



Pontificia Universidad  
**JAVERIANA**  
Bogotá

ESPECIALIZACIÓN EN  
ARQUITECTURA EMPRESARIAL DE SOFTWARE

# Patrones de Software

Juan Carlos Cerón Barreto

[ceron.juan@javeriana.edu.co](mailto:ceron.juan@javeriana.edu.co)



# SOA Patterns

- Foundational Inventory Patterns
- Logical Inventory Layer Patterns
- Inventory Centralization Patterns
- Inventory Implementation Patterns
- Inventory Governance Patterns
- Foundational Service Patterns
- Service Implementation Patterns
- Service Security Patterns

# SOA Patterns

- Service Contract Design Patterns
- Legacy Encapsulation Patterns
- Service Governance Patterns
- Capability Composition Patterns
- Service Messaging Patterns
- Composition Implementation Patterns
- Service Interaction Security Patterns
- Transformation Patterns
- REST-inspired Patterns



# Foundational Inventory

Canonical  
Schema

Domain  
Inventory

Enterprise  
Inventory

Service  
Layers

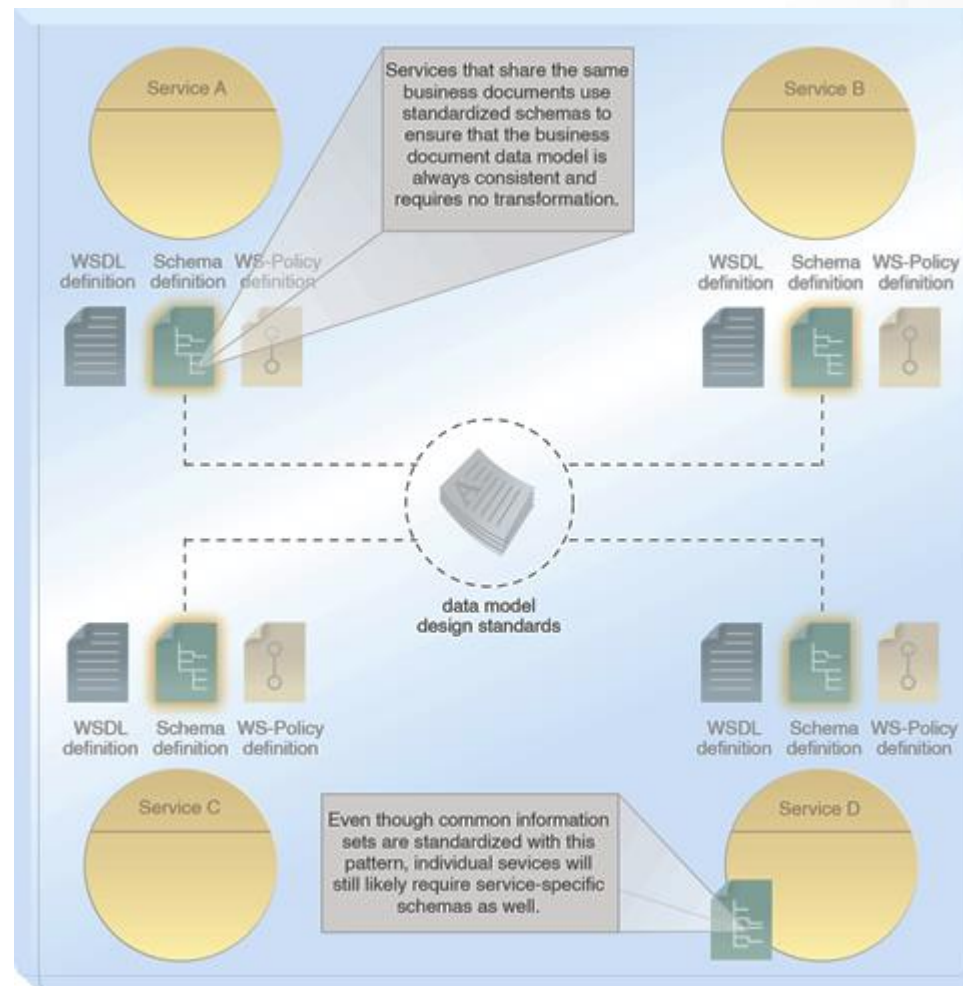


# Canonical Schema

- Como se pueden diseñar los servicios para evitar transformación de datos?
- Los modelos de datos para conjuntos de información comunes están estandarizados a través de contratos de servicio dentro de un límite de inventario.

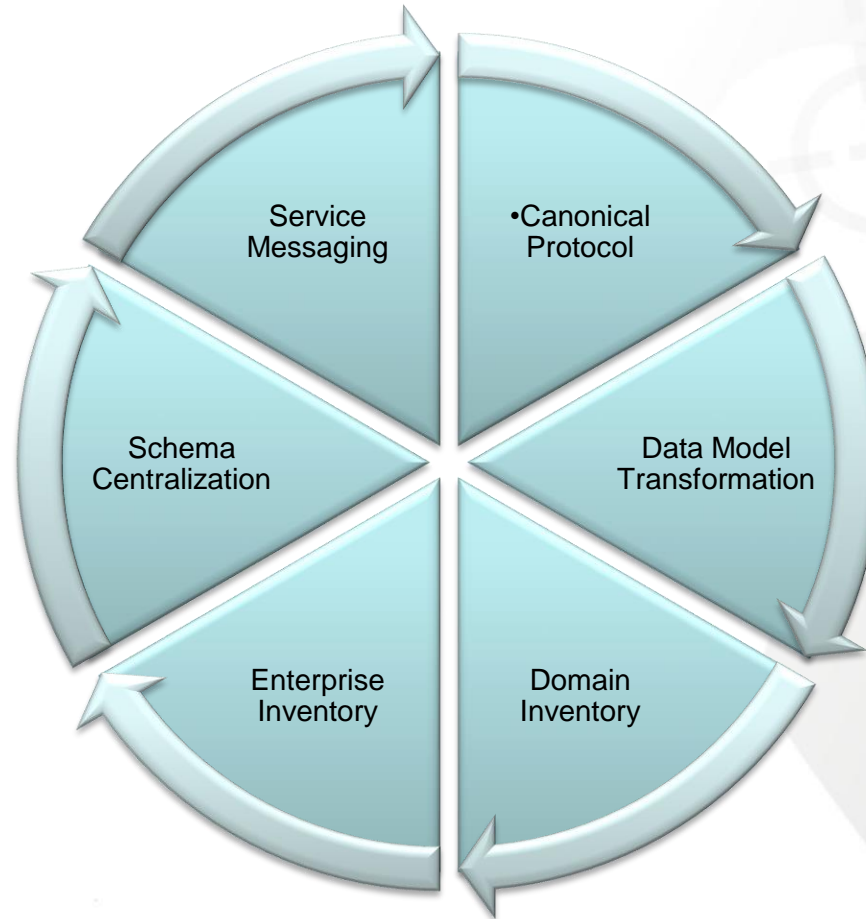


# Canonical Schema





# Canonical Schema





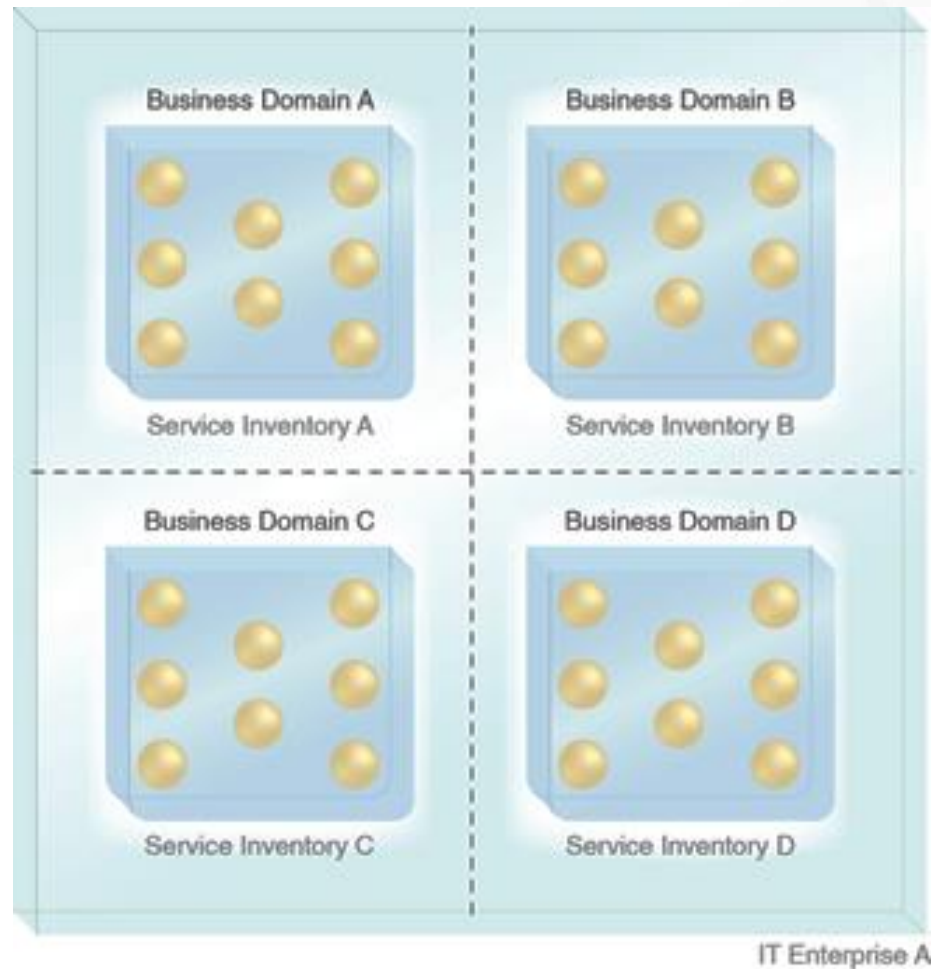
# Domain Inventory

- Como se pueden entregar los servicios para habilitar la recomposición cuando la estandarización de la empresa no es posible?
- Los servicios se pueden agrupar en inventarios de servicios específicos del dominio los cuales pueden ser estandarizados y gobernados de forma independiente





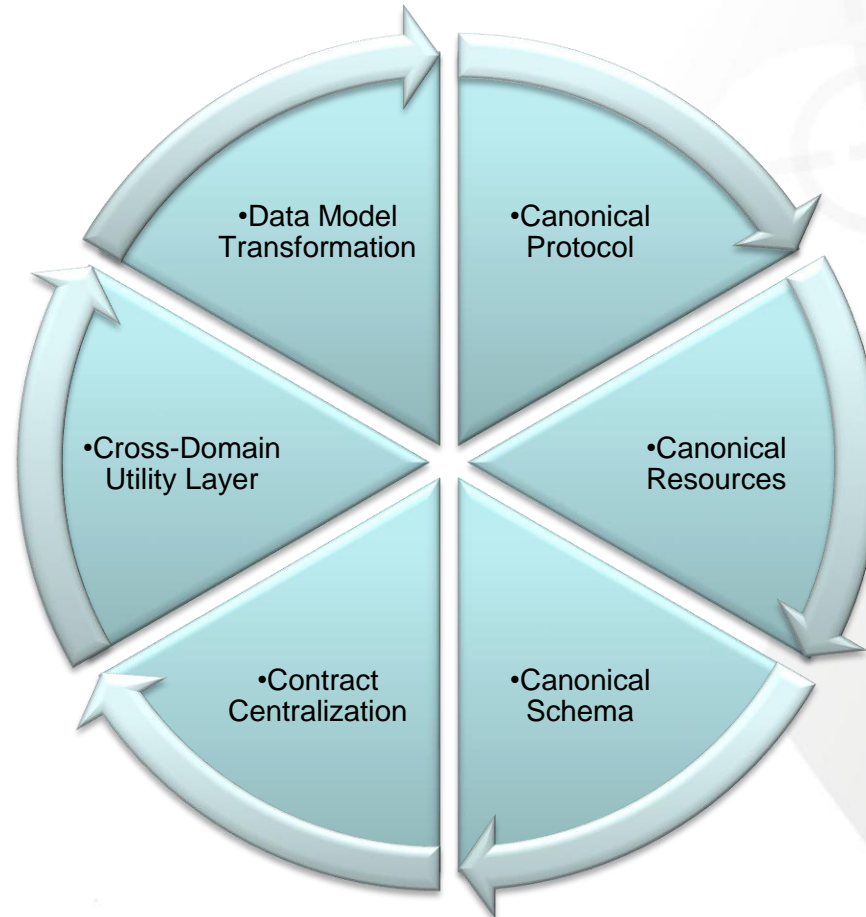
# Domain Inventory





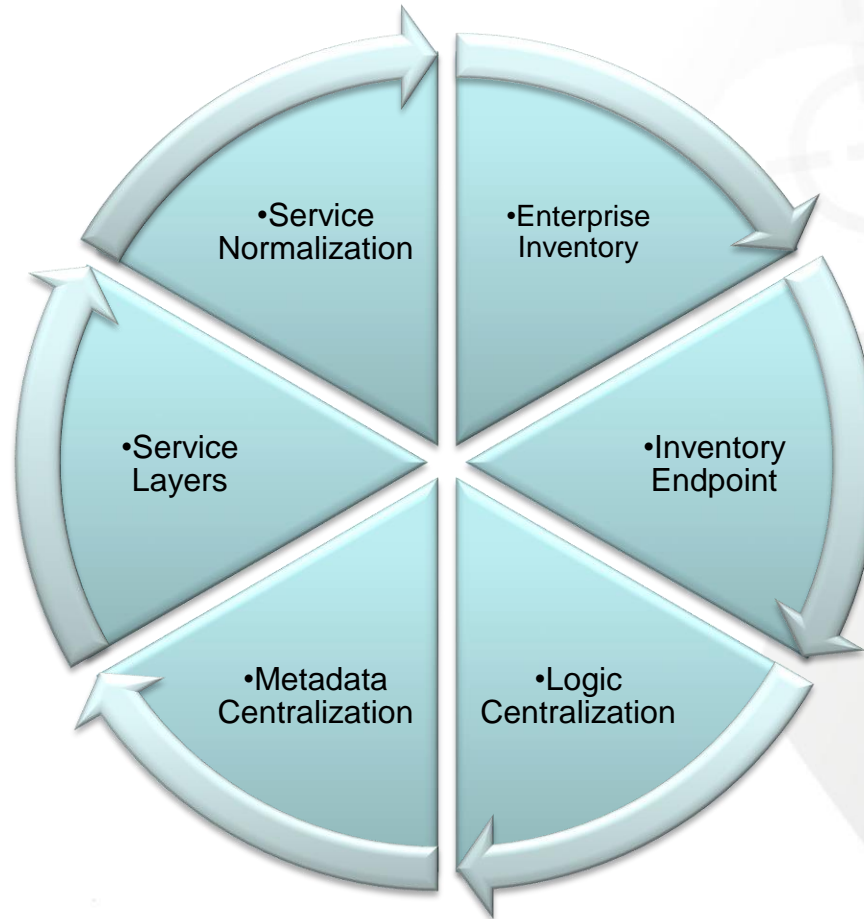


# Domain Inventory





# Domain Inventory





# Enterprise Inventory

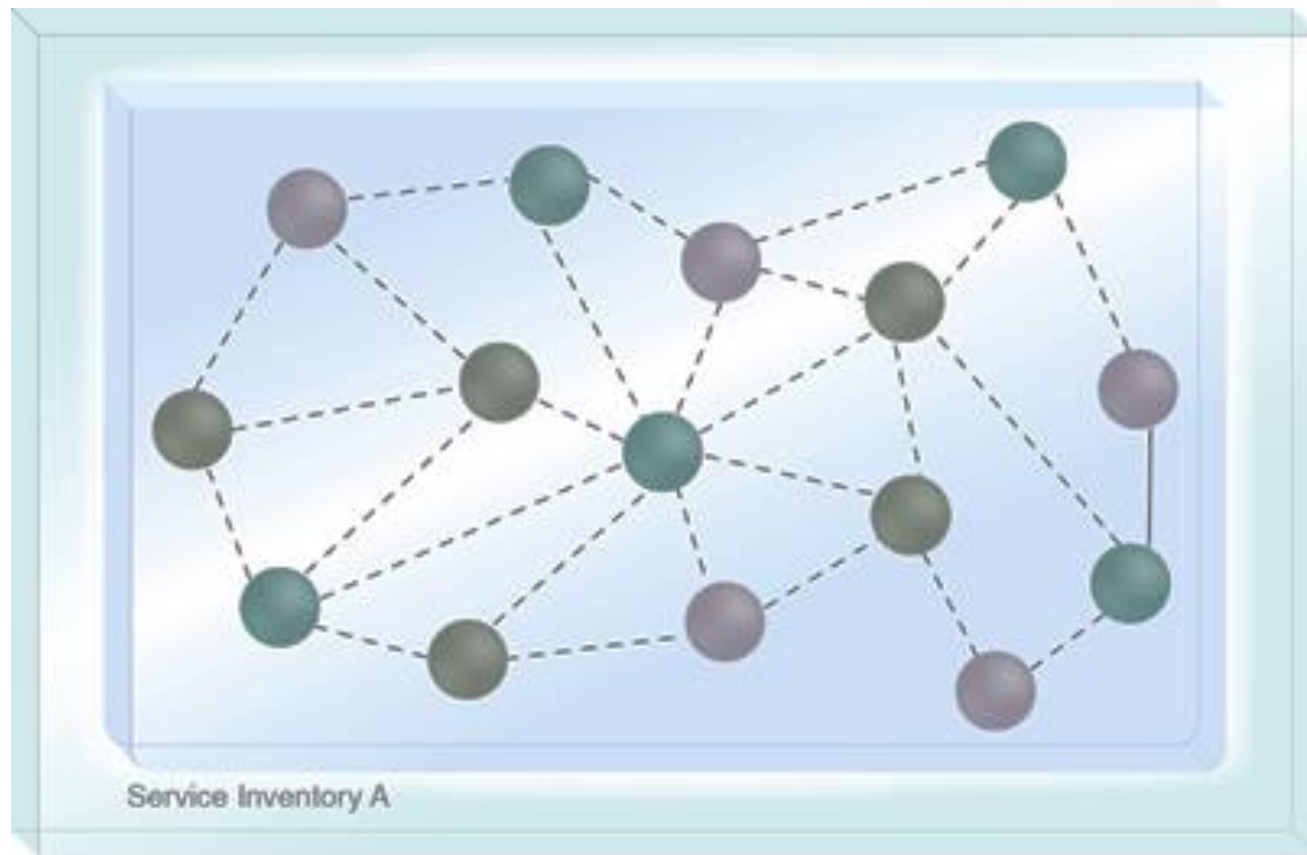
- Como se pueden entregar los servicios para maximizar la la recomposición?
- Los servicios para múltiples soluciones se pueden diseñar para su entrega dentro de una arquitectura de inventario estandarizada para toda la empresa



Pontificia Universidad  
**JAVERIANA**  
Bogotá

ESPECIALIZACIÓN EN  
ARQUITECTURA EMPRESARIAL DE SOFTWARE

# Enterprise Inventory

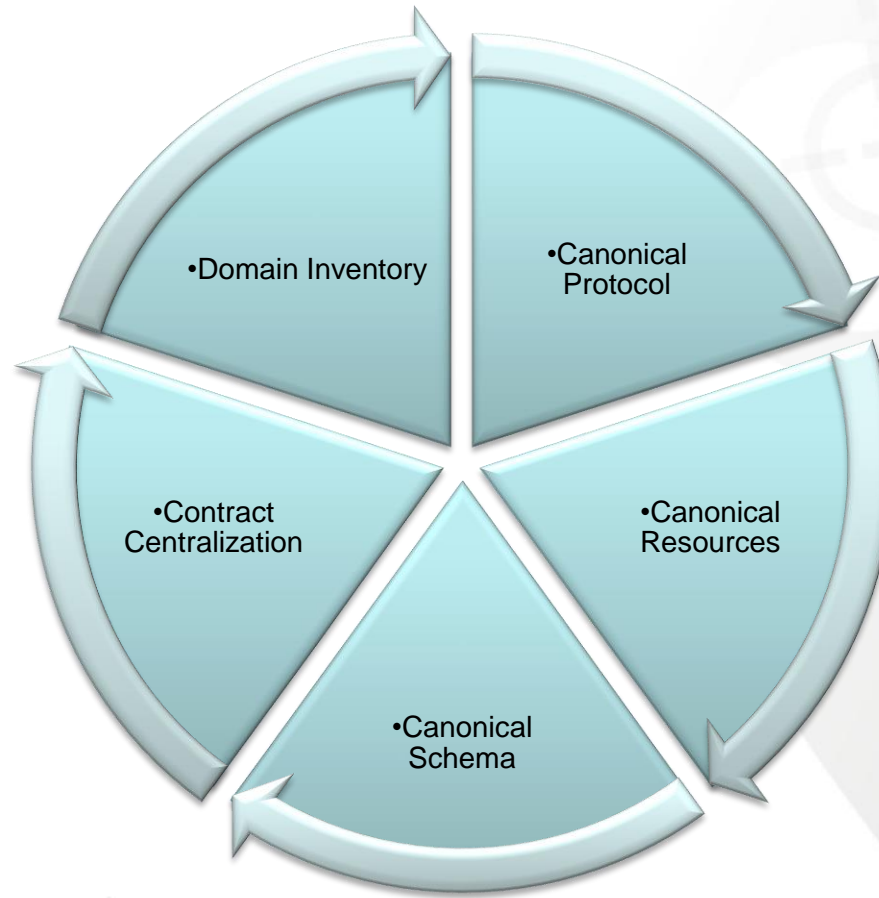


Service Inventory A

IT Enterprise A

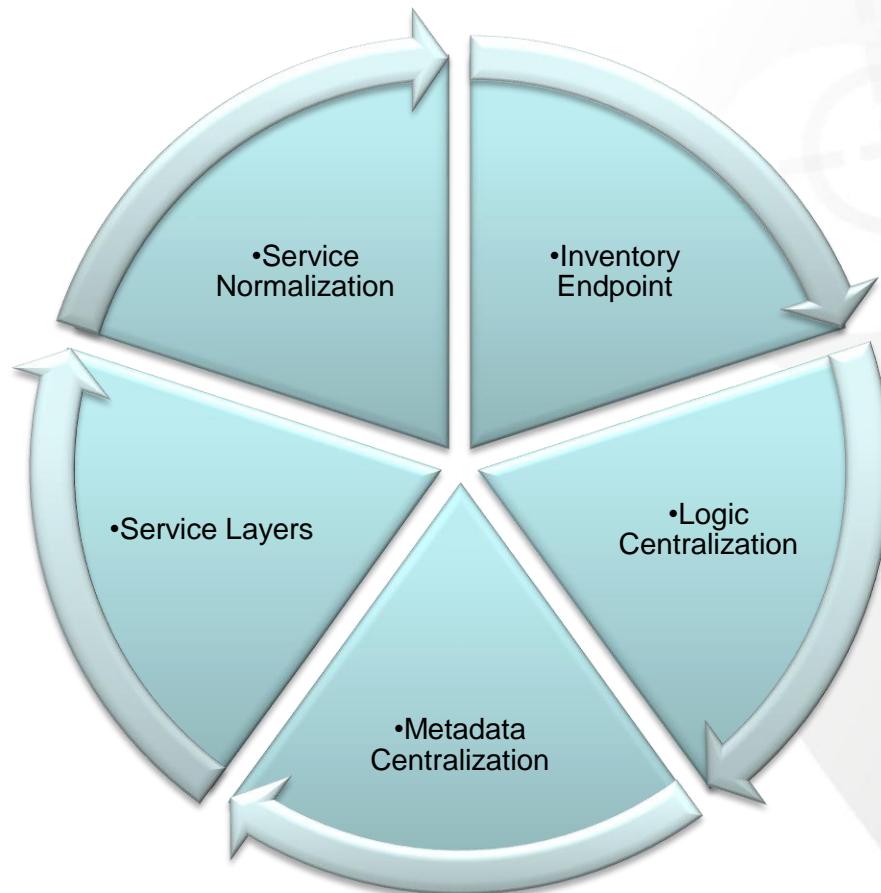


# Enterprise Inventory





# Enterprise Inventory





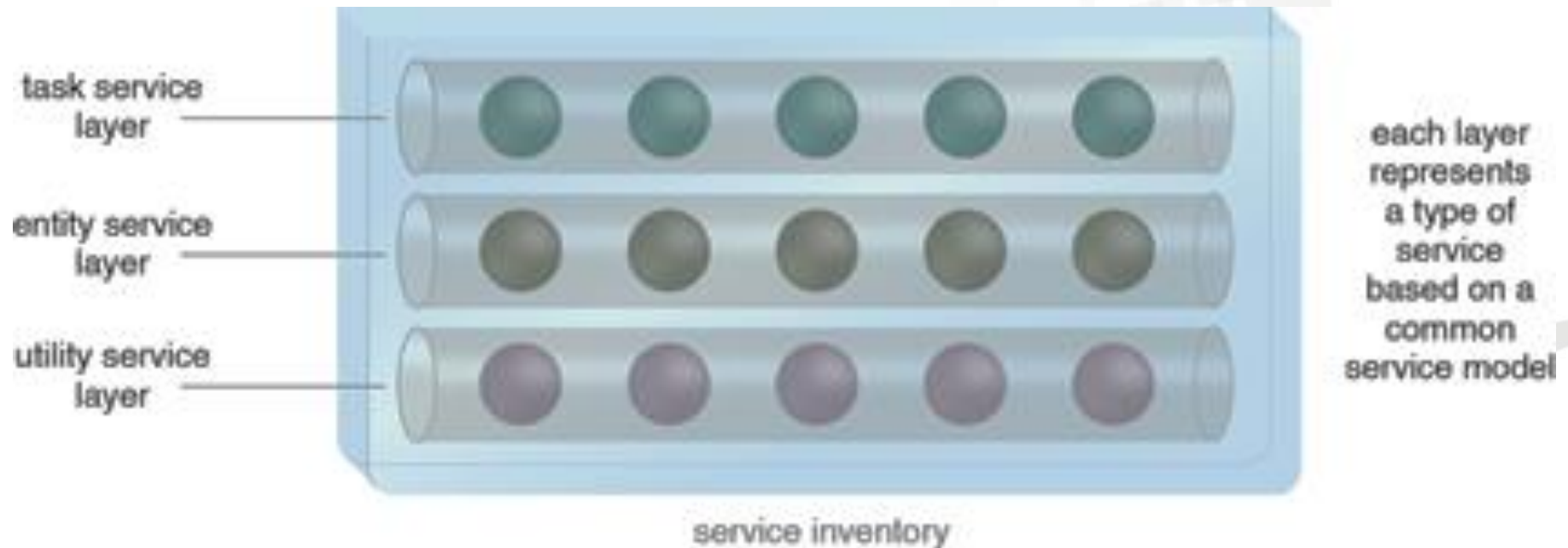


# Service Layer

- ¿Cómo se pueden organizar los servicios en un inventario en base a la funcionalidad común?
- El inventario está estructurado en dos o más capas de servicio lógicas, cada una de las cuales es responsable de abstraer la lógica basada en un tipo funcional común.

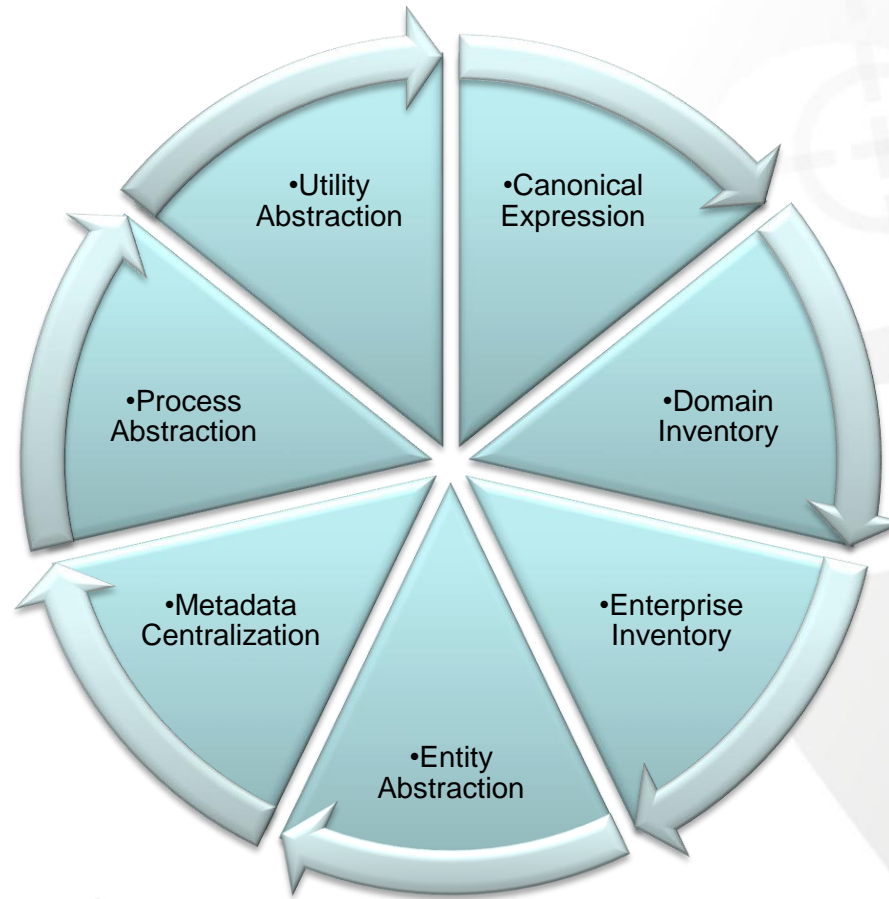


# Service Layer



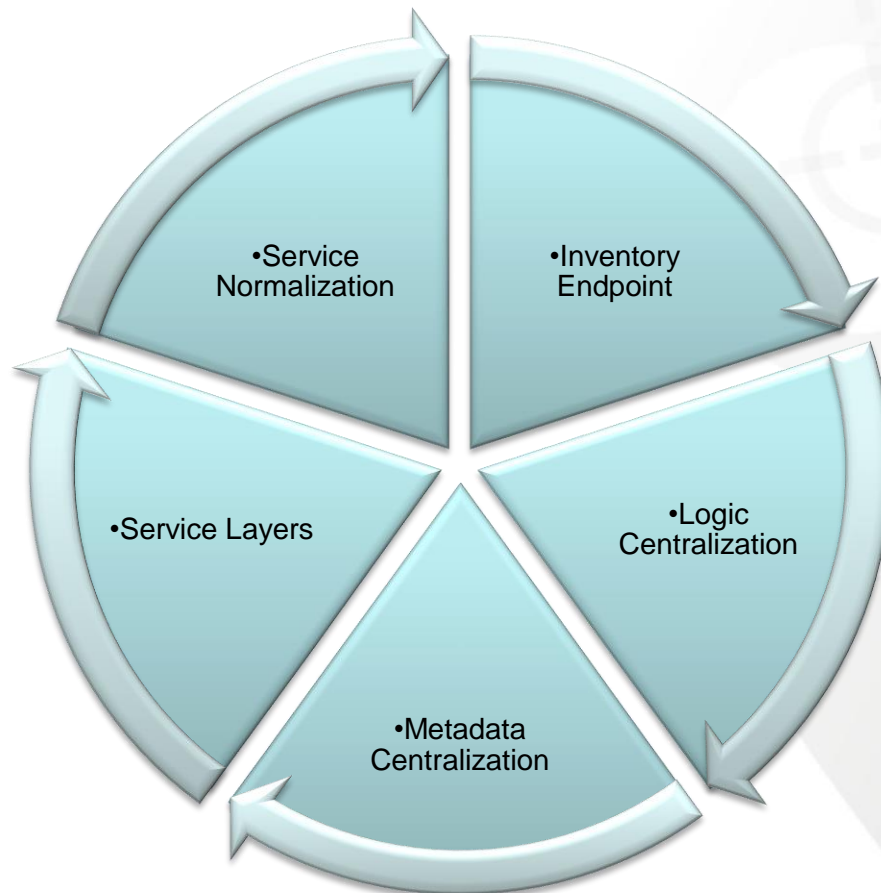


# Service Layer





# Enterprise Inventory





# Inventory Centralization Patterns

Policy  
Centralization

Process  
Centralization

Rules  
Centralization

Schema  
Centralization



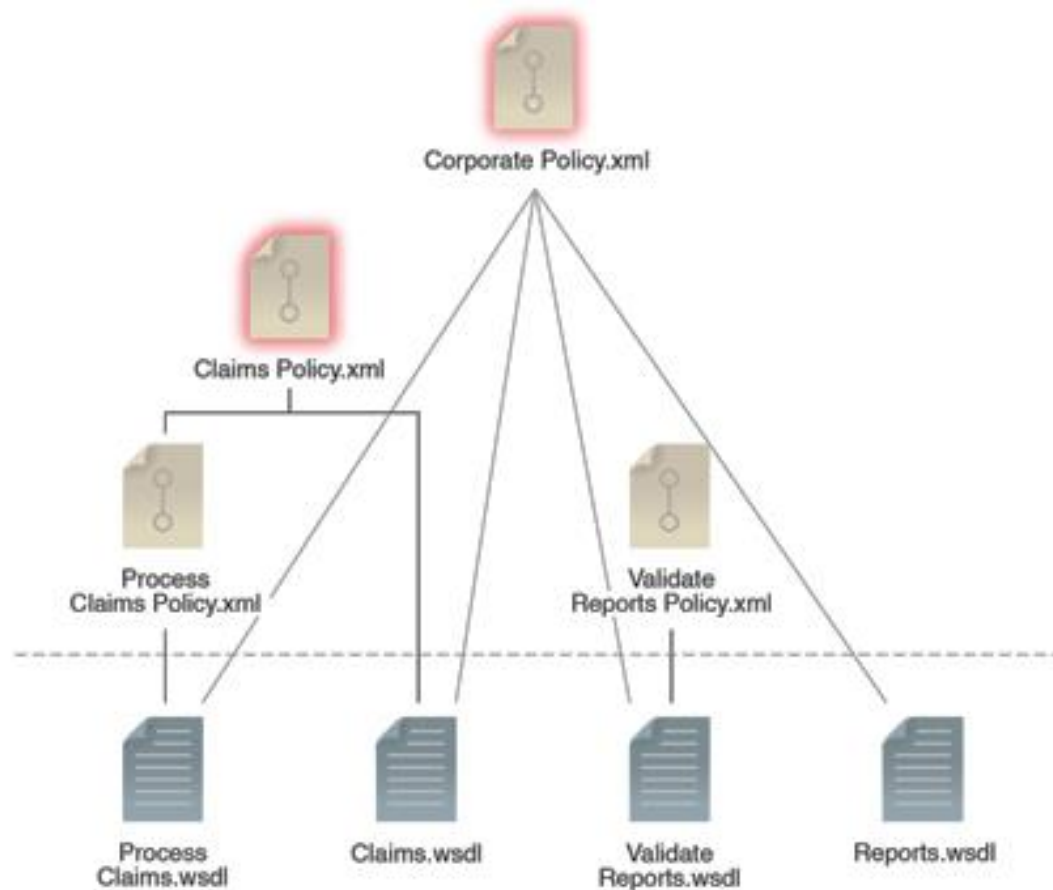
# Policy Centralization

- ¿Cómo se puede normalizar las políticas y aplicarlas en múltiples servicios?
- Las políticas globales o específicas de dominio se pueden aislar y aplicar a múltiples servicios.



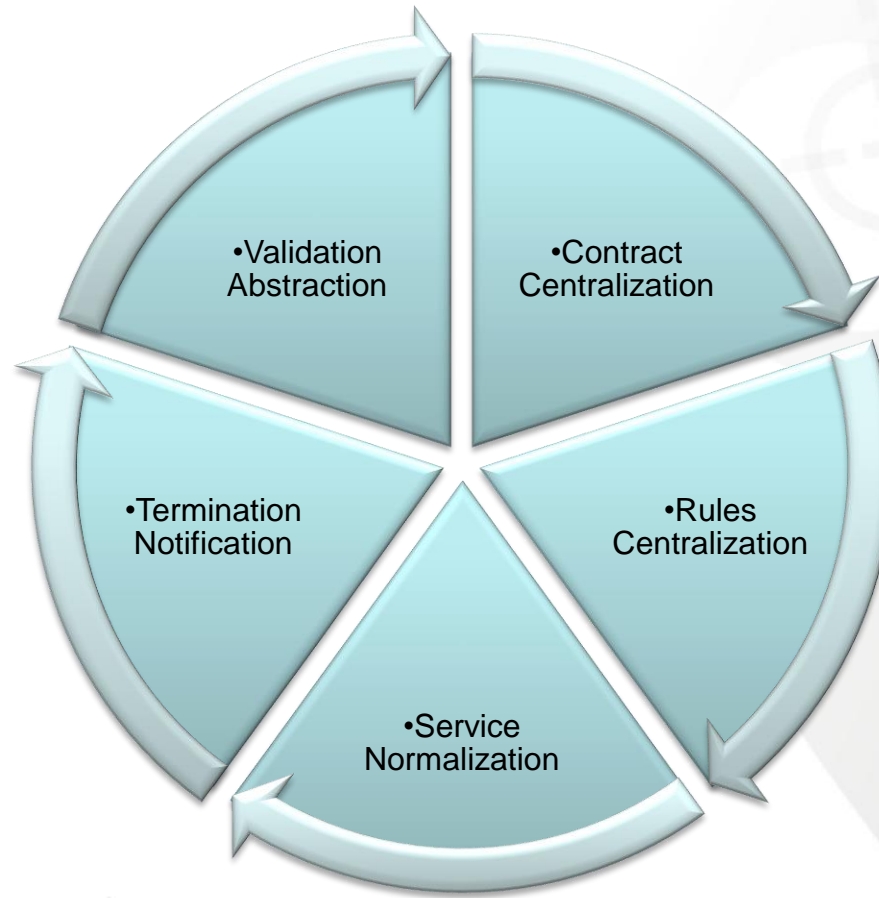


# Policy Centralization





# Policy Centralization



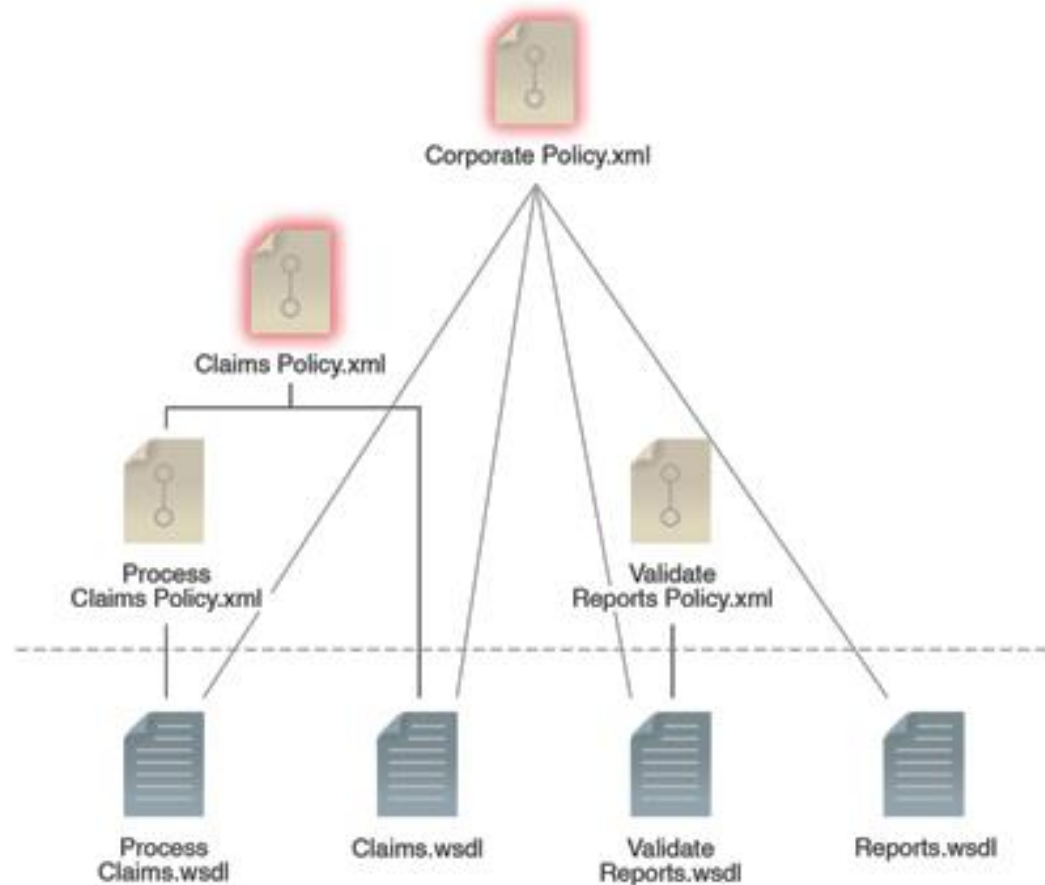


# Process Centralization

- ¿Cómo se puede normalizar las políticas y aplicarlas en múltiples servicios?
- Las políticas globales o específicas de dominio se pueden aislar y aplicar a múltiples servicios.

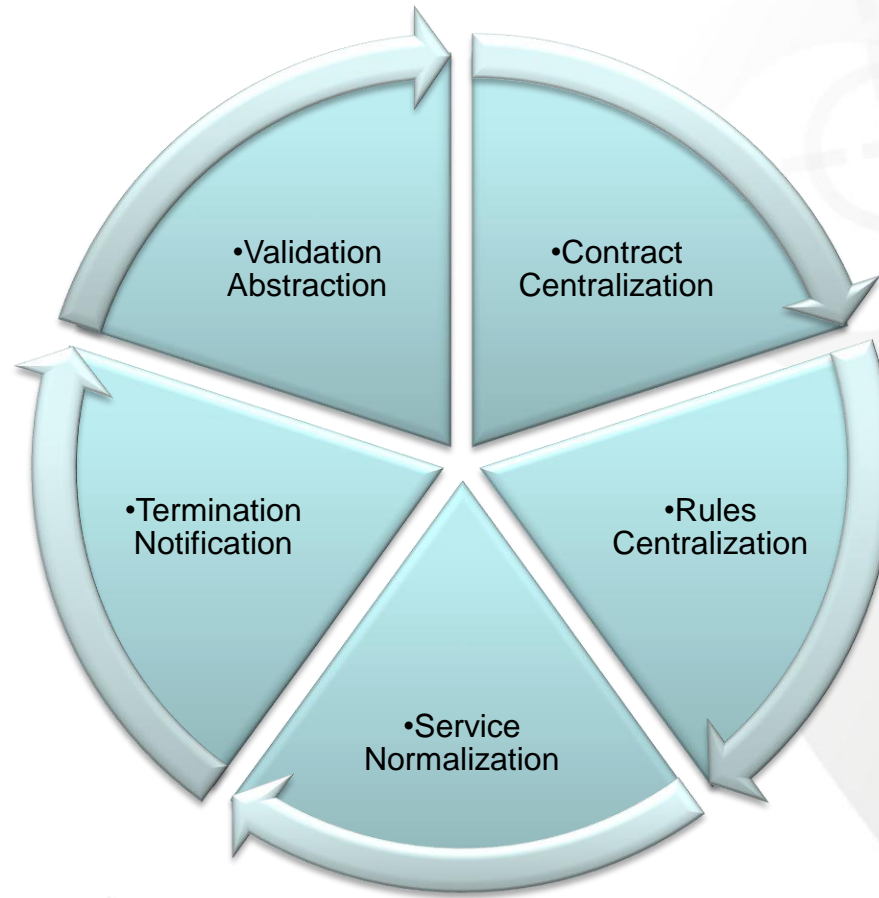


# Process Centralization





# Process Centralization





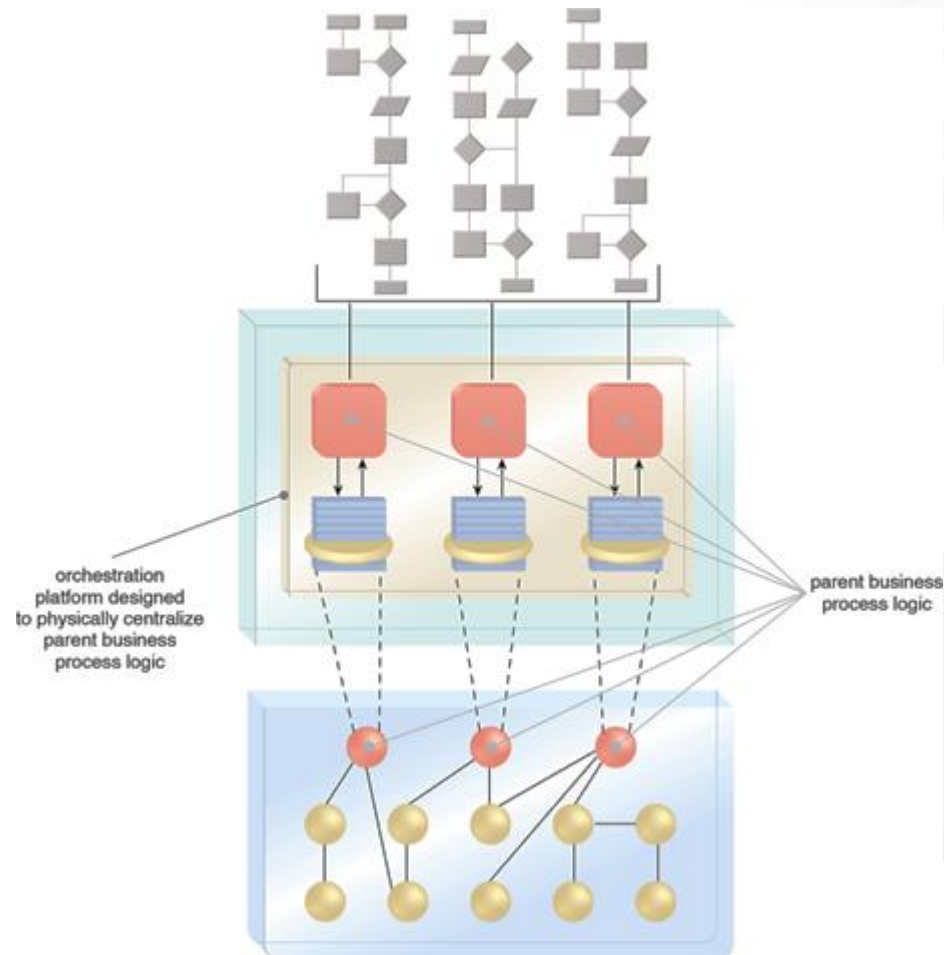
# Rules Centralization

- ¿Cómo se puede gobernar de manera centralizada la lógica del proceso?
- La lógica que representan procesos de negocio se pueden implementar y gobernar centralmente



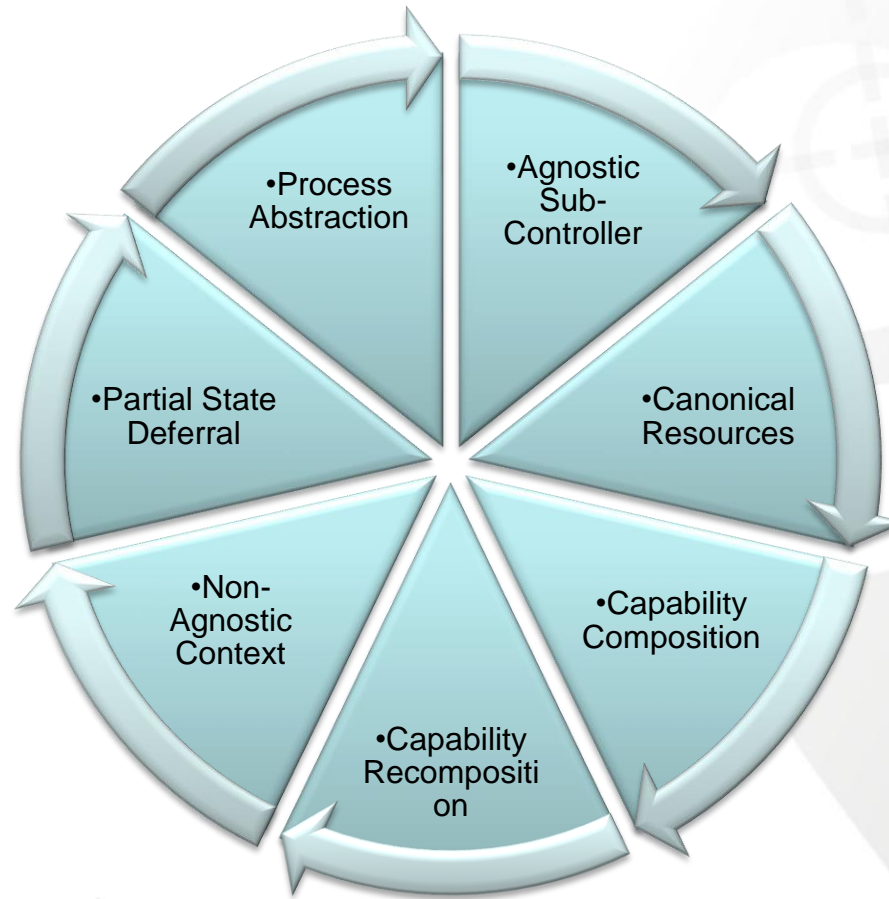


# Rules Centralization





# Rules Centralization





# Schema Centralization

- ¿Cómo se pueden diseñar los contratos de servicio para evitar la representación redundante de datos?
- Seleccione esquemas que existen como partes físicamente separadas del contrato de servicio que se comparten a través de contratos múltiples

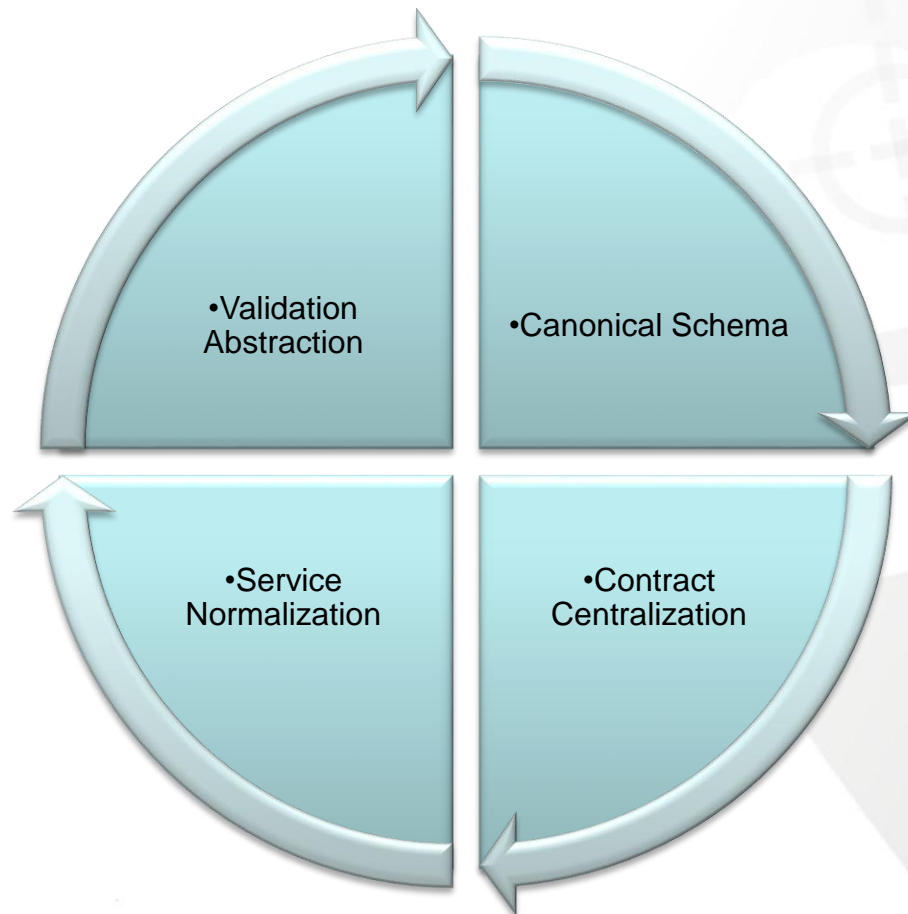


# Schema Centralization





# Schema Centralization





# Service Security Patterns

Exception  
Shielding

Message  
Screening

Trusted  
Subsystem



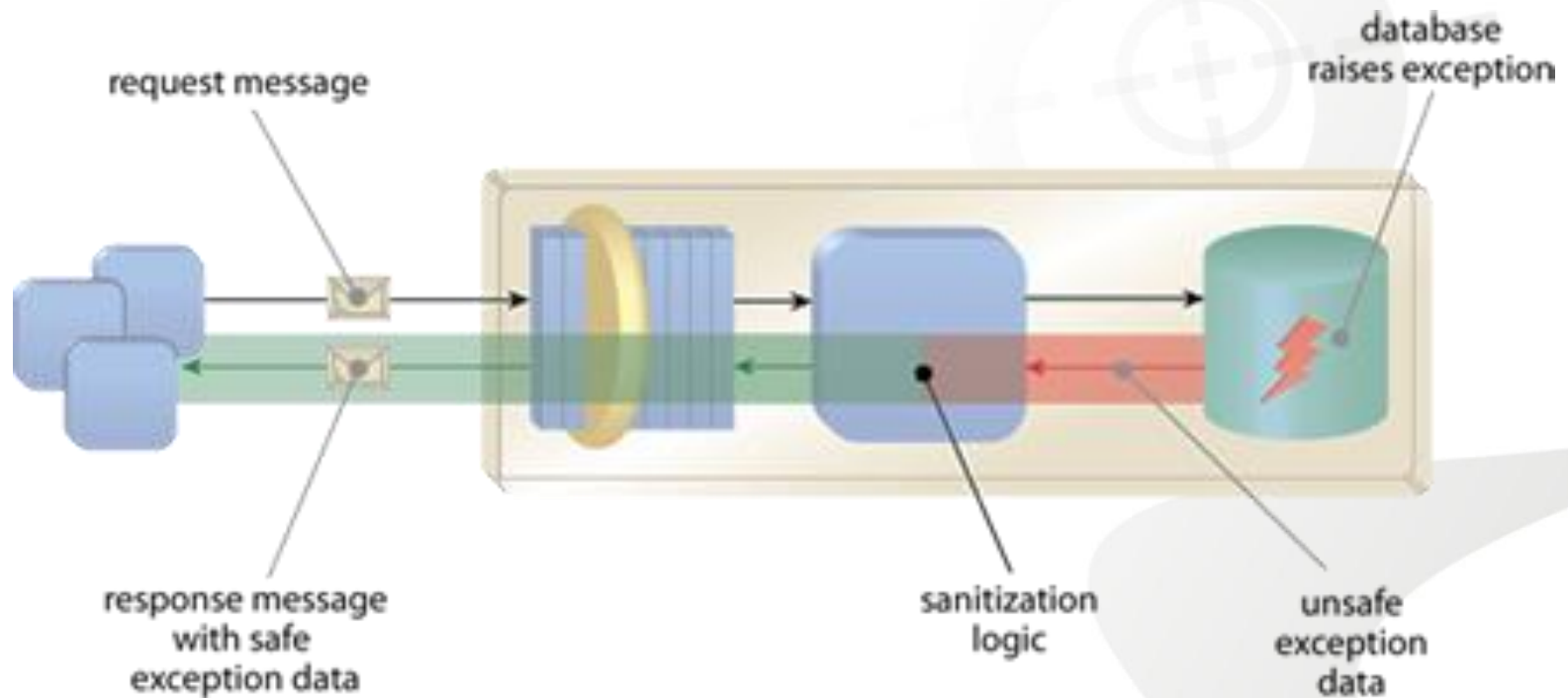


# Exception Shielding

- ¿Cómo puede un servicio evitar la divulgación de información sobre su implementación interna cuando se produce una excepción?
- Los datos de excepción potencialmente inseguros deben ser controlados y reemplazados con datos de excepción que son seguros antes de que estén disponibles para los consumidores.



# Exception Shielding





# Exception Shielding



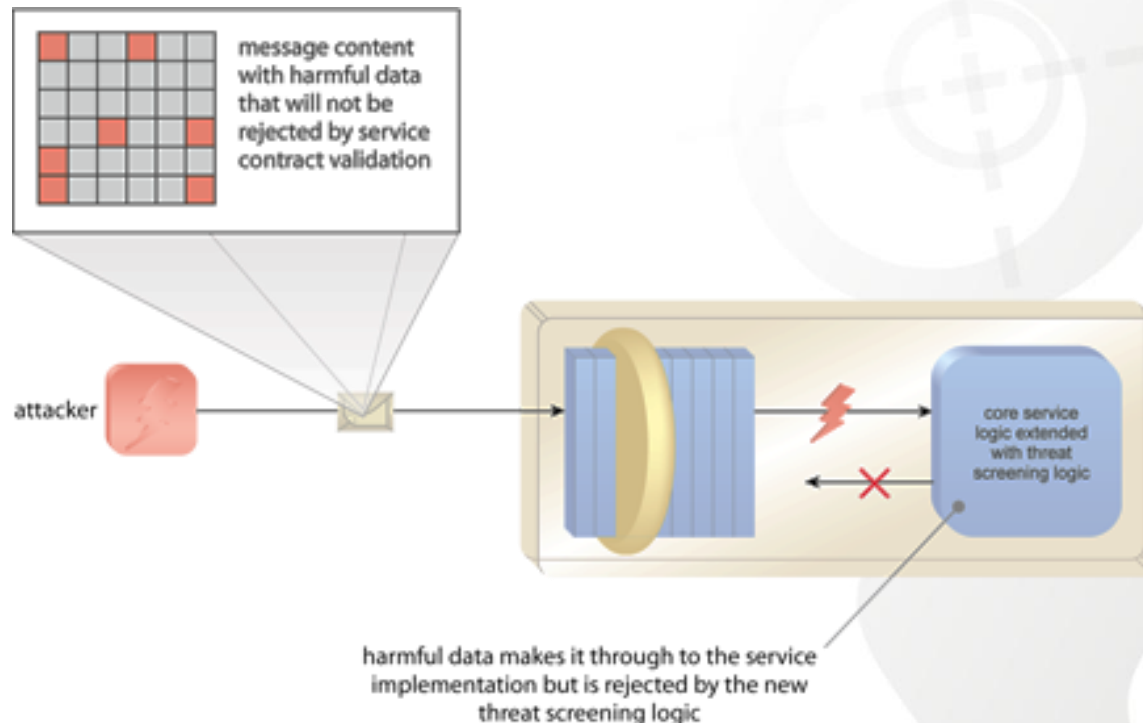


# Message Screening

- ¿Cómo se puede proteger un servicio de una entrada mal formada o maliciosa?
- El servicio está complementado con rutinas de detección especiales que suponen que todos los datos de entrada son dañinos hasta que se demuestre lo contrario.



# Message Screening





# Message Screening





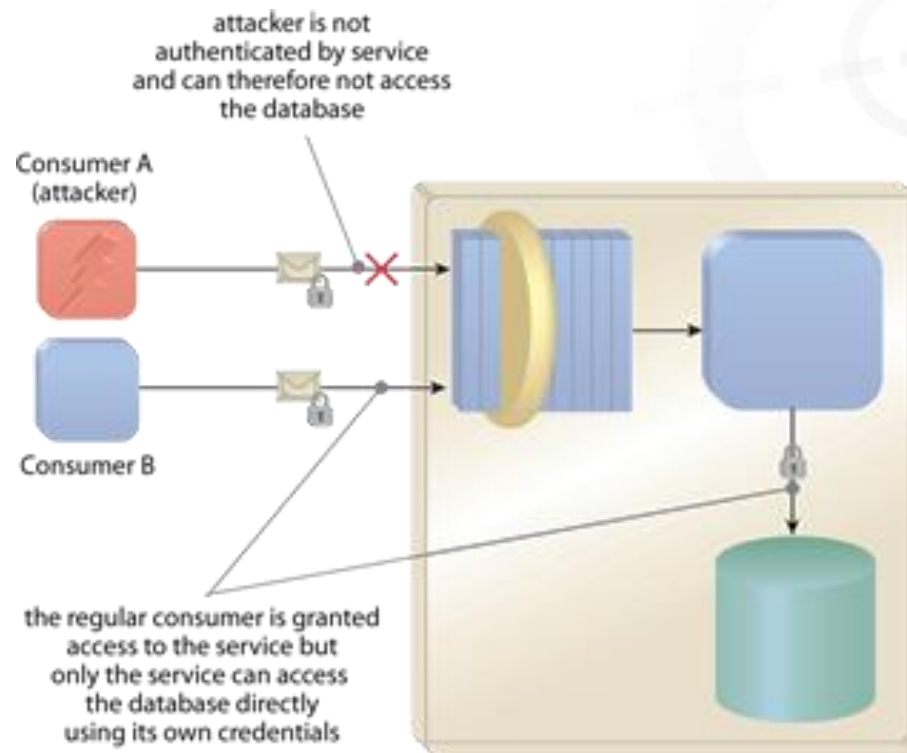


# Trusted Subsystem

- ¿Cómo se puede evitar que un consumidor eluda un servicio y acceda directamente a sus recursos?
- El servicio está diseñado para usar sus propias credenciales para autenticación y autorización con recursos de back-end en nombre de los consumidores.



# Trusted Subsystem





# Trusted Subsystem





# Service Contract Design

Concurrent  
Contracts

Contract  
Centralization

Decoupled  
Contract

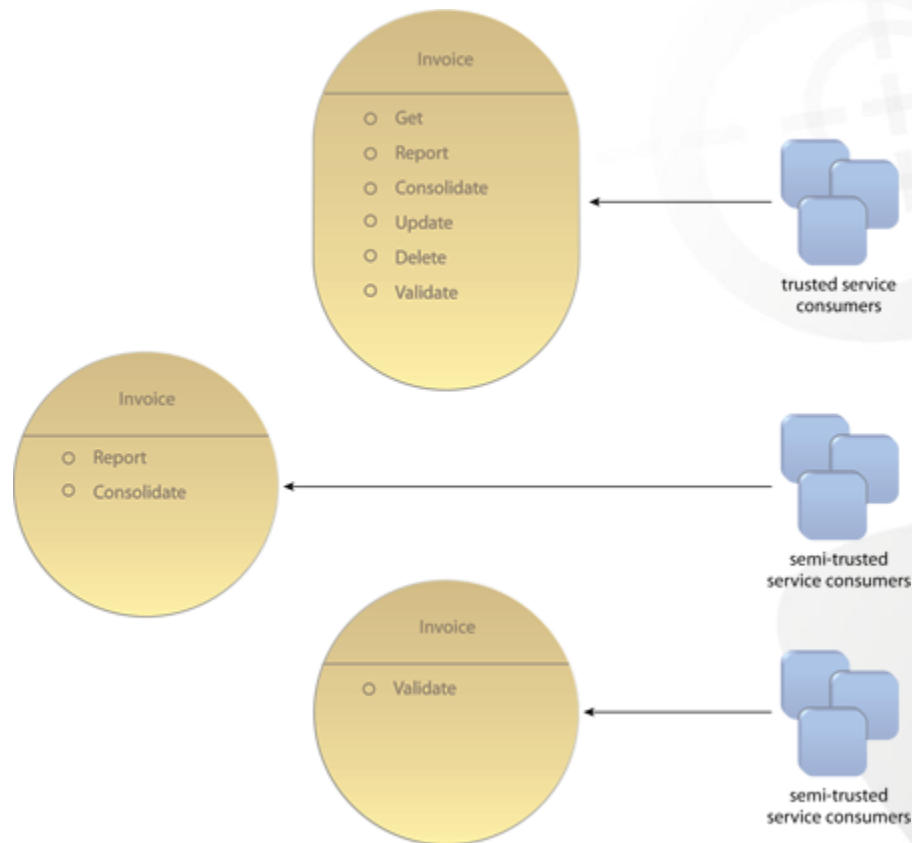


# Concurrent Contracts

- ¿Cómo puede un servicio facilitar los requisitos de acoplamiento de múltiples consumidores y la abstracción al mismo tiempo?
- Se pueden crear contratos múltiples para un solo servicio, cada uno dirigido a un tipo específico de consumidor.



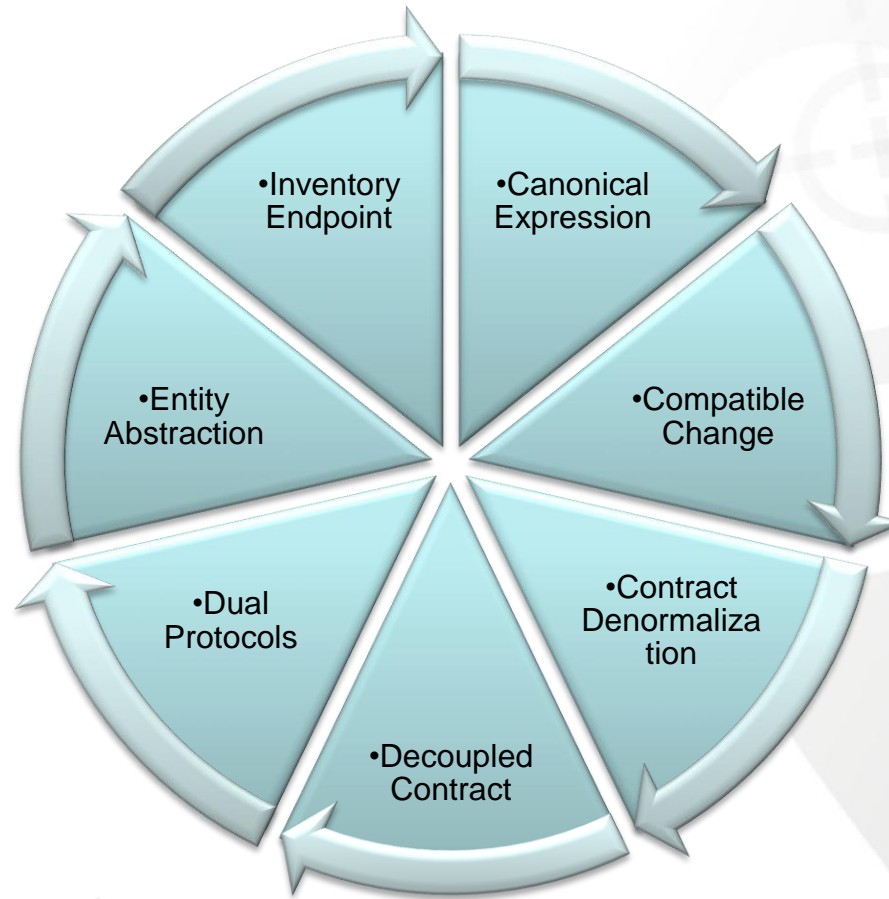
# Concurrent Contracts







# Concurrent Contracts



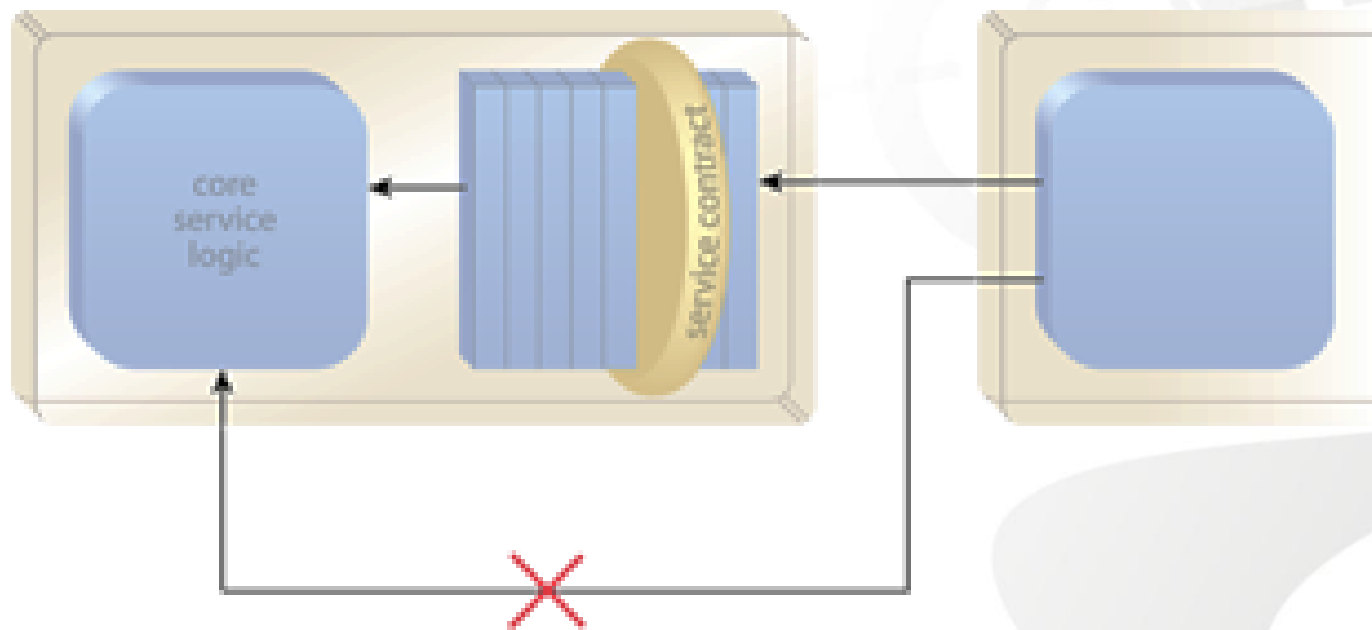


# Contract Centralization

- ¿Cómo se puede evitar el acoplamiento directo entre el consumidor y la implementación?
- El acceso a la lógica del servicio se limita al contrato de servicio, lo que obliga a los consumidores a evitar el acoplamiento de implementación.

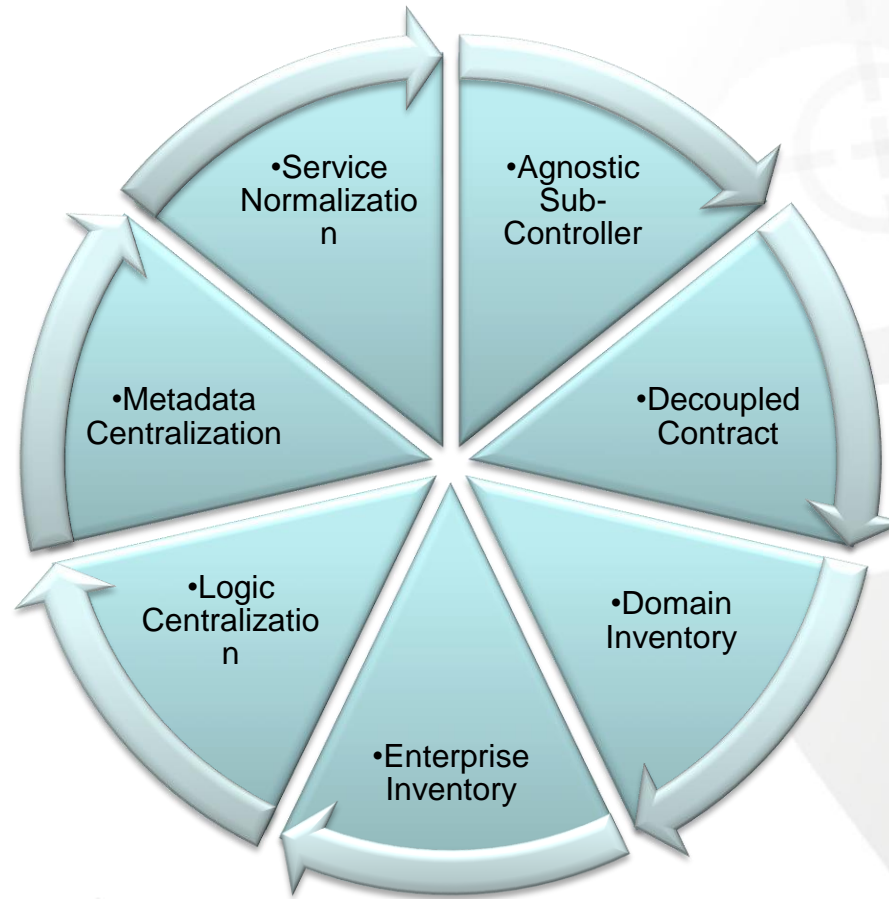


# Contract Centralization





# Contract Centralization



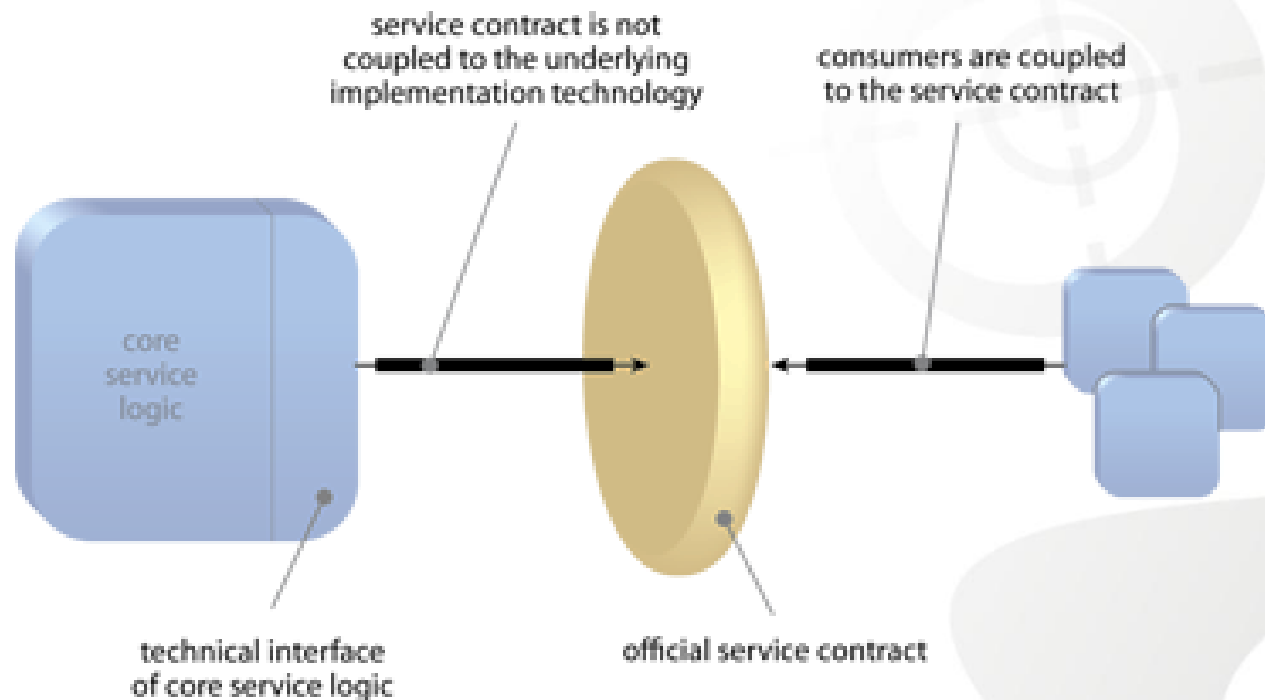


# Decoupled Contract

- ¿Cómo puede un servicio expresar sus capacidades independientemente de su implementación?
- El contrato de servicio está físicamente desacoplado de su implementación.



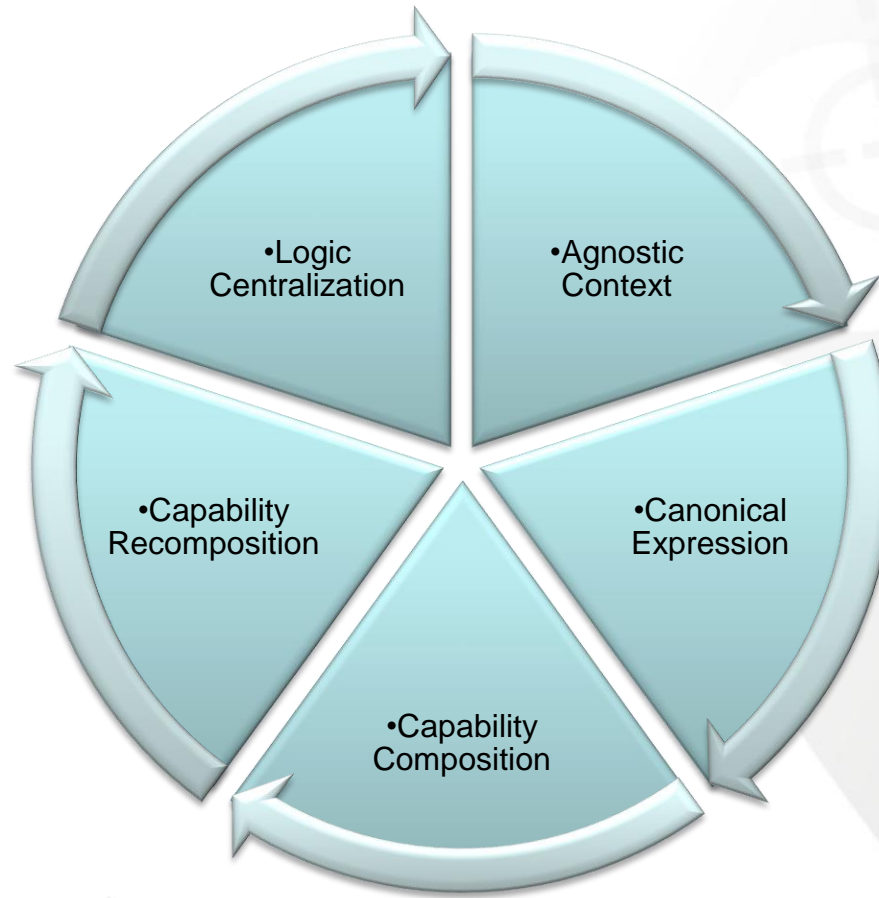
# Decoupled Contract














# Decoupled Contract












### Foundational Inventory Patterns >

-  Canonical Protocol
-  Canonical Schema
-  Domain Inventory
-  Enterprise Inventory
-  Logic Centralization
-  Service Layers
-  Service Normalization



### Inventory Governance Patterns >

-  Canonical Expression
-  Canonical Versioning
-  Metadata Centralization




### Logical Inventory Layer Patterns >

-  Entity Abstraction
-  Micro Task Abstraction
-  Process Abstraction
-  Utility Abstraction










### Foundational Service Patterns >

-  Agnostic Capability
-  Agnostic Context
-  Functional Decomposition
-  Non-Agnostic Context
-  Service Encapsulation









### Inventory Centralization Patterns >

-  Policy Centralization
-  Process Centralization
-  Rules Centralization
-  Schema Centralization





### Service Implementation Patterns >

-  Containerization
-  Microservice Deployment
-  Partial State Deferral
-  Partial Validation
-  Redundant Implementation
-  Reference Data Centralization
-  Service Data Replication
-  Service Façade
-  UI Mediator

### Inventory Implementation Patterns >






-  Augmented Protocols
-  Canonical Resources
-  Cross-Domain Utility Layer
-  Dual Protocols
-  Inventory Endpoint
-  Service Grid
-  State Repository
-  Stateful Services

### Service Security Patterns >

-  Exception Shielding
-  Message Screening
-  Service Perimeter Guard
-  Trusted Subsystem



### Service Contract Design Patterns >

-  Concurrent Contracts
-  Contract Centralization
-  Contract Denormalization
-  Decoupled Contract
-  Validation Abstraction



### Legacy Encapsulation Patterns >

-  File Gateway
-  Legacy Wrapper
-  Multi-Channel Endpoint











### Service Governance Patterns >

-  Compatible Change
-  Decomposed Capability
-  Distributed Capability
-  Proxy Capability
-  Service Decomposition
-  Service Refactoring
-  Termination Notification
-  Version Identification

### Capability Composition Patterns >

-  Capability Composition
-  Capability Recomposition





### Service Messaging Patterns >

-  Asynchronous Queuing
-  Event-Driven Messaging
-  Intermediate Routing
-  Messaging Metadata
-  Reliable Messaging
-  Service Agent
-  Service Callback
-  Service Instance Routing
-  Service Messaging
-  State Messaging



### Composition Implementation Patterns >

-  Agnostic Sub-Controller
-  Atomic Service Transaction
-  Compensating Service Transaction
-  Composition Autonomy

### Service Interaction Security Patterns >

-  Brokered Authentication
-  Data Confidentiality
-  Data Origin Authentication
-  Direct Authentication

### Transformation Patterns >

-  Data Format Transformation
-  Data Model Transformation
-  Protocol Bridging



Pontificia Universidad  
**JAVERIANA**  
Bogotá

ESPECIALIZACIÓN EN  
ARQUITECTURA EMPRESARIAL DE SOFTWARE

# Gracias