Software as a Service

Dr. Vu Thi Huong Giang

Covered topics

- Services
- Service workflow
- Service composition
- SOA
- Web service patterns

Objectives

- After this lesson, students will be able to:
  - Distinguish the concepts of service workflow, SOA and service composition
  - Conduct web service patterns for a given software

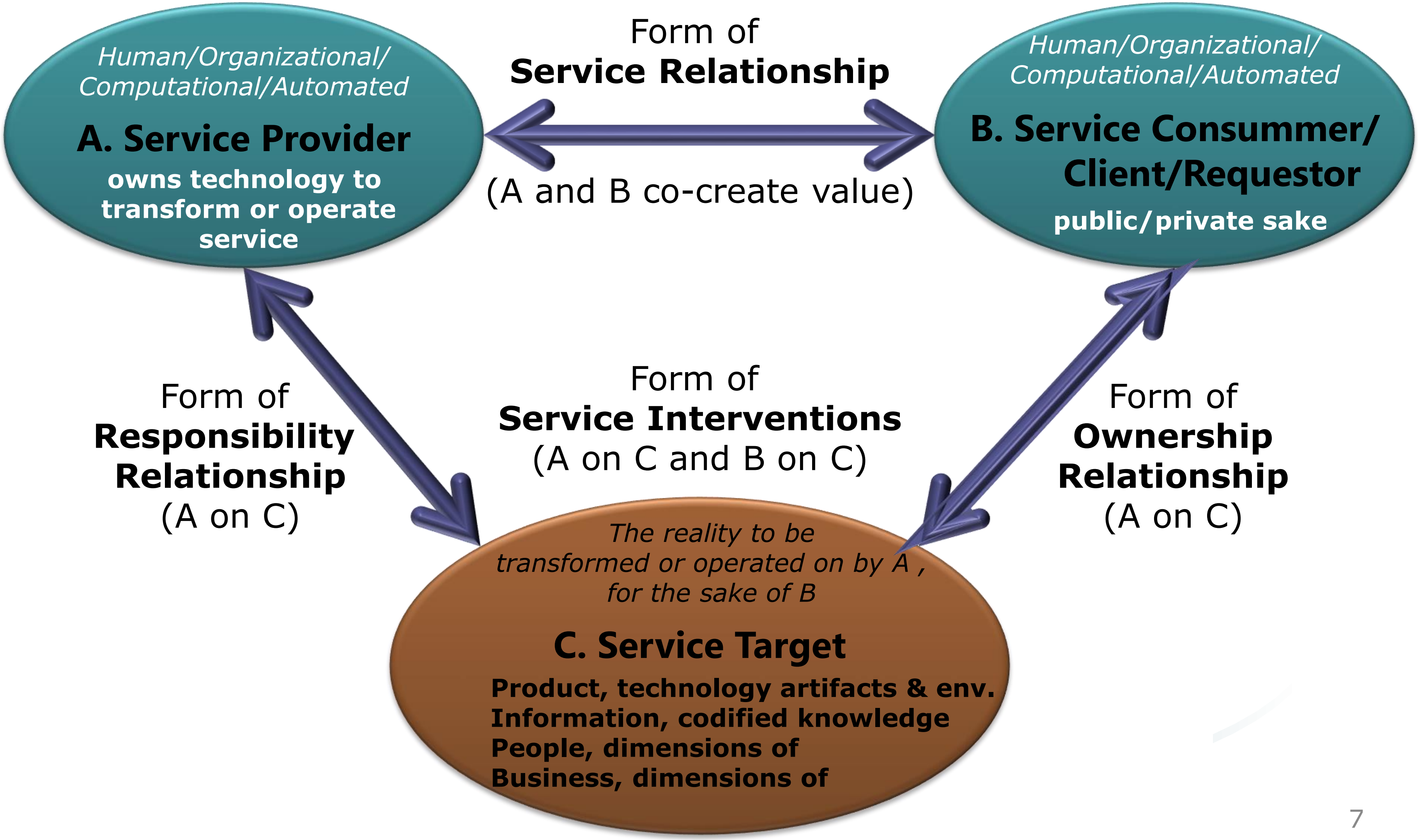
I. SERVICE

1. Definition
2. Characteristics
3. Service communication
4. Classification

What is a service ?

- The Economist:
  - Services are anything sold in trade that cannot be dropped on your foot
- Most people understand (or think they do) what makes something a product or goods:
  - Goods are things that can be owned, traded, and distributed to different places at different times without changing their identity
- But there is much less agreement about the definition of service:
  - Intangible
  - Perishable
  - Experiential
  - Co-produced between the service provider and service consumer

Service as a system of relationships



Human/Organizational/ Computational/Automated

A. Service Provider

owns technology to transform or operate service

Form of **Service Relationship**

(A and B co-create value)

Human/Organizational/ Computational/Automated

B. Service Consummer/ Client/Requestor

public/private sake

Form of **Responsibility Relationship** (A on C)

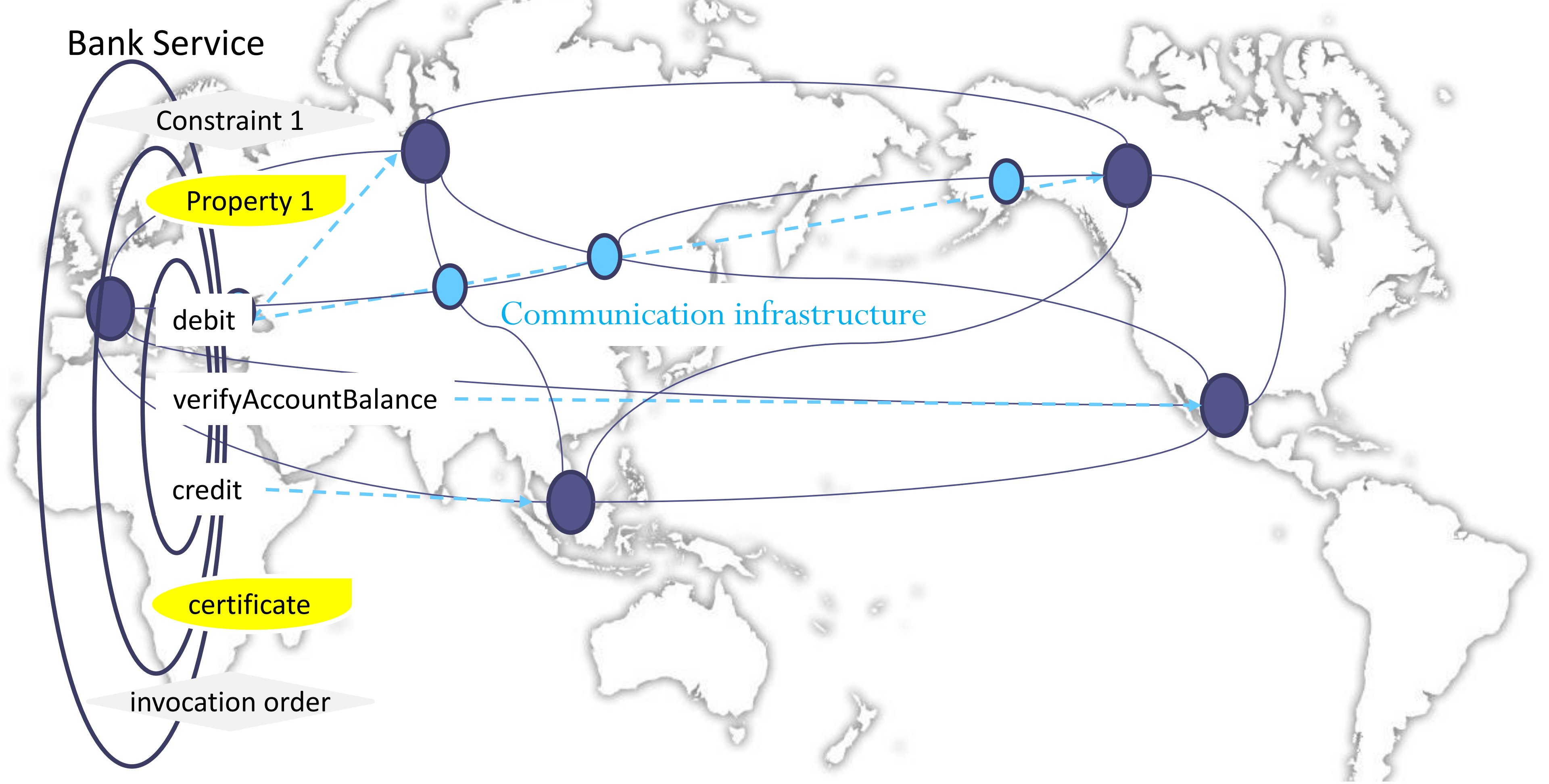
Form of **Service Interventions** (A on C and B on C)

Form of **Ownership Relationship** (A on C)

The reality to be transformed or operated on by A , for the sake of B

C. Service Target

Product, technology artifacts & env.
Information, codified knowledge
People, dimensions of
Business, dimensions of

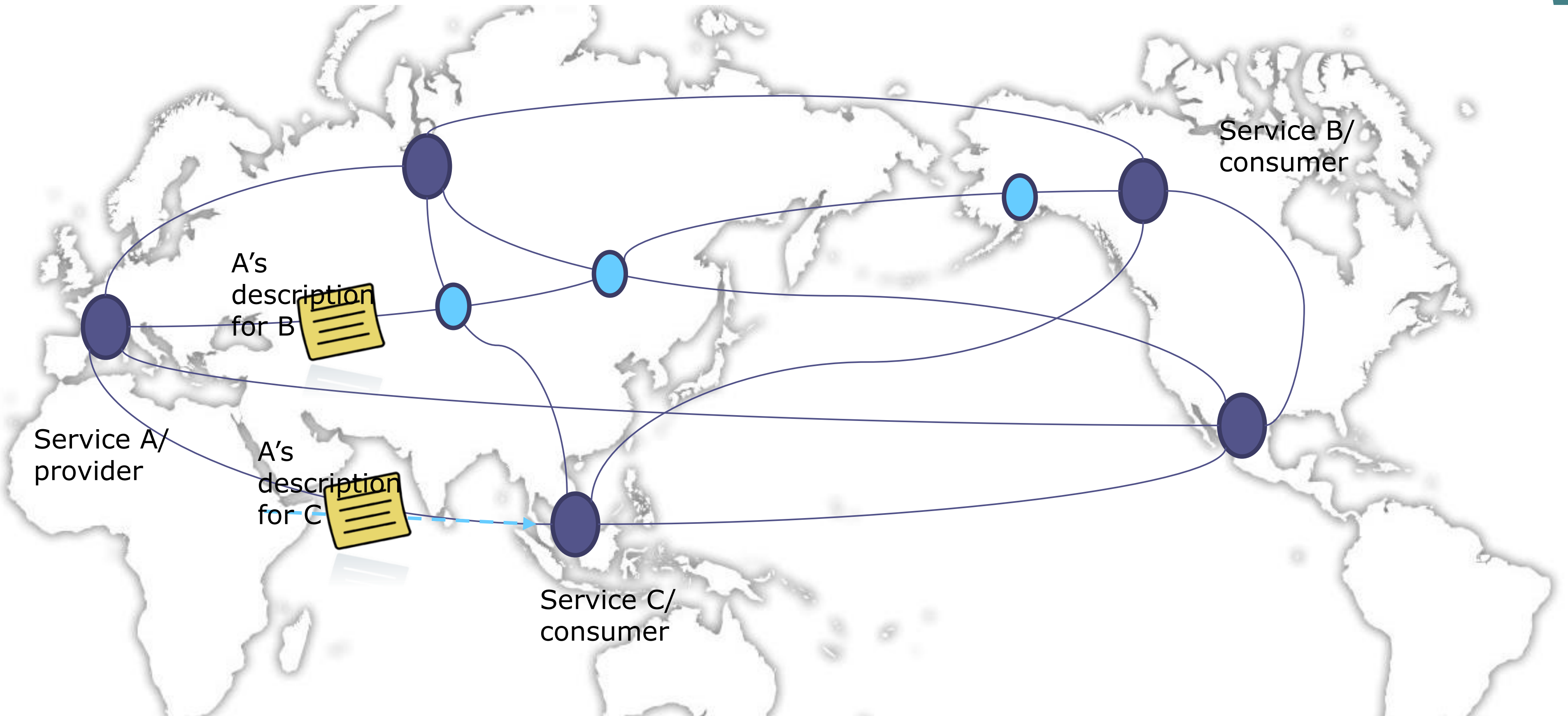


Service characteristics

- Name
- Interface: functions
- Manual: properties + constraints

→ A further abstract from object and components

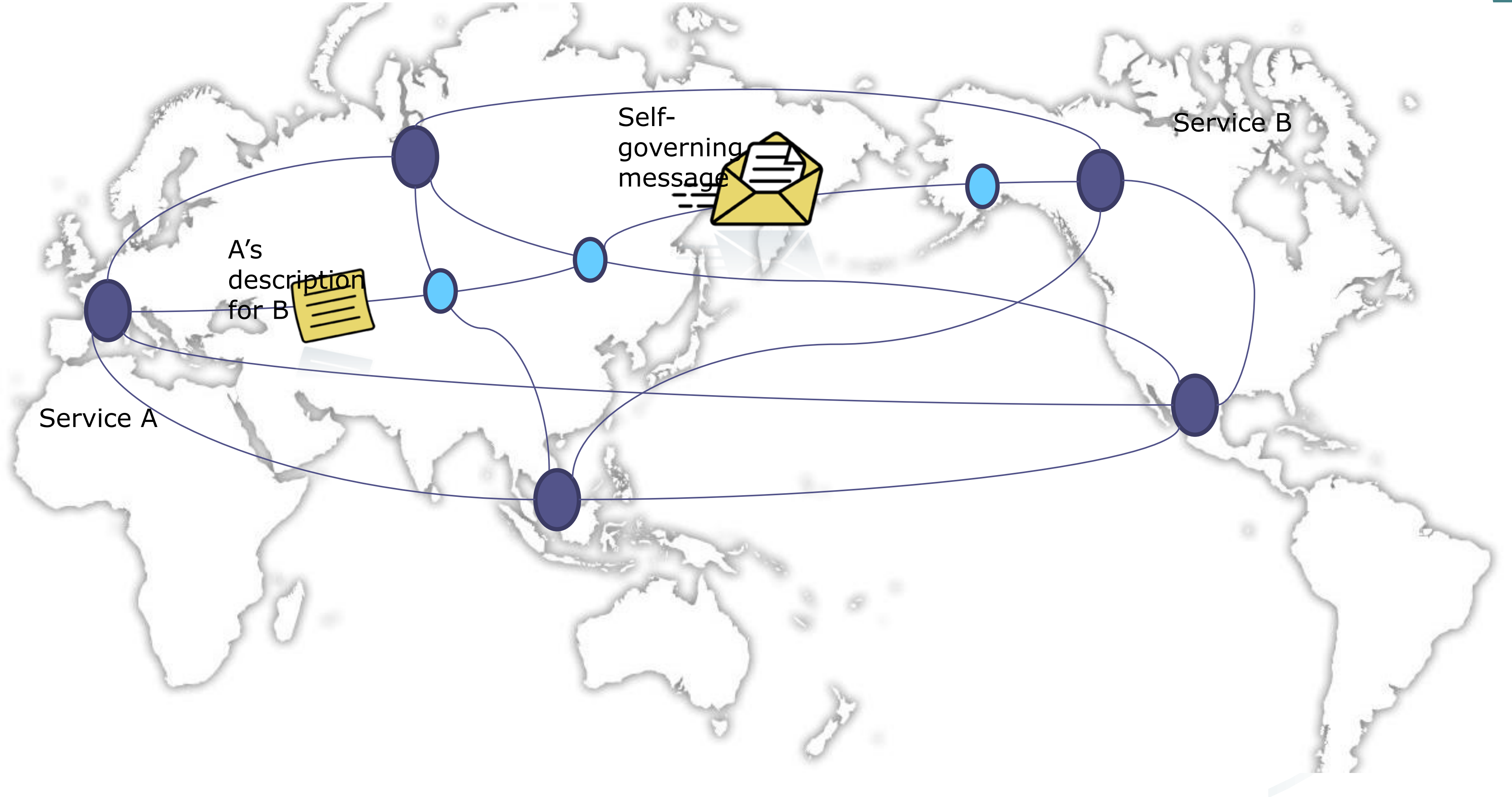
How services relate ?



A is built in a way that is independent of the context in which they are used

→ the service provider and the consumers are **loosely coupled**

How services communicate ?

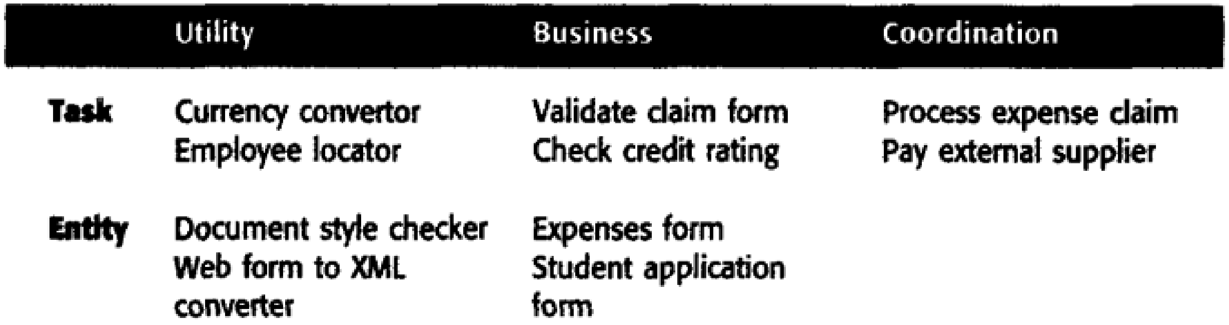


Service classification - Based on users / usage objectives

- Utility services :
  - Implement some general functionality that may be used by different business processes.
  - Example: currency conversion service to compute the conversion of one currency (e.g. dollars) to another (e.g. euros).
- Business services:
  - Associated with a specific business function.
  - Example: registering of students for a course (a business function in a university)
- Coordination or process services:
  - Support a more general business process which usually involves different actors and activities.
  - Example: ordering service that allows orders to be placed with suppliers, goods accepted and payments made (service in a company)

Service classification
- Based on tasks / entities

- Task-oriented services:
  - Associated with some activity whereas
  - Coordination services are always task-oriented
- Entity-oriented services:
  - Associated with some business entity (like objects)
  - Example: a job application form.
  - Utility and business services are task or entity-oriented



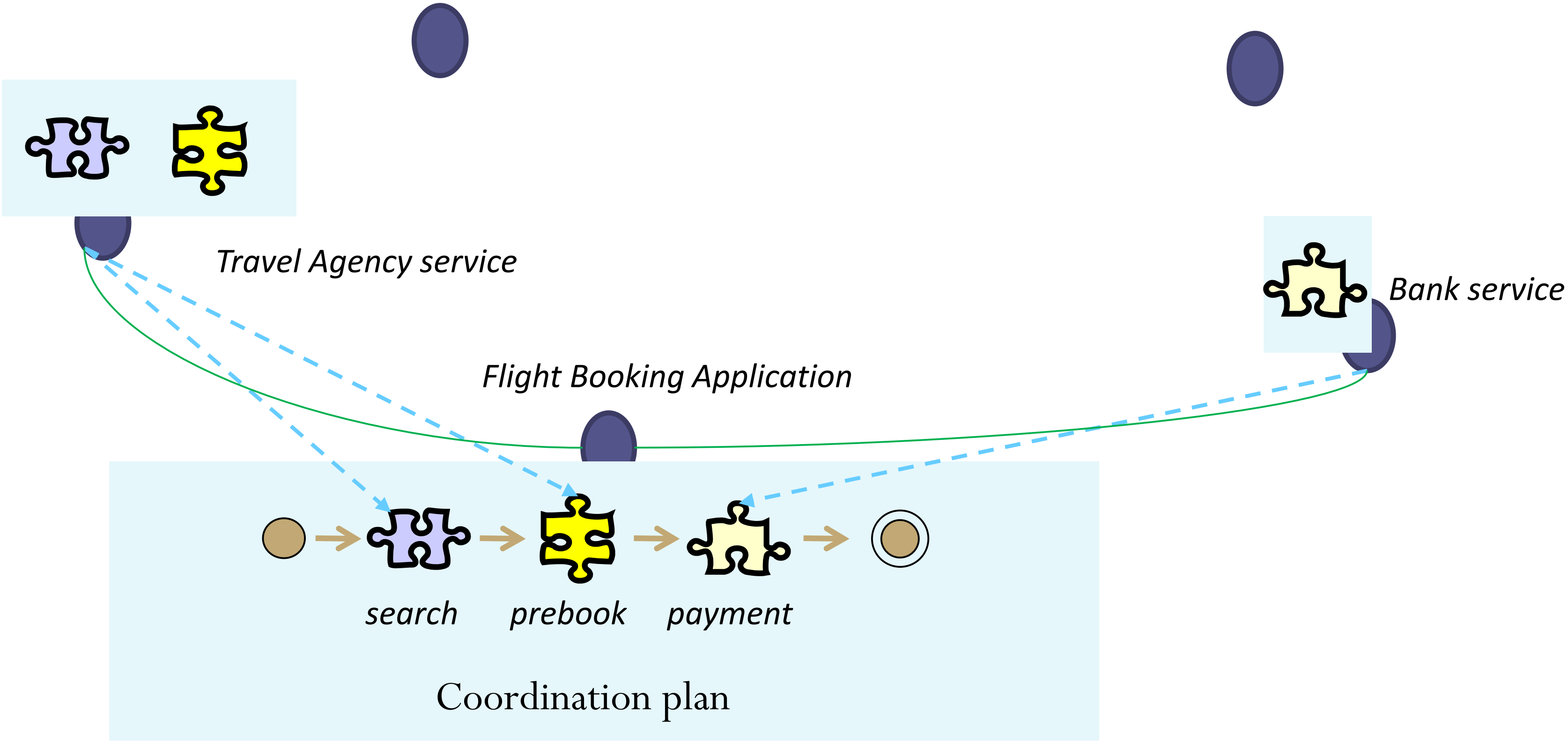
	Utility	Business	Coordination
Task	Currency convertor Employee locator	Validate claim form Check credit rating	Process expense claim Pay external supplier
Entity	Document style checker Web form to XML converter	Expenses form Student application form	

II. SERVICE WORKFLOW

Introduction

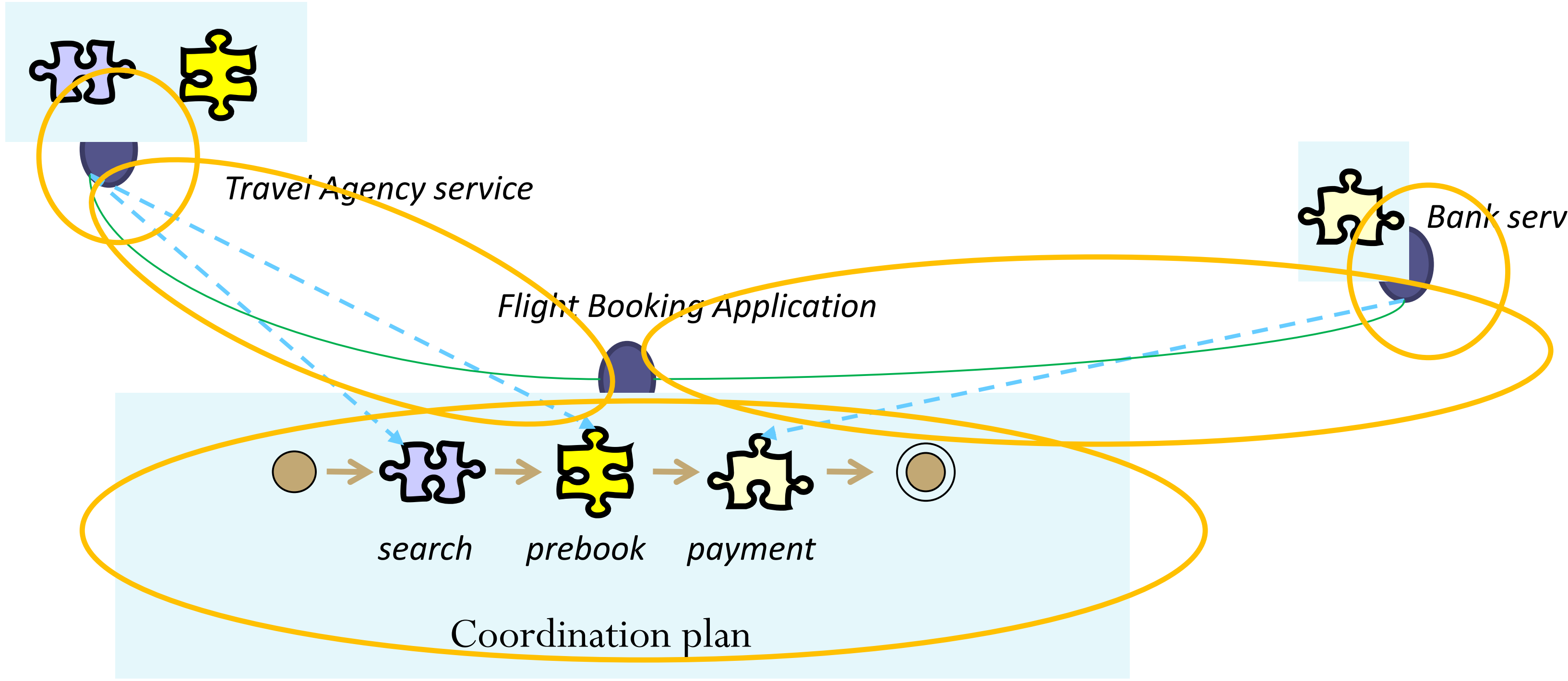
- Compose and configure existing services to create new, composite services:
  - New composite service = service that is used as components in some other service composition
  - New composite service = web application: involved services are integrated with a web user interface
- The services involved in the composition may be:
  - specially developed for the application
  - business services developed within a company
  - services from some external provider

Construction viewpoint : Service-based application



- built out of service's function invocations
- 1 **activity** of the application = 1 service's function invocation
- The execution of activities respects a **coordination plan** (application logic)

Deployment viewpoint : Software as a Service (SaaS)



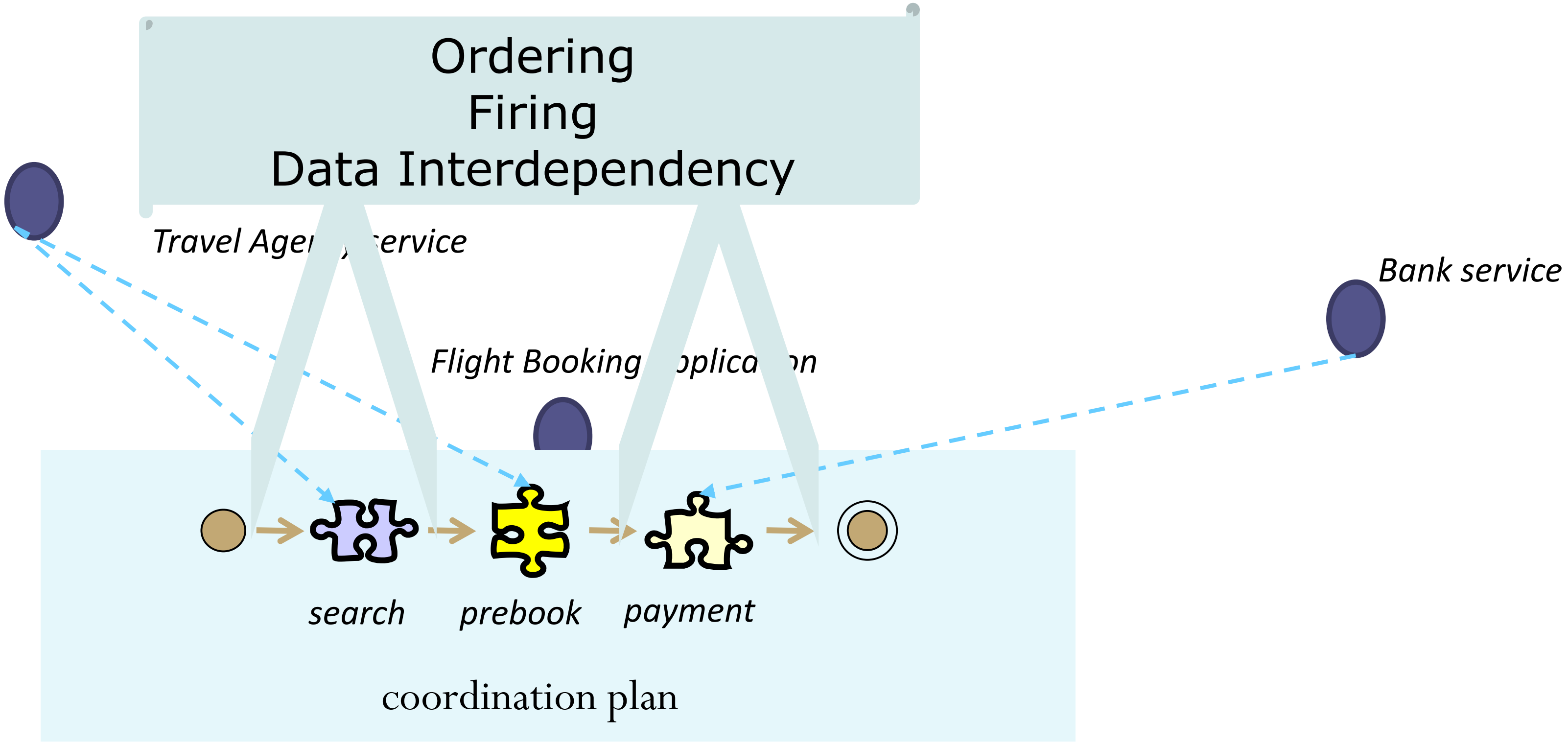
- How to describe / construct :
  - as a service-based application
- How to package service's function :
  - deployed as a hosted service
- How to deliver service's function :
  - accessed over the Internet using a standard web browser

→ Coordination plan

→ Services

→ Communication

Coordination



Ordering
Firing
Data Interdependency

Travel Agency service

Flight Booking Application

Bank service

search

prebook

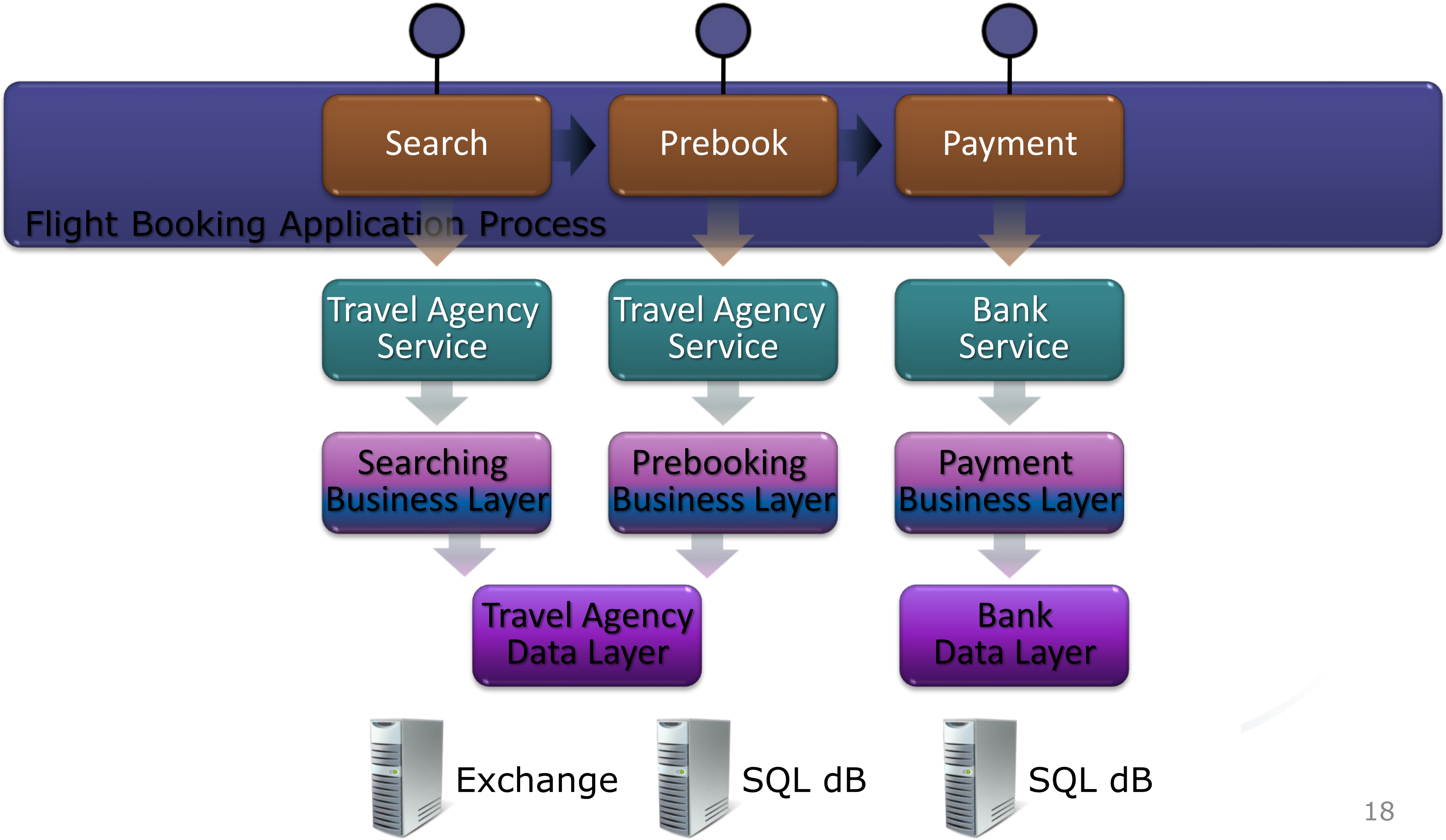
payment

coordination plan

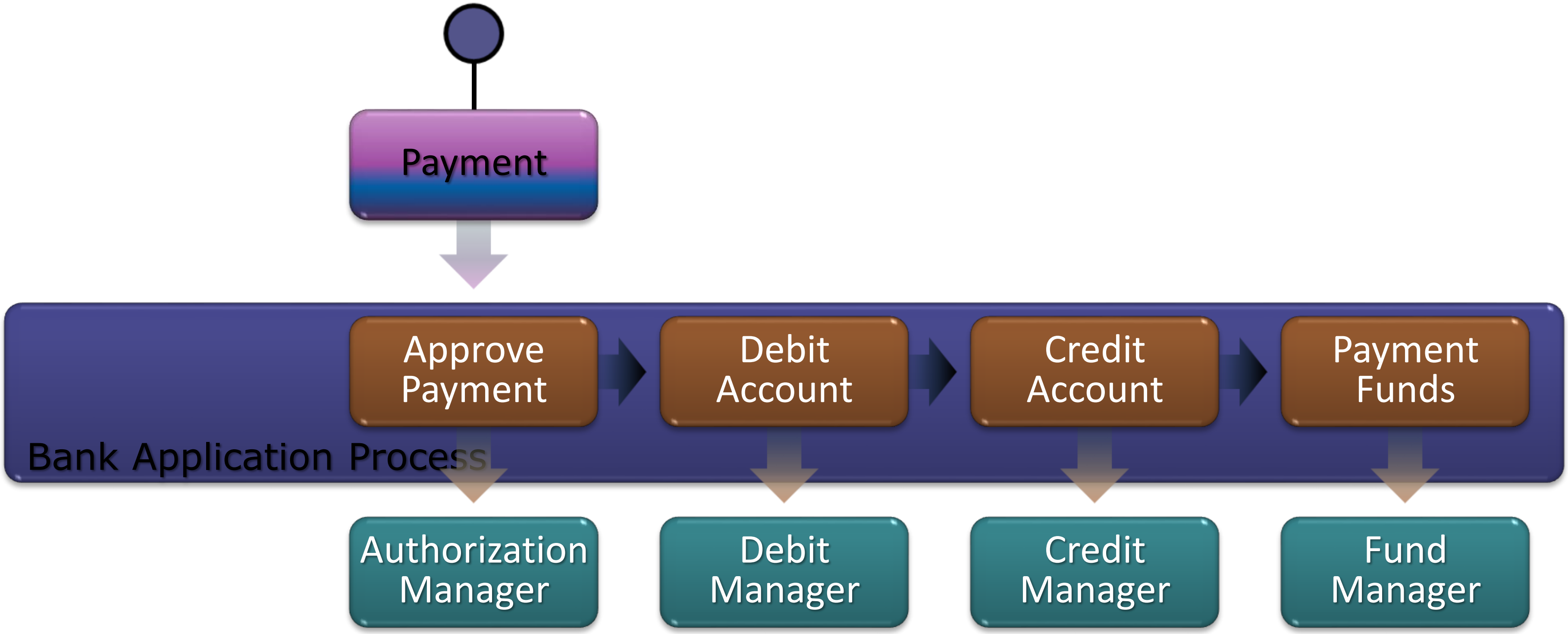
- Languages: Semantic expression about the coordinated execution of services
- Architectures: Managing the coordinated execution of services

→ Instantiate a practical engine for the coordination plan of application’s activities.

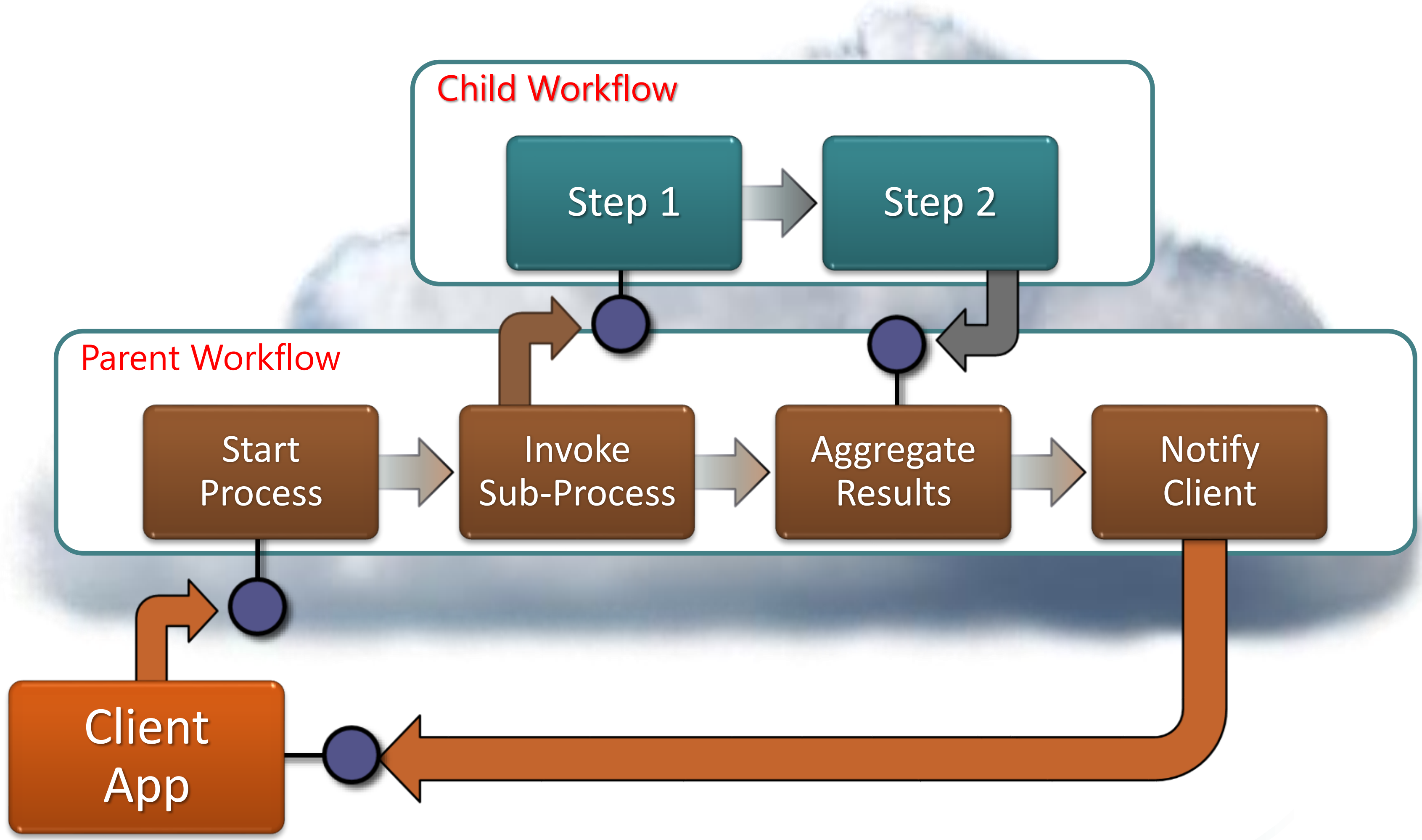
Workflow as coordination plan



Workflow as service operation



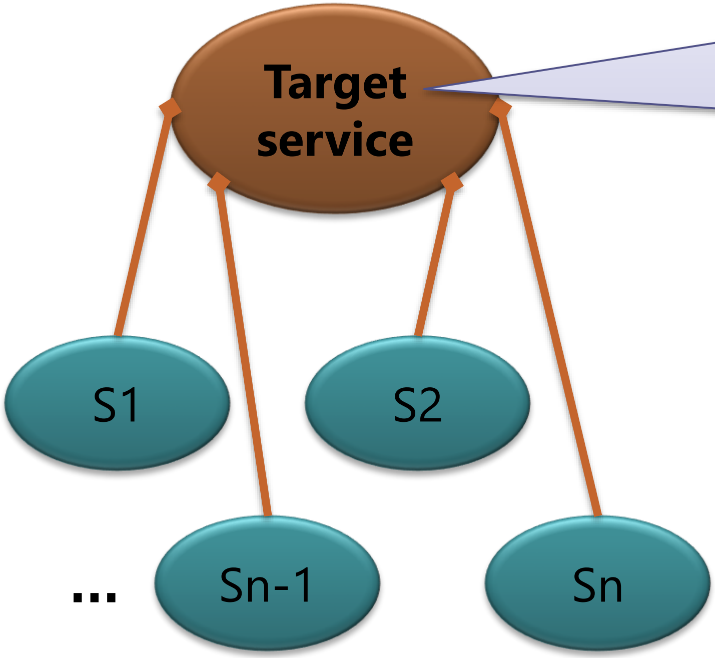
Workflows in the Cloud



III. SERVICE COMPOSITION

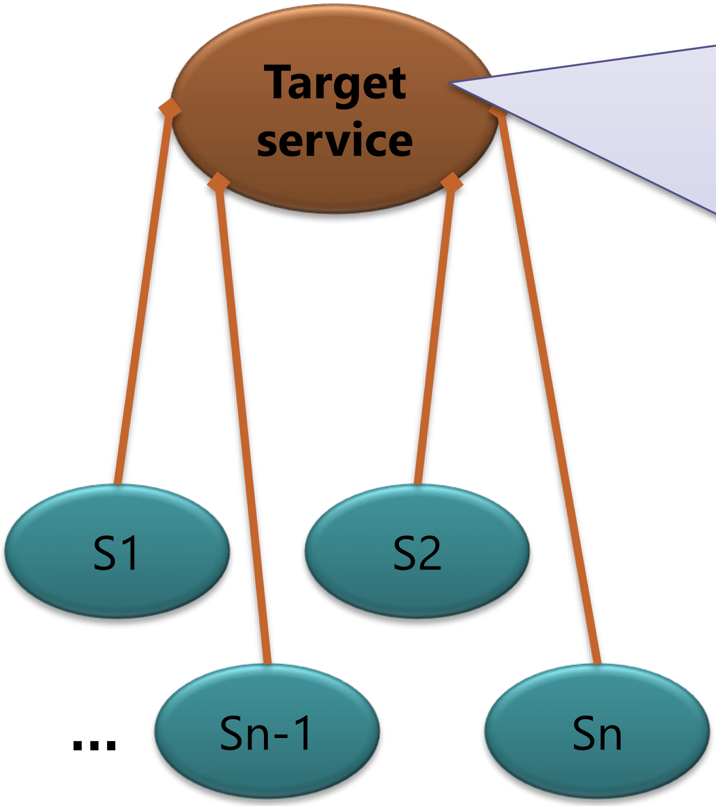
Service composition

- The service composition encompasses necessary **roles** and **functionalities** for the aggregation of multiple services into a single composite service.



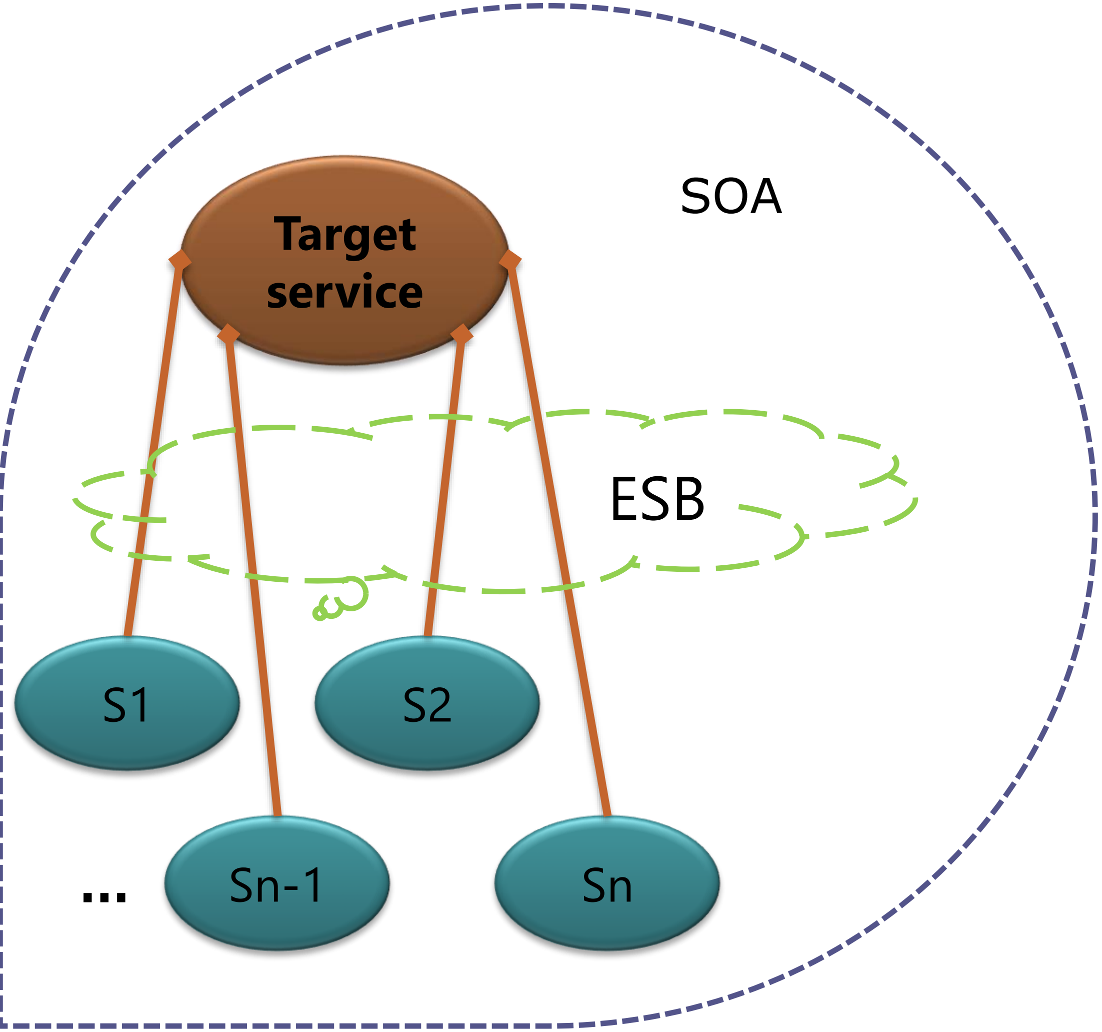
•provide functions to be used in further service compositions
•offered as complete applications / solutions to service clients
(by means of service aggreators)

Service composition



- Service aggregators:
- Play the role of service providers : publishing the service descriptions of the composite service they create.
- Develop specifications and/or code that permit the composite service to perform functions that are based on features such as meta-data descriptions, standard terminology and reference models and service conformance.
- Perform service coordination to control the execution of the composite services, services transactions
- Manage both the dataflow as well as the control flow between composite services.
- Enforce policies on aggregate service invocations.

Service composition



- Service-Oriented Architecture and the Enterprise Service Bus (ESB) helps discharge many of the obligations induced by requirements

IV. SOA

SOA

- Conceptual architecture, in which application functions are built as services that are loosely coupled and well-defined to support interoperability and to improve flexibility and reuse
- Design a philosophy to think about "what a business does‟: A business (application logic) can be a mix of core:
  - internal functions that it does itself
  - outsourced ones provided by other businesses

What is a SOA ?

- The widest technical (and rather minimalist) definition:
  - An enterprise-wide IT architecture that promotes loose coupling, reuse, and interoperability between systems.
- A moderately complex technical definition:
  - An application architecture in which
    - All functions or services are defined using a description language and have callable interfaces that are called to perform business processes.
    - Each interaction is independent of each and every other interaction and the interconnect protocols of the communicating devices.
    - Because interfaces are platform independent, a client can use the service from any device using any operating system in any language.

SOA Tenets

- Service boundaries are explicit
- Services are autonomous
- Services share schema and contract, not type
- Services are compatible based on policy

SOA & Web service

,

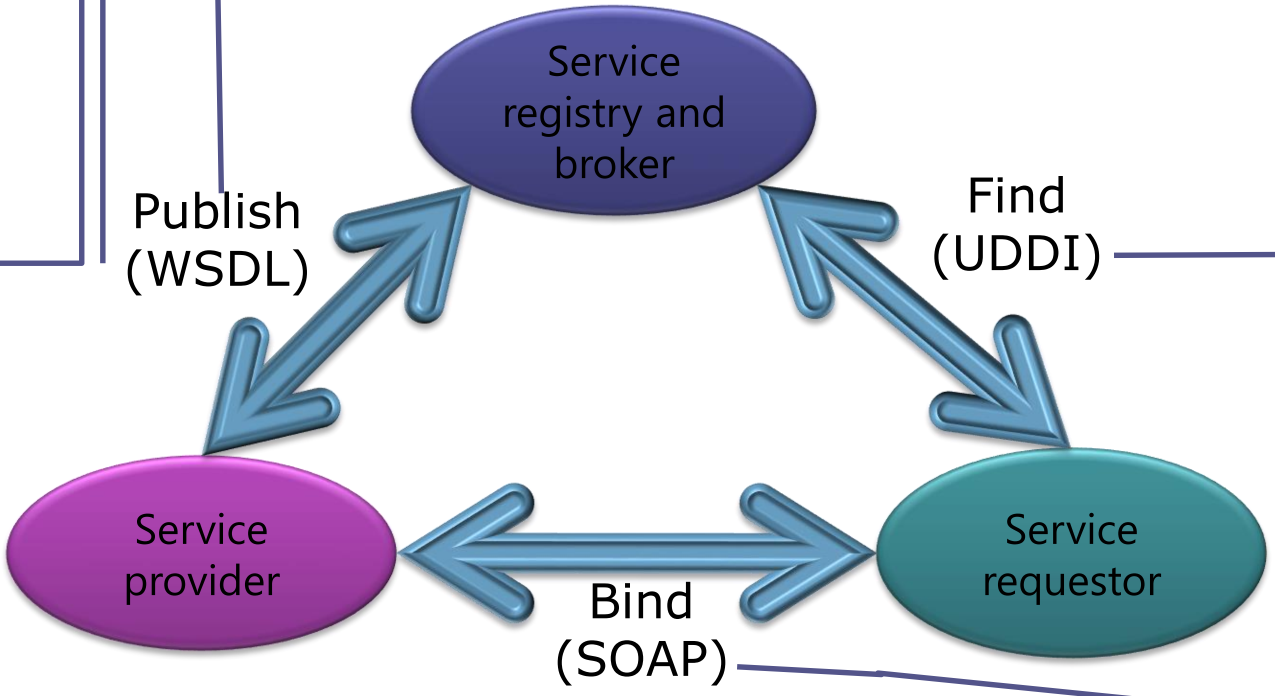
- MBAs and CIOs talk about SOAs
  - A set of business, process, organizational, governance, and technical methods to reduce or eliminate frustrations with IT and to quantifiably measure the business value of IT while creating an agile business environment for competitive advantage.
  - Provide the flexibility to treat elements of business processes and the underlying IT infrastructure as secure, standardized components (services) that can be reused and combined to address changing business priorities.

- Software architects and developers talk about Web services
  - Autonomous, platform-independent computational elements, possibly managed by different organizations
  - Described, published, discovered, coordinated and programmed to build networks of collaborating applications, distributed both within and across organizational boundaries

Web service - oriented architecture

- Define the way in which service providers should define the interface to these services.
- Allow the interface of a service and its bindings to be defined in a standard way

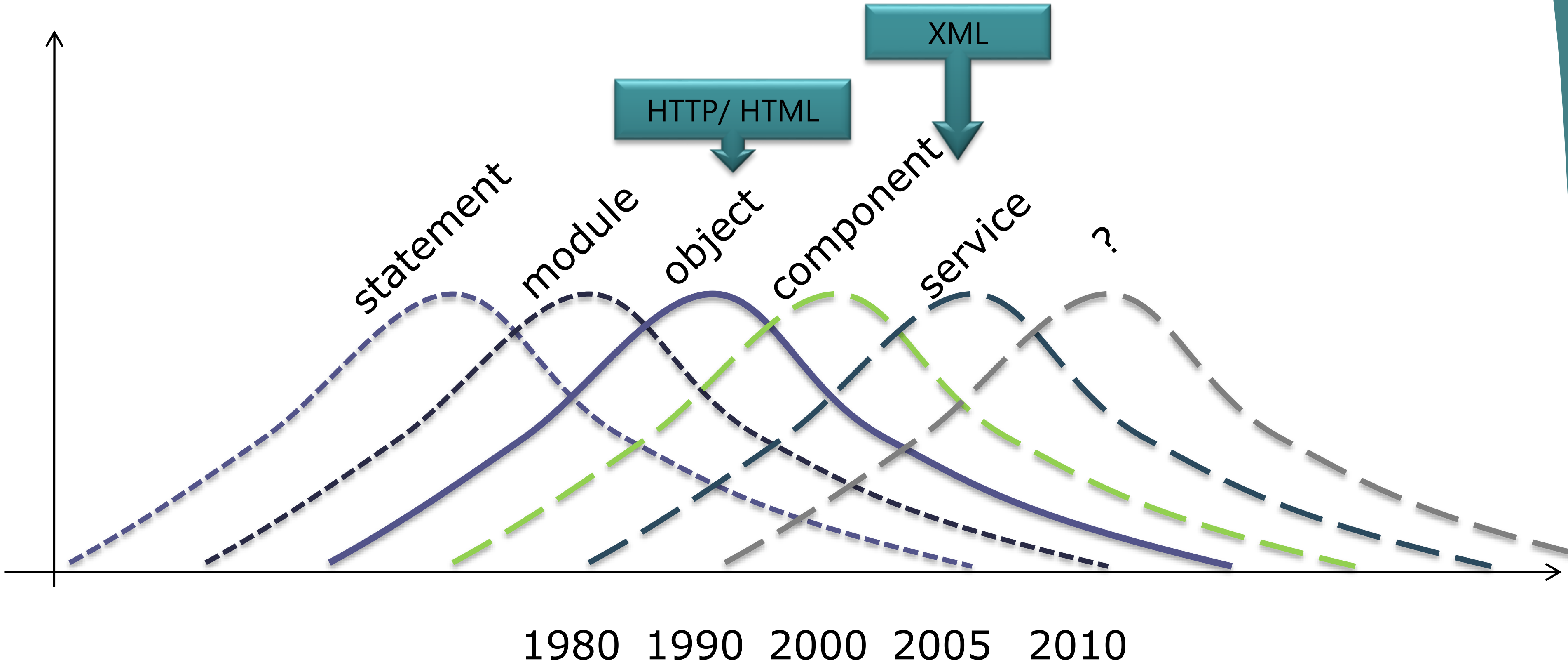
•Define the components of a service specification that may be used to discover the existence of a service: information about the service provider & provided service, location of the service description , business relationships.
- Enable potential users of a service to discover what services are available.



- Message interchange standard supporting the communication between services
- Define the essential and optional components of messages passed between services

V. Web service patterns

Programming paradigms revisited



1st generation of e-Bussiness: B2C & B2B

2nd generation of e-Bussiness: B2B & B2G

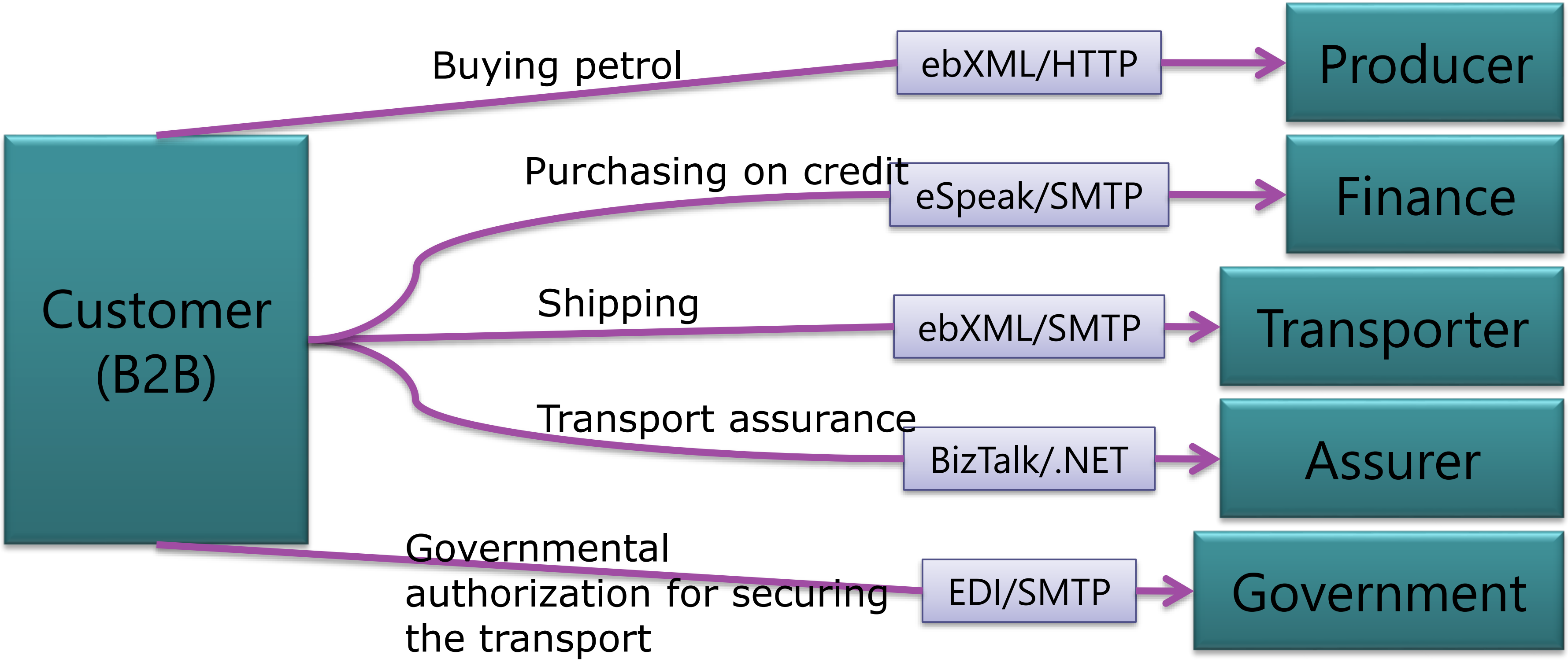
3th generation of e-Bussiness: M2M

1st generation of E-Business : B2B

- B2B (Business-to-business)
  - Yahoo
  - Google
    - Provides search, cache, and spelling services
    - Accessible via Internet
    - Across different client platforms
    - Can be integrated into your application
    - Can use while developing in any environment
  - eBay
    - Allows third-party vendors to sell on site
  - Amazon
    - Interacts with external services
    - Provide many more complex services, e.g. access to product data, content from customers, seller information, and shopping carts

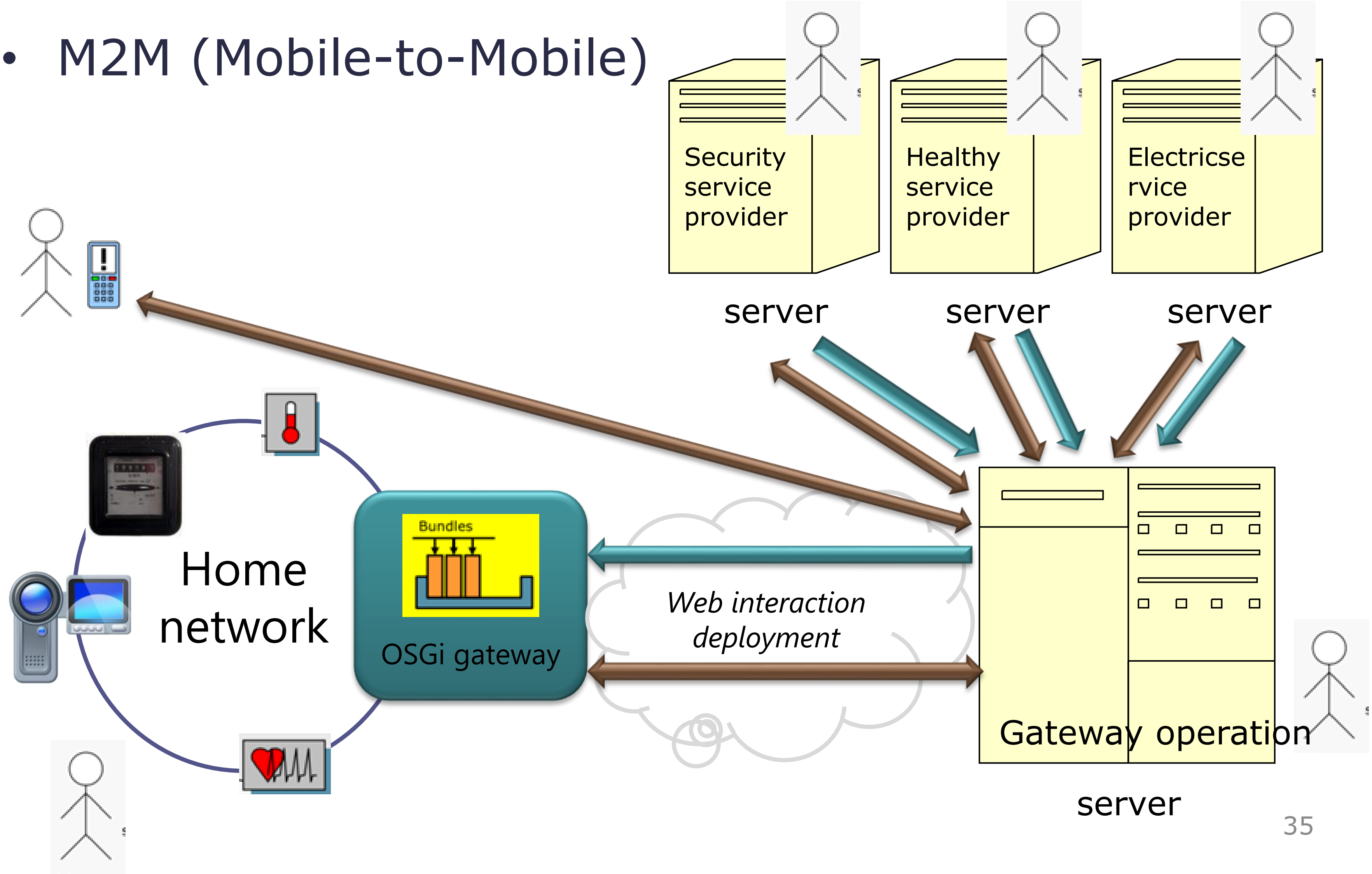
2nd generation of E-Business : B2B + B2G

- B2G (Business-to-government)
  - Third-party supports
  - Example:



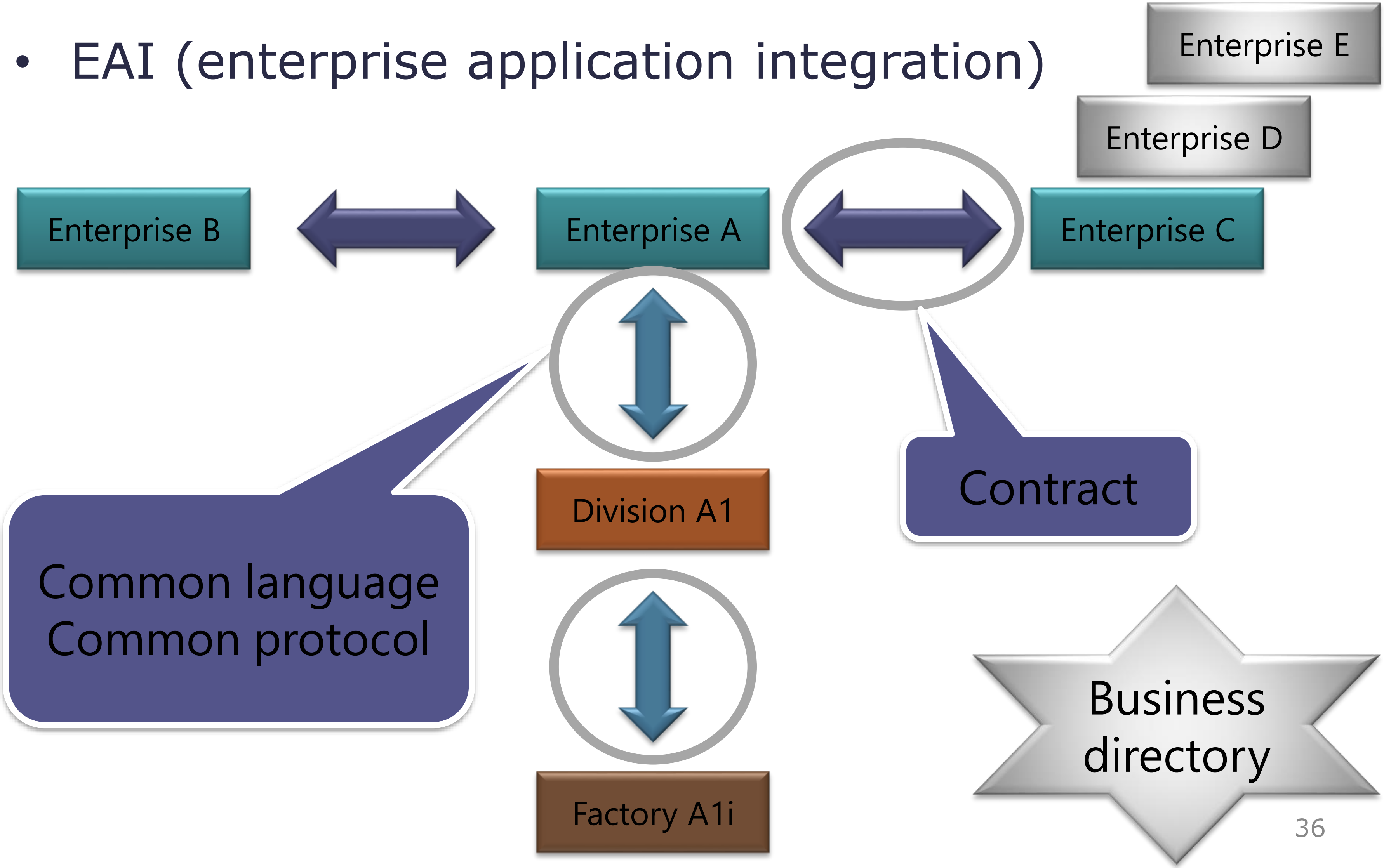
3th generation of E-Bussiness: M2M

- M2M (Mobile-to-Mobile)



Needs

- EAI (enterprise application integration)



Enterprise E

Enterprise D

Enterprise B

Enterprise A

Enterprise C

Division A1

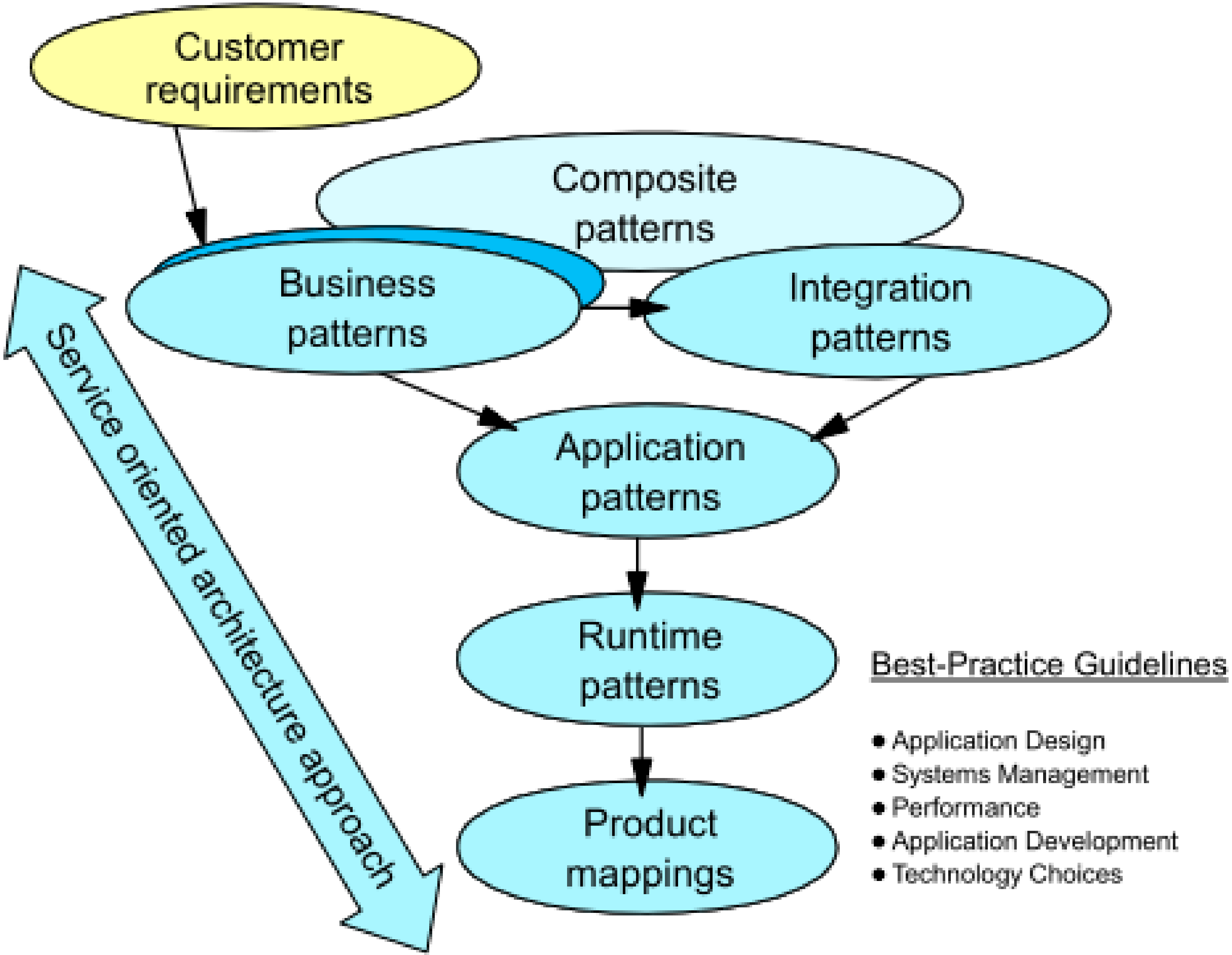
Factory A1i

Common language
Common protocol

Contract

Business directory

How to do ?



Customer requirements

Composite patterns

Business patterns

Integration patterns

Application patterns

Runtime patterns

Product mappings

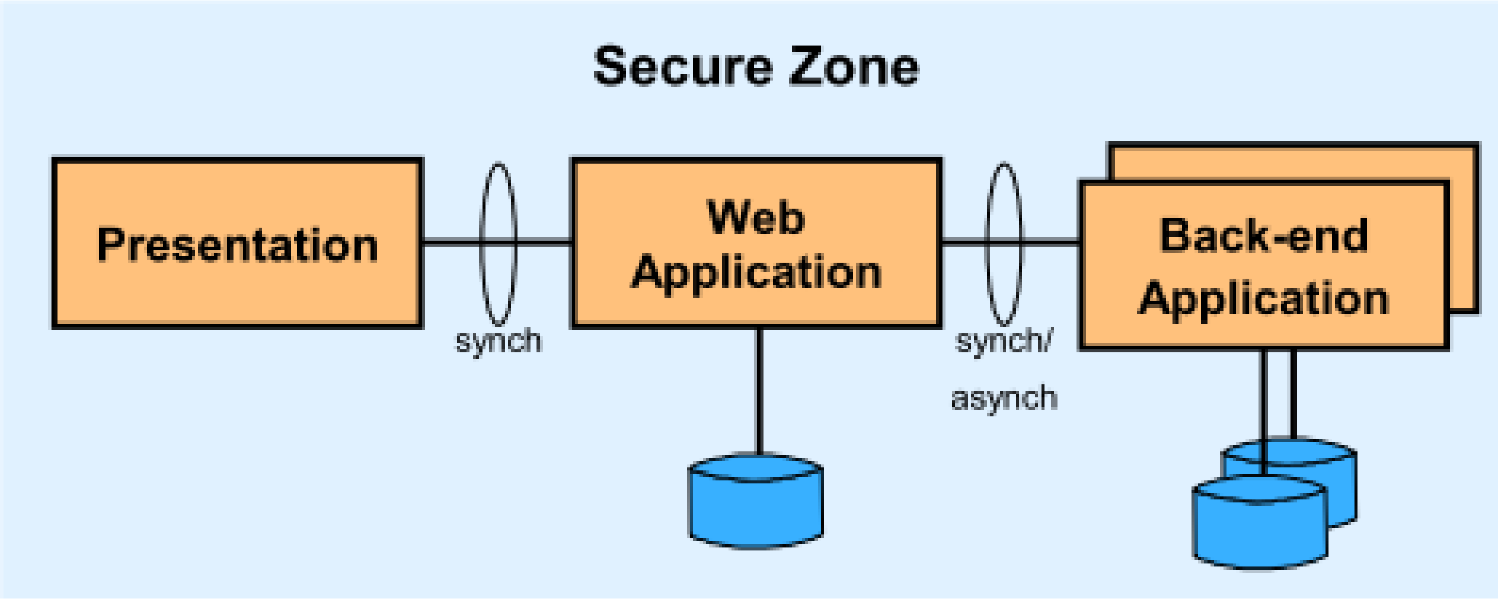
Service oriented architecture approach

Best-Practice Guidelines

- Application Design
- Systems Management
- Performance
- Application Development
- Technology Choices

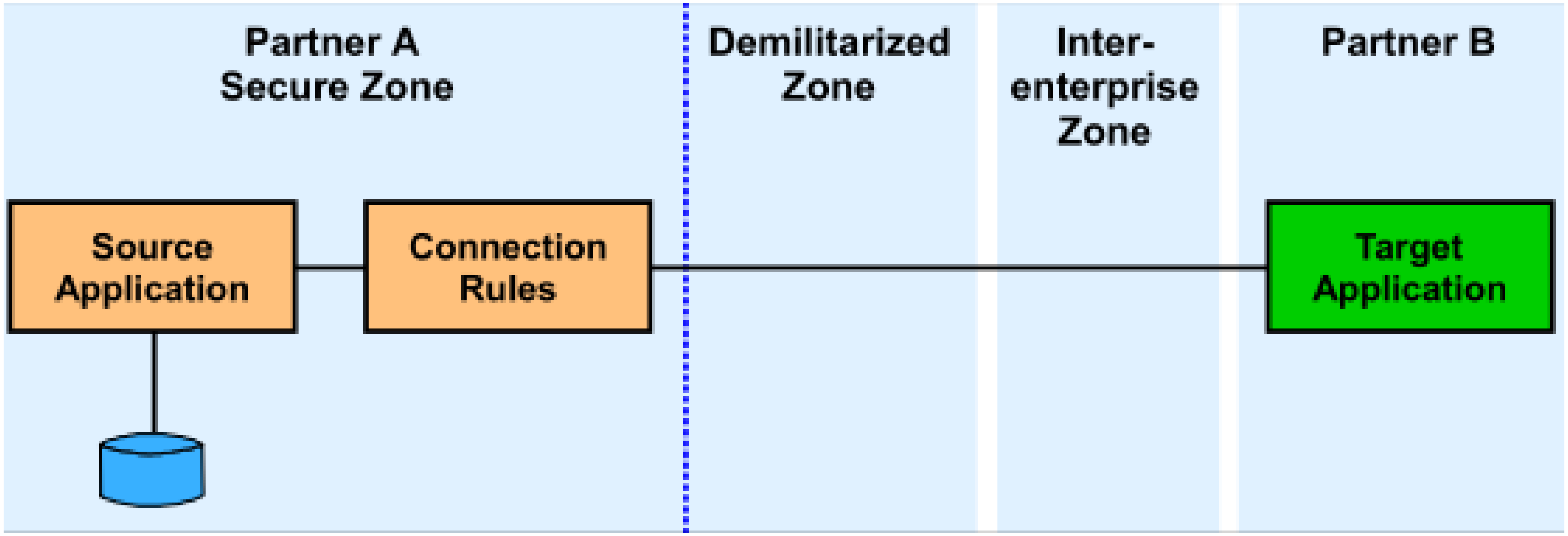
Self-Service business pattern

- Capture the direct interactions between users and an enterprise, e.g., simple information access, complex updates involving core enterprise systems and data.
- In the SOA context:
  - Service consumers : customers, business partners, stockholders and employees
  - Service provider: enterprise



Extended Enterprise business pattern

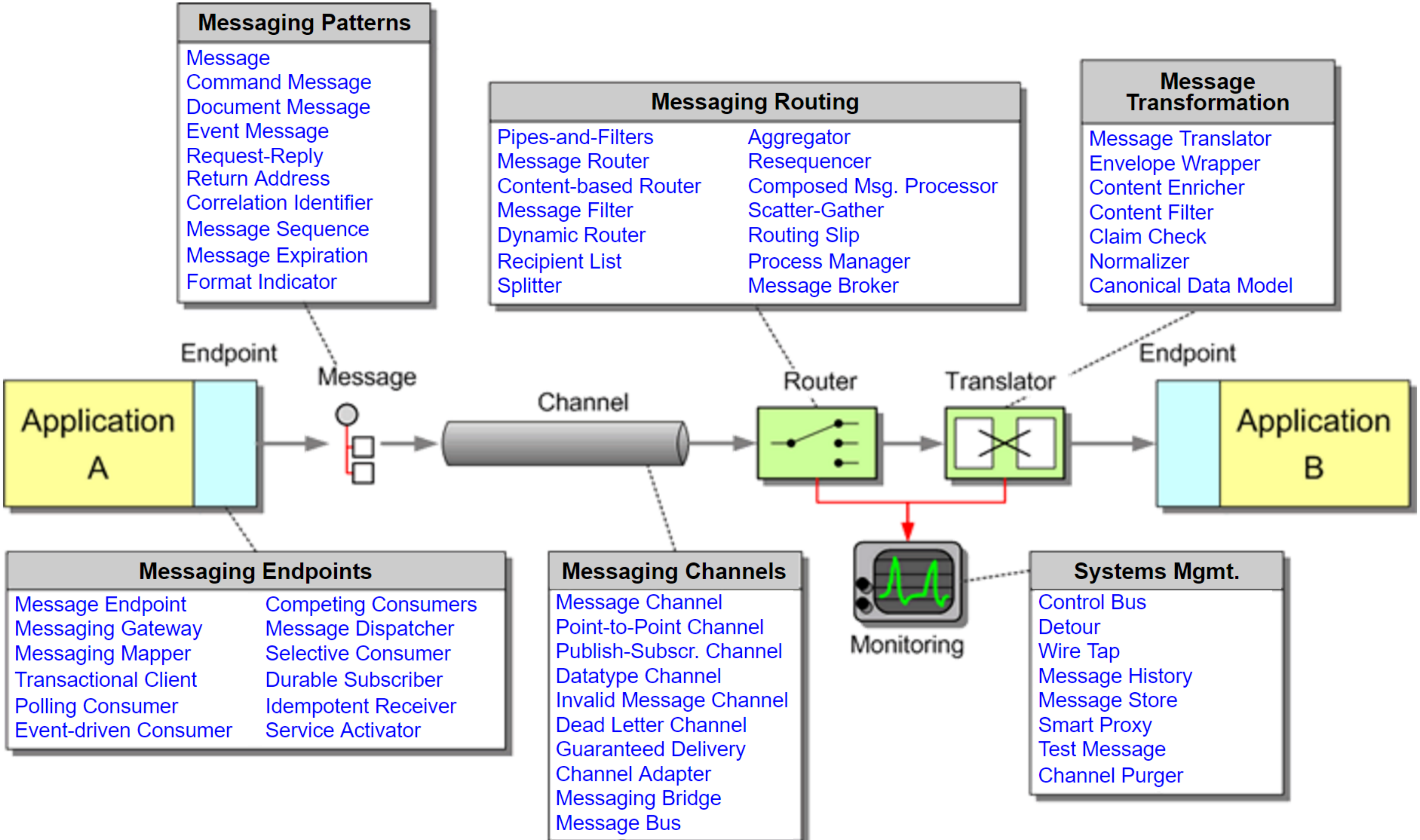
- Address interactions and collaborations between business processes in separate enterprises.
- Connect inter-enterprise applications or services by programmable interfaces



Application integration pattern

- Capture best practices around back-end integration of applications and data, process automation, and workflow implementations involving human interactions.
  - Application integration
  - Direct connection
  - Broker
  - Serial process
  - Parallel process

Application integration pattern



Messaging Patterns

- Message
- Command Message
- Document Message
- Event Message
- Request-Reply
- Return Address
- Correlation Identifier
- Message Sequence
- Message Expiration
- Format Indicator

Messaging Routing

- Pipes-and-Filters
- Message Router
- Content-based Router
- Message Filter
- Dynamic Router
- Recipient List
- Splitter
- Aggregator
- Resequencer
- Composed Msg. Processor
- Scatter-Gather
- Routing Slip
- Process Manager
- Message Broker

Message Transformation

- Message Translator
- Envelope Wrapper
- Content Enricher
- Content Filter
- Claim Check
- Normalizer
- Canonical Data Model

Messaging Endpoints

- Message Endpoint
- Messaging Gateway
- Messaging Mapper
- Transactional Client
- Polling Consumer
- Event-driven Consumer
- Competing Consumers
- Message Dispatcher
- Selective Consumer
- Durable Subscriber
- Idempotent Receiver
- Service Activator

Messaging Channels

- Message Channel
- Point-to-Point Channel
- Publish-Subscr. Channel
- Datatype Channel
- Invalid Message Channel
- Dead Letter Channel
- Guaranteed Delivery
- Channel Adapter
- Messaging Bridge
- Message Bus

Systems Mgmt.

- Control Bus
- Detour
- Wire Tap
- Message History
- Message Store
- Smart Proxy
- Test Message
- Channel Purger

Runtime pattern

- Use nodes to group functional and operational components. The nodes are interconnected to solve a business problem.
- Each Application pattern leads to one or more underpinning Runtime patterns.
- We can overlay the Application pattern over the Runtime pattern to identify where business logic is deployed on nodes. The Runtime patterns covered give some typical examples of possible solutions, but should not be considered exhaustive.

Quiz and Exercises

- Now let’s go over what you have learned through this lesson by taking a quiz.
- When you’re ready, press Start button to take the quiz

Start