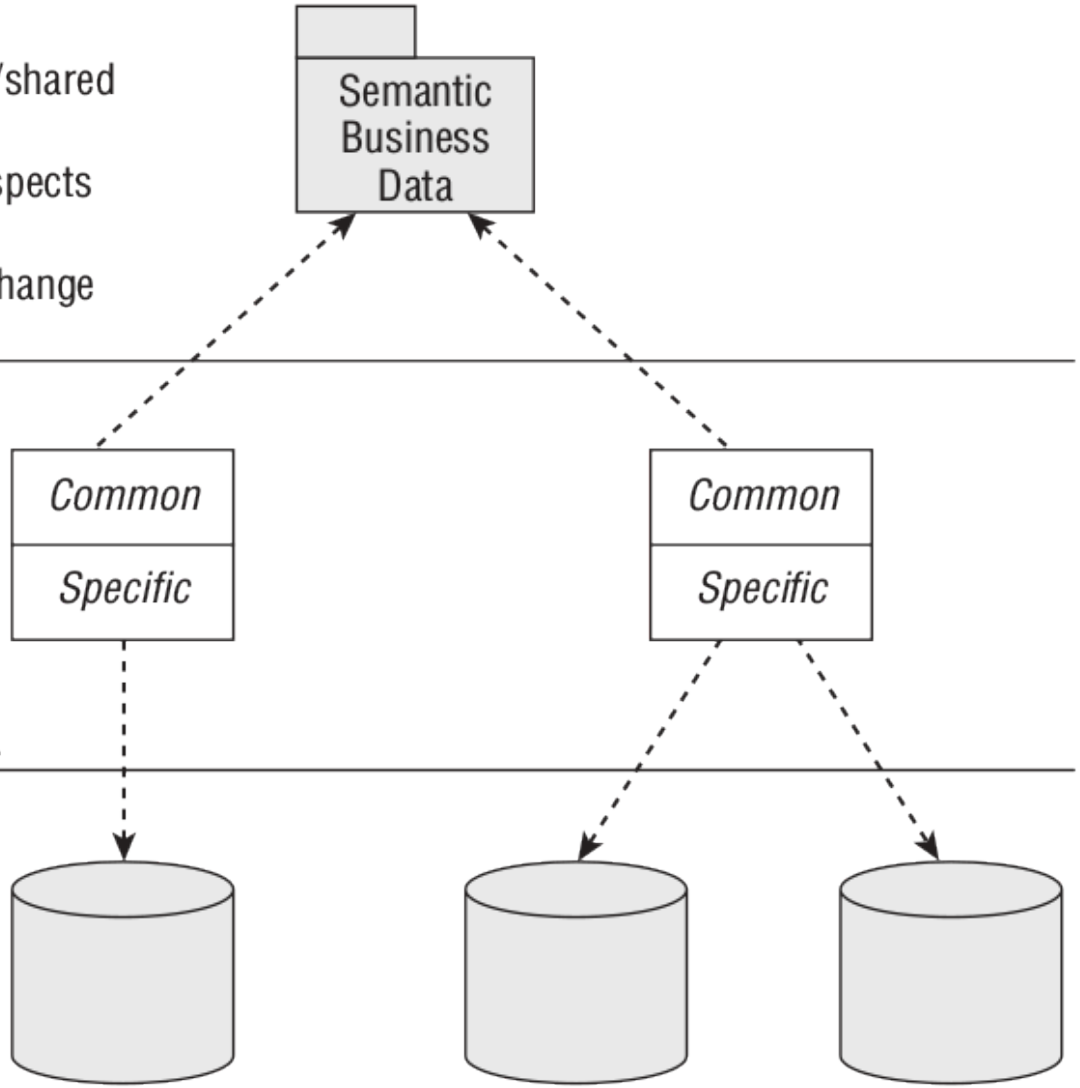


Data in business process

Semantic Data:
- Described by a common/shared information model.
- A view of the common aspects of services.
- Used for information exchange through interfaces.

Domain Data:
- Described by an internal data model.
- A view of the physical data.
- Used for implementation.

Physical Data:
- Described by a database schema.
- Used for persistence.



Services characteristics

Granularity

Business processes are decomposed into modular services that are self-contained.
Coarse-grained services provide a greater level of functionality within a single service operation (reduce network overhead).
Fine-grained service operations provide the exchange of small amounts of information to complete a specific discrete task.

Encapsulation

Hides the service's internal implementation details and data structures from the published interface operations and semantic model. Service interface (what a service does); service implementation (how it is done)

Services characteristics

Loose coupling

describes the **number of dependencies** between a service consumer and provider
The degree of coupling directly **affects the flexibility** and extensibility of a system

Isolation of responsibilities

Services are **responsible for discrete tasks** or the management of specific resources.

Autonomy

Is the characteristic that allows services to be **deployed, modified, and maintained** independently from each other. An autonomous services **life cycle is independent** of other services.

Services characteristics

Reuse

In other words, services are **shared and reused** as building blocks in the construction of processes or composite services.

Dynamic discovery and binding

Services can be **discovered** at design time through the use of a design-time service repository or at run-time using the same repository and binding dynamically.

Stateless

The service **neither remember the last thing** they were asked to do nor care what the next is.
Stateless services provide better flexibility, scalability, and reliability.

Services characteristics

Self-describing

The service contract provides a complete description of the service **interface**, its **operations**, the **input** and **output parameters**,and **schema**

Composable

Services can be **composed from other services** in order to create new services or new business process.

Governed by policy

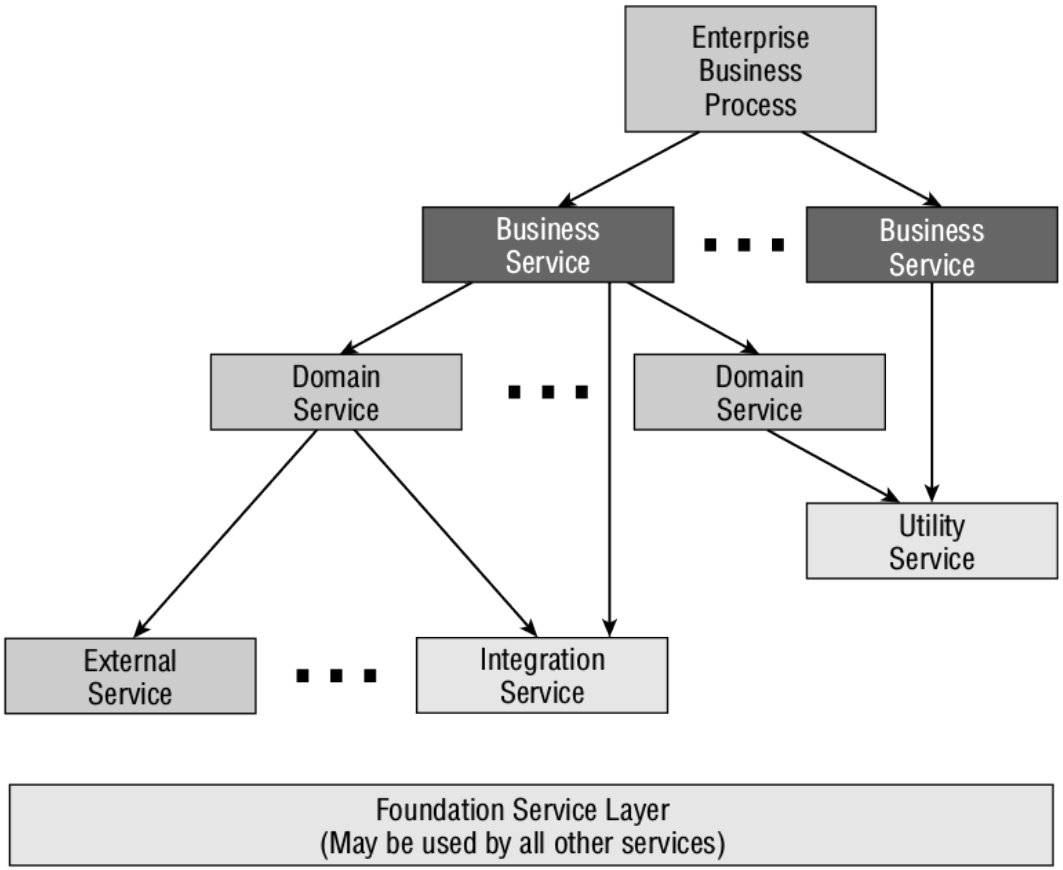
Policies and service level agreements (SLAs) describe how different consumers are allowed to interact with the service.

Independent of location, language, and protocol

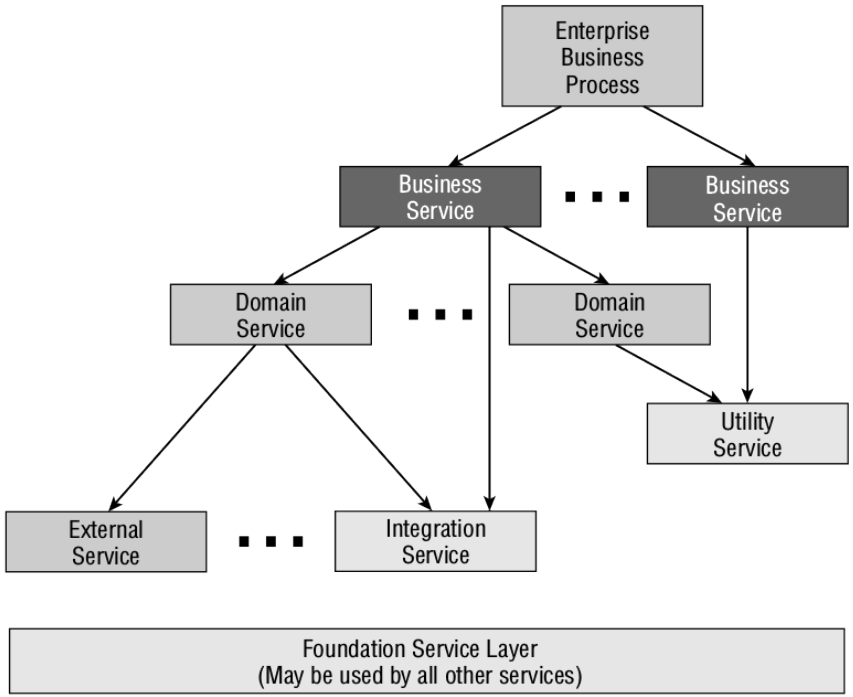
Designed to be location-transparent and protocol/platform independent. They are accessible to any authorized user, on any platform, from any location

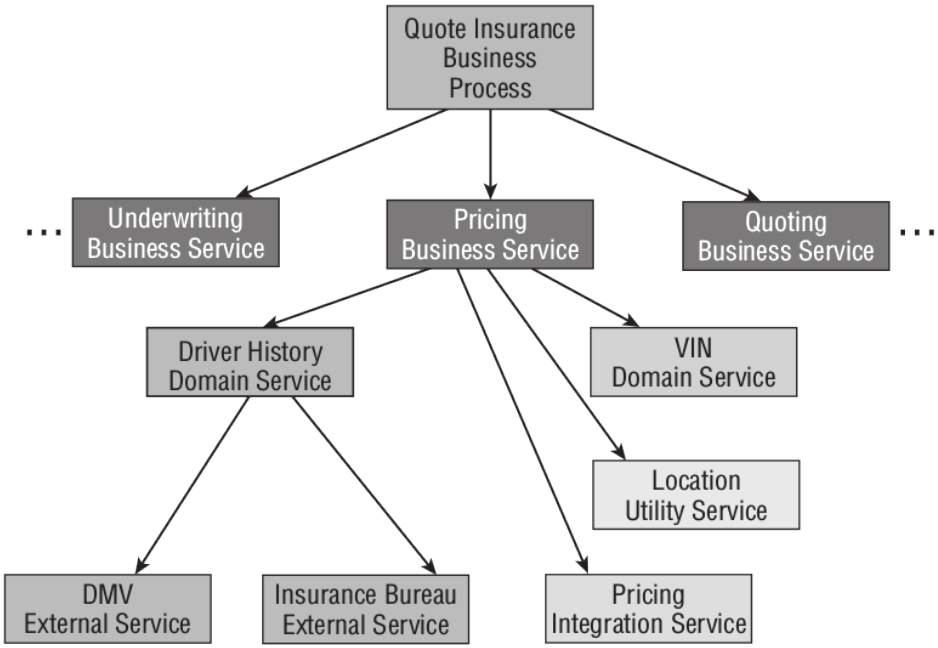
Granularity

Granularity describes the size of a service. This doesnt mean size in terms of kilobytes of code. It means the **amount of business function that is performed in a single request/response exchange of messages**.



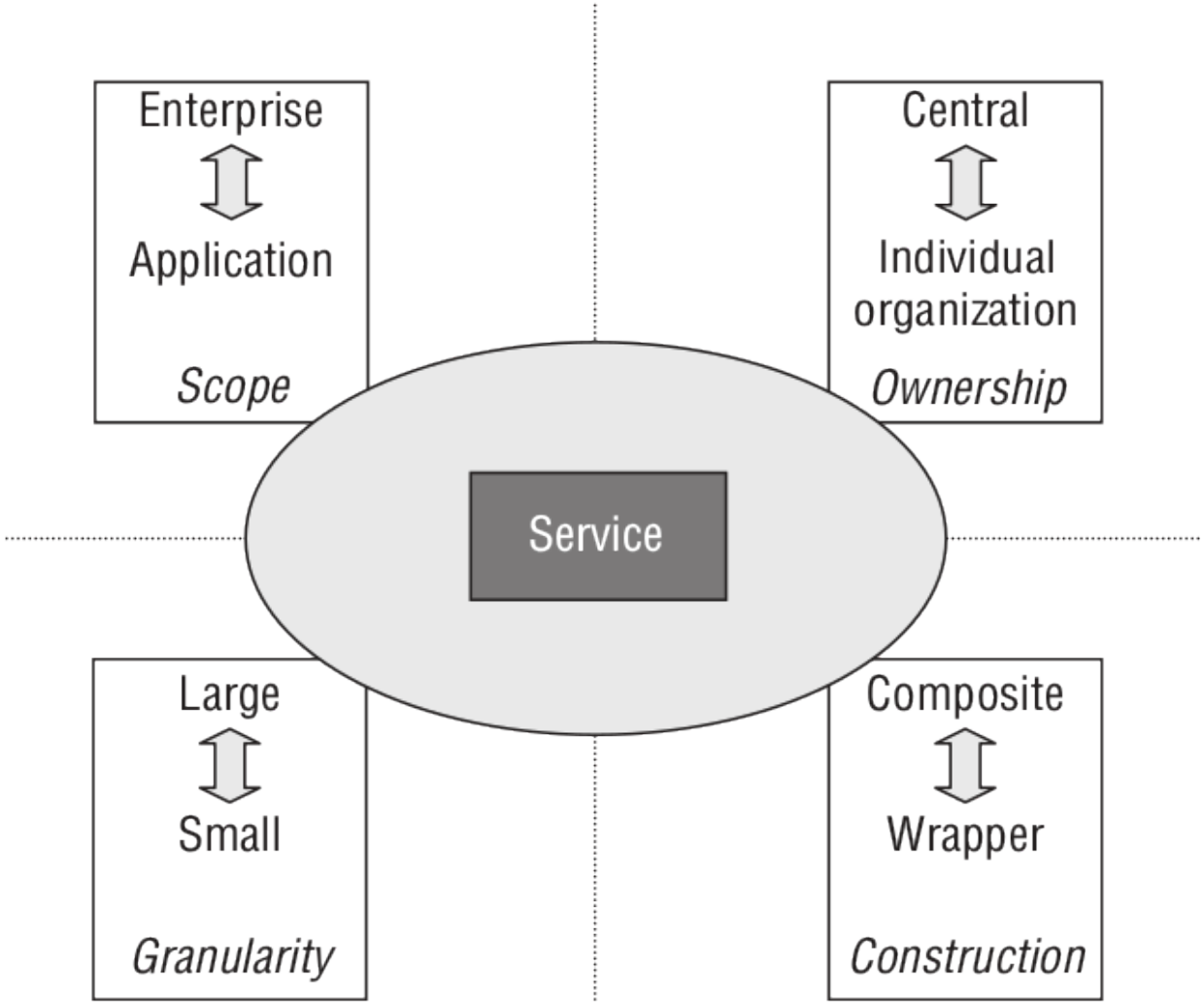
Granularity an example





Service dimension

Size is not the only important characteristic that determines how a service is used.



Service dimension

Scope

defines the organizational **boundaries** that the service is expected to operate (limit the scope of service increase the liability)

Ownership

defines the organizational unit that is responsible for support a services

Granularity

amount of business function performed in a single request/response session.

Construction

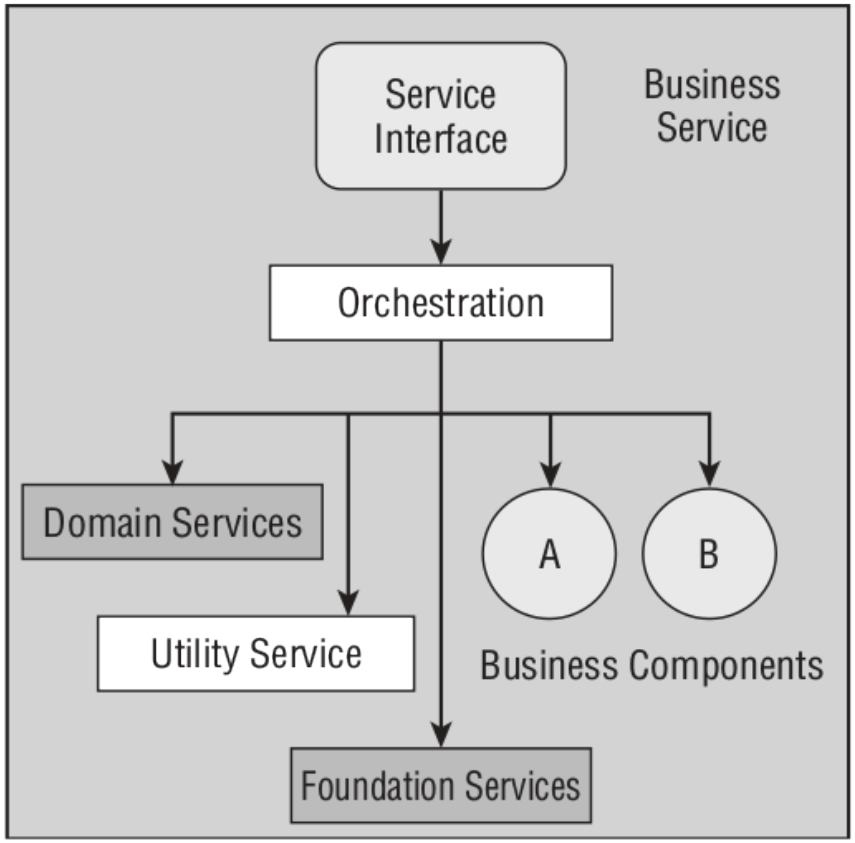
refer how the service has been implemented (composite, atomic, foundation, business, domain).

Metrics for loose coupling

Independent systems (low coupling) but concentration of responsibilities on the same class (high cohesion)

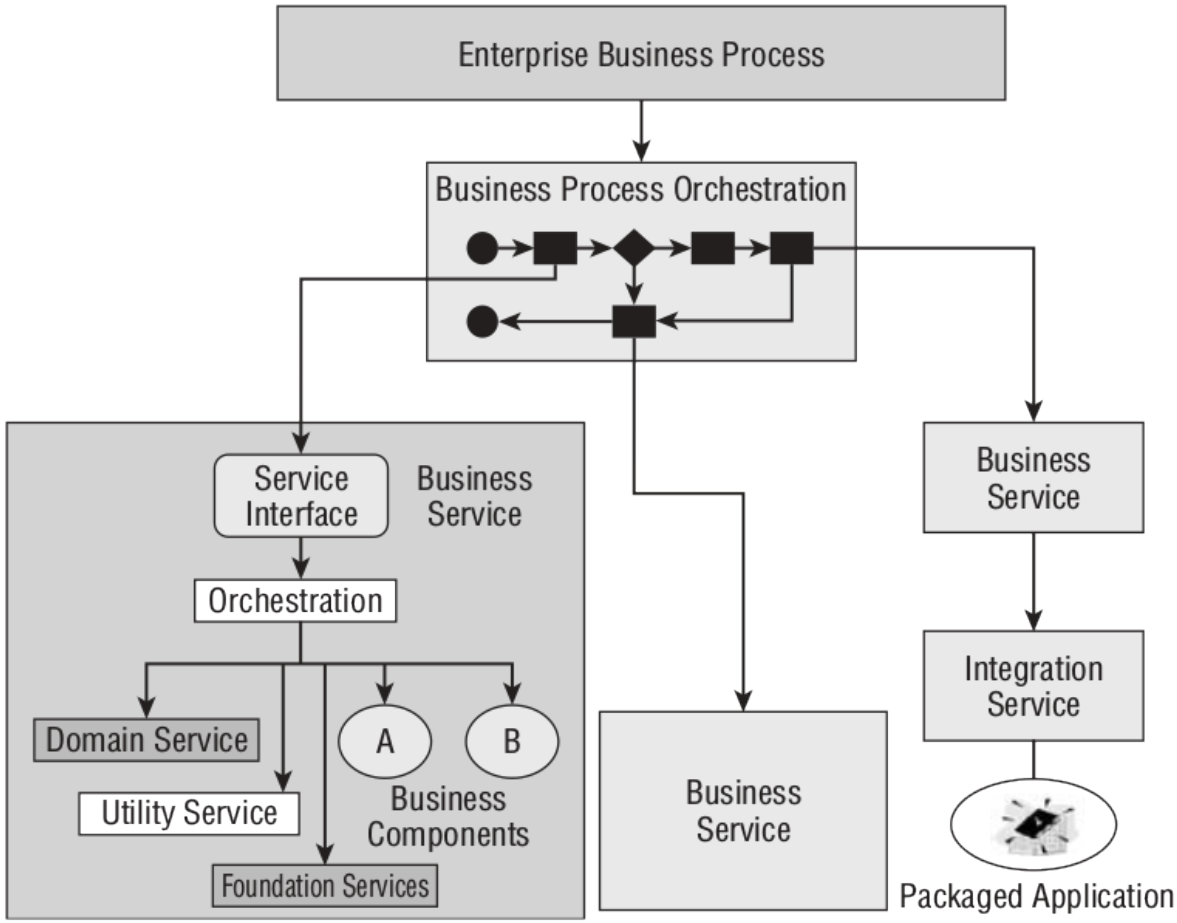
- **Location transparency**: service registry
- **Interface and implementation**: decoupling of interface and implementation
- **Data**: public view of the data (the semantic data); the internal view (the domain data)
- **Versioning**: minor enhancement and major enhancement; backward compatibility
- **Interoperability and platform independence**:communication mechanism cross-plattform
- **Usage, assumptions, and knowledge**: no assumptions on the purpose, or technical or business characteristics, of the service consumer.

Service Patterns



Orchestration is used to define how these **different services are combined** to form the business service

Service Patterns



the systems do not already conveniently **provide their functionality** as services, so they are wrapped by an integration service.

Questions?