# Understanding Service Layers

hungdn@ptit.edu.vn

# Outlines

1. Introduction to Service Layers
2. Breaking Down the Business Problem
3. Building Up the Service-Oriented Solution

# 5.1 Introduction to Service Layers

# Service model

- A service model is a classification used to indicate that a service belongs to one of several pre-defined types

- Classification based on
  - The type of logic it contains
  - The reuse potential of the logic
  - How the service may relate to elements of the actual business logic

# Common Service Model

- **Task Service**
  - Non-agnostic functional context, compose several other services to complete its task
  - Single-purpose
- **Entity Service**
  - Agnostic functional context associated with one or more related **business entities**
  - A reusable service
- **Microservice**
  - A non-agnostic service in small functional scope
  - Have single-purpose with specific requeriments
  - Typically not reusable
- **Utility Service**
  - Agnostic functional context encapsulates low-level technology-centric functions
  - A reusable service

# service layer

- **A given service inventory will usually contain multiple services that are grouped based on each of these service models**

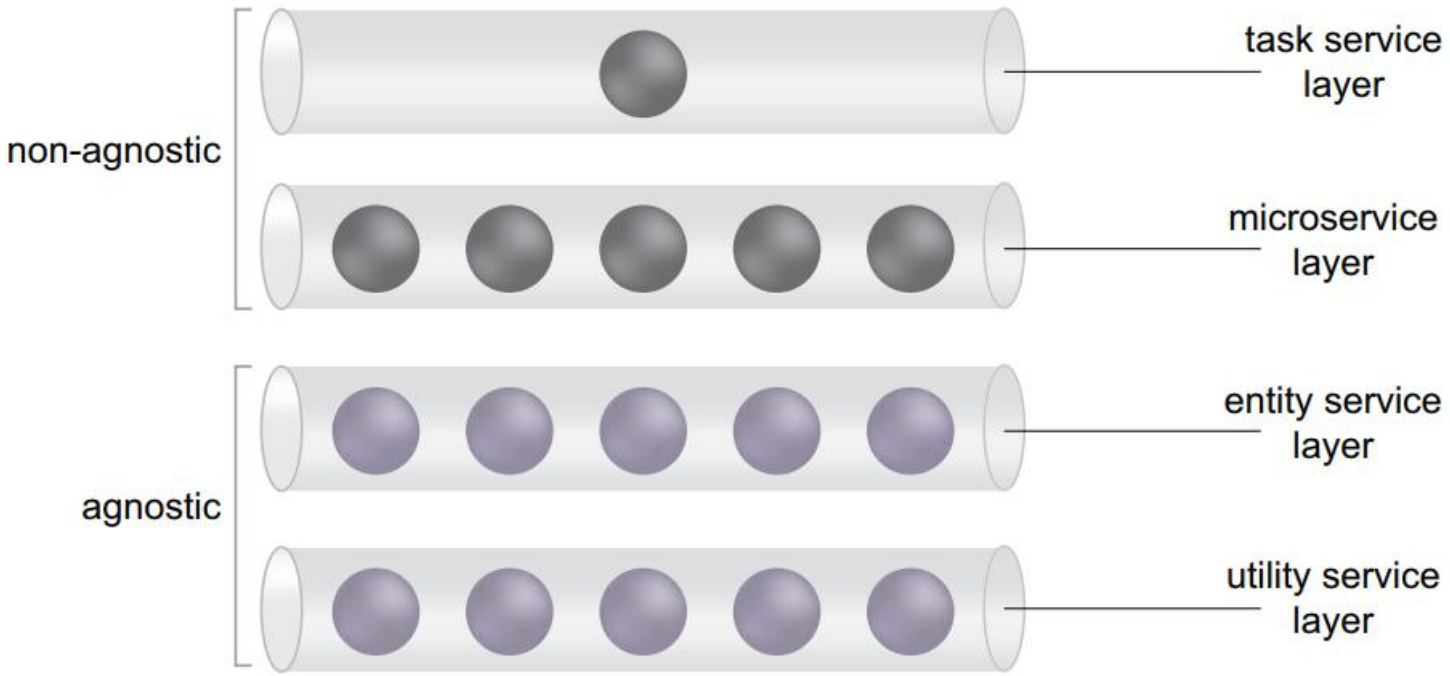



**Figure 5.2**
The common service layers, each of which is based on a service model.

- **Each of these groupings is referred to as a service layer**

# Service and Service Capability Candidates

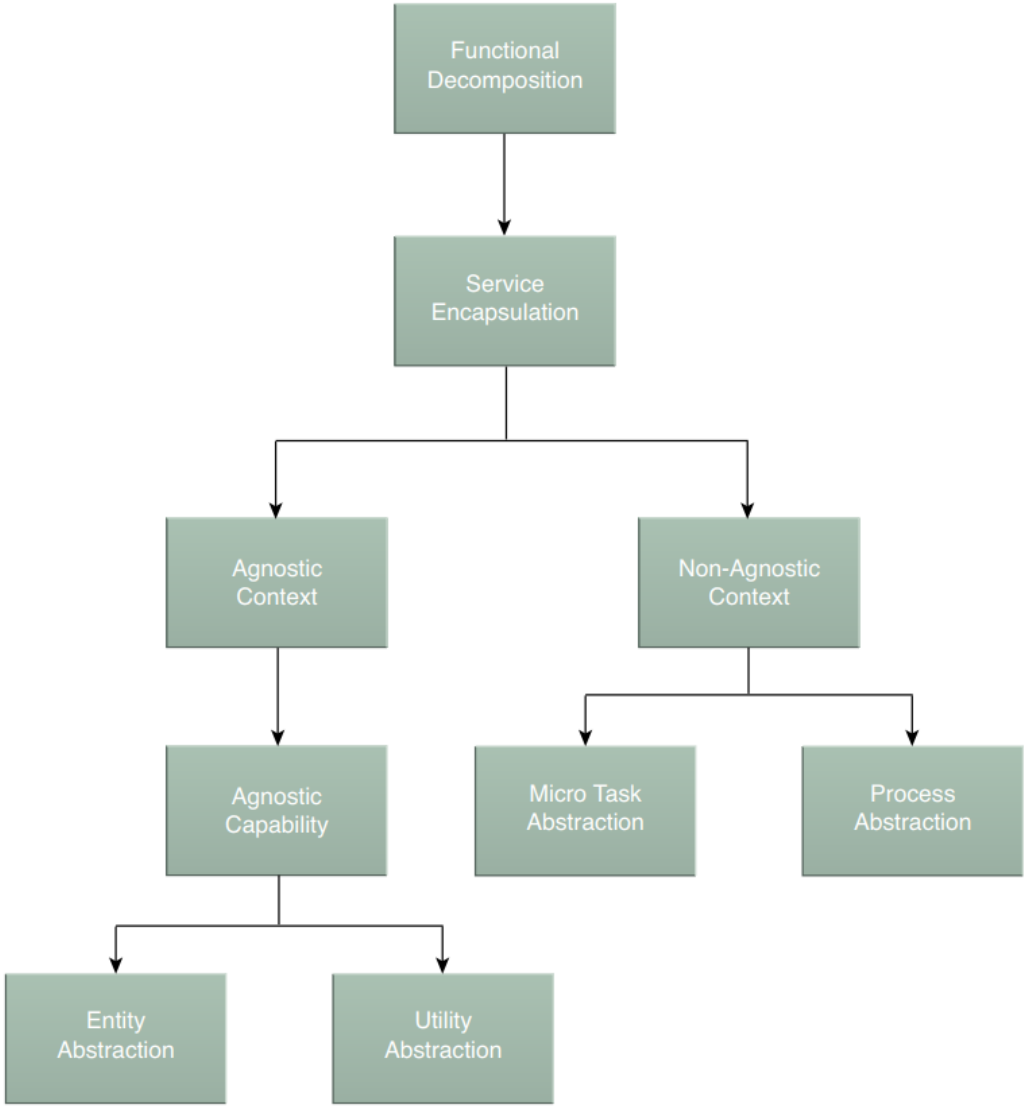
- The terms “service candidate” and “service capability candidate” are used to distinguish conceptualized service logic from service logic that has already been implemented.

# 5.2 Breaking Down the Business Problem

# Introduction

- **A primitive service modeling process aim to the definition of candidate services and capabilities.**

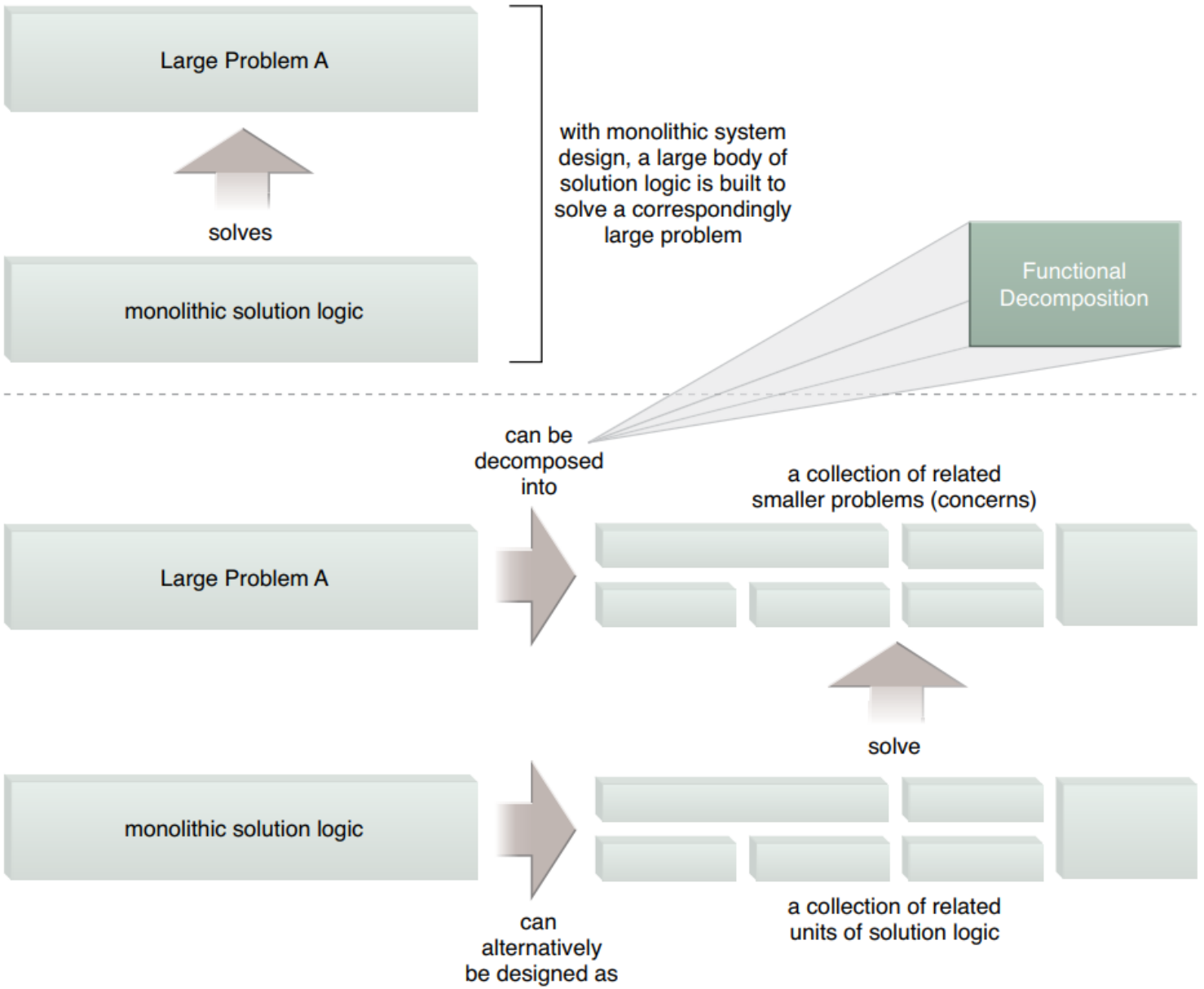
- **Applied to the early stages of service modeling and service design**



# Functional Decomposition

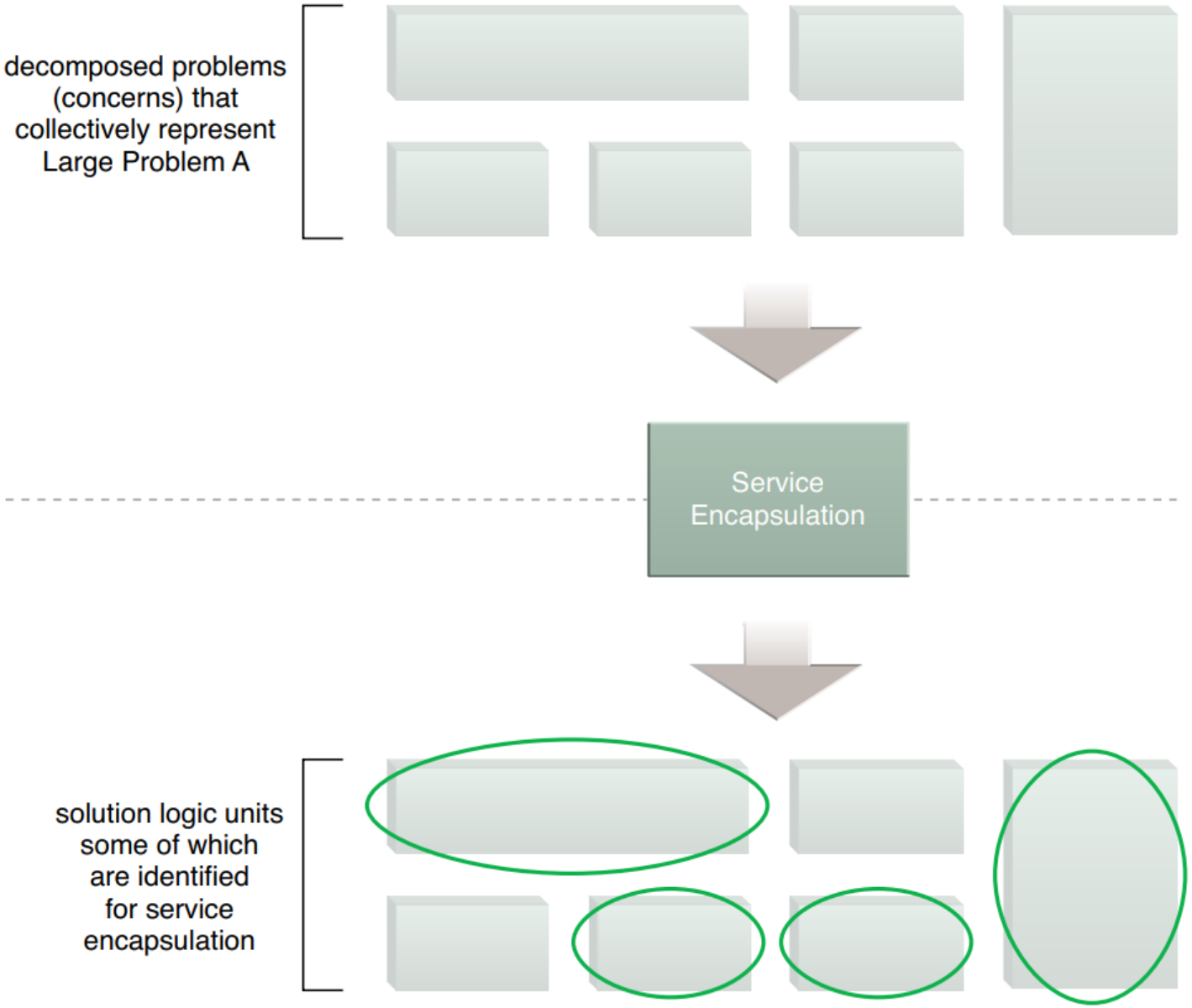
- **The decomposition of a larger problem into smaller problems, called "concerns" - a unit of solution logic can be built**

- **Functional decomposition**
  - First step in service modeling process (**Figure 5.1**)



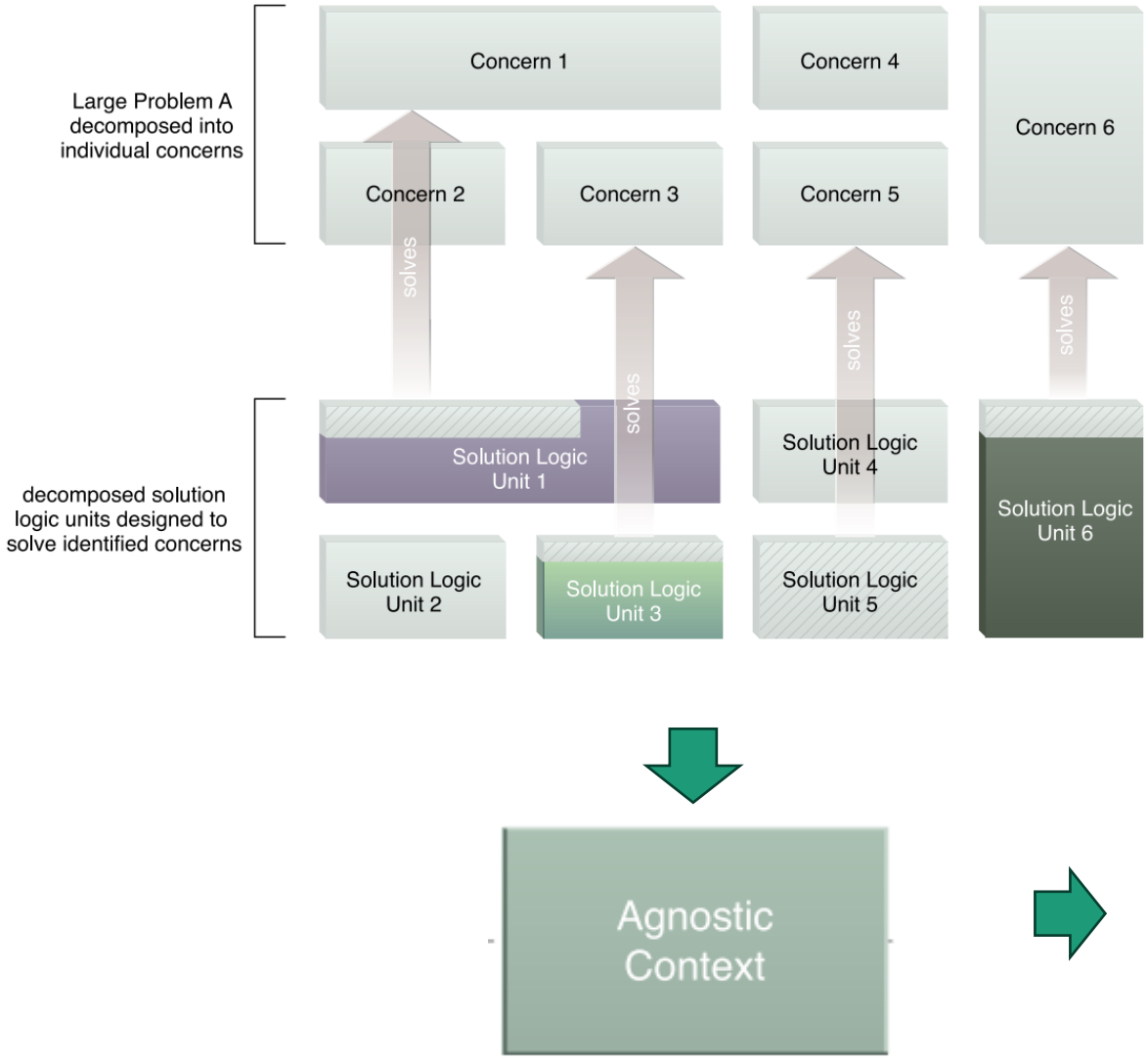
# Service Encapsulation

- **During the service encapsulation step, we identify the parts of the logic required that are suitable for encapsulation by services**



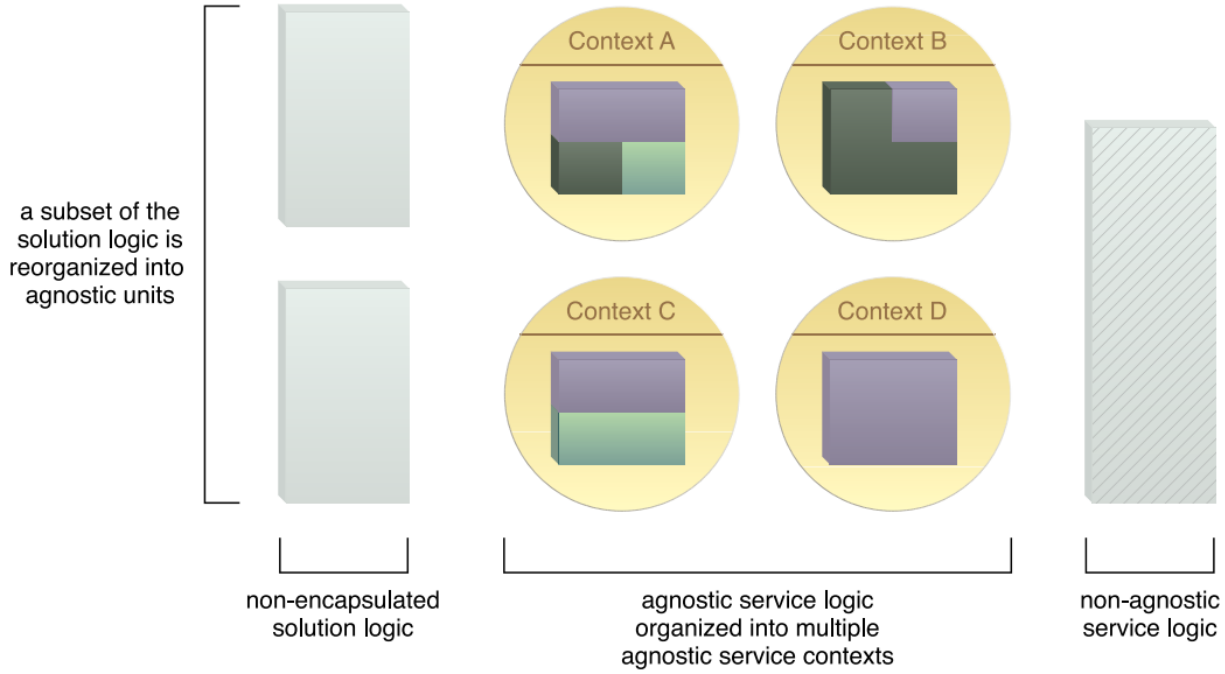

- **Service Encapsulation**
  - Second step in service modeling process (**Figure 5.1**)

# Agnostic Context




1. Identify the parts of logic that are not specific concerns

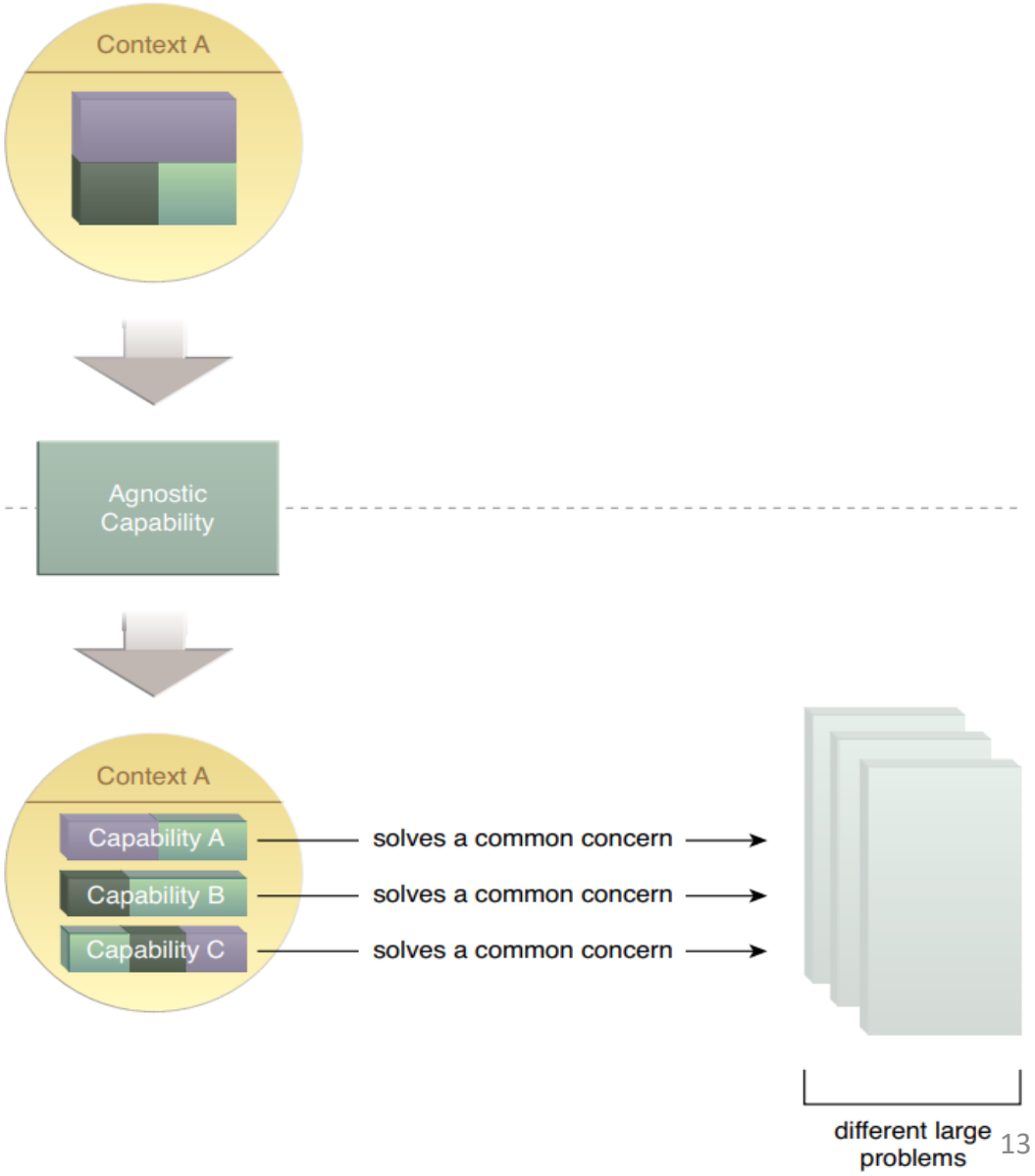
2. Separate and reorganize the appropriate logic into a set of agnostic contexts



# Agnostic Capability

- **Within each agnostic service context, the logic is further organized into a set of agnostic service capabilities**

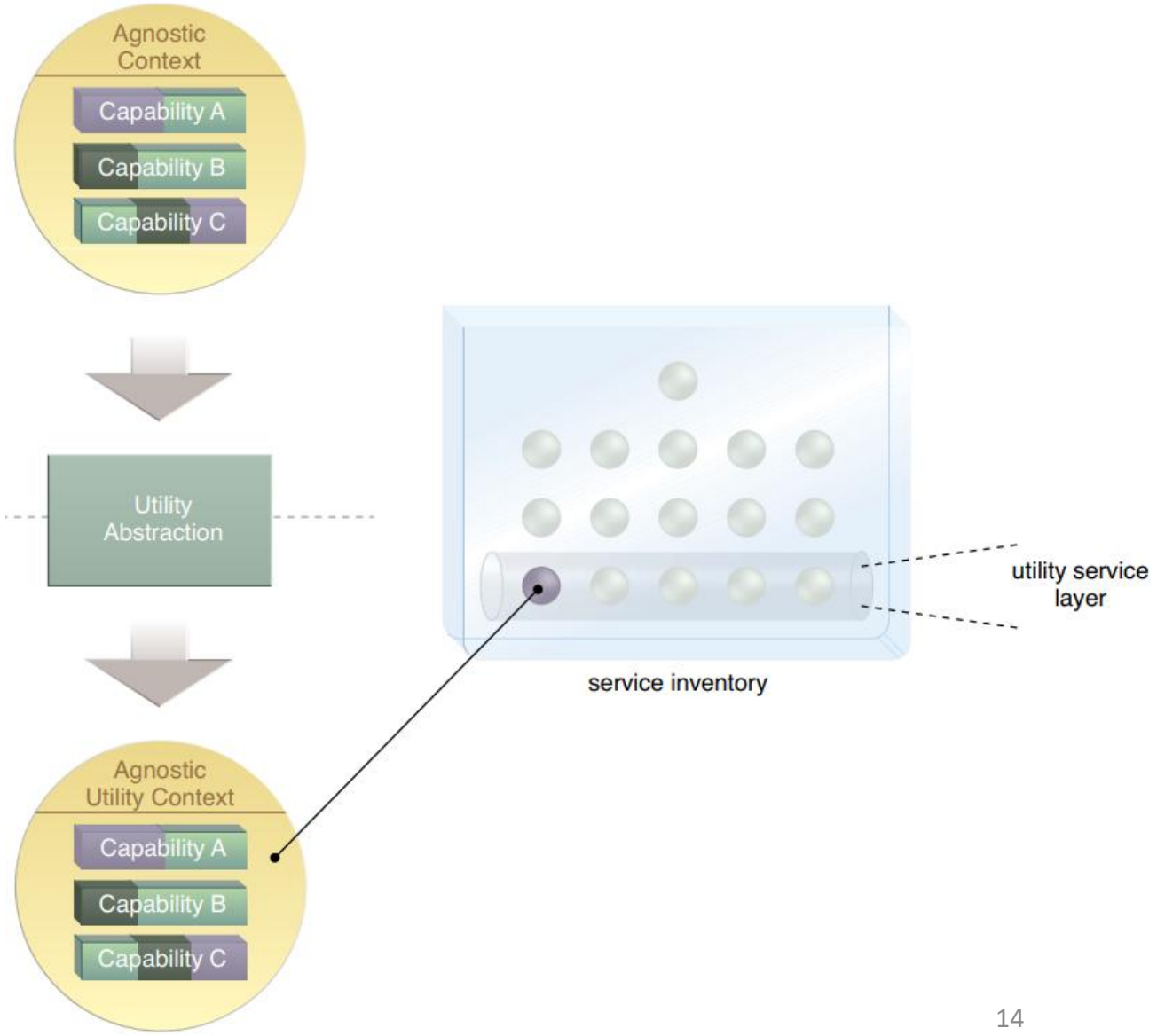
- **Because they are agnostic, the capabilities are multipurpose and can be reused to solve multiple concerns**



# Utility Abstraction

- **Utility Abstraction step to separate common, cross-cutting functionality that is neither specific to a business process nor a business entity**

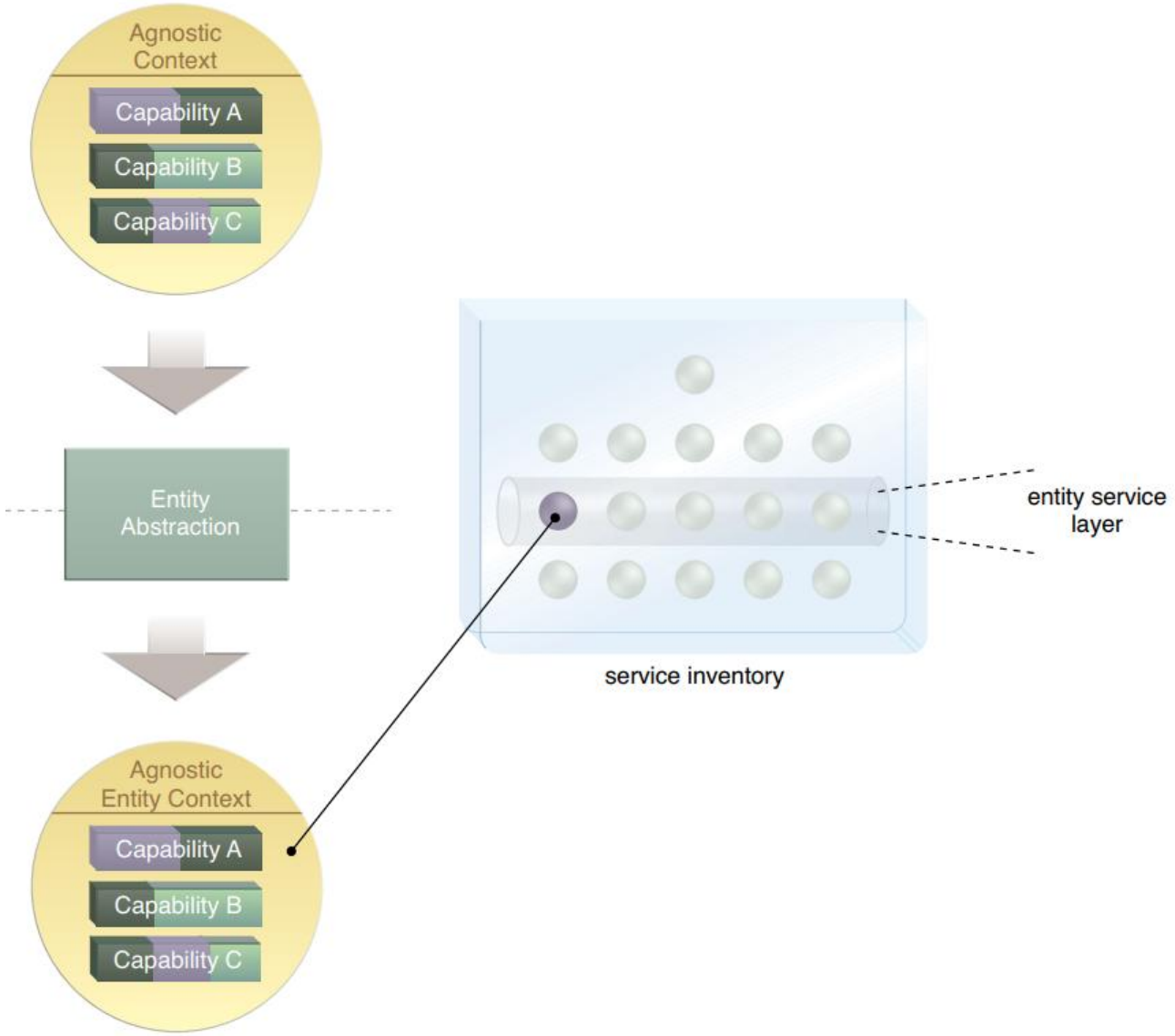
- **Repeating this step can result in the creation of multiple utility service candidates for the utility service layer**



# Entity Abstraction

- **Entity Abstraction step is focused on shaping the functional context of a service so that it is limited to logic that relates to one or more business entities.**

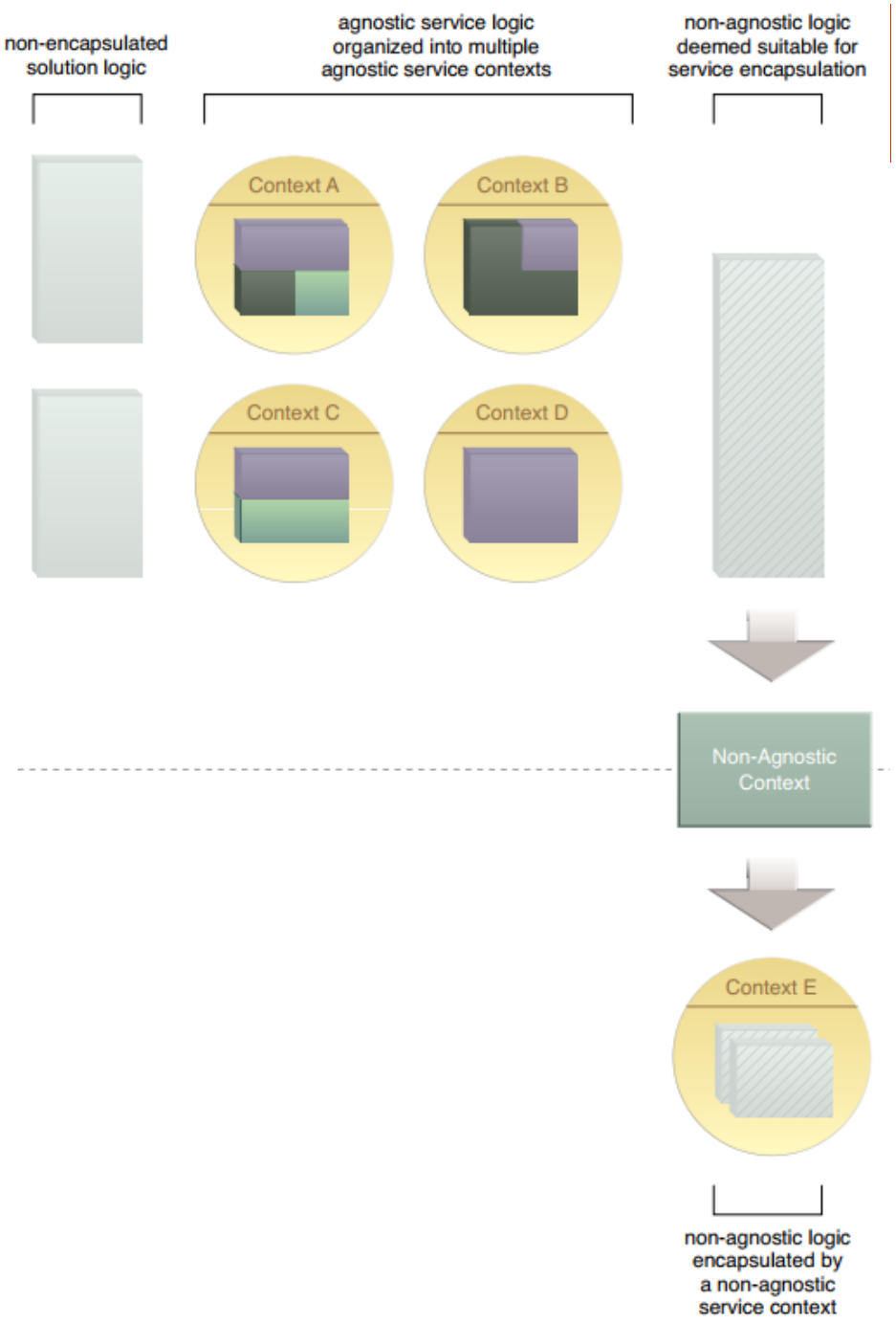
- **As with utility abstraction, repeating this step can result in establishing entity service layer**



# Non-Agnostic Context

- **After the multi-purpose logic has been separated, the remaining logic is specific to the business process.**

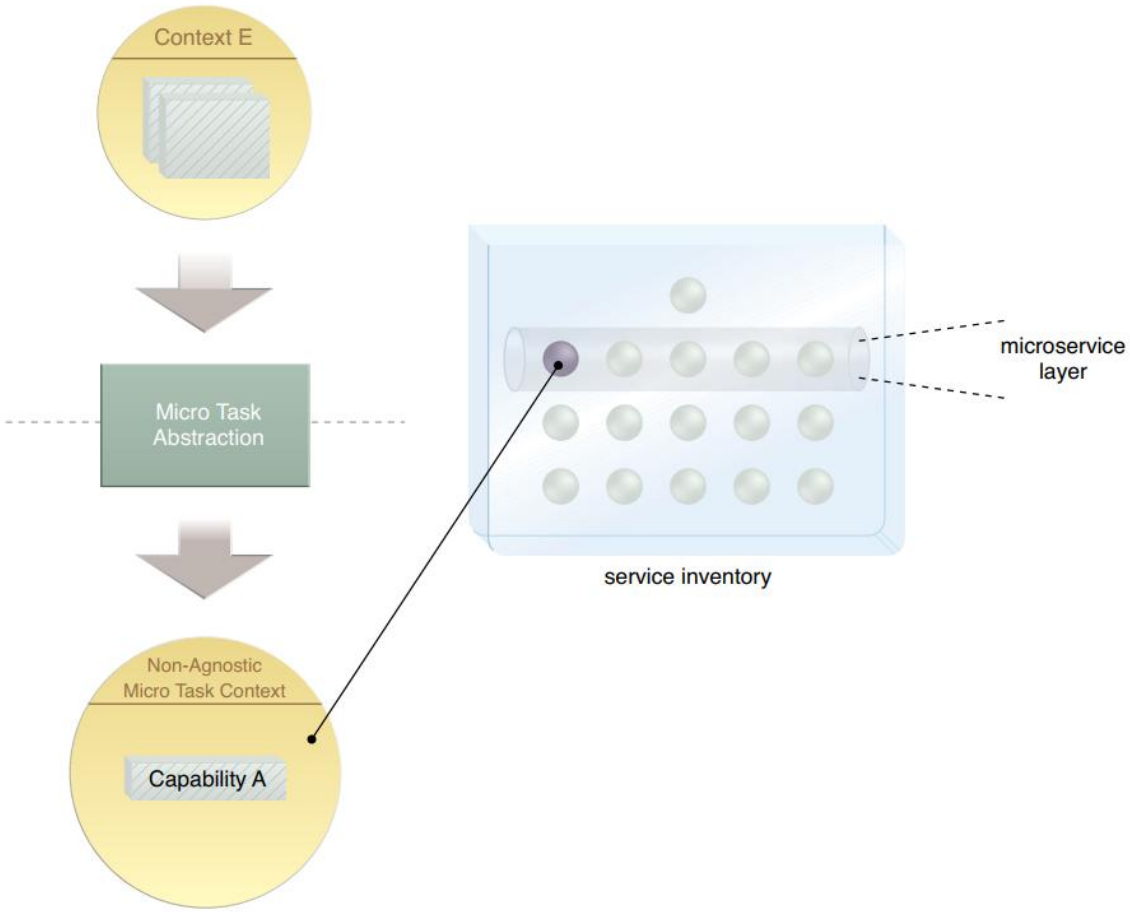
- **These logic are considered single-purpose in nature, it is classified as non-agnostic**



# Micro Task Abstraction

- **The subsets of non-agnostic logic (or "micro tasks") may have specific performance or reliability requirements.**

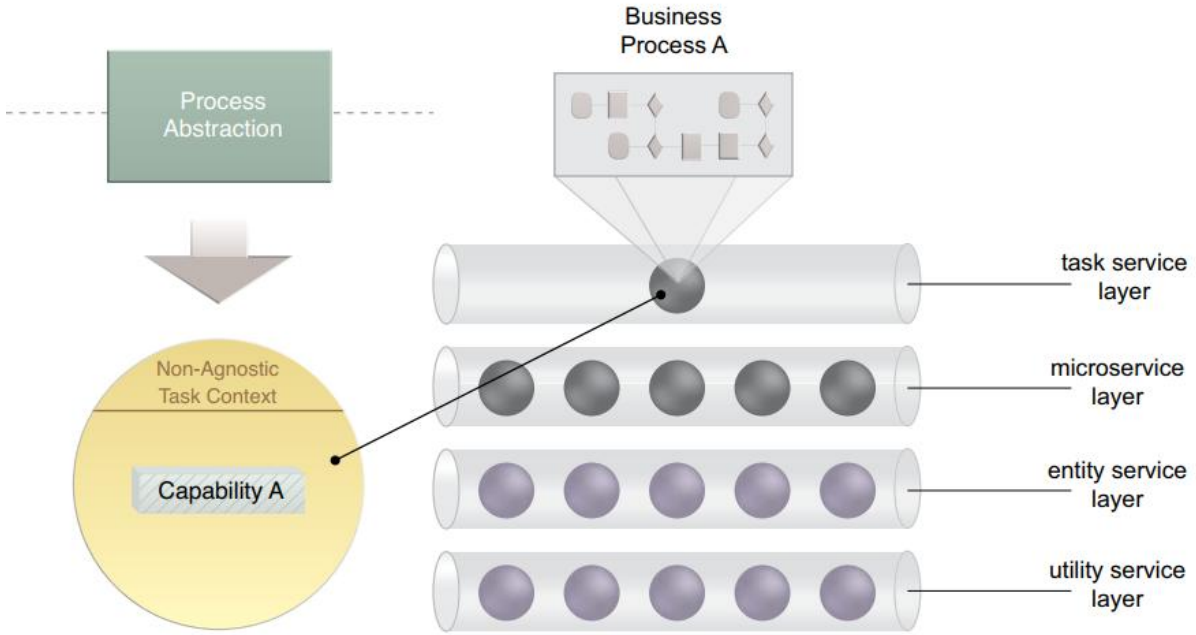
- **This type of processing logic can be abstracted into a separate service layer, called microservice.**



# Process Abstraction and Task Services

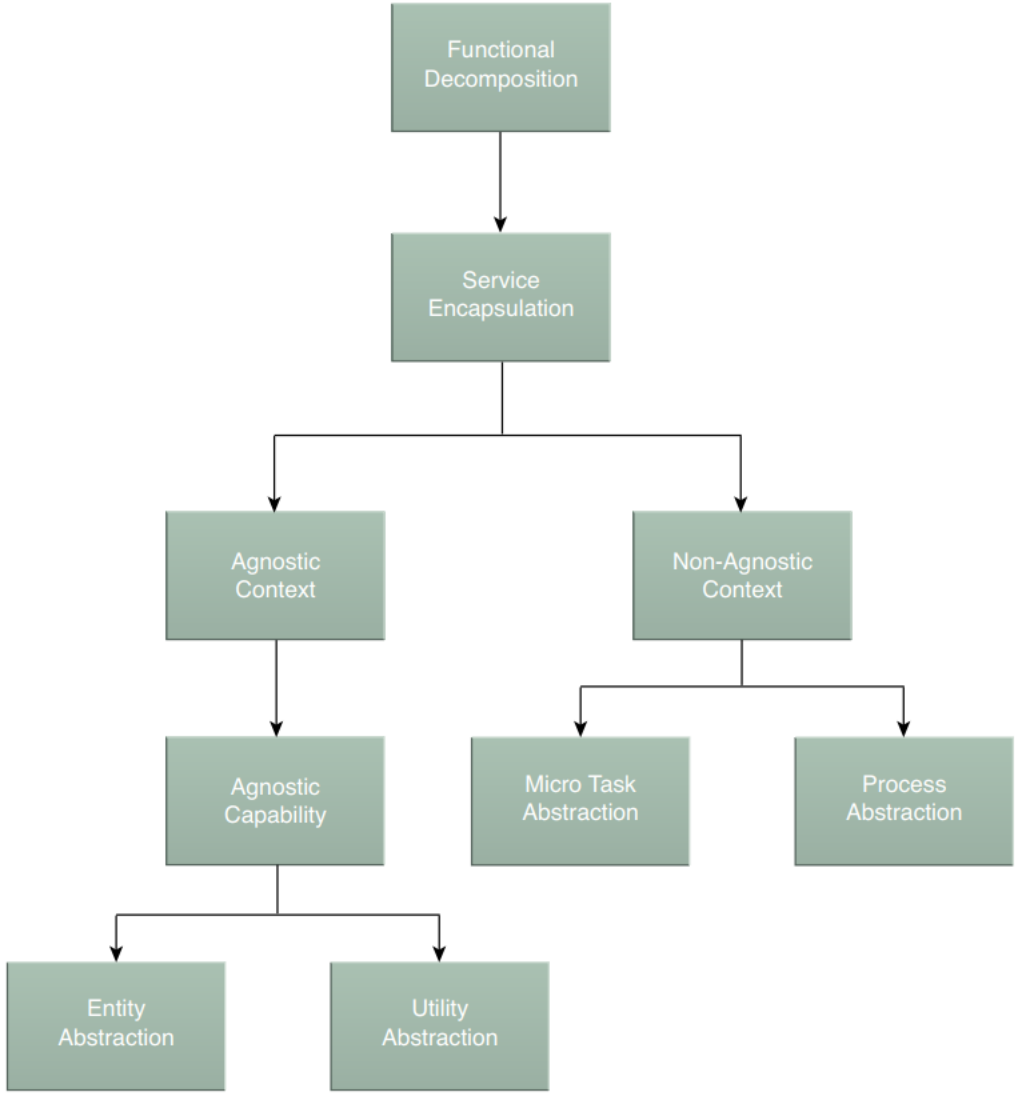
- **Abstracting the remaining business process-specific logic will typically result in the creation of a task service**

- **Task services generally encapsulate composition logic or other forms of logic which responsible automate the business process.**
  - As a role known as the composition controller



# Breaking Down the Business Problem

1. **Functional Decomposition**
2. **Service Encapsulation**
3. **Agnostic Context**
   - 3.1. Agnostic Capability
   - 3.2 Entity Abstraction
   - 3.3 Utility Abstraction
4. **Non-Agnostic Context**
   - 4.1 Micro task Abstraction
   - 4.2 Process/Task Abstraction

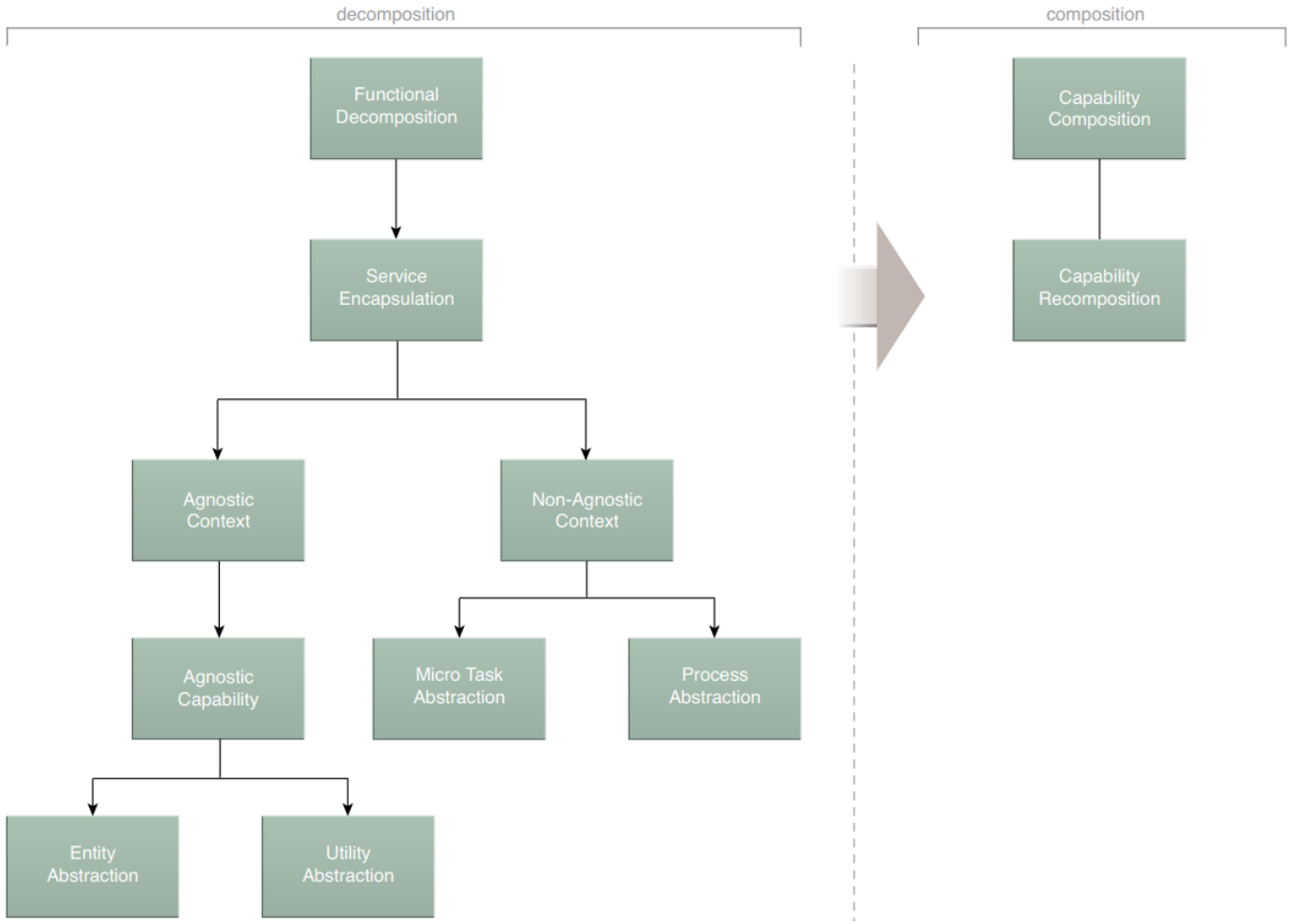


# 5.3 Building Up the Service-Oriented Solution

# Capability Composition and Capability Recomposition

- **In previous process steps, logic has only been separated into individual functional contexts and capabilities.**

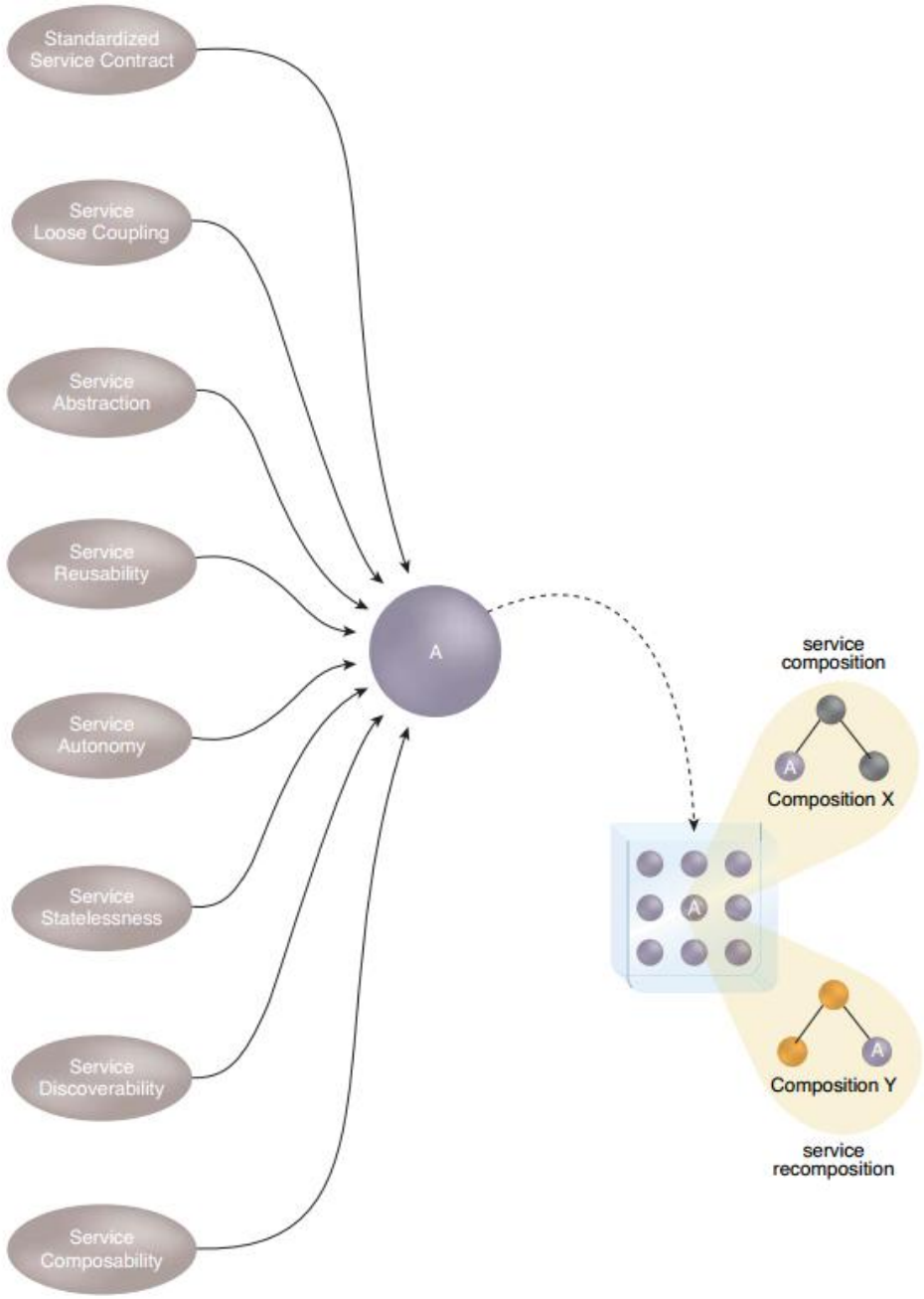
- **The next process steps are focused on building process via the composition and recomposition of service capability candidates**



# Service-Orientation and Service Composition

- **The service-orientation design paradigm is naturally focused on enabling flexible composition**

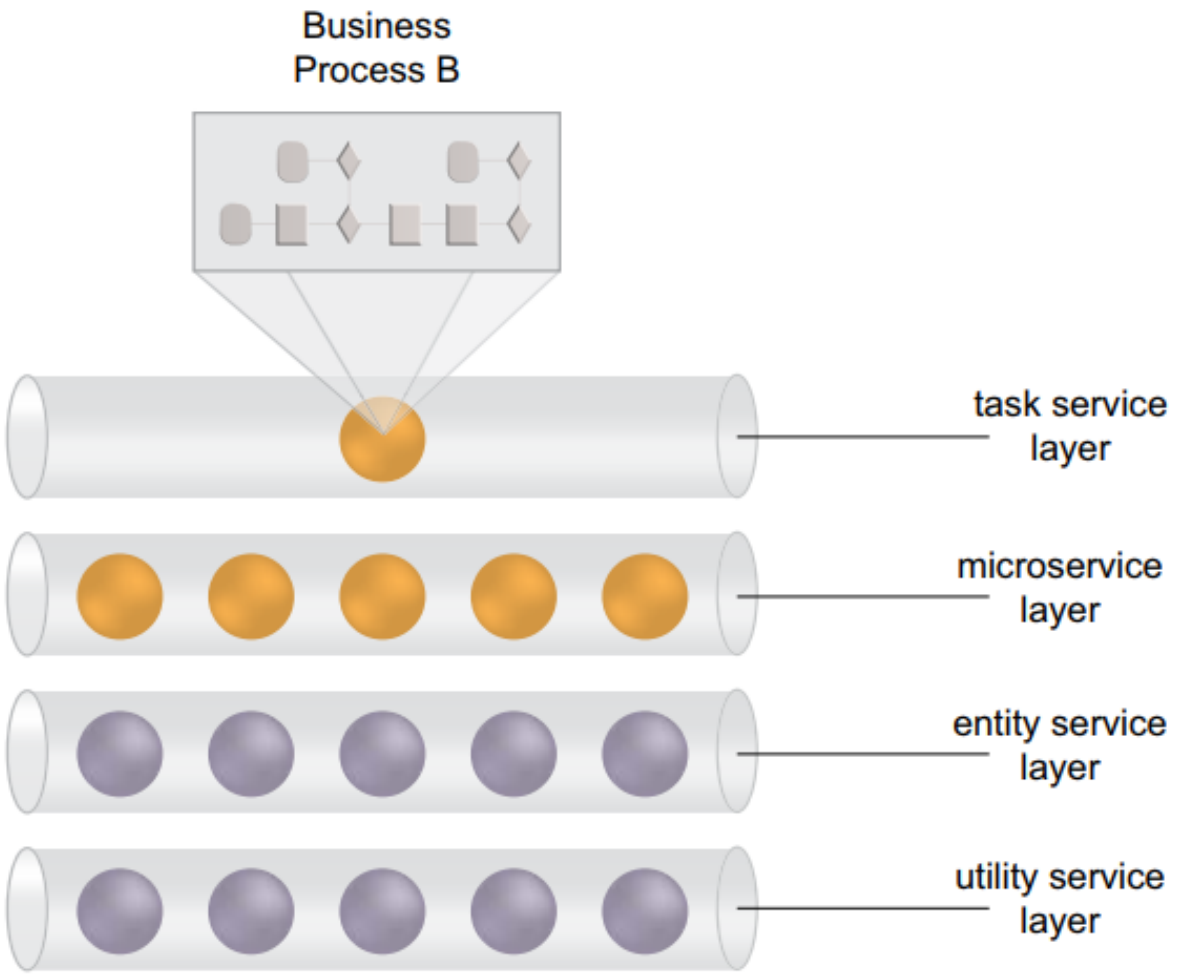
- **The difference between SOA with another distributed technology architecture is how it positions agnostic services in the way they are repeatedly composable**



# Service-Orientation and Service Composition

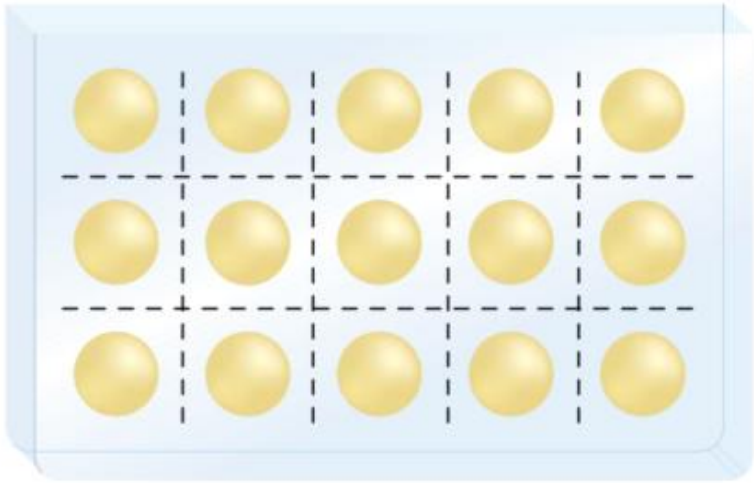
- **Service-Orientation ensures Services are naturally interoperable and designed for participation in complex service compositions**

- **Enabling them to fulfill new business requirements and automate new business processes**



# Logic Centralization and Service Normalization

- **The less functional overlap that is allowed in a service inventory, the less redundant logic exists -> The more normalized the service inventory becomes**



normalized service inventory

- **Logic centralization is a technique that supports service inventory normalization**

# Q & A