www.cbdiforum.com

Series: Business Service Architecture

Module 2: Introduction

Independent Guidance for
Service Architecture and Engineering

Everware-CBDI

Agenda

Background & Context

- Service-Oriented Architecture (SOA)
- Service Architecture and Engineering (SAE)

This Module

- What is the Service Architecture?
  - Basic Structure
  - Rationale
- Service Classification
- Service Layers and Dependencies

Where we are going next

- Identifying Services
- Specifying Services
- Implementing and Deploying Services
- Architectural and Governance Policies

Learning Objectives

- To understand the different types of services and their contribution to SOA.

Motivation

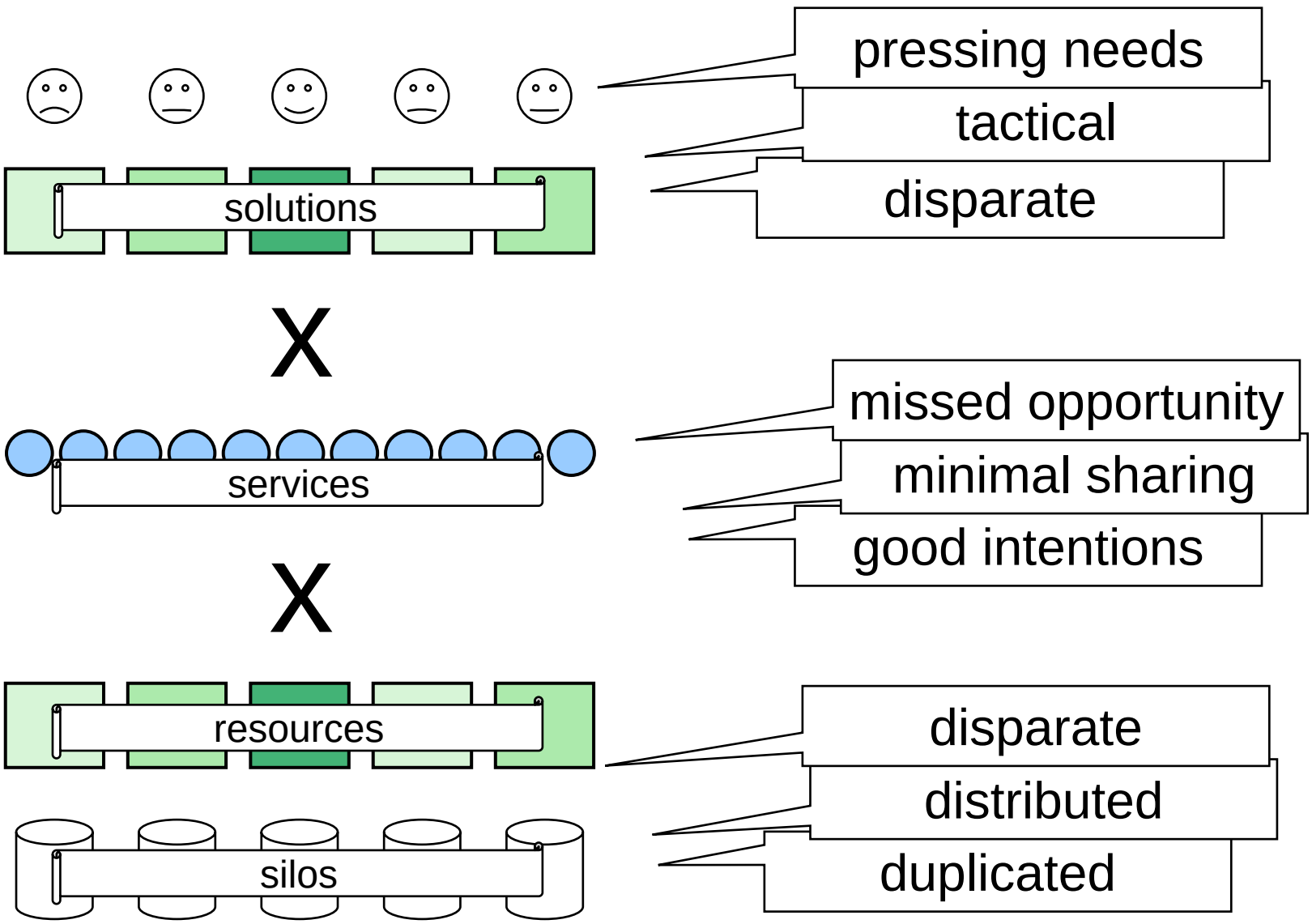
Without an architecture …

- Service Anarchy
- JBOWS (just a bunch of web services)

With an architecture …

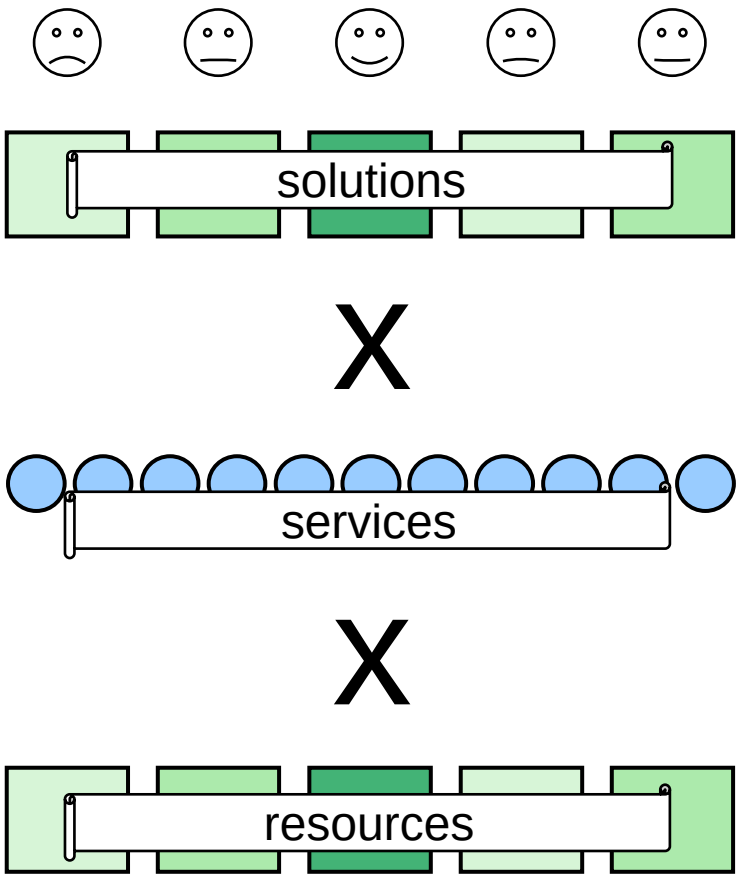
- Business-IT alignment
- Economics of scale (shared services, reuse, repurpose)
- Flexibility, Quality

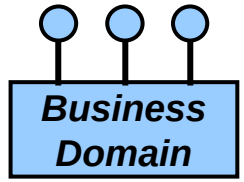
Typical Approach to Services Today



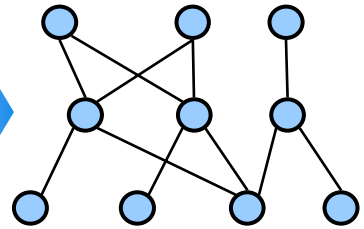
Focusing on the A in SOA - Service Architecture

Typical Approach to Services Today

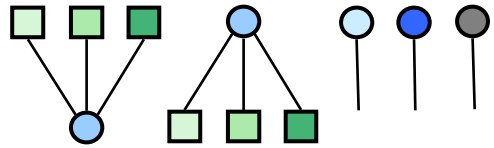




Services Grouped by Domain



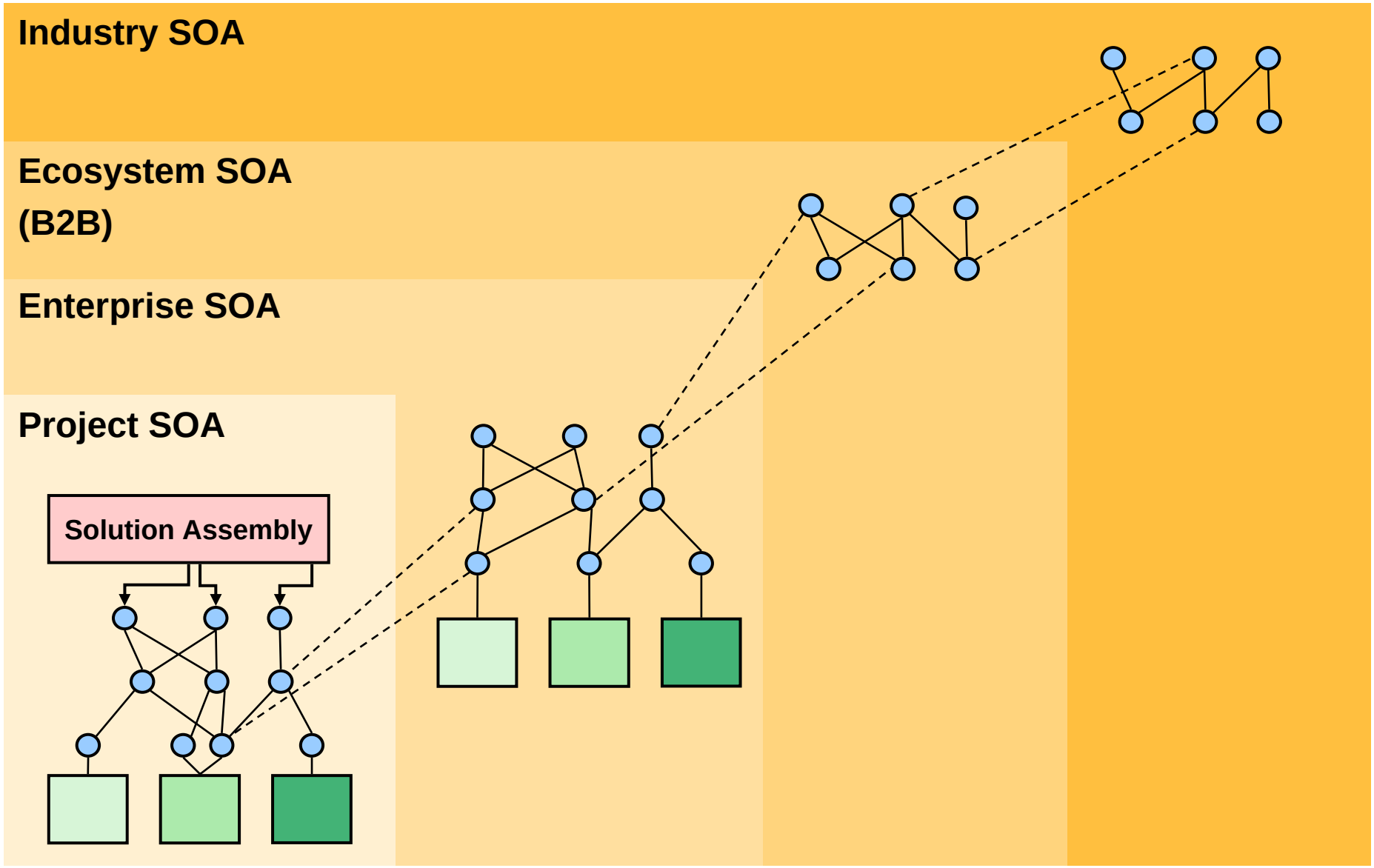
Services Organized into layers by Purpose and Type



Services Selected for Sharing, Aggregation or Differentiation

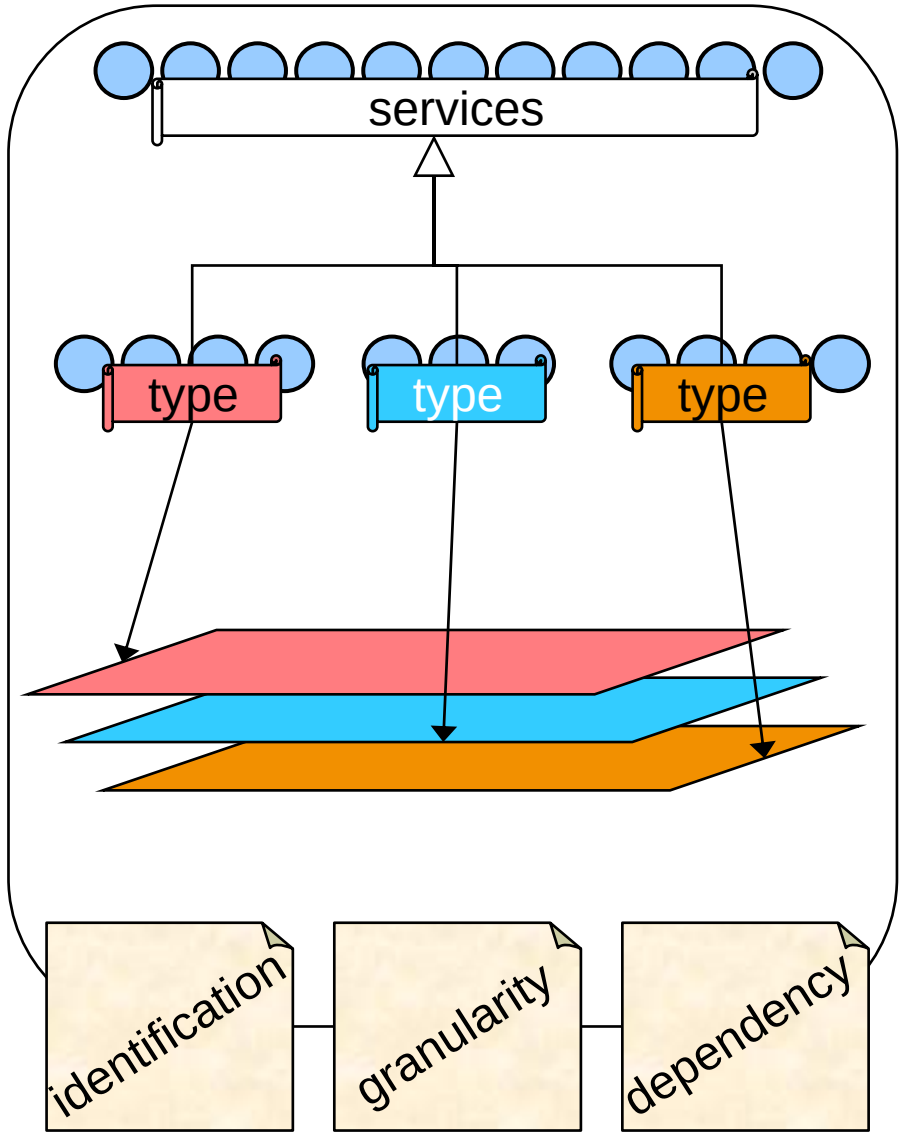
Application of Architectural Concepts

Scope – How Widely the Service or Service Architecture is Used



What is a Service Architecture?

- A collection of services
- classified into types
- arranged into layers
- governed by architectural patterns and policies



five Views of Service Architecture

Business View

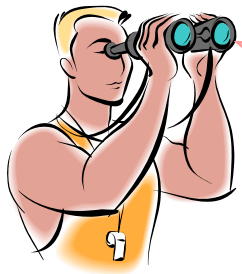
(for enterprise planners)

Infrastructure View

(for technology planners)

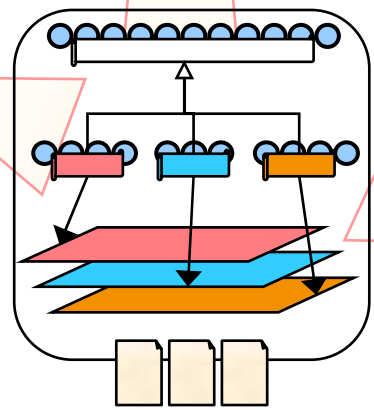
Specification View

(for consuming developers)



Deployment View

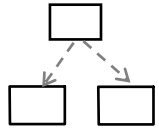
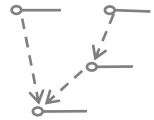
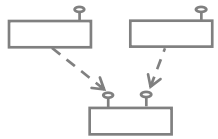
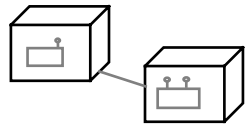
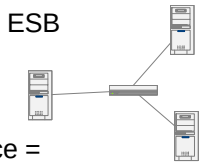
(for operations)



Implementation View

(for service developers)

Reference Architecture Spans Business to Operations

View	Sample Artifacts	Service Perspective
Business	SO Business Plan SO Business Model	Business Service, Context for Software Services  Business Service = Services offered by Organizational Units
Specification	Service Specification Architecture Service Specification	Logical Specification of Software Services  Service = Service Specification
Implementation	Service Implementation Architecture Automation Unit Specification Service Implementation	Service Packaging into Automation Units  Service = Software Service Impl.
Deployment	Service Deployment Architecture Service Platform Design Specification	Deployment of Automation Units  Service = Run-Time Software Service
Infrastructure	Logical Network Services Physical Network	e.g. Network Layout, ESB  Infrastructure Service = Run-Time Platform

SO Security Architecture

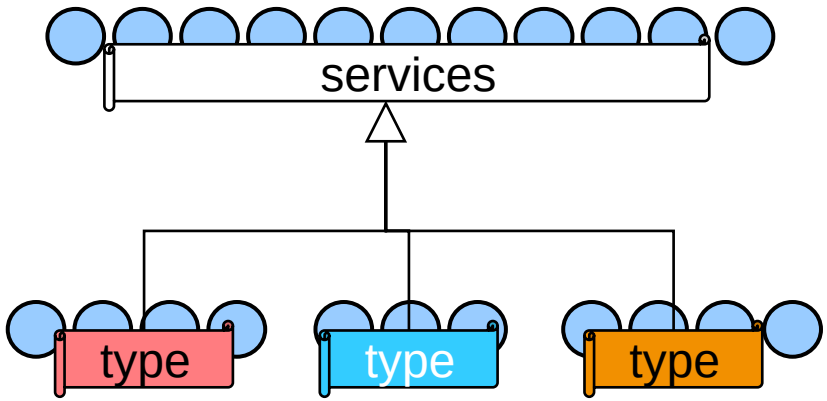
Service Portfolio Plan

Service Catalog

Service Level Agreement

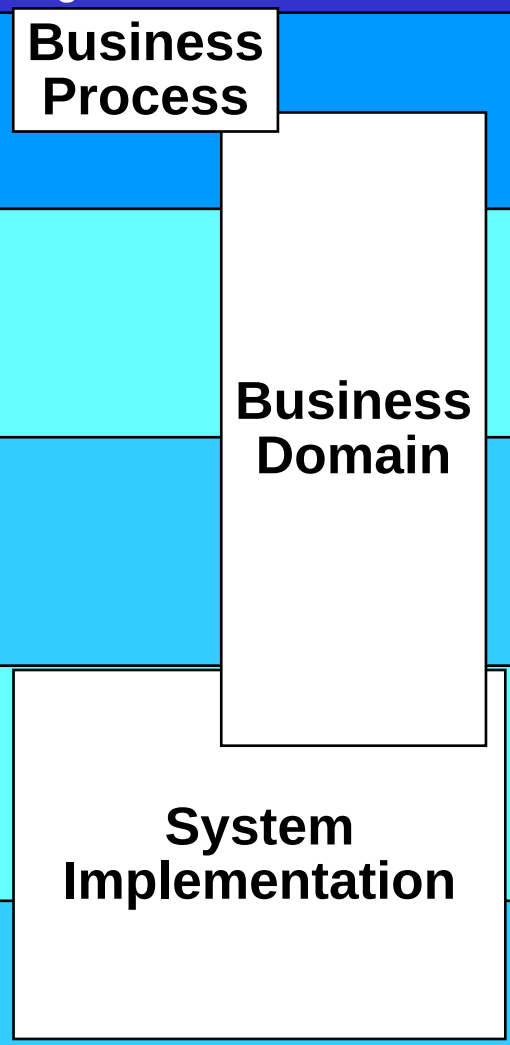
Technical Architecture

Classification:
How are Services Classified and Organized?



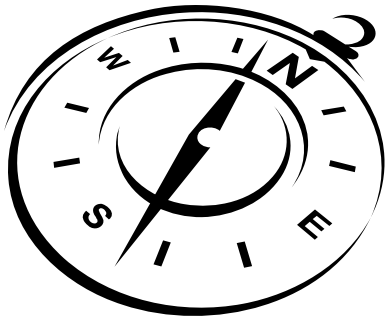
Organize Services by Purpose and Type

Type	Purpose	Coupling
Process Services	Orchestration	
Capability Services	Business functions	
Core Business Services	Business types (“entity services)	
Underlying Services	Not exposed as well-formed service	
Utility Services	Shared by Core Business Services	



Core Business Services

Purpose



- Business representation of business resource - **360° view**
- Information capture and retrieval
- Shared across multiple processes
- Common business rules

Similar Concepts

- Entity Services
- Core Services
- Domain Services

Examples

- Employees Services
- Products Services
- Purchase Orders Services
- Customers Services
- General Ledger Services
- Locations Services

Core Business Services

Specification View

- Initially derived from business type models
- Typically offers CRUD operations
- Highly generic, applicable to many lines of business
- Apply enterprise-wide business rules

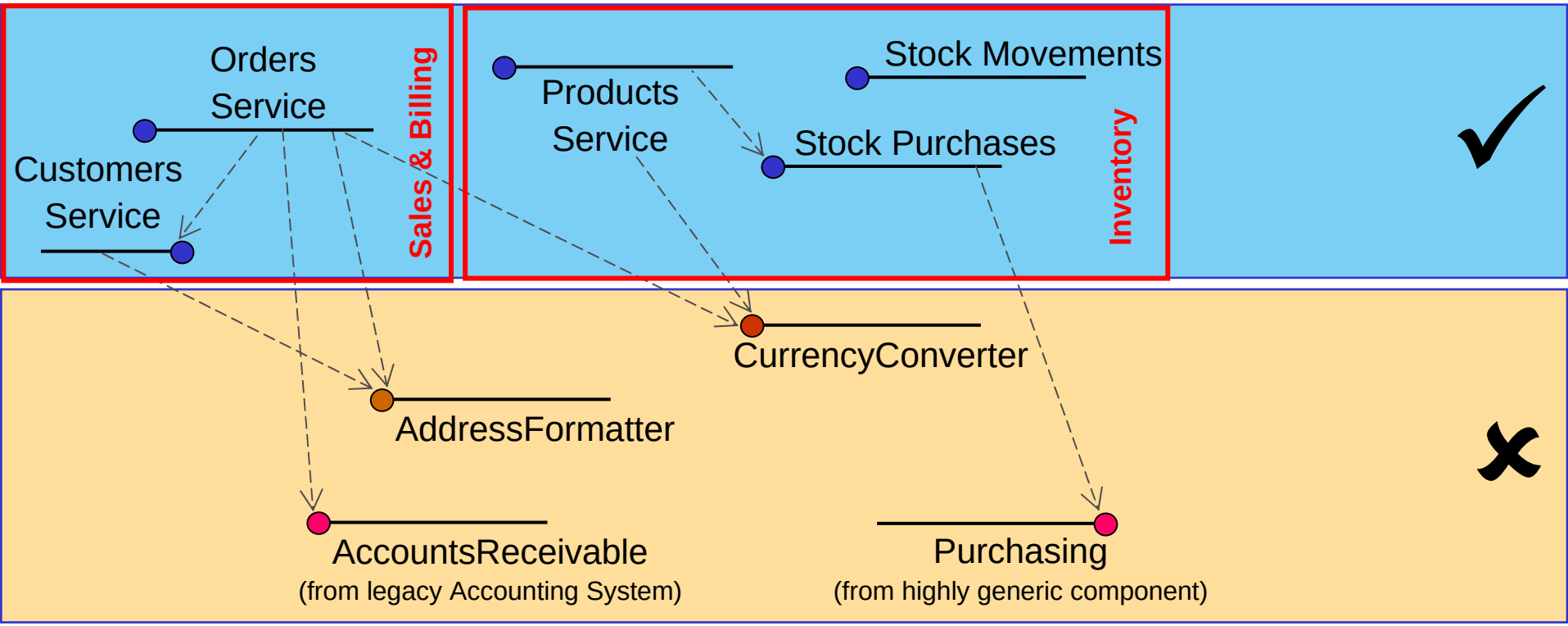
Implementation View

- Normally maintain the principal data stores, unless this delegated to Underlying Services
- Operations are UI- and business process- independent, so can be used in different contexts

Deployment View

- Concurrency
- Multi-Tenancy

Core Business Service - Examples



Capability Services
Encapsulated, Independent

Purpose

- Support Business Capabilities
  - It defines **what** your business is able to do, but not **how** it is done
- Process-independent business know-how
- Maximum business agility

Examples

- Bank – Payment
- Logistics – Shipment
- Sector neutral – Auction, Procurement

Specification View

- Encapsulates entire Business Capability.
- May include software, resources, physical and electronic processes

Implementation View

- UI- and process independent,
  - *designed to support one particular 'business capability'*
- Development capable of being offshored or outsourced
- Standalone …
  - *stores own data;*

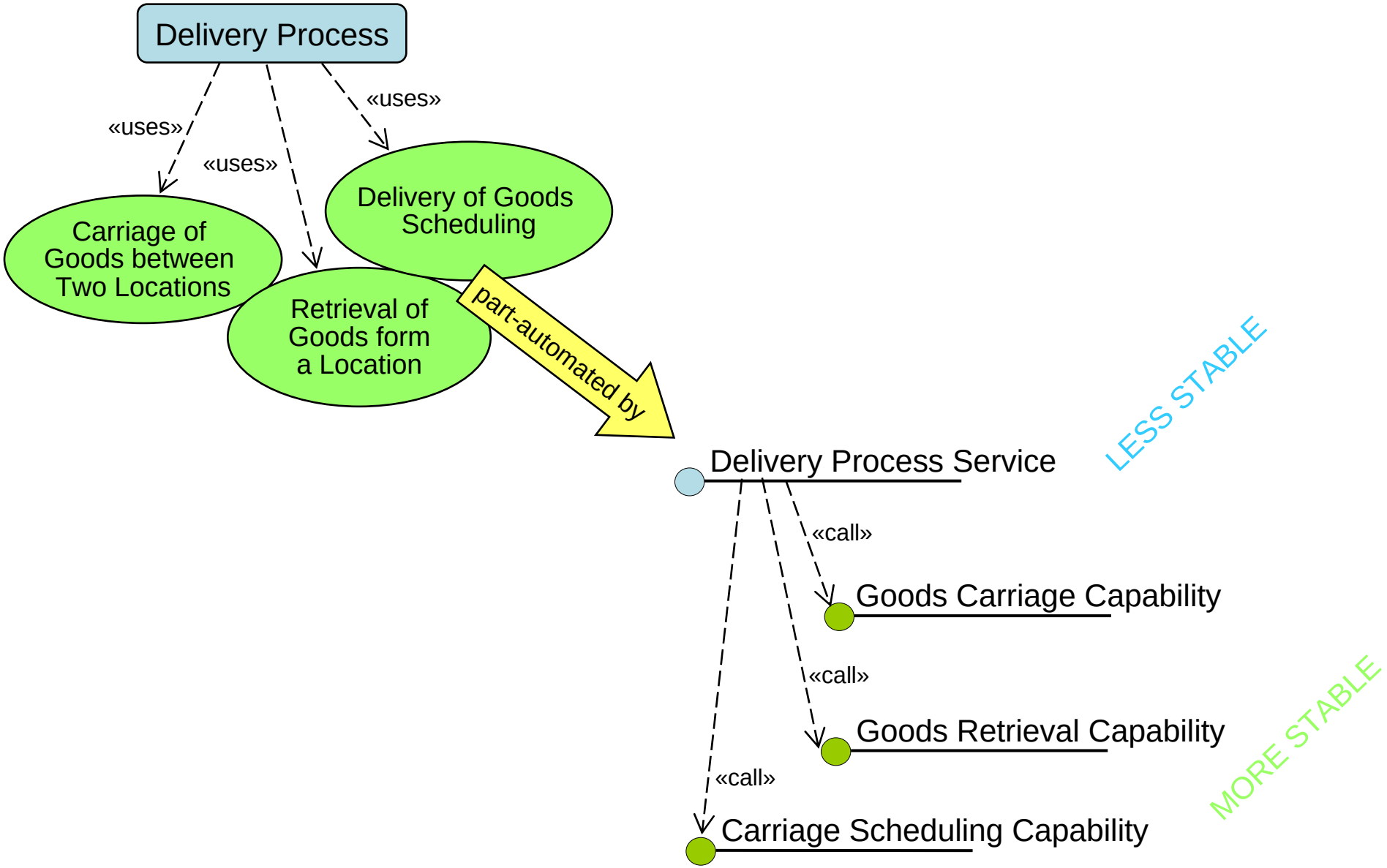
… or integrated

  - *may store any data not managed by core business & lower services.*

Deployment View

- Execution capable of being offshored or outsourced

Capability Service - Example



Process Services

Purpose

- Orchestration: process rules and logic, manages process state,
- Coordinates other services

Implementation View

- May be developed as a BPEL script or workflow
- May store transient data (including process state)

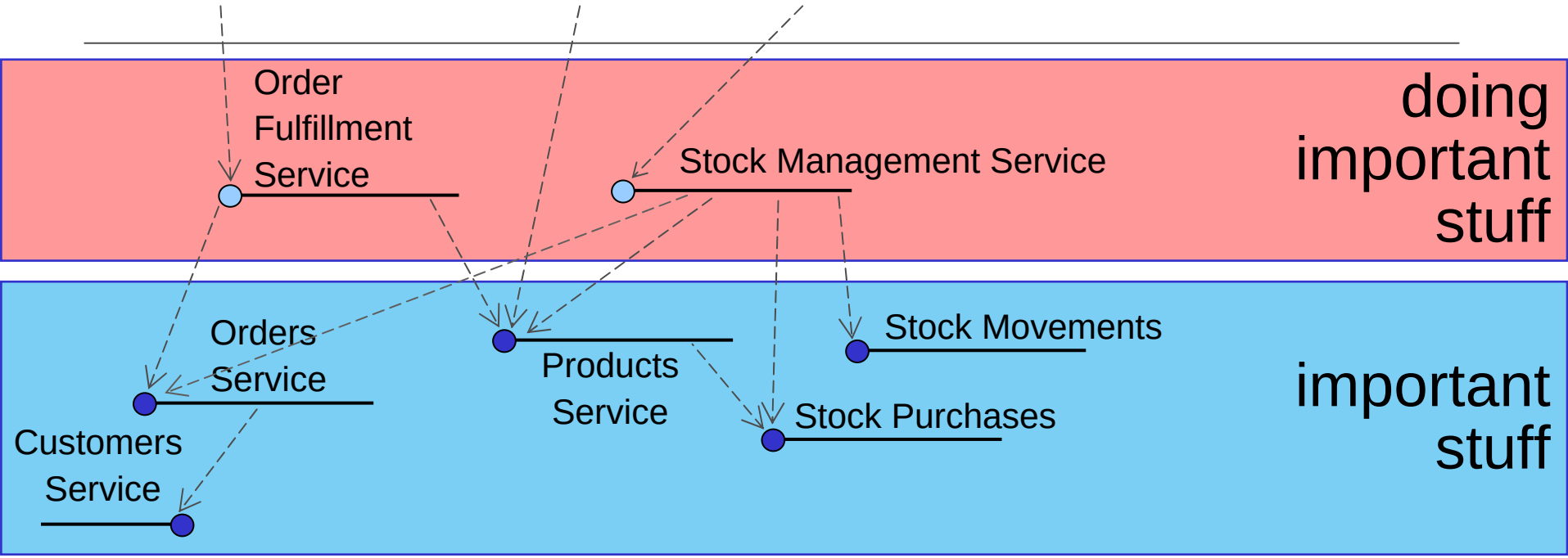
Specification View

- Based on Business Process Model
- May use BPM tools
- UI-independent
- Encapsulates hierarchy of services

Deployment View

- May be delivered via ESB or process engine

Process Service - Example



Utility Services

Business Utility

- Common business logic & algorithms
- Specialized business functionality
- Directory or reference data

System Utility

- Extend technical functionality of platform / infrastructure
- E.g. identity, presence, security

General Characteristics

- Shared?
- Volatile?
- Technology coupling?
- No business coupling?

Specification View

- Highly generalized
- Highly specialized

Implementation View

- Often built using off-the-shelf components.
- Data storage
  - Directories
  - Look-up tables
  - Audit trails
- Functionality may move to platform

Deployment View

- Engineered for ubiquitous usage

Underlying Services – The Façade Pattern

Starting Point

- Functionality that is difficult to use or expose as a service.
  - *Highly generalized*
  - *Non-standard terminology*

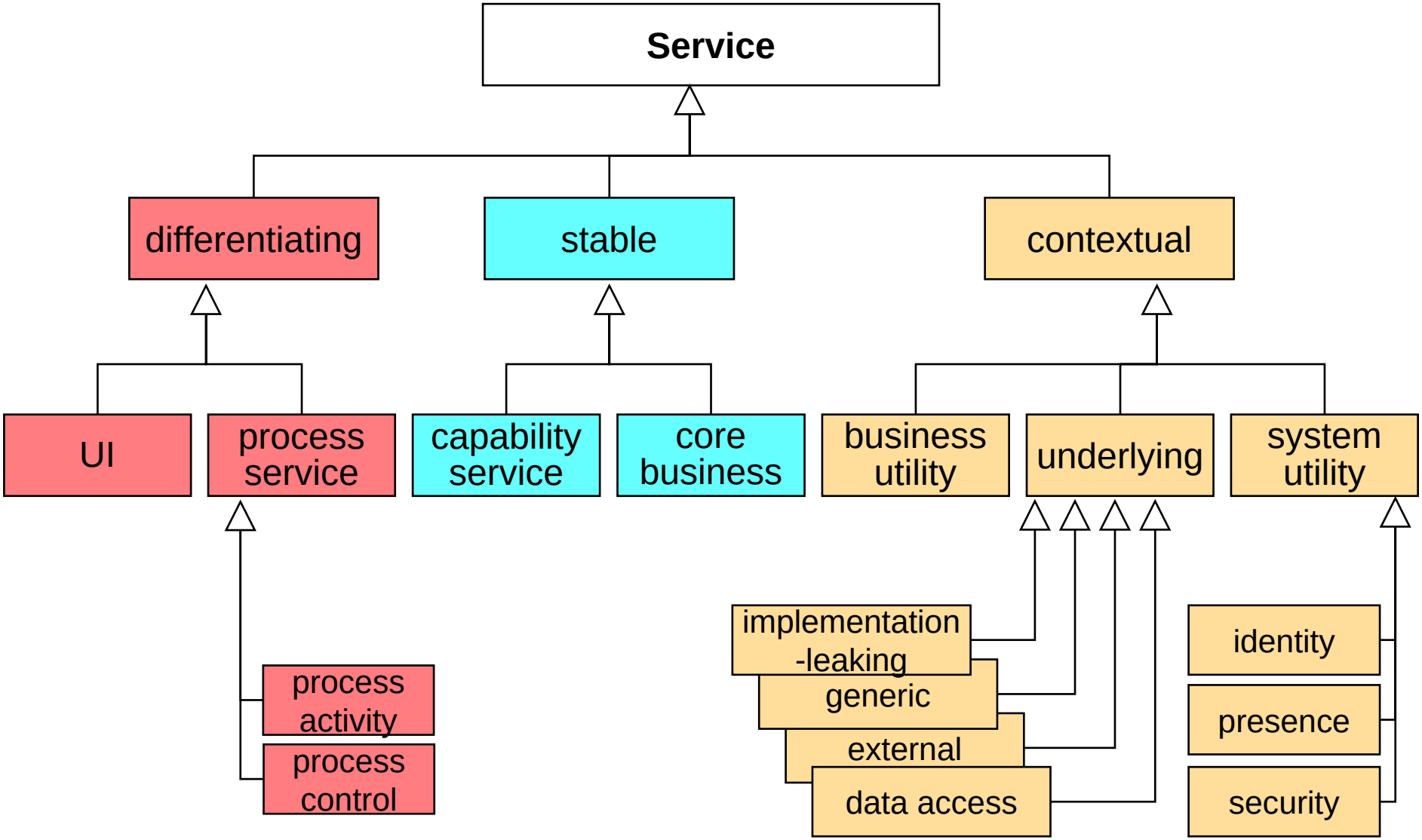
Therefore

- Hide underlying service behind a façade
  - translates data items to core business service's information model
  - filters out unnecessary options & data,
  - combines data and functions from several sources
  - does not include business logic

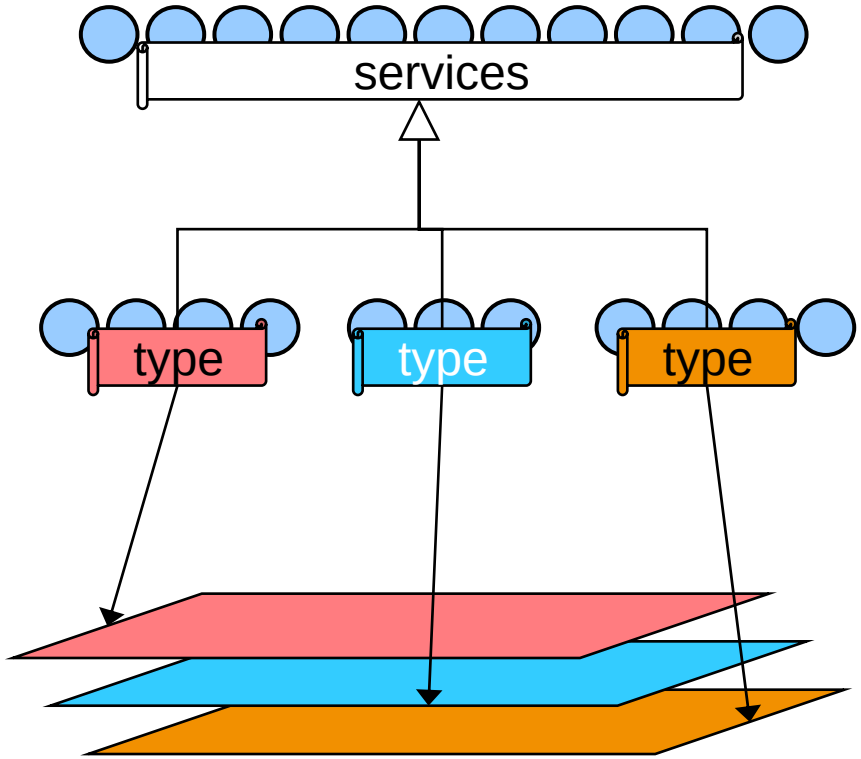
Consumption

- Underlying Services might be consumed directly by Solution or Process layers
  - before core business services become available
  - since pretty straightforward to use
  - provides much better performance
- Underlying services often maintain or access significant data stores
- Core services call underlying services to access this data & its processing rules

Recap: A taxonomy of services



Stratification:
How and Why are Services Assigned to Layers?



Service Layering

Benefits of Layering

Achieve SOA Objectives

- Higher degrees of reuse/sharing
- Flexibility in assembly of services from lower layers

Basis for Policy & Patterns

- Determine policies by layer
- Apply patterns by layer
- Separation of concerns

Basis for Sourcing

- Standardization in lower layers
- Customization in higher layers

Layering Guidelines

- Different characteristic rates of change
- Different granularity
- Standardization versus Differentiation
- Inside / Outside

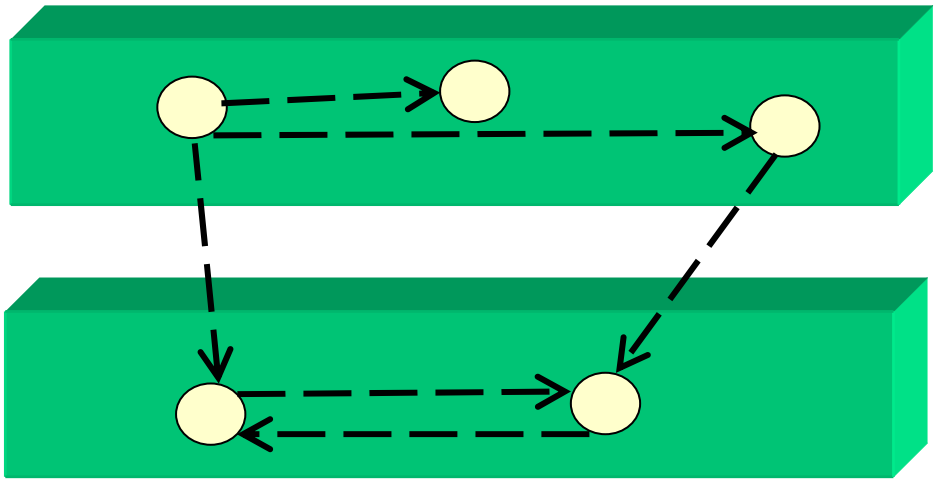
Assigning Services to Logical Layers

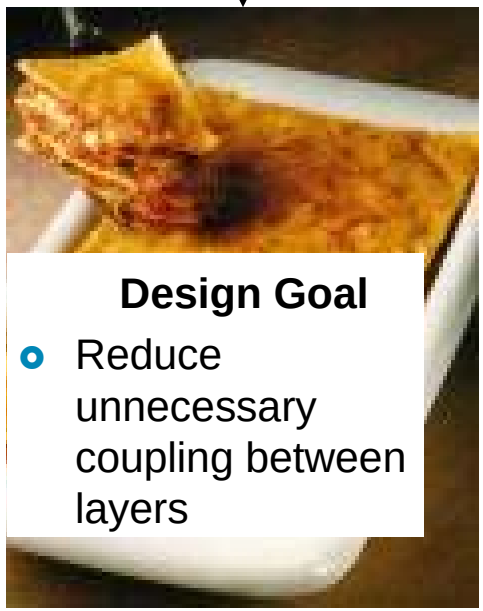
Division of Responsibilities

- Clear system function for each layer

Dependencies

- Upwards?
- Downwards?
- Sideways?

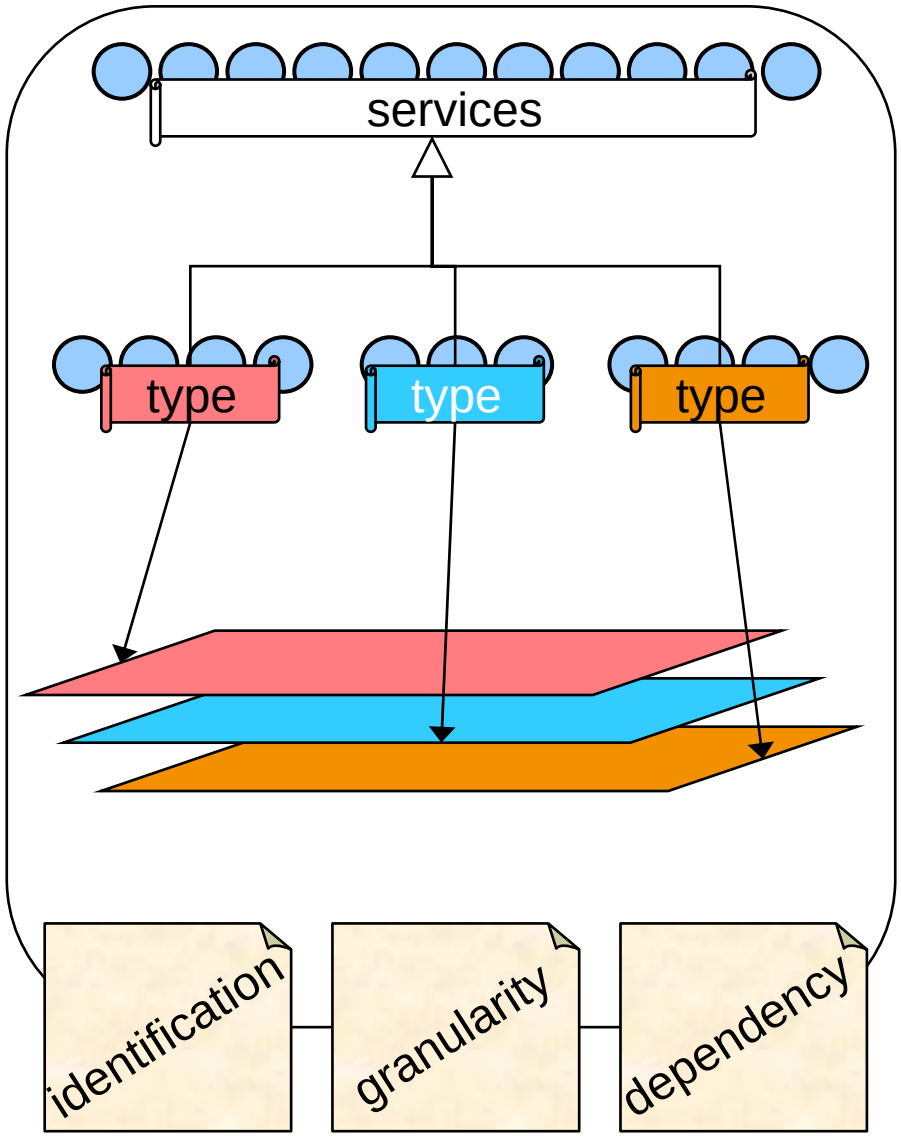




Design Goal

- Reduce unnecessary coupling between layers

What policies govern the Service Architecture?



including …

Identification / Variation

- Differentiation / Standardization

Granularity

- Coarse-grained / Fine-grained?

Dependency

- Upwards?
- Downwards?
- Sideways?

Non-Functional Requirements

- Different expectations in each layer

Service Dependency

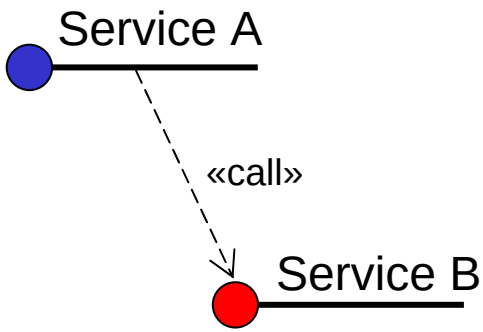
Specification View

Logical Dependency

- Service A needs Service B

Examples:

- **Invariant Dependency** - A creates/updates business type instances which are constrained by instances in B
- **Creation Dependency** - A cannot function without B creating business type instances for A first



Implementation View

Call Dependency

- Means service A requests service B to perform one or more of its operations

Event-Based Dependency

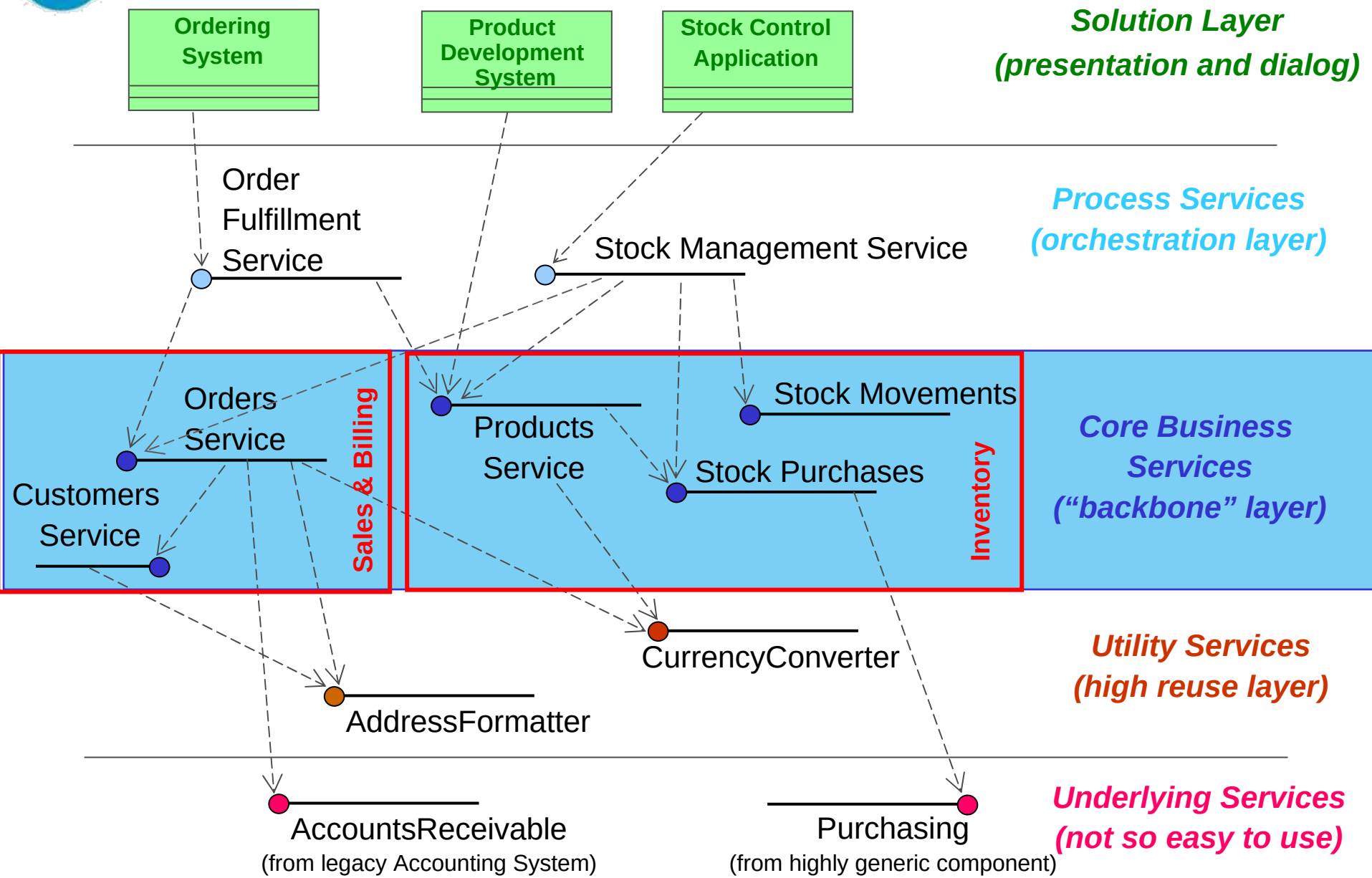
- Service A waits for an event generated by Service B

Publish/Subscribe

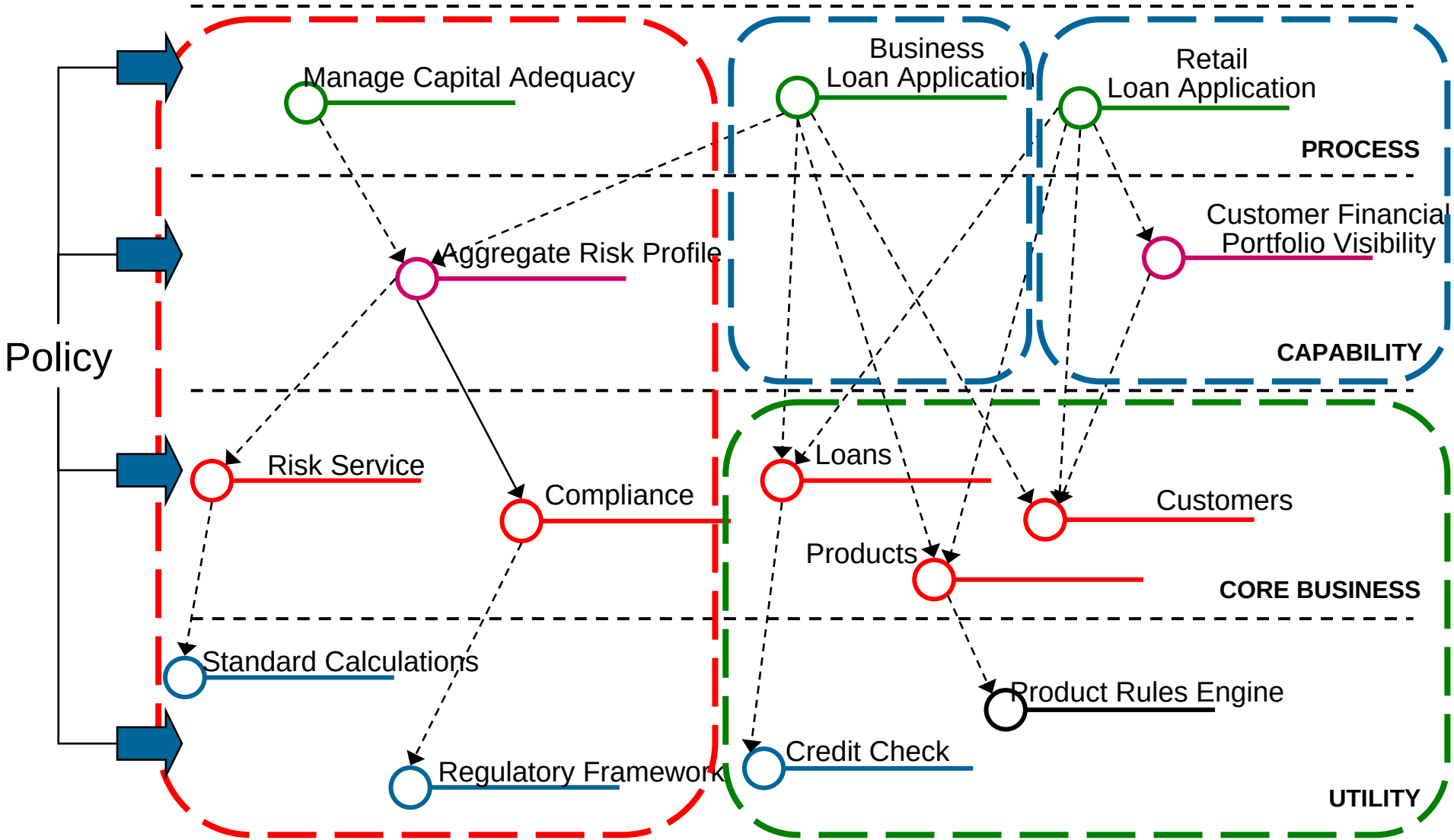
- Service A subscribes to some data or other document published by Service B

Specification View

Services Organized into Layers



Specification View



Recap: CBDI proposes six basic layers

What?

- **Solution layer**
  - only consumes services
  - does not offer any services
- **Process Service layer**
  - value chain rules
  - coordinates other services
- **Capability Services**
  - encapsulates entire business function
- **Core Business Service layer**
  - reusable enterprise services
  - maintain data
- **Utility Service layer**
  - commonly needed logic
  - highly shareable
- **Underlying Service layer**
  - services that are difficult to consume

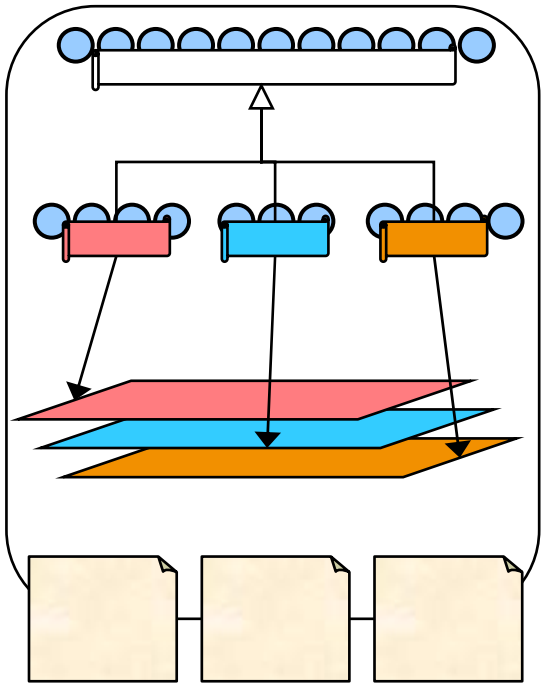
Why?

- **Classification**
  - useful scheme
  - manage different services differently
- **Layering**
  - improved maintainability
  - higher degrees of reuse / sharing
  - specialization of labor
  - standardization in lower layers
  - differentiation in higher layers
- **Result**
  - More productive process
  - More agile architecture

Actions

What you have learned so far

- Understand the different types of services and their contribution to SOA.



Using what you have learned so far

Architects

- Establish service layers
- Set policies
- Set scoping strategy,
- Establish project rules

Project / Program Managers

- Consistency
- Policies
- Patterns for agility
- Lifetime cost of ownership

Further study

eLearning Modules

- Identifying Services (by Type)
- Building the Architecture

CBDI Portal

- Service Architecture and Engineering

CBDI Reports

- Exploring Business Capability
- Project Driven Service Portfolio
- Practical Service Specification and Design
- Service Portfolio Planning Revisited

www.cbdiforum.com

Independent Guidance for Service Architecture and Engineering

Everware-CBDI

www.everware-cbdi.com