Enterprise Architecture

Dr. Adnan Albar

Faculty of Computing & Information Technology
King AbdulAziz University - Jeddah

Enterprise Architecture Methods

Lecture 5

Week 5 Slides

King AbdulAziz University - FCIT

Overview

- Description Languages for Business & IT Domains
- IDEF
- BPMN
- Testbed
- ARIS
- Unified Modeling Language
- Service-Oriented Architecture (SOA)

Description Languages

- In domains such as business process design and software development, we find established description languages for modeling these domains.
- For software modeling, UML is of course, the single dominant language.
- In organization and process modeling, on the other hand, a multitude of languages are in use: there is no standard for models in this domain.
- We will focus on languages that either find widespread use or have properties that are interesting from the perspective of our goals in developing an enterprise architecture language.

IDEF – Integrated DEFinition Methods

- IDEF is a family of languages
- Used to perform enterprise modeling and analysis
- Currently, there are 16 IDEF methods. Of these methods, IDEF0, IDEF3, and IDEF1X (‘the core’) are the most commonly used.

IDEF – The Scope it Covers

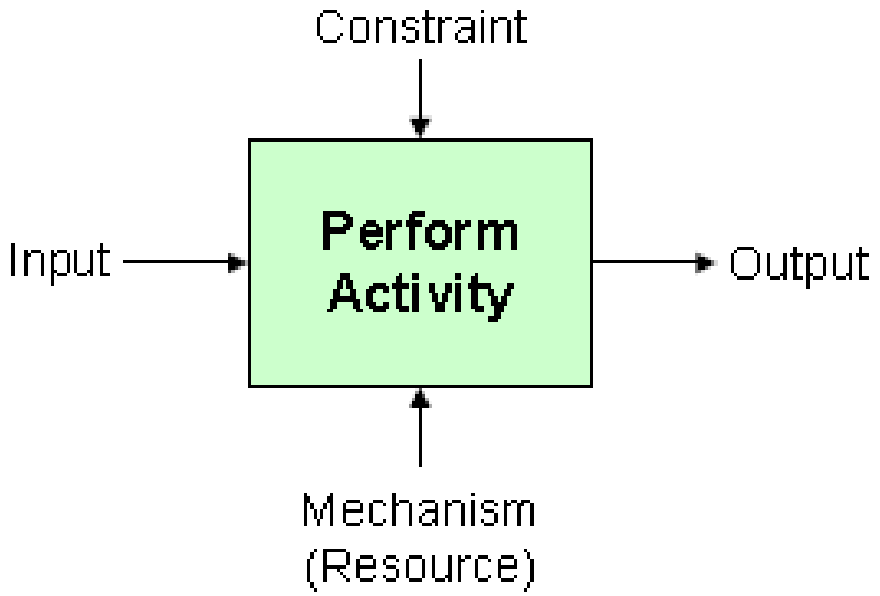
- ***Functional modeling***, **IDEF0: The idea behind IDEF0 is to model the elements controlling the execution of a function, the actors performing the function, the objects or data consumed and produced by the function, and the relationships between business functions (shared resources and dependencies).**

- ***Process modeling***, **IDEF3: IDEF3 captures the workflow of a business process via process flow diagrams. These show the task sequence for processes performed by the organization, the decision logic, describe different scenarios for performing the same business functions, and enable the analysis and improvement of the workflow.**

IDEF – The Scope it Covers

- ***Data modeling***,

IDEF1X: IDEF1X is used to create logical data models and physical data models by the means of logical model diagrams, multiple IDEF1X logical subject area diagrams, and multiple physical diagrams.



IDEF0 representation

BPMN – Business Process Modeling Notation

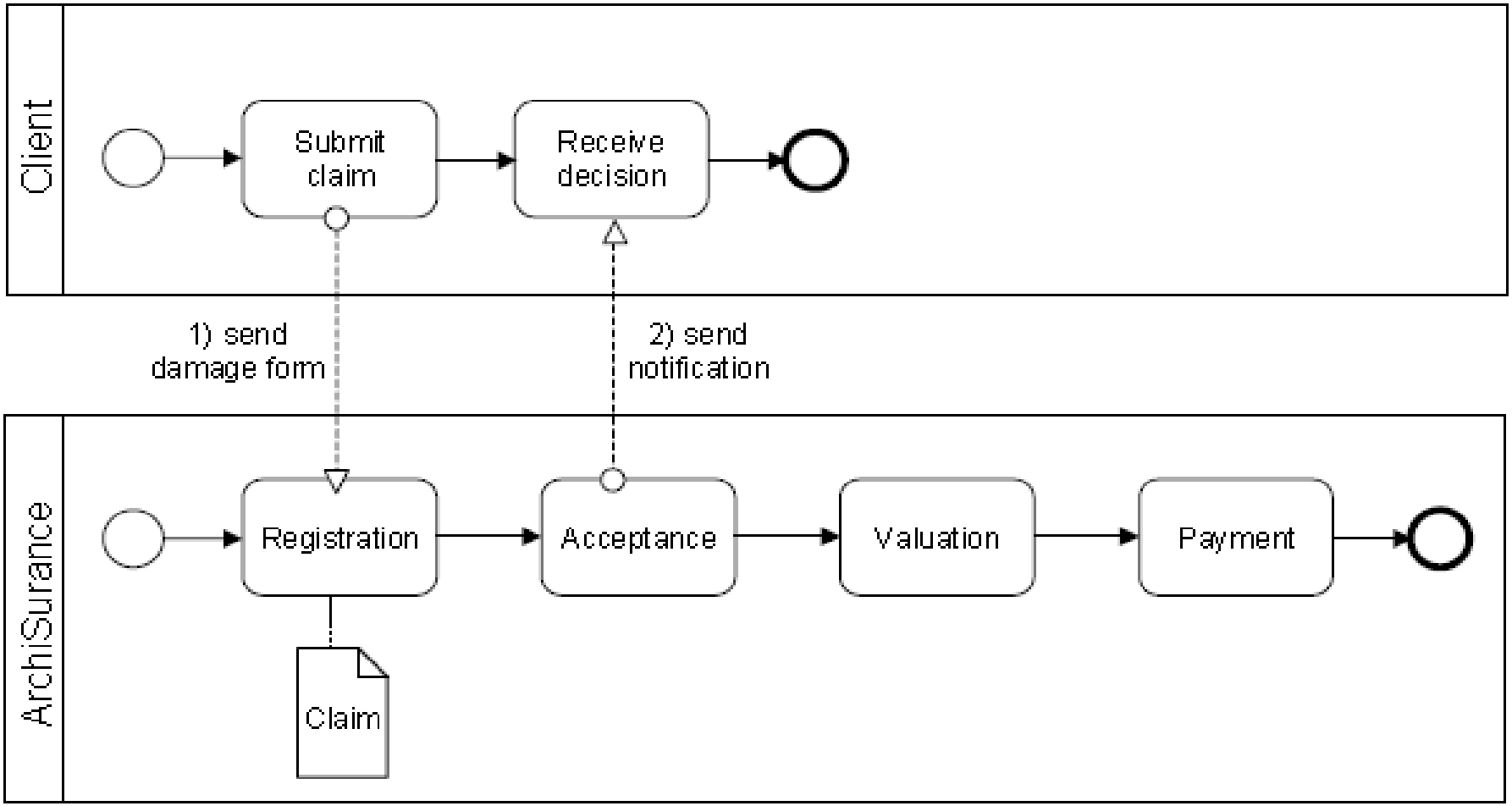
- developed by Business Process Management Initiative (BPMI)

- The BPMN standard (Object Management Group 2009) specifies a graphical notation that serves as a common basis for a variety of business process modeling and execution languages

- BPMN is restricted to process modeling

- BPMN provides a uniform notation for modeling business processes in terms of activities and their relationships

BPMN – Business Process Modeling Notation



Example model in BPMN

Testbed

- Developed by the Telemetric Institute together with a consortium of companies

- Its target users are mostly business consultants

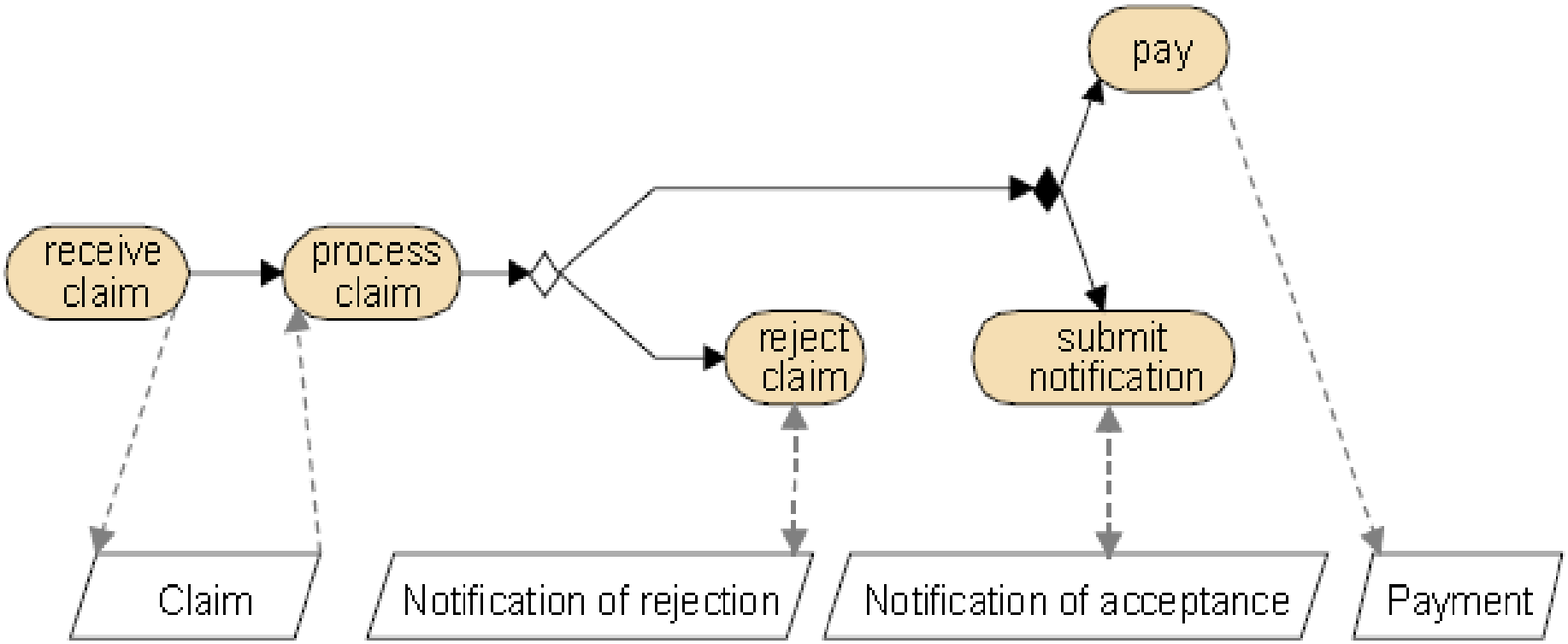
Testbed recognizes three aspect domains:

- the *actor domain*, which describes the resources for carrying out business activities

- the *behaviour domain*, which describes the business processes performed by the resources

- the *item domain*, which describes the data objects handled by business processes

Testbed



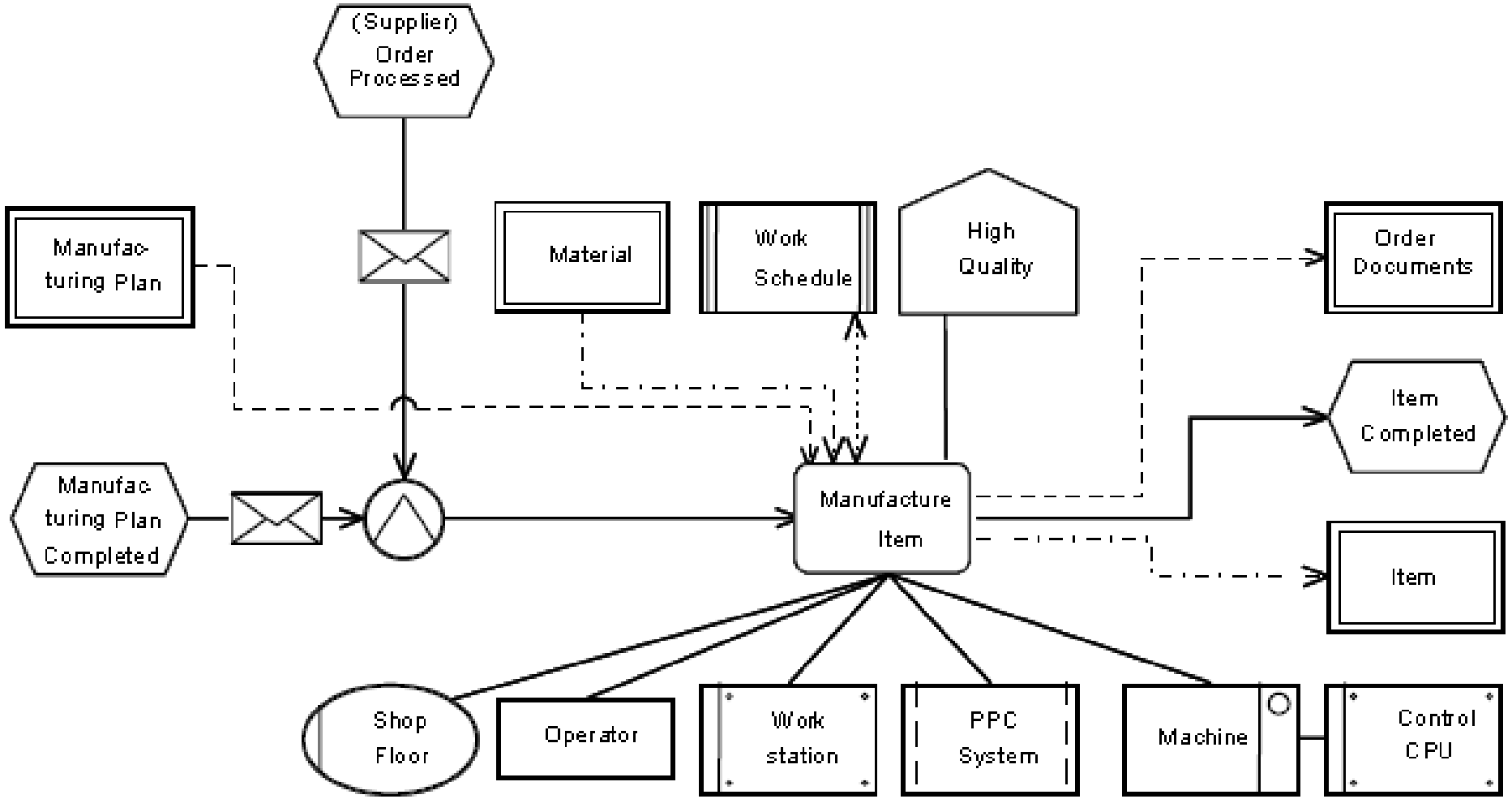
Example of a business process model in Testbed

ARIS - Architecture of Integrated Information Systems

- A well known approach to enterprise modeling
- ARIS is a business modeling method
- ARIS provides a modeling language known as event-driven process chains (EPCs)

The ARIS Toolset includes various editors that can be used to design and edit several types of diagrams. The most important are value-added chain diagrams, organizational charts, interaction diagrams, function trees, and EPCs

ARIS - Architecture of Integrated Information Systems



Events, functions and control flows in ARIS

Unified Modeling Language

- Currently the most important industry-standard language
- Specifies, visualizes, constructs, and document the artifacts of software systems
- UML covers all possible modeling domains one can think of
- UML is a rich combination of 13 sublanguages

Unified Modeling Language

The 13 diagrams in UML, can be grouped in three categories:

- ***structure:*** package diagrams, class diagrams, object diagrams, composite, structure diagrams;

- ***behaviour:*** use case diagrams, state diagrams, sequence diagrams, timing diagrams, communication diagrams, activity diagrams, interaction overview diagrams;

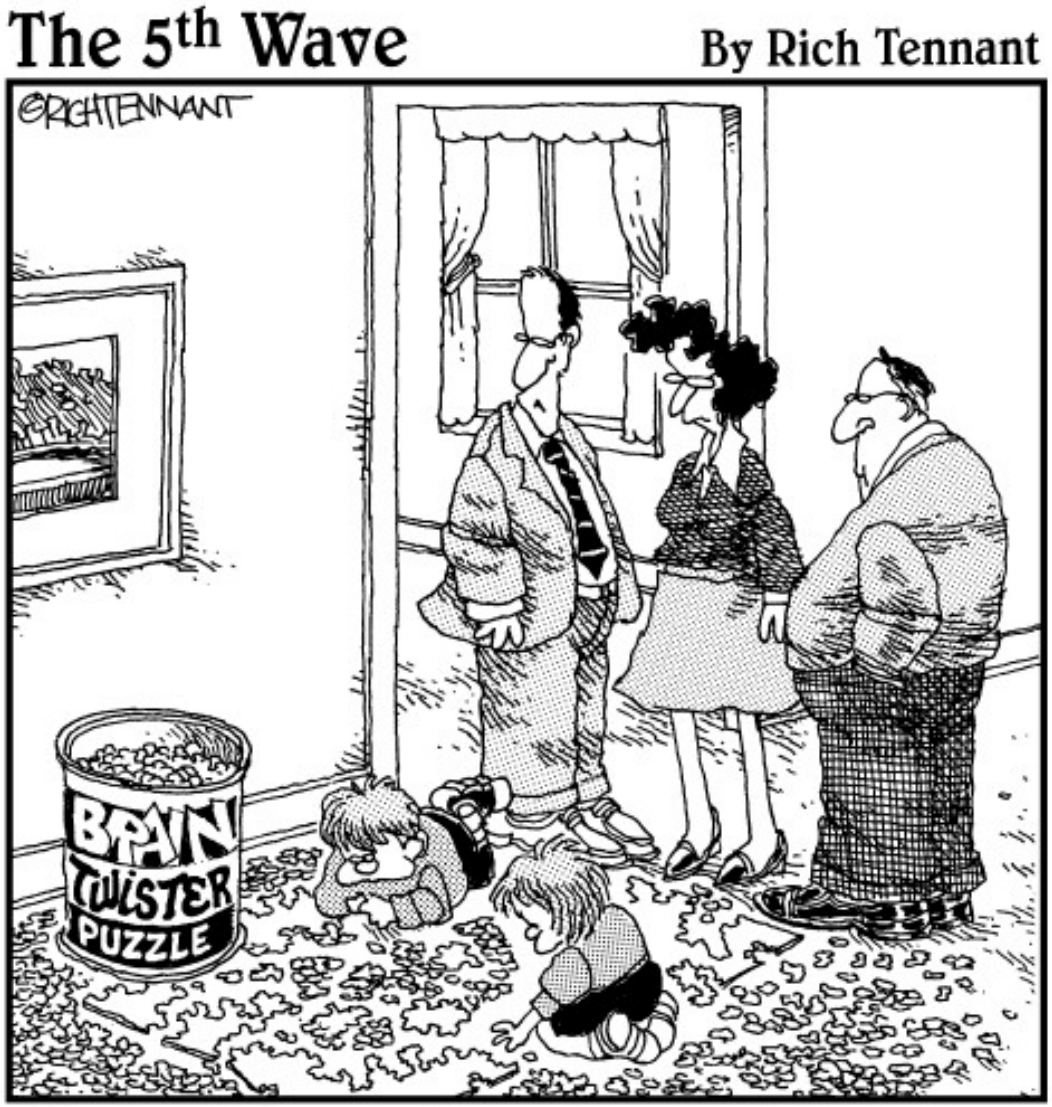
- ***implementation:*** component diagrams, deployment diagrams

SOA – Service Oriented Architectures

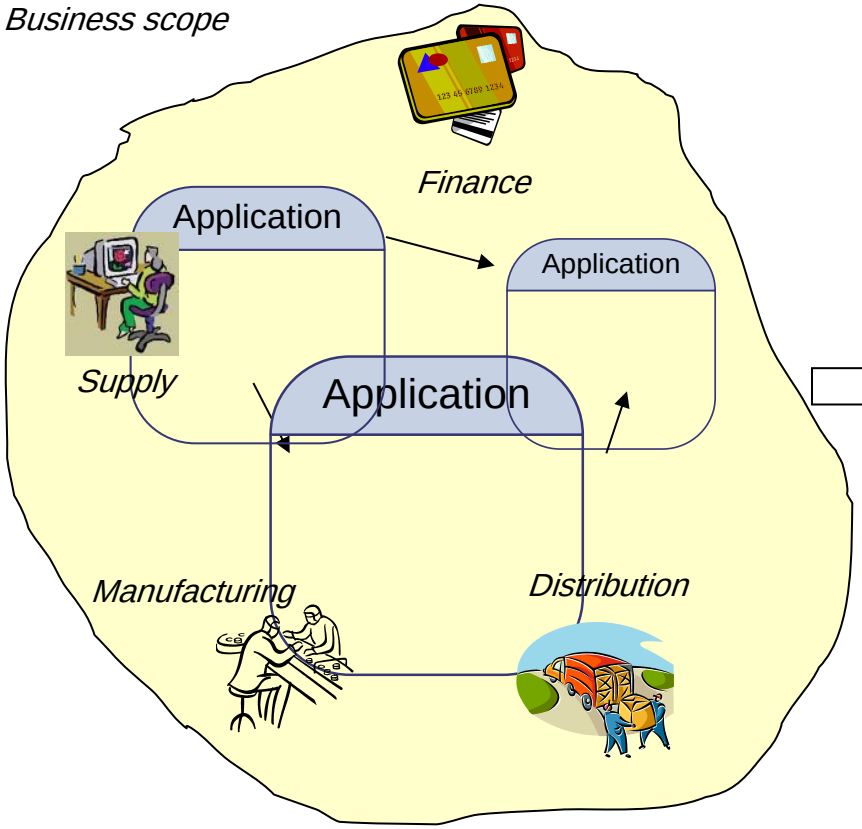
Overview:

- Motivation for Service Oriented Architecture (SOA)
- SOA Defined
- SOA Principles
- Applying SOA

Current Environment

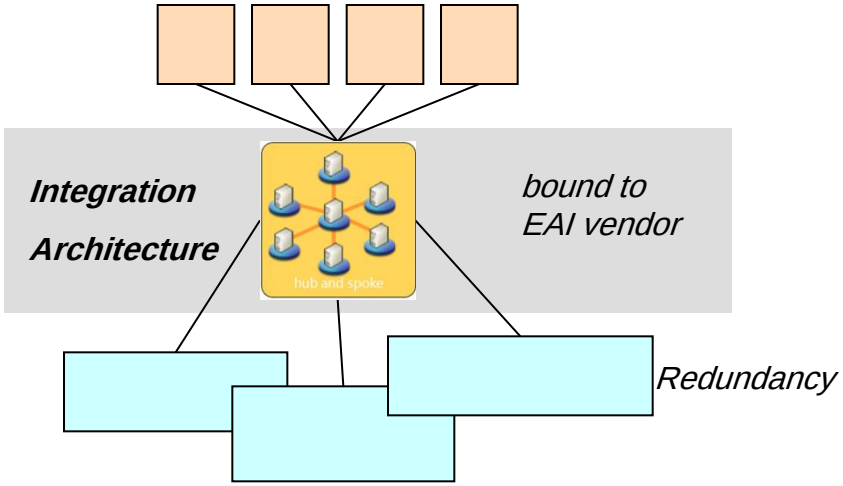


Application Centric



Business functionality is duplicated in each application that requires it.

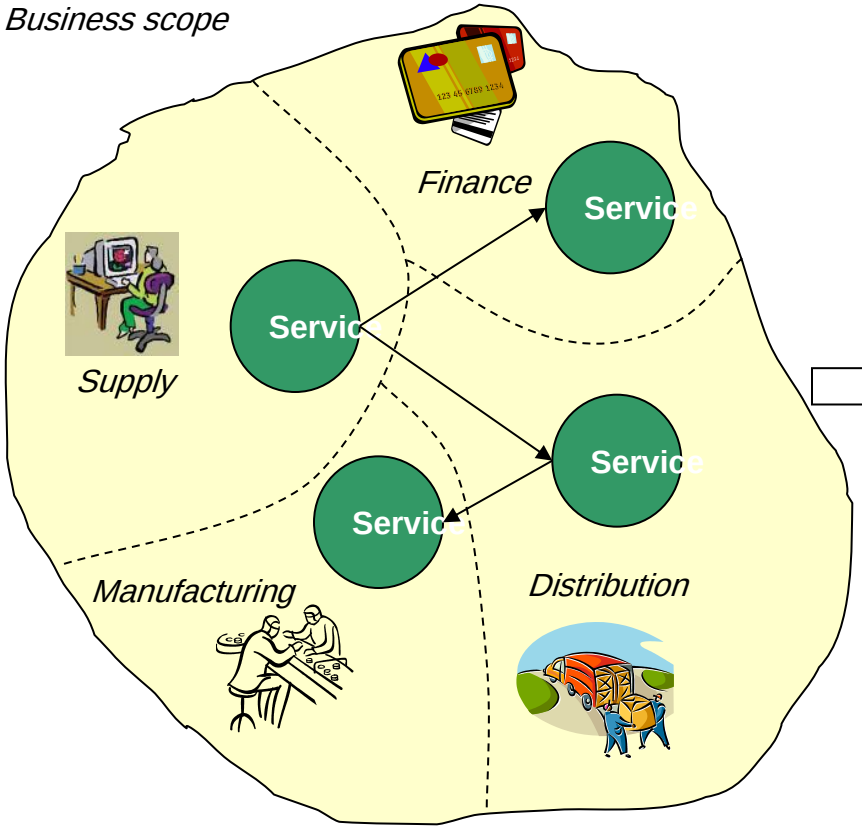
Narrow Consumers
Limited Business Processes



Overlapped resources
Overlapped providers

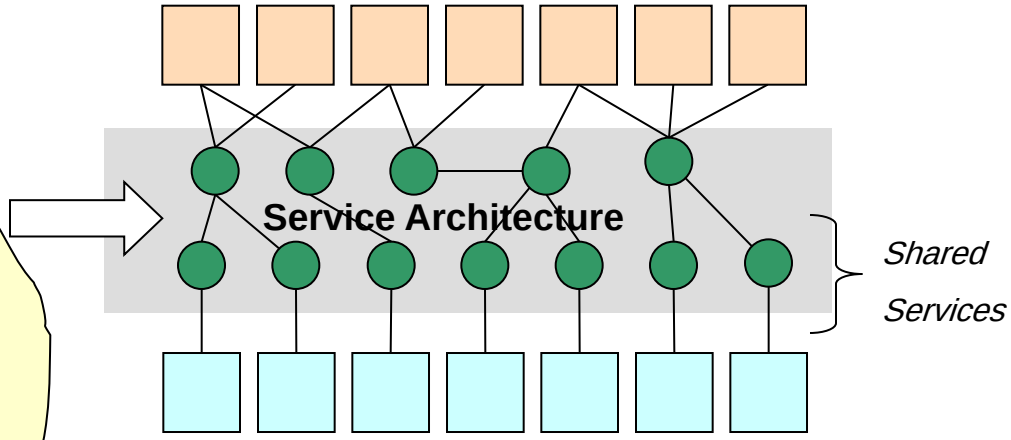
EAI 'leverage' application silos with the drawback of data and function redundancy.

Service Centric



SOA structures the business and its systems as a set of capabilities that are offered as Services, organized into a Service Architecture

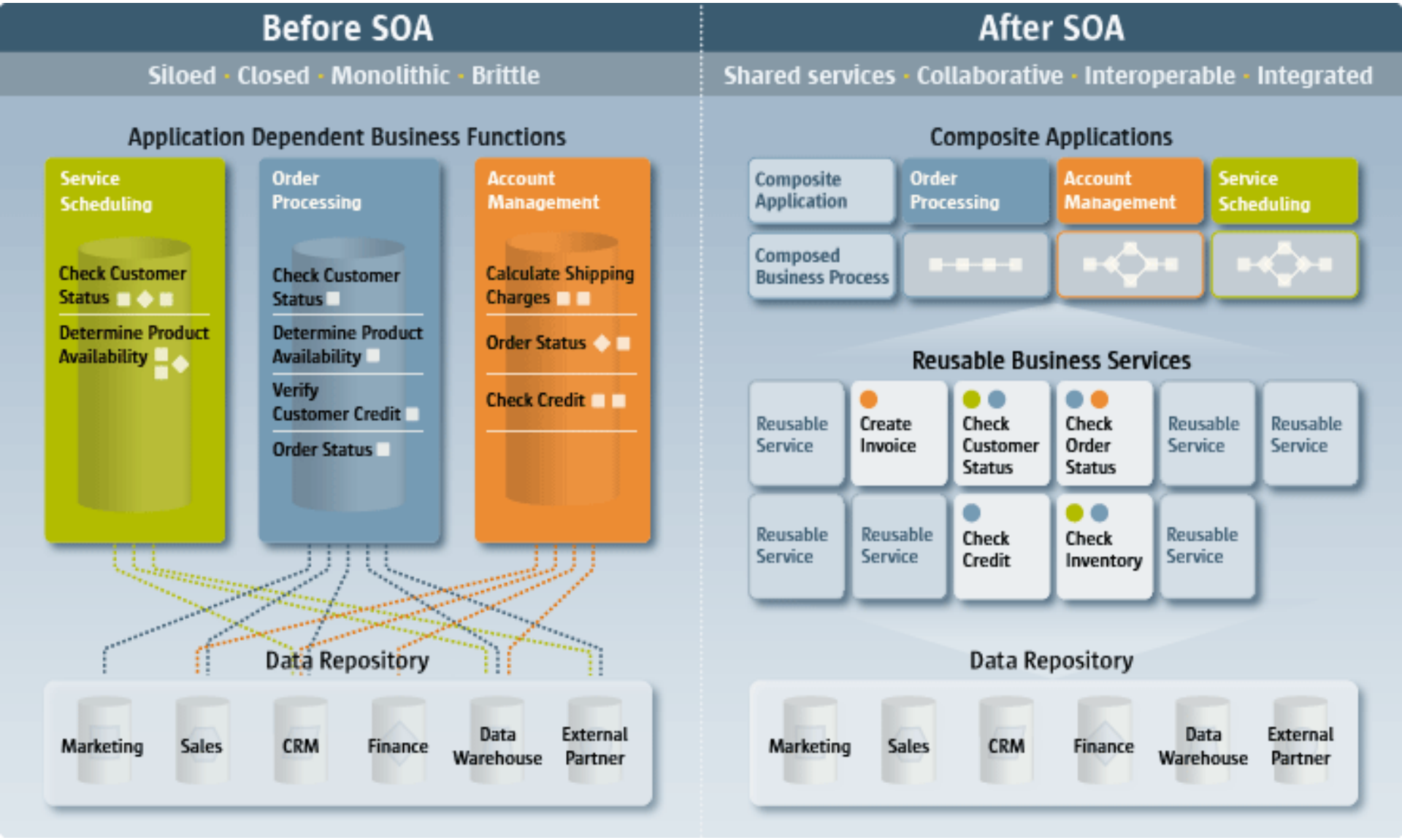
Multiple Service Consumers
Multiple Business Processes



Multiple Discrete Resources
Multiple Service Providers

Service virtualizes how that capability is performed, and where and by whom the resources are provided, enabling multiple providers and consumers to participate together in shared business activities.

Before SOA – After SOA

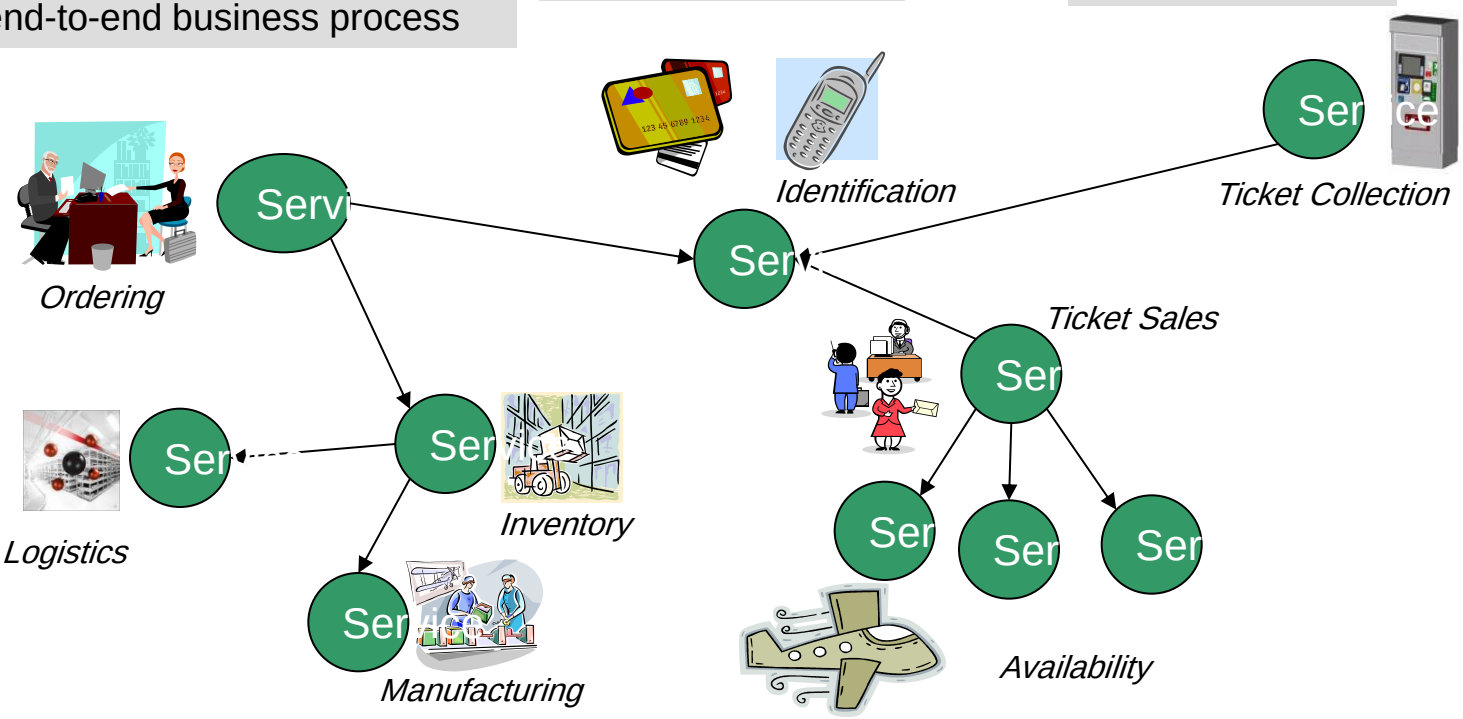


Why SOA? To enable Flexible, Federated Business Processes

Enabling a virtual federation of participants to collaborate in an end-to-end business process

Enabling alternative implementations

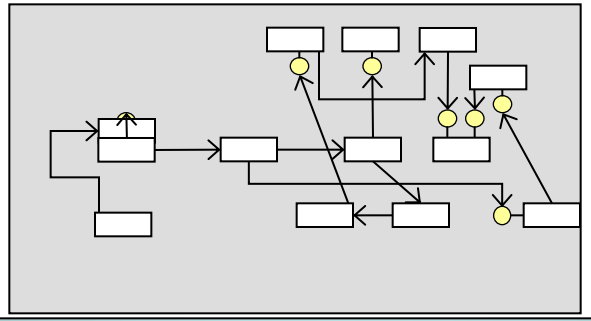
Enabling reuse of Services



Enabling virtualization of business resources

Enabling aggregation from multiple providers

Why SOA? To enable Business Process Optimization and the Real Time Enterprise (RTE)



BPM Expressed in terms of Services Provided/Consumed

Seamless End to End Process

Service to Customers

Service from Multiple Suppliers

Enterprise

Smart Clients

Stores POS

Mobile

3rd Party Agents

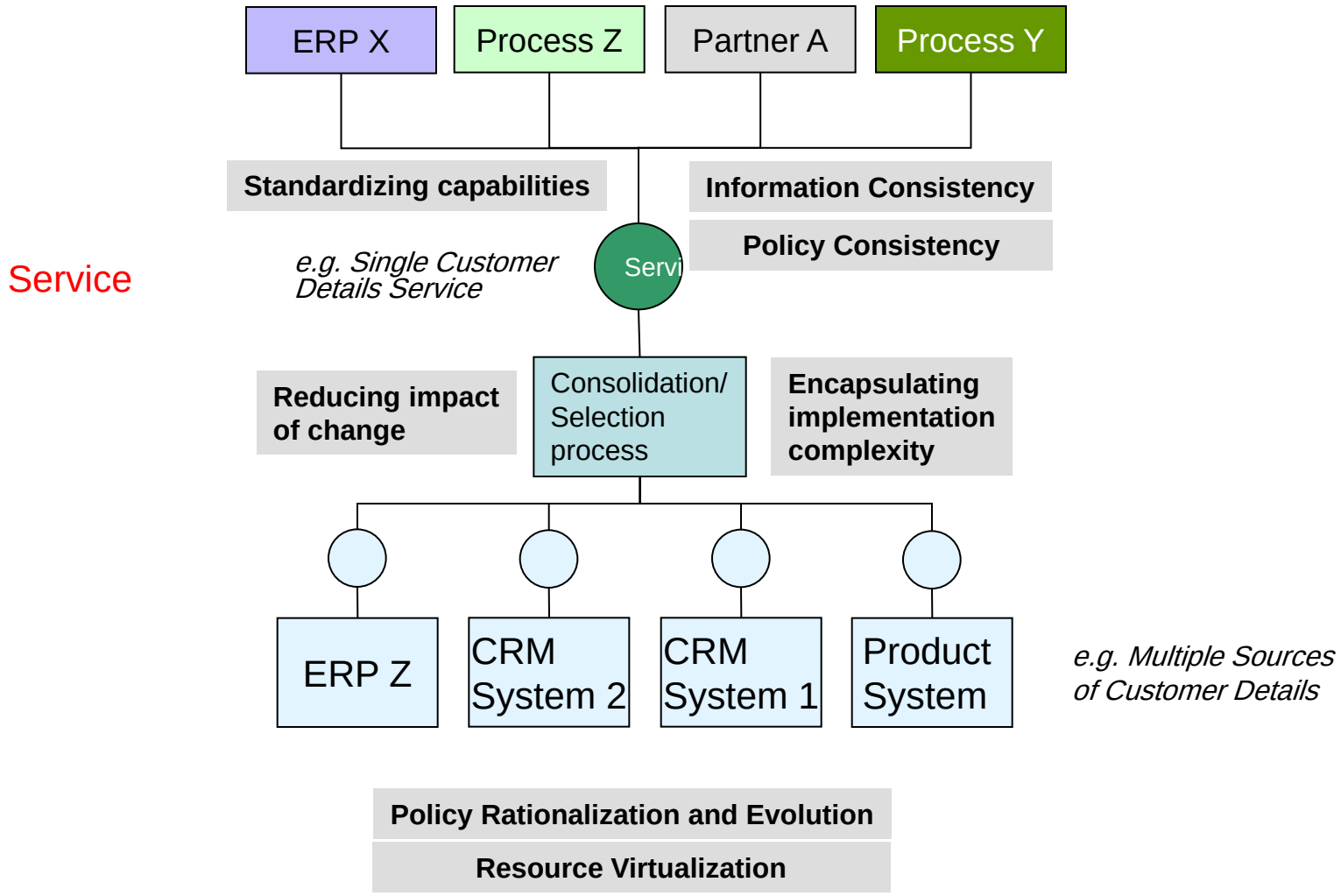
Portal

Internal Systems

SOA Patterns: Single, Multi-Channel Service for consistency

SOA Pattern: Standardized Service provided by multiple suppliers

Why SOA?
Enable Structural Improvement



SOA Defined

- SOA is a software architecture model
  - in which business functionality are logically grouped and encapsulated into
    - self contained,
    - distinct and reusable units

      called **<u>services</u> that**
    - represent a high level business concept
    - can be distributed over a network
    - can be reused to create new business applications
    - contain contract with specification of the purpose, functionality, interfaces (coarse grained), constraints, usage

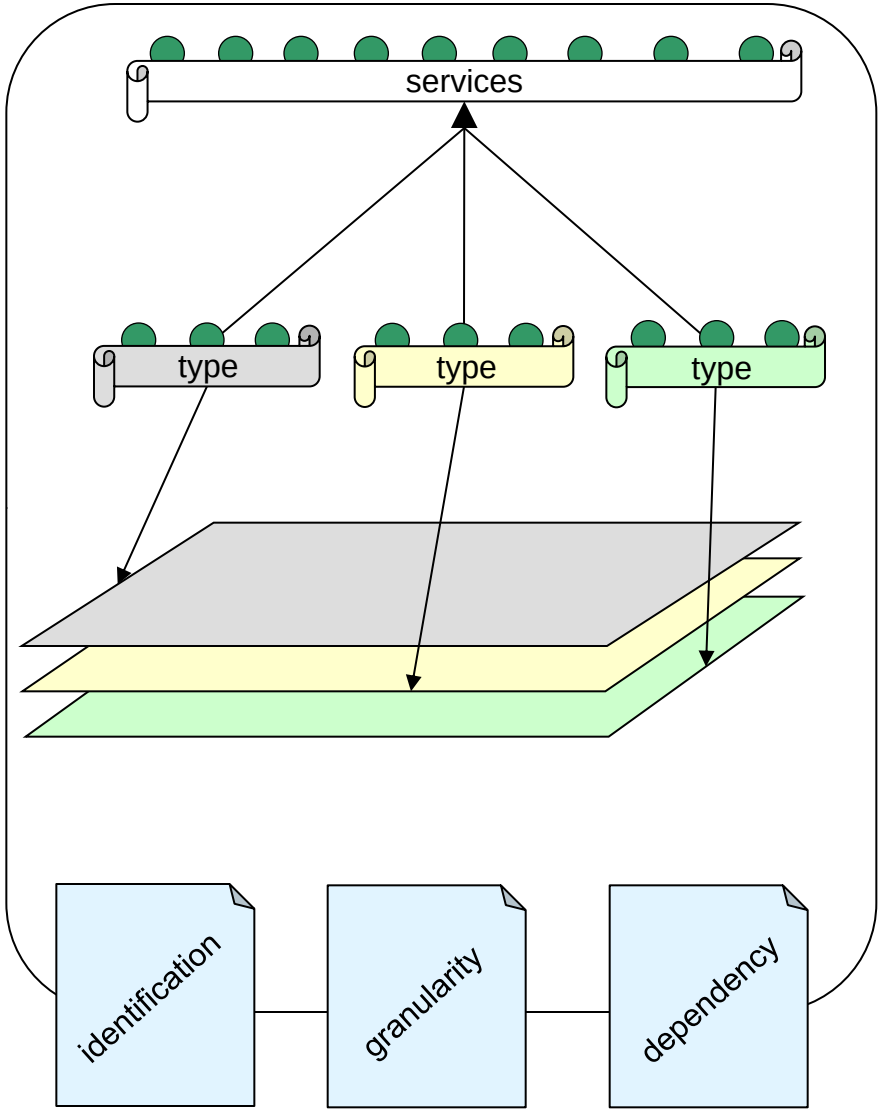
      ... of the business functionality

Services are autonomous, discrete and reusable units of business functionality exposing its capabilities in a form of contracts.

Services can be independently evolved, moved, scaled even in runtime.

What is Service Architecture?

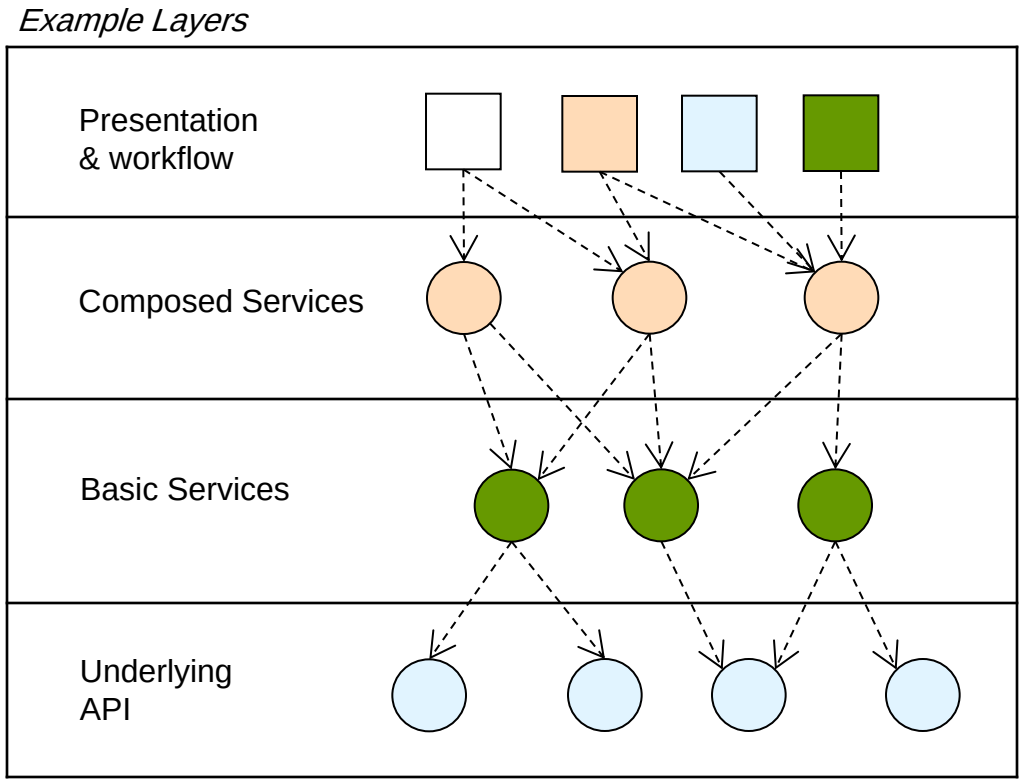
- A collection of services
- classified into types
- arranged into layers
- Governed by architectural patterns and policies



Service Architecture Organized by Layers

Reasons for Layering

1. Flexible composition.
2. Reuse.
3. Functional standardization in lower levels
4. Customization in higher layers
5. Separation of Concerns.
6. Policies may vary by Layer



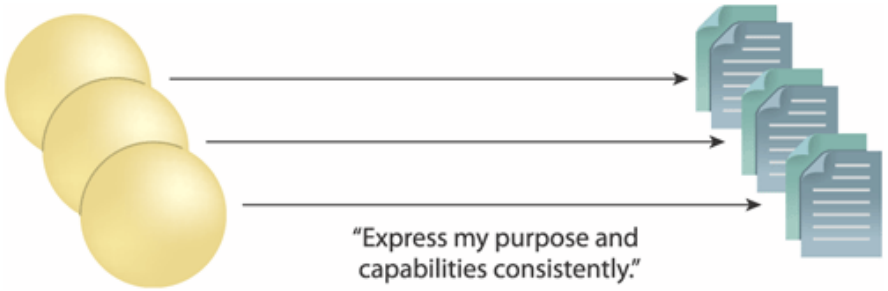
SOA Principles

- Standardized Service Contracts
- Loose Coupling
- Abstraction
- Reusability
- Autonomy
- Statelessness
- Discoverability
- Composability

Standardized Service Contracts

"*Services within the same service inventory are in compliance with the same contract design standards*."

- Services use service contract to
  - Express their purpose
  - Express their capabilities

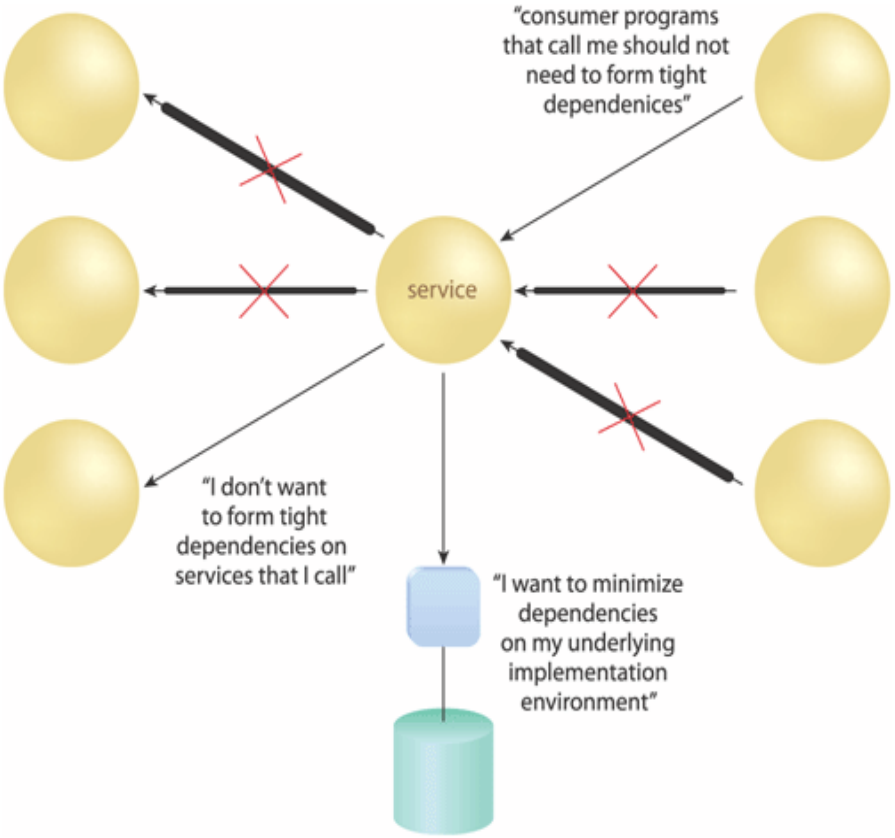


- Use formal, standardized service contracts
- Focus on the areas of
  - Functional expression
  - Data representation
  - Policy

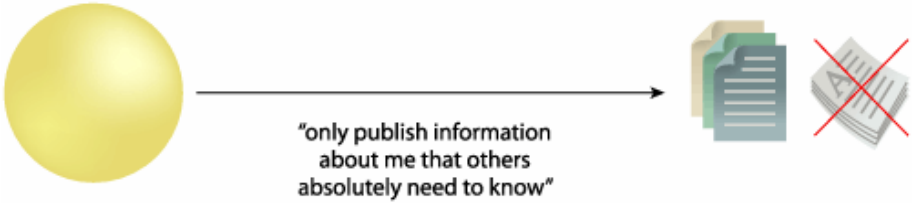
Loose Coupling

- "*Service contracts impose low consumer coupling requirements and are themselves decoupled from their surrounding environment."*

- Create specific types of relationships within and outside of service boundaries with a constant emphasis on reducing ("loosening") dependencies between
  - Service contract
  - Service implementation
  - Service consumers



Abstraction

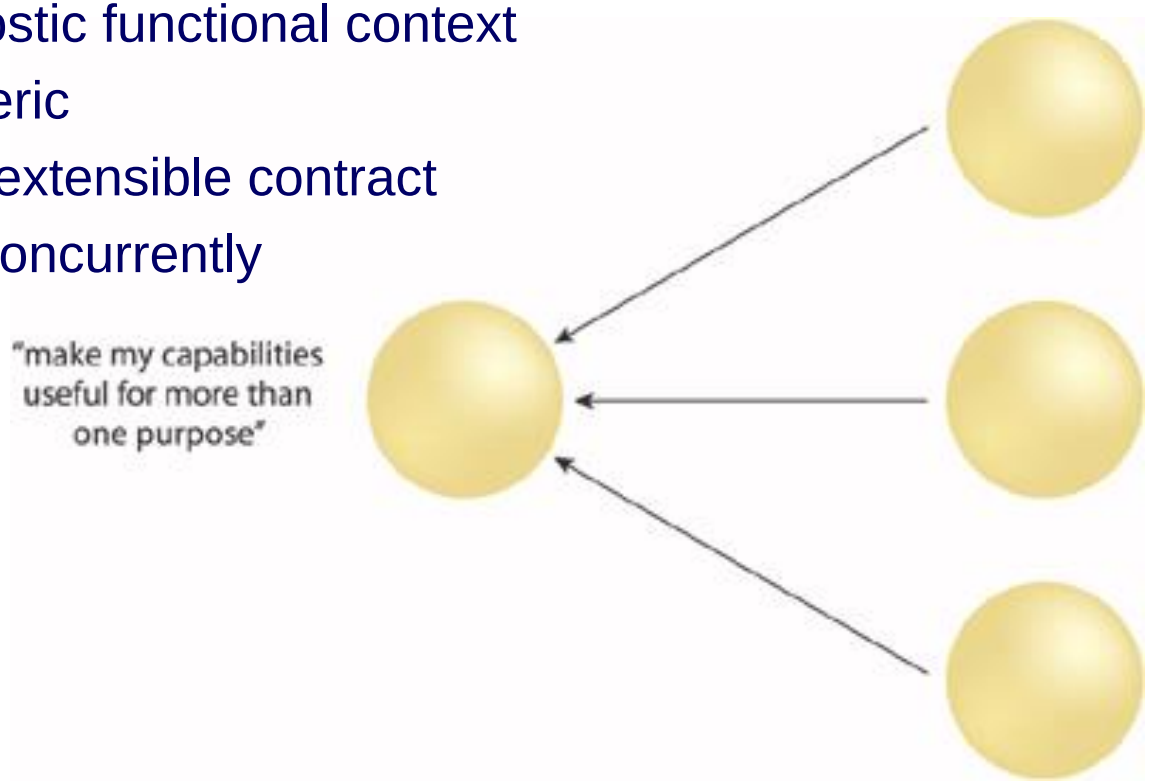


- "*Service contracts only contain essential information and information about services is limited to what is published in service contracts*"
- Avoid the proliferation of unnecessary service information, meta-data.
- Hide as much of the underlying details of a service as possible.
  - *Enables and preserves the loosely coupled relationships*
  - *Plays a significant role in the positioning and design of service compositions*

Reusability

- "*Services contain and express agnostic logic and can be positioned as reusable enterprise resources."*

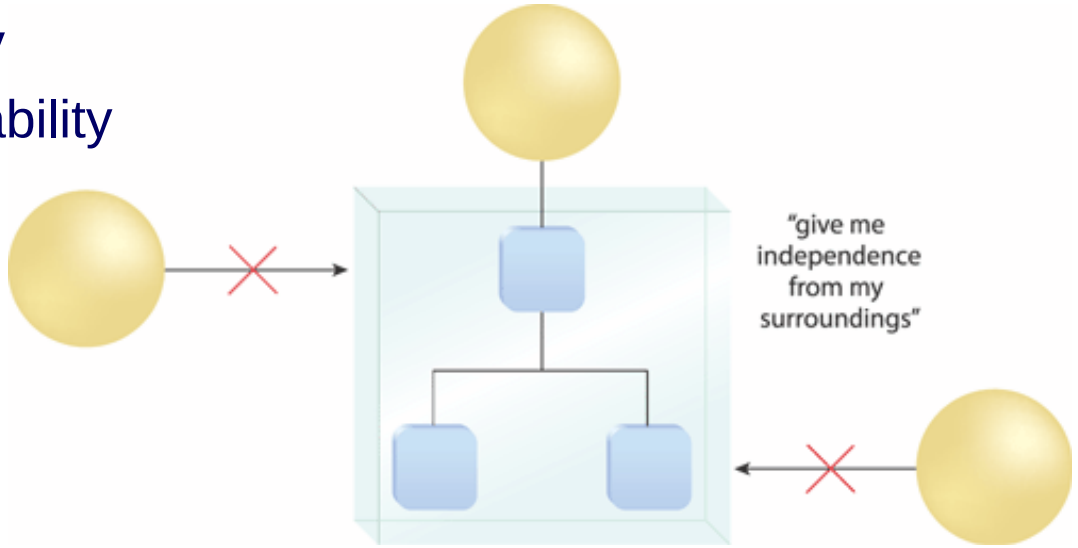
- Reusable services have the following characteristics:
  - Defined by an agnostic functional context
  - Logic is highly generic
  - Has a generic and extensible contract
  - Can be accessed concurrently



Autonomy

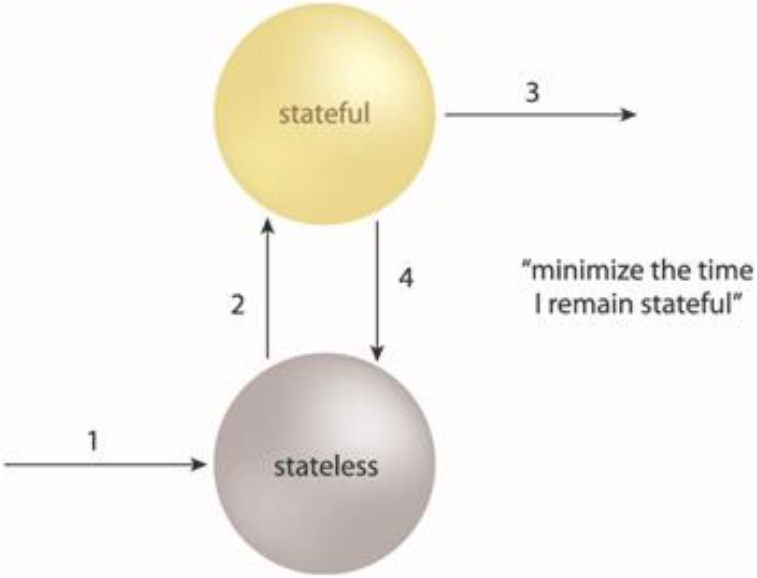
- *"Services exercise a high level of control over their underlying runtime execution environment."*

- Represents the ability of a service to carry out its logic independently of outside influences
- To achieve this, services must be more isolated
- Primary benefits
  - Increased reliability
  - Behavioral predictability

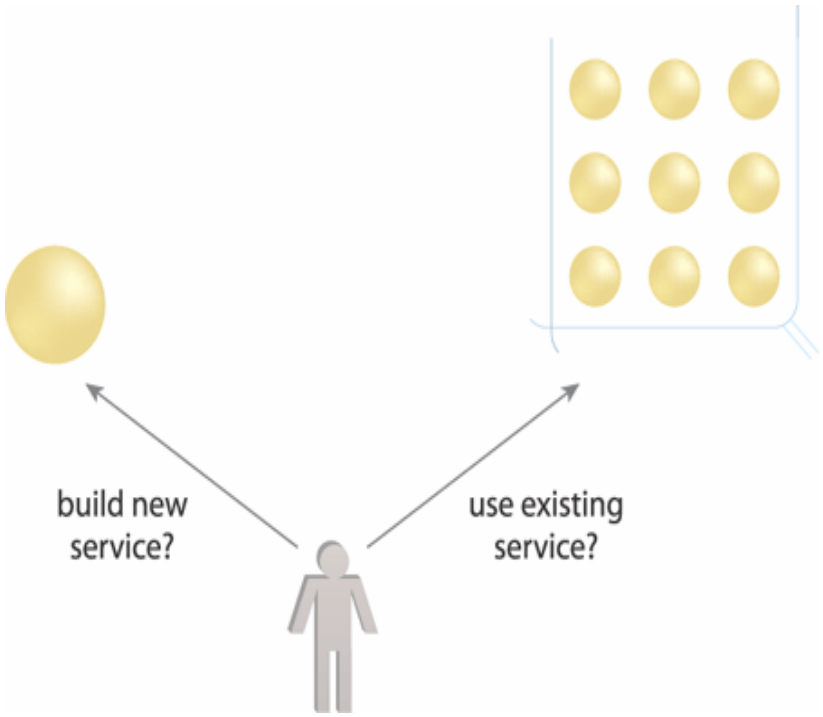


Statelessness

- *"Services minimize resource consumption by deferring the management of state information when necessary."*
- Incorporate state management deferral extensions within a service design
- Goals
  - Increase service scalability
  - Support design of agnostic logic and improve service reuse



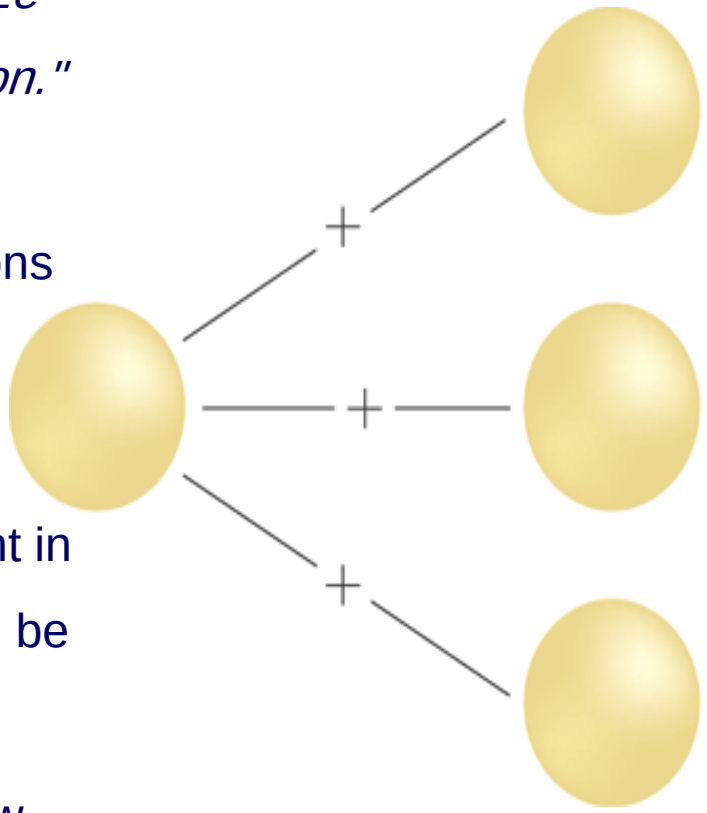
Discoverability



- *"Services are supplemented with communicative meta data by which they can be effectively discovered and interpreted."*
- Service contracts contain appropriate meta data for discovery which also communicates purpose and capabilities to humans
- Store meta data in a service registry or profile documents

Composability

- *"Services are effective composition participants, regardless of the size and complexity of the composition."*
- Ensures services are able to participate in multiple compositions to solve multiple larger problems
- Related to Reusability principle
- Service execution should efficient in that individual processing should be highly tuned
- Flexible service contracts to allow different types of data exchange requirements for similar functions



"allow my capabilities to be repeatedly combined with those of other services"

Applying SOA - Governance

- **Governance is a program that makes sure people do what is 'right'**
- **In conjunction with software, governance controls the development and operation of software**

Goal: Establish SOA organization governance (SOA Board) that governs SOA efforts and breaks down capabilities into non-overlapping services

Applying SOA - Governance

- Policies
  - Codification of laws, regulations, corporate guidelines and best practices
  - Must address all stages of the service lifecycle (technology selection, design, development practices, configuration management, release management, runtime management, etc.)
- Processes
  - Enforce policies
  - System-driven processes (code check-in, code builds, unit tests)
  - Human-driven process (requests, design reviews, code reviews, threat assessment, test case review, release engineering, service registration, etc.)
- Metrics
  - Measurements of service reuse, compliancy with policy, etc.
  - Organization
  - Governance program should be run by SOA Board, which should have cross-functional representatives

Applying SOA - Challenges

- Service Orientation
- Reuse
- Sharing of Responsibilities
- Increased complexity!

Business functionality has to be made available as services. Service contracts must be fixed

Implemented services must be designed with reuse in mind. This creates some overhead.

Potential service users must be involved in the design process and will have influence on the service design