SHAPE

Service-oriented Heterogeneous Architecture and Platforms Engineering

# SoaML Tutorial

# Service Modelling with SoaML

**Brian Elvesæter, SINTEF ICT**

**brian.elvesater@sintef.no**

# Tutorial outline

- Introduction
  - What you will learn
  - Online material
- SoaML language
  - Overview
  - Core concepts
- SoaML methodology
  - Business architecture modelling guidelines
  - System architecture modelling guidelines


# Introduction



# What you will learn

- Today
  - Core concepts of the SoaML language
  - SoaML modelling guidelines
- SHAPE extra material (on your own)
  - Website: http://rd.softeam.com/demos/soaml
    - BPMN and SoaML languages
    - SoaML methodology guidelines
    - Modelio demonstration and hands-on
    - SHAPE language extensions for SoaML
      - Flexible Business Models
      - Semantic Web Services
      - Multi-Agent Systems

# Online material

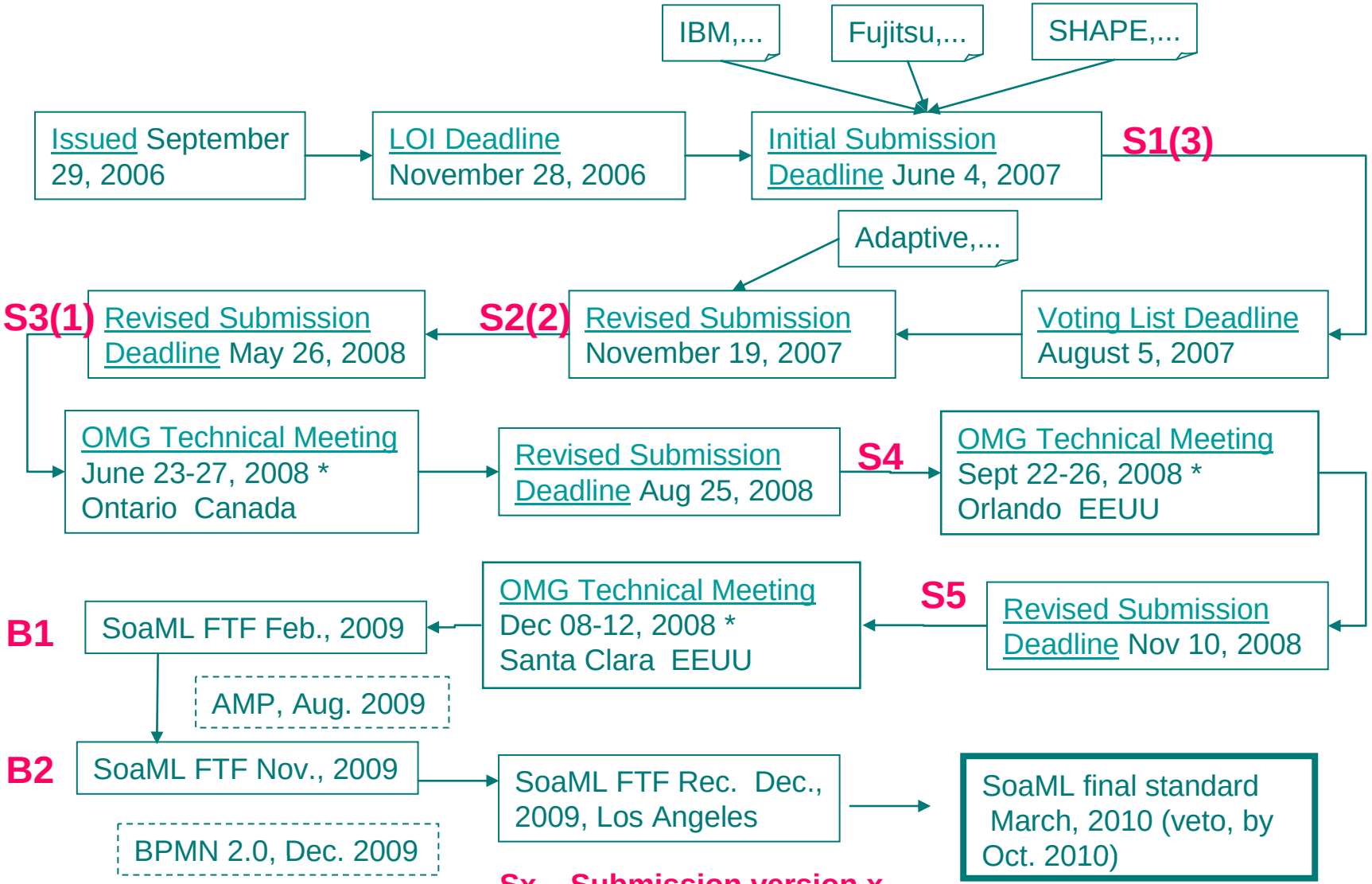
- SHAPE project website:
  - http://www.shape-project.eu/
- SHAPE Methodology
  - http://www.shape-project.eu/download-area/SHAPE-Methodology_OnlineLibrary_final/index.htm
- SoaML specification
  - http://www.omg.org/spec/SoaML/
- SoaML wiki
  - http://www.omgwiki.org/SoaML/doku.php
- SHAPE hands-on tutorial given at ECMFA 2010 (extra material)
  - http://rd.softeam.com/demos/soaml
- Modelio Enterprise Edition v1.1.1 modelling tool (30 day evaluation)
  - http://modeliosoft.com
- SoaML Designer and SoaML Engine modules for Modelio
  - http://rd.softeam.com/prototypes/


# SoaML language

# What is SoaML?

- Service oriented architecture Modeling Language (SoaML)
  - Extensions to UML2 to support service concepts.
  - Focuses on basic service modelling concepts.
  - A foundation for further extensions and integration with BPMN, BMM and other metamodels.

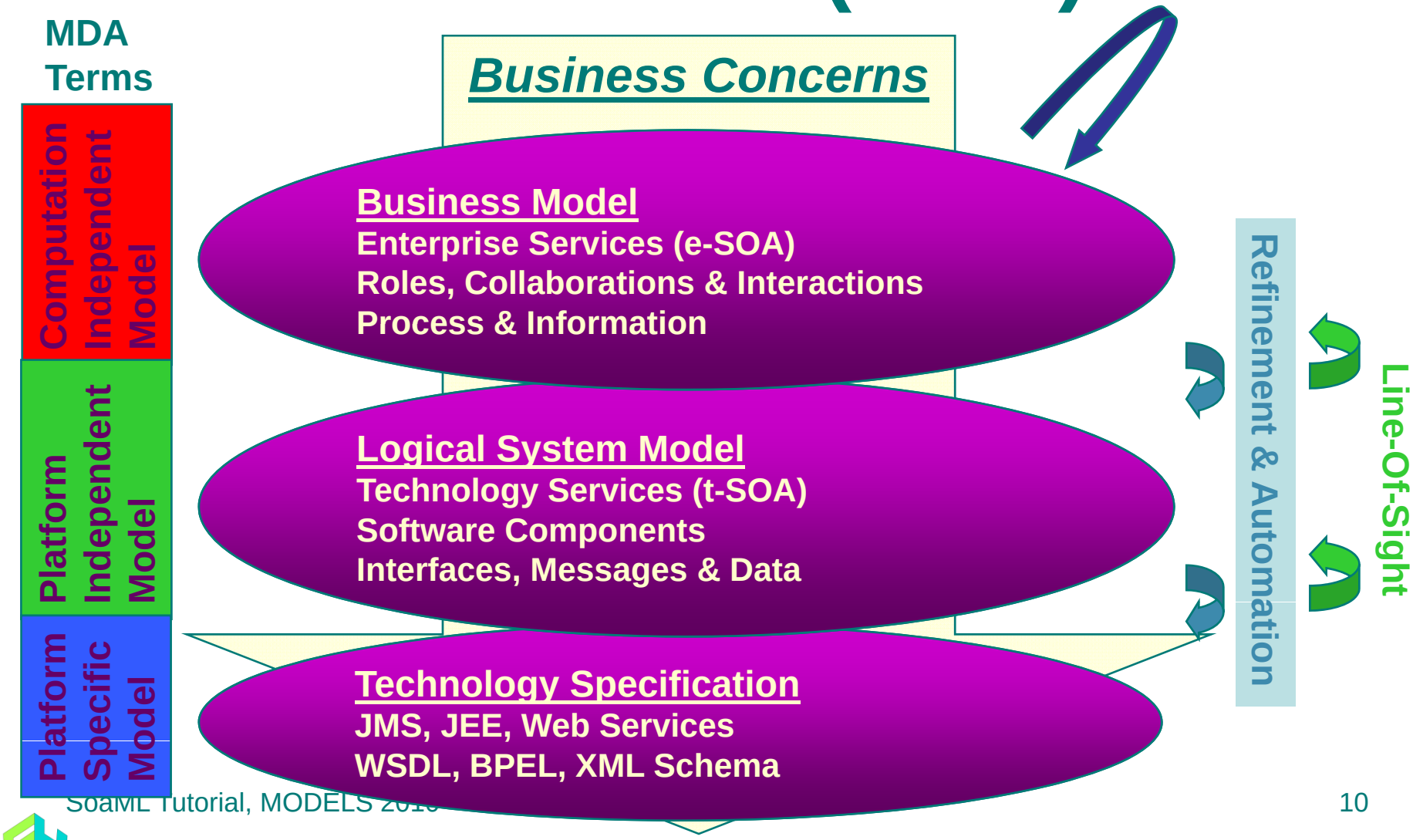
# SoaML – History



# SoaML – Goals

- Model Driven Architecture (MDA).
- Intuitive and complete service modelling in UML.
- Bi-directional asynchronous services.
- Services architectures where parties provide and use multiple services.
- Services defined to contain other services.
- Mapped to and part of a business process specification.
- Compatibility with UML and BPMN.
- Direct mapping to Web services.
- Top-down, bottom up or meet-in-the-middle modelling.
- Design by contract or dynamic adaptation of services.
- Service capability and its contract.
- No changes to UML.

# SOA in Model Driven Architecture (MDA)



# SoaML – Service and SOA

- ***”A service is value delivered to another through a well-defined interface and available to a community (which may be the general public). A service results in work provided to one by another. “***
- Service Oriented Architecture (SOA) is a way of describing and understanding organizations, communities and systems to maximize agility, scale and interoperability.
- SOA is an architectural paradigm for defining how people, organizations and systems provide and use services to achieve results.
- SoaML provides a standard way to architect and model SOA solutions using the Unified Modeling Language (UML).

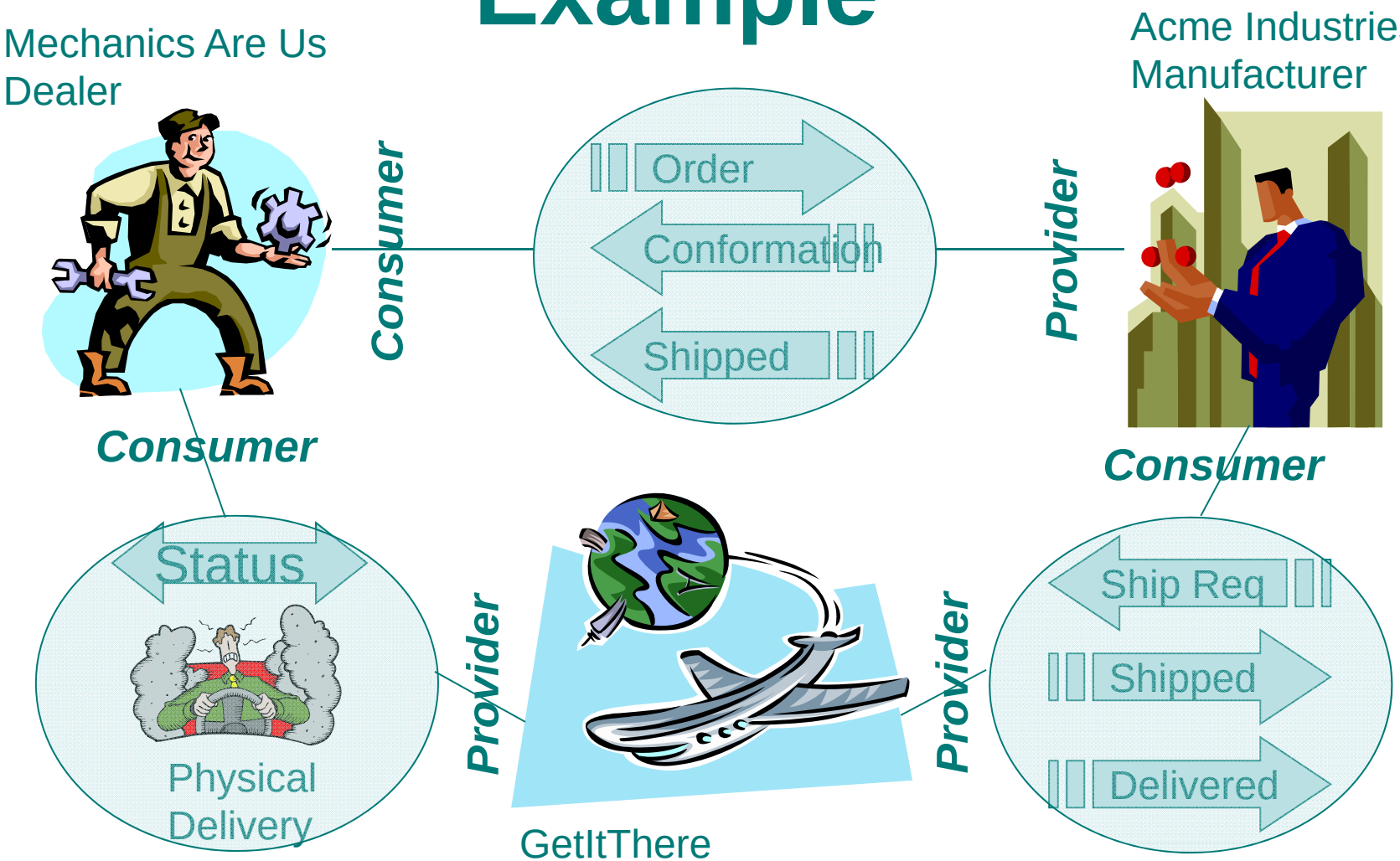
# SoaML – Capabilities

- UML extensions to support service modelling:
  - identifying services
  - specifying services
  - defining service consumers and providers
  - policies for using and providing services.
  - defining classification schemes
  - defining service and service usage requirements and linking them to related OMG metamodels such as the BMM and BPMN.

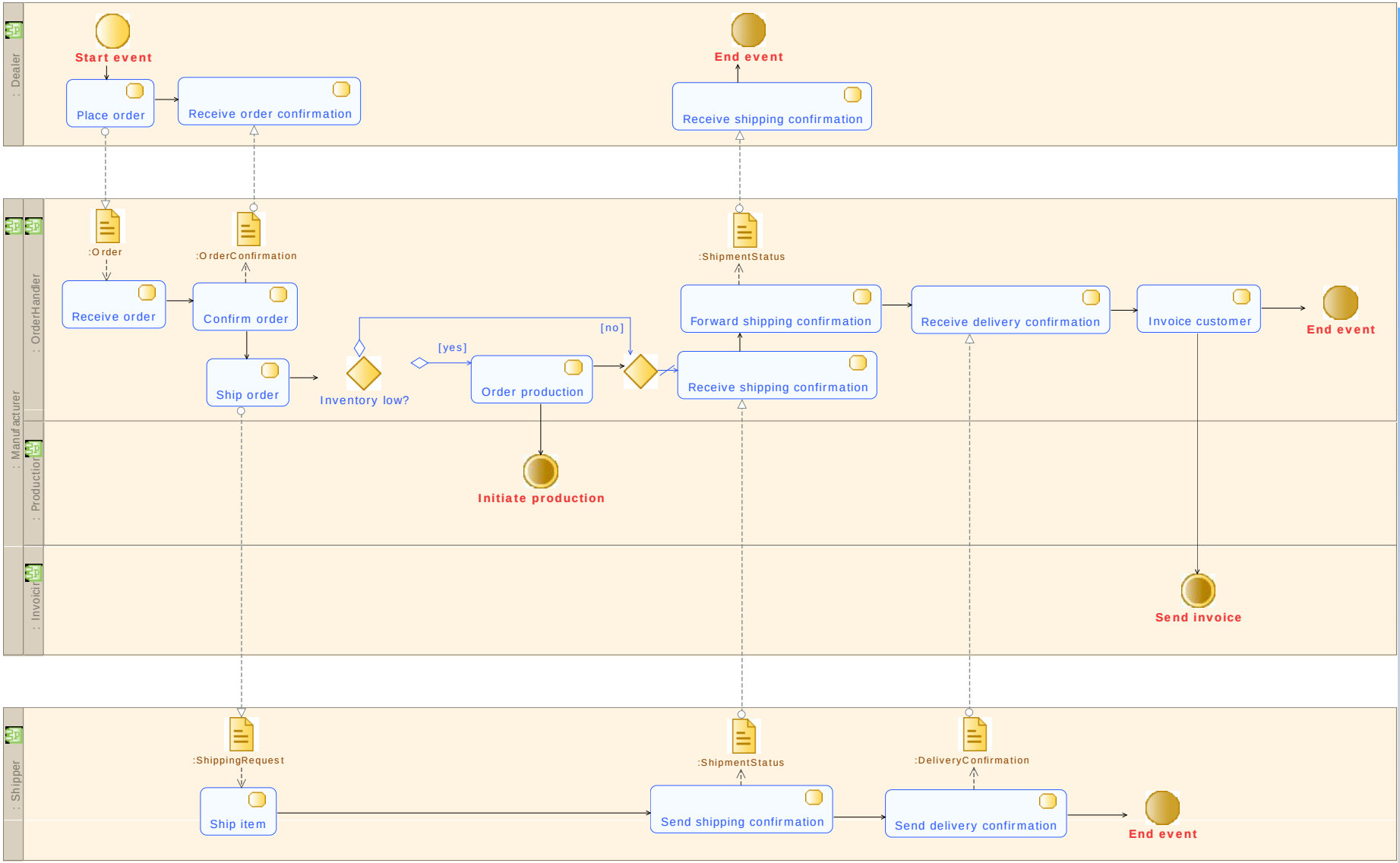
# SoaML – Concepts

- Agent
- Attachment
- Capability
- **Consumer**
- Collaboration
- CollaborationUse
- Expose
- **MessageType**
- Milestone
- **Participant**
- Port
- Property
- **Provider**
- Request
- ServiceChannel
- **ServiceContract**
- **ServiceInterface**
- Service
- **ServicesArchitecture**

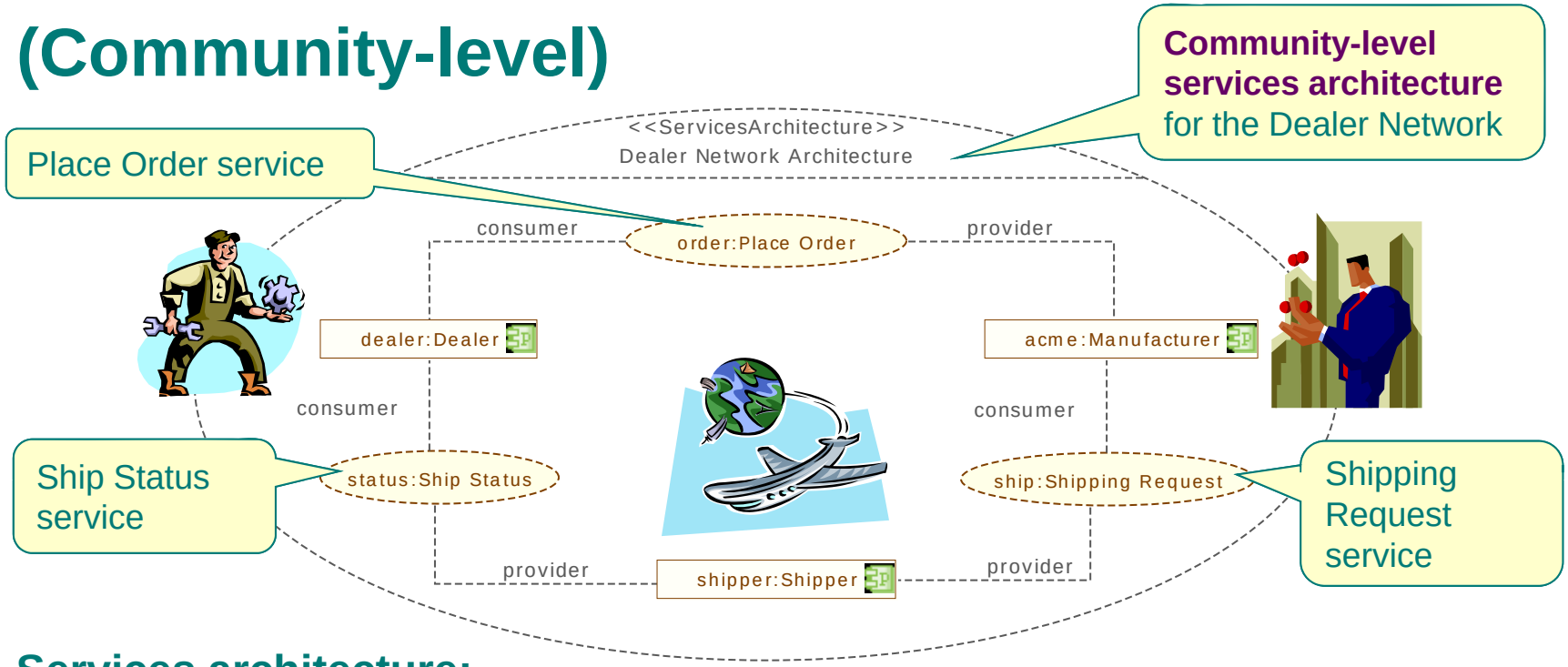
# Marketplace Services Example



# Business process overview



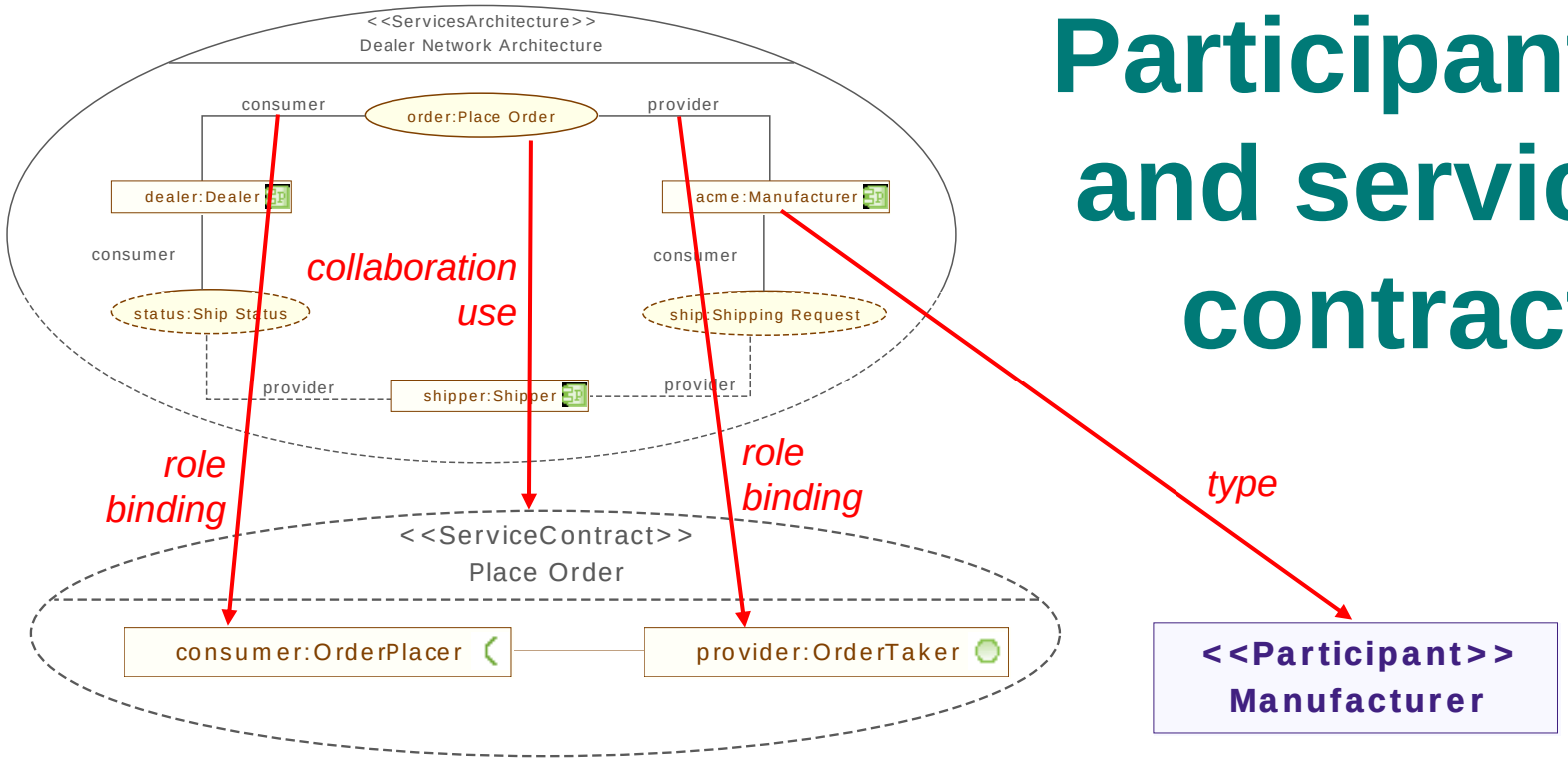
# Services architecture
## (Community-level)




**Services architecture:**

- High level description of how **participants** work together for a purpose by providing and using services expressed as **service contracts**.
- UML collaboration stereotyped «ServicesArchitecture».

# Participants and service contracts




## Service contract:

- Service specifications that define the **roles** each participant plays in the service and the **interfaces** they implement to play that role.
- UML collaboration stereotyped «ServiceContract».

## Participant:

- Represent logical or real people or organizational units that participate in services architectures and/or business processes.
- UML class stereotyped «Participant».

# Services architecture
## (Participant-level)

Order

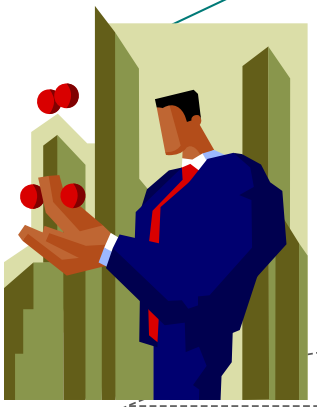
Conformation

Shipped

Ship Req

Shipped

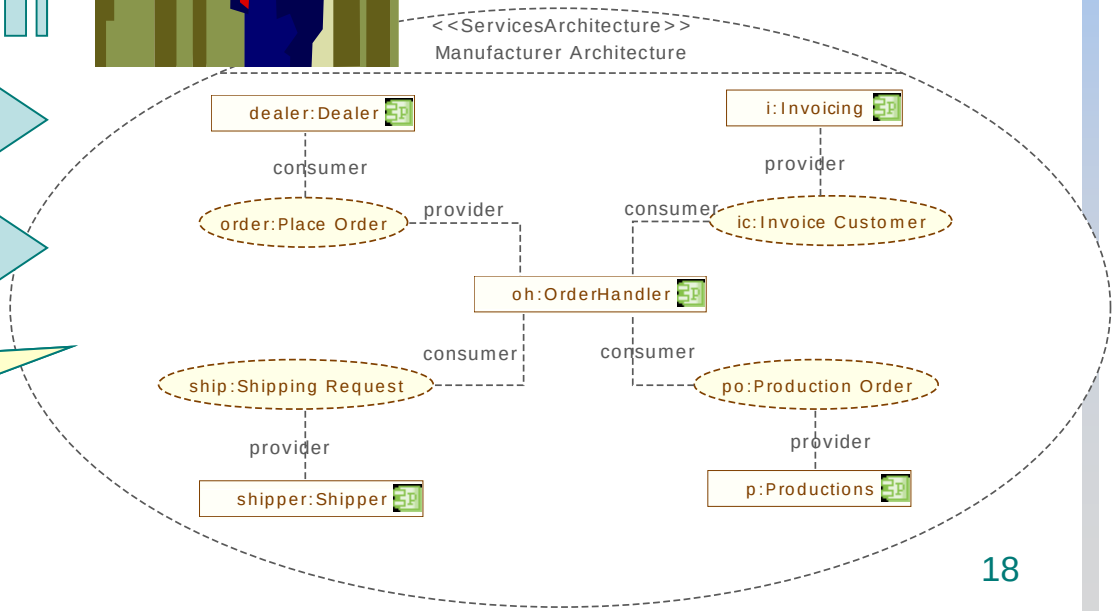
Delivered



OrderHandler

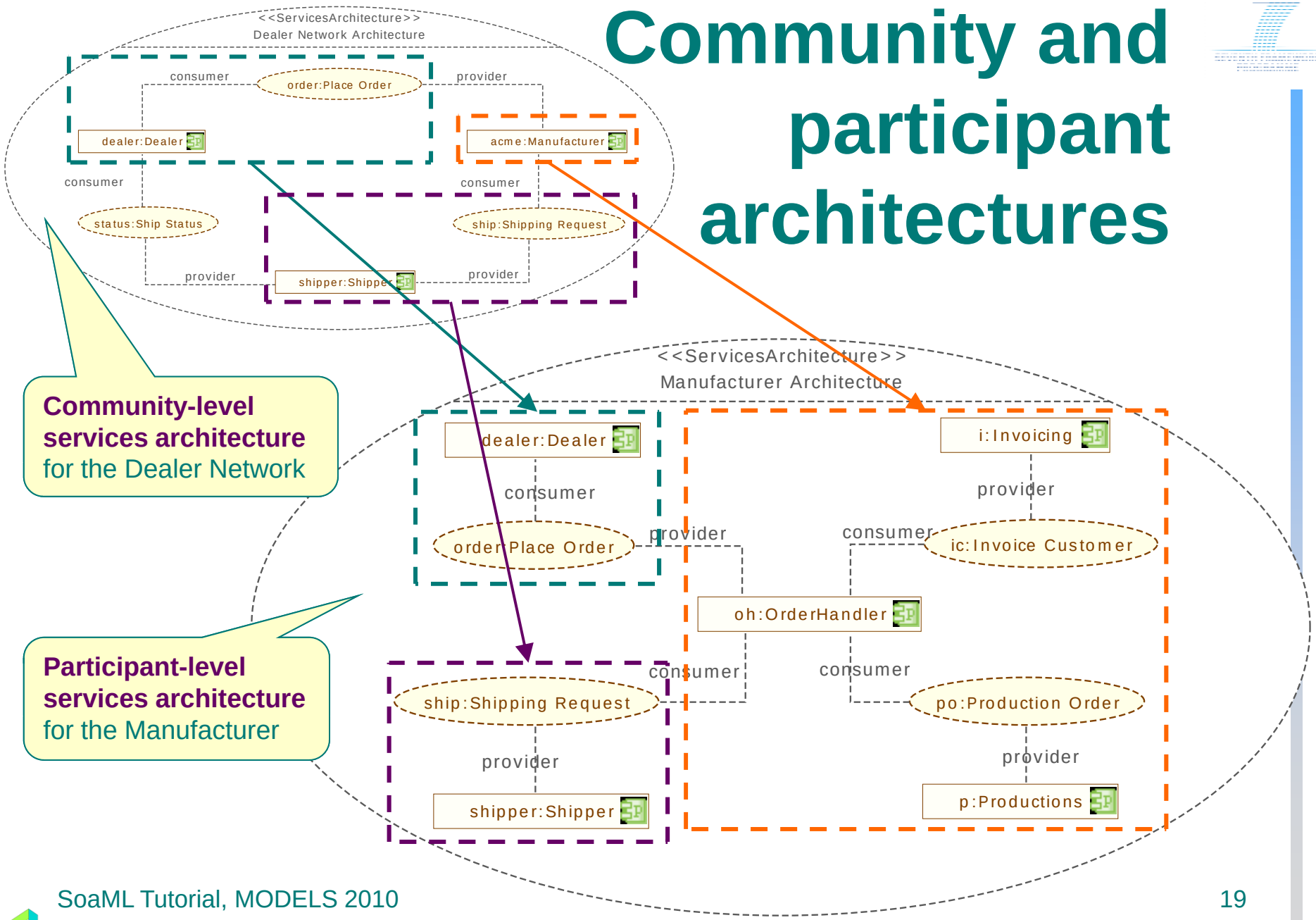
Invoicing

Productions

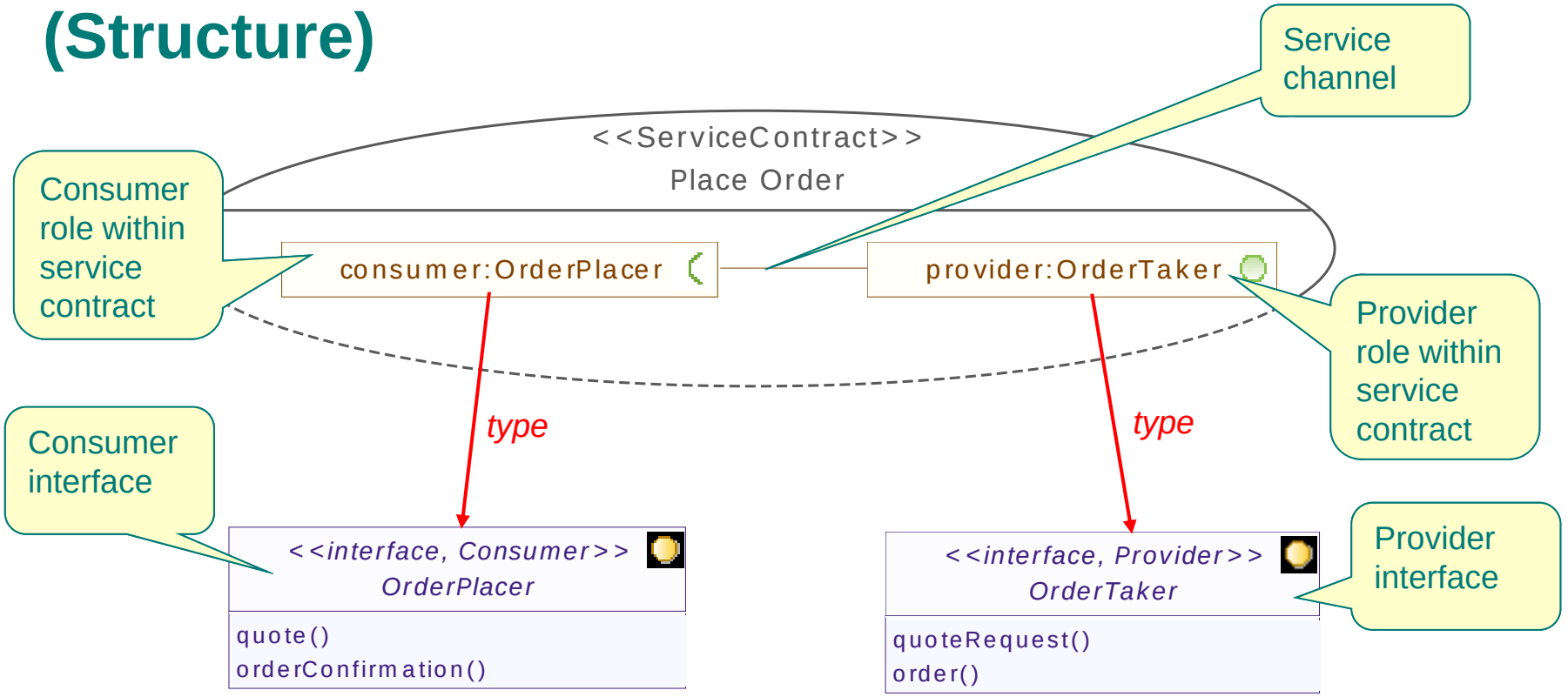



**Participant-level services architecture** for the Manufacturer

# Community and participant architectures

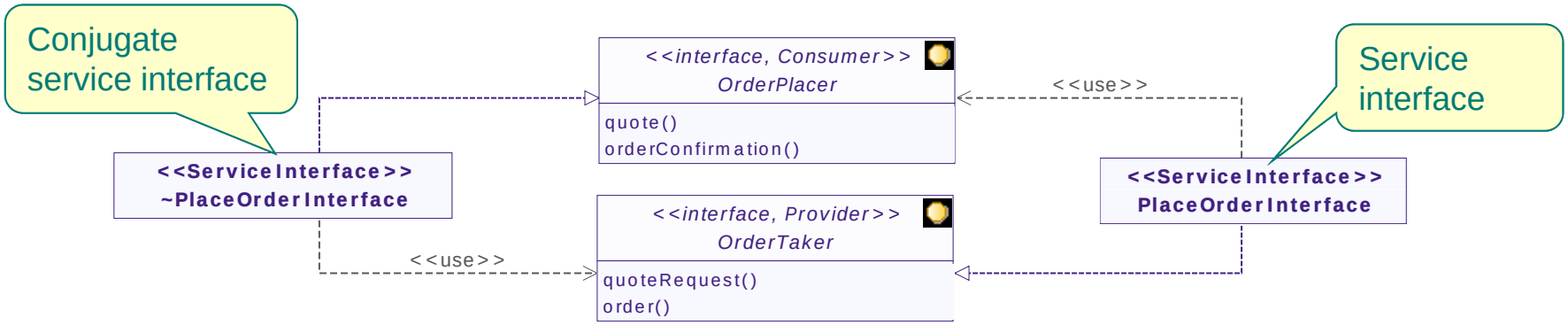


# Service contract: Place order (Structure)




- The service contract represents an **agreement** between the involved participants for how the service is to be provided and consumed.
- The agreement includes the **interfaces**, **choreography** and any other terms and conditions.

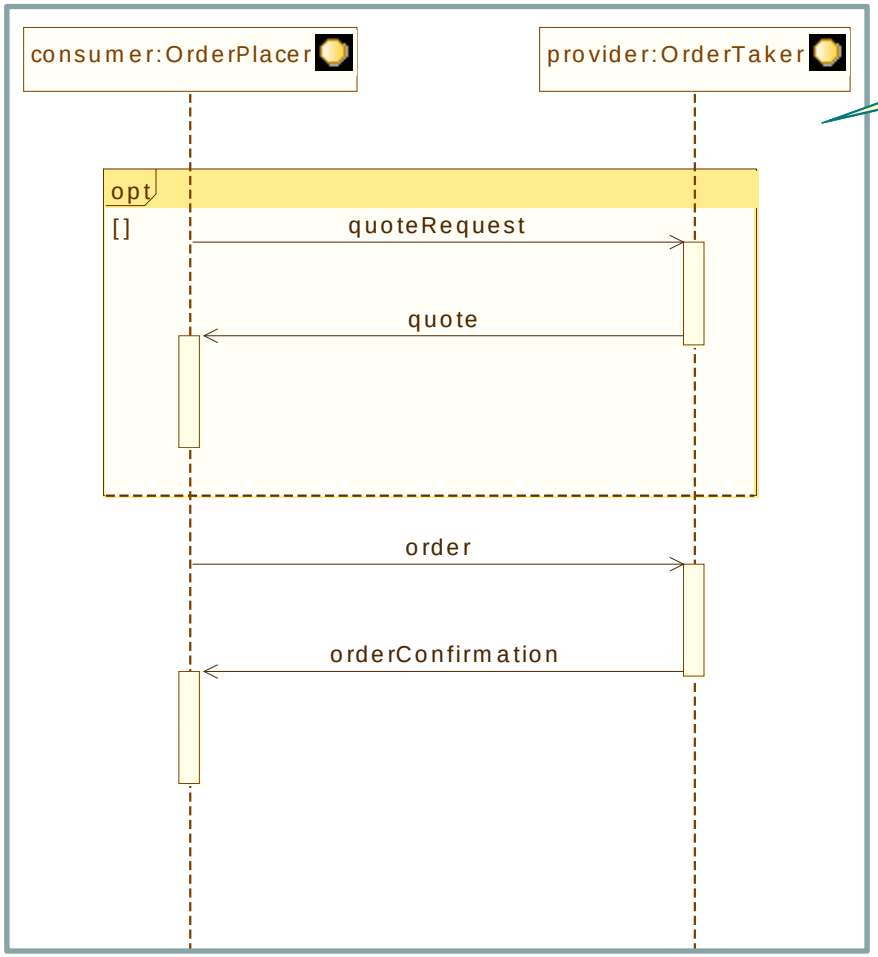
# Service interface: Place order (Structure)



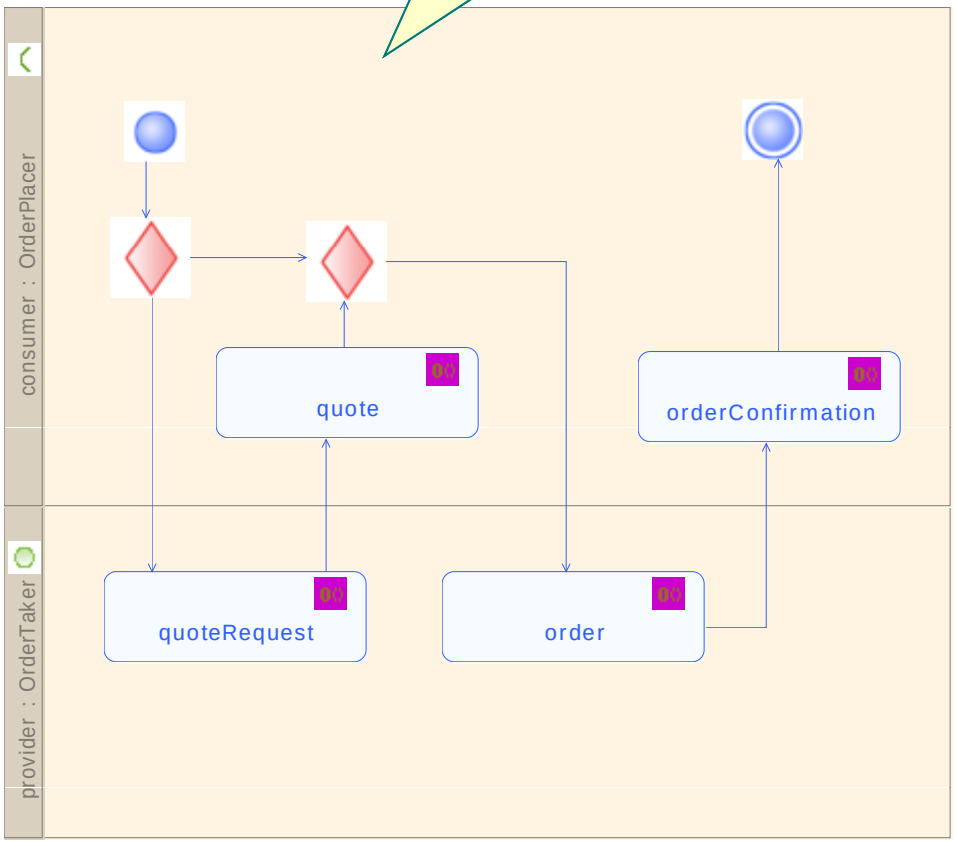

## Service interface:

- Refines the service contract.
- Defines the provided and required interfaces for a service.
- A **service interface** is the type of a **service port** on a participant or component.
  - UML class stereotyped «ServiceInterface».
- A **conjugate service interface** is the type of a **request port** on a participant or component.
  - UML class stereotyped «ServiceInterface» and the names starts with a '**~**'

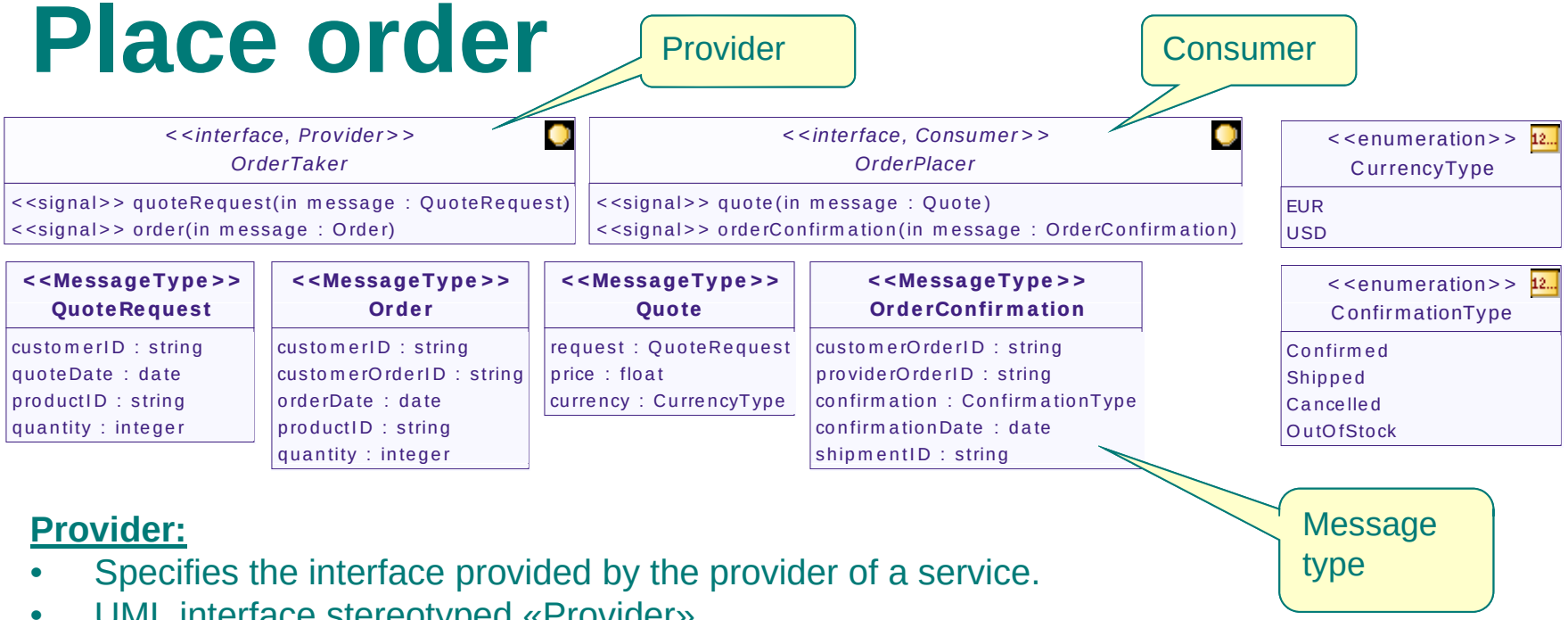
# Choreography: Place order
## (Service contract behaviour)
## (Service interface behaviour)




Service choreography can be specified using any UML behaviour, e.g, interaction or activity



# Interfaces and message types: Place order




**Provider:**

- Specifies the interface provided by the provider of a service.
- UML interface stereotyped «Provider».

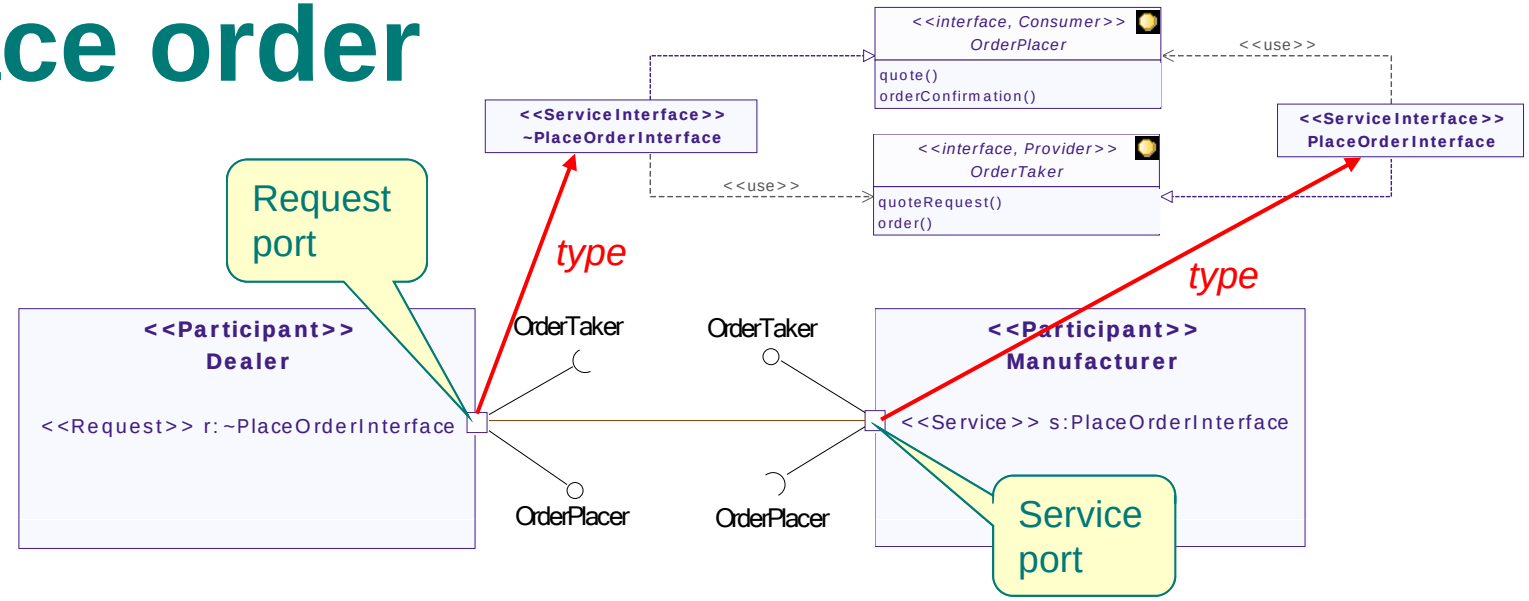
**Consumer:**

- Specifies the interface provided by the consumer of a service.
- UML interface stereotyped «Consumer».

**Message type:**

- Specifies the information exchanged between service consumers and providers.
- UML class stereotyped «MessageType».

# Service and request ports: Place order



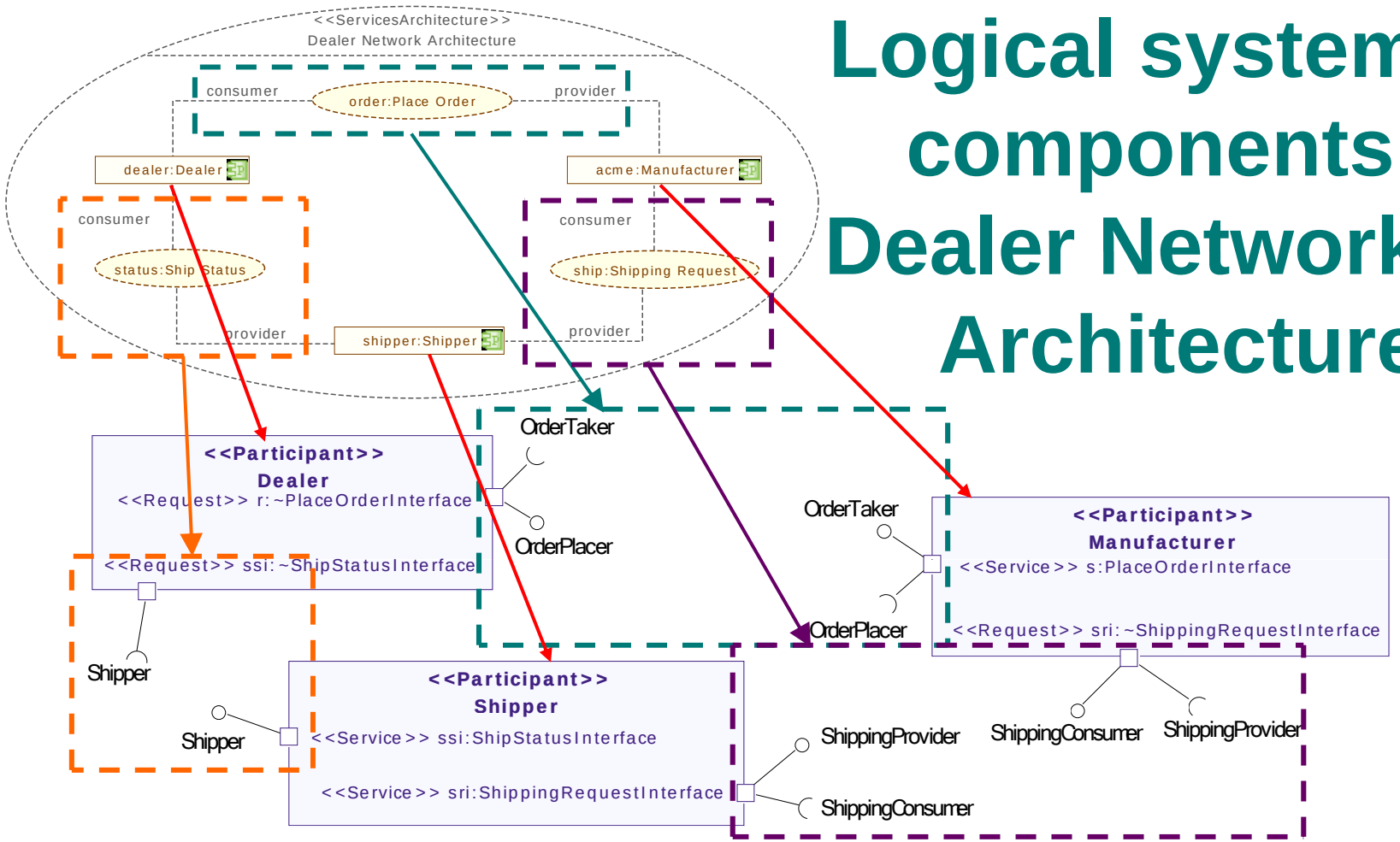

**Service:**

- Specifies a feature of a participant that is the offer of a service by one participant to others using well defined terms, conditions and interfaces.
- UML port stereotyped «Service» on a participant or component.

**Request:**

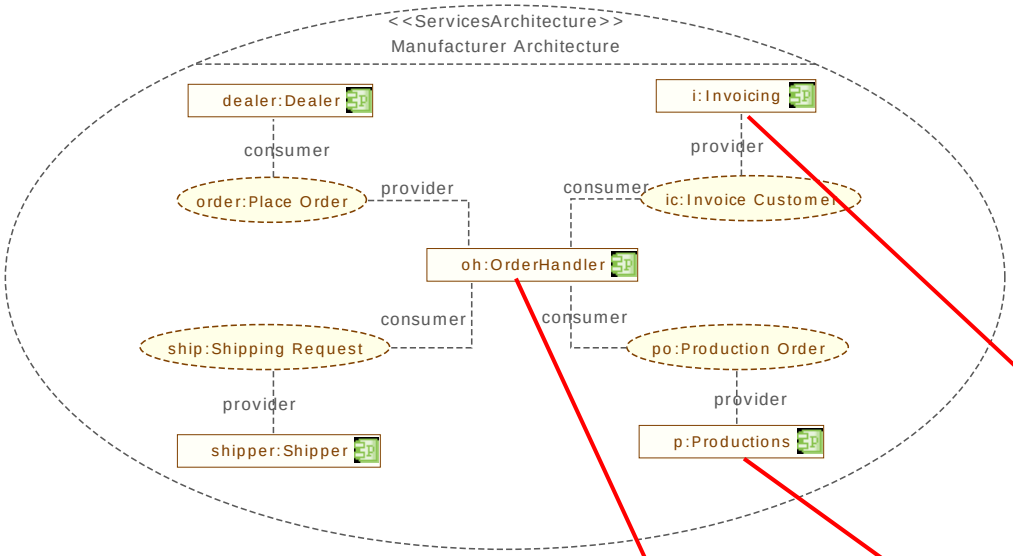
- Specifies a feature of a participant that represents a service the participant needs and consumes from other participants.
- UML port stereotyped «Request» on a participant or component.

# Logical system components: Dealer Network Architecture

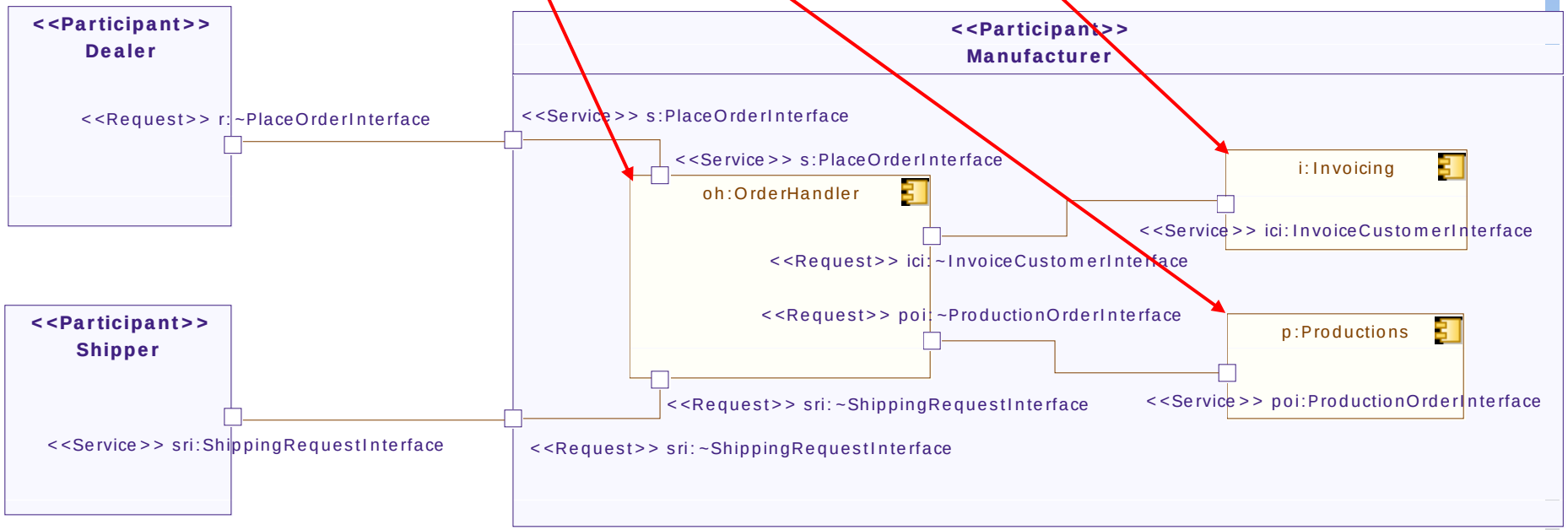



- Components implement the service interfaces providing the link to systems. Participants and services may be used in multiple architectures.

# Software components: Manufacturer Architecture




*Mapped to software components*




# SoaML methodology

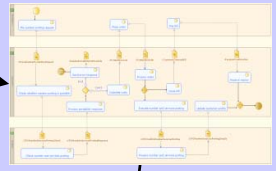
# SoaML Methodology

**Flexible Business Model (CIM)**

Business Processes, Business Data, Business Rules, Organization

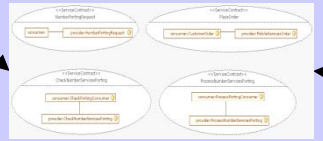
**Business Architecture Model (BAM)**

Business Goals



Business Processes

Capabilities

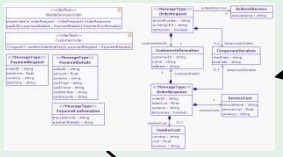


Services Architectures

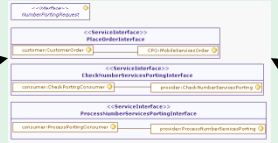
Service Contracts and Choreographies

Model to Model (M2M) Transformation

**System Architecture Model (SAM)**

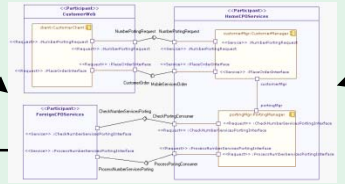


Interfaces and Messages



Service Interfaces

Service Choreographies



Service Orchestrations

Software Components

Model to Text (M2T) Transformation

**Platform-Specific Model (PSM)**

Cloud, Web Services, JEE, MAS, P2P/Grid, SWS

CIM
PIM
PSM

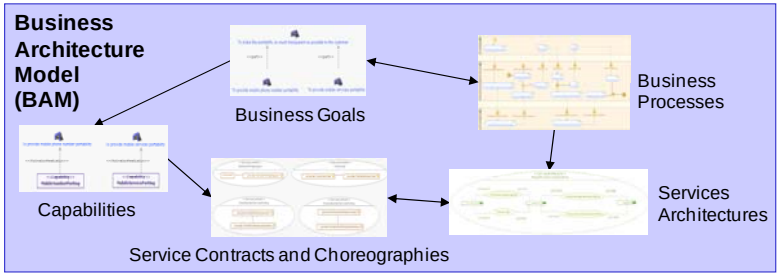
**CIMFlex**

**Modelio**

- Business perspective on SOA
- IT perspective on SOA

**Modelio**

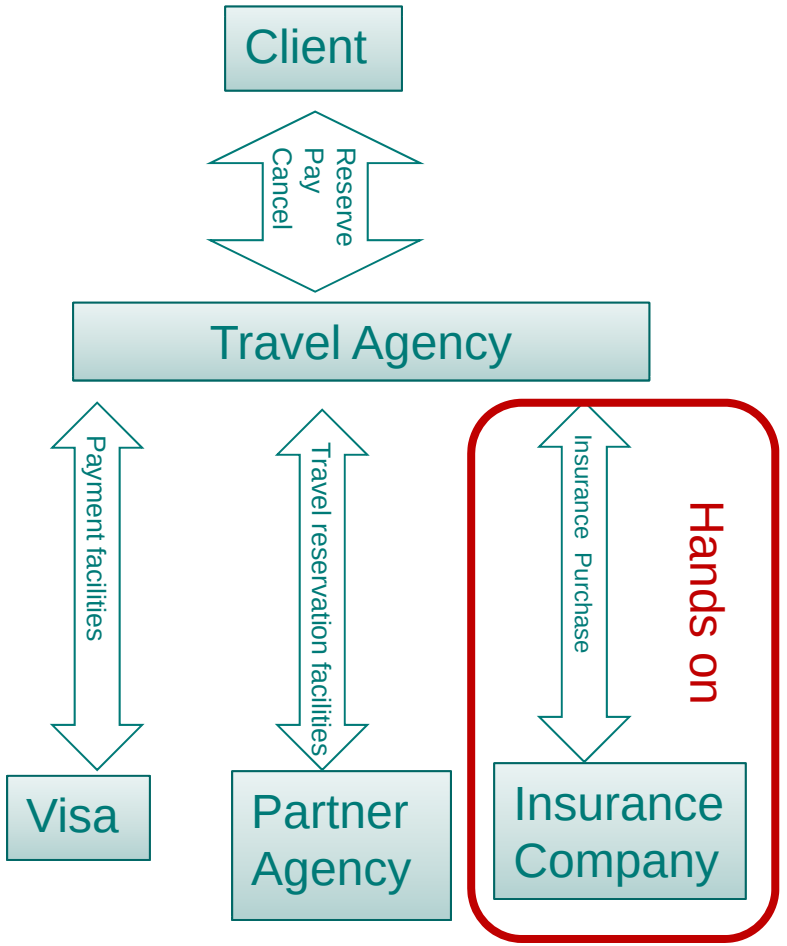
# Business Architecture Model (BAM)



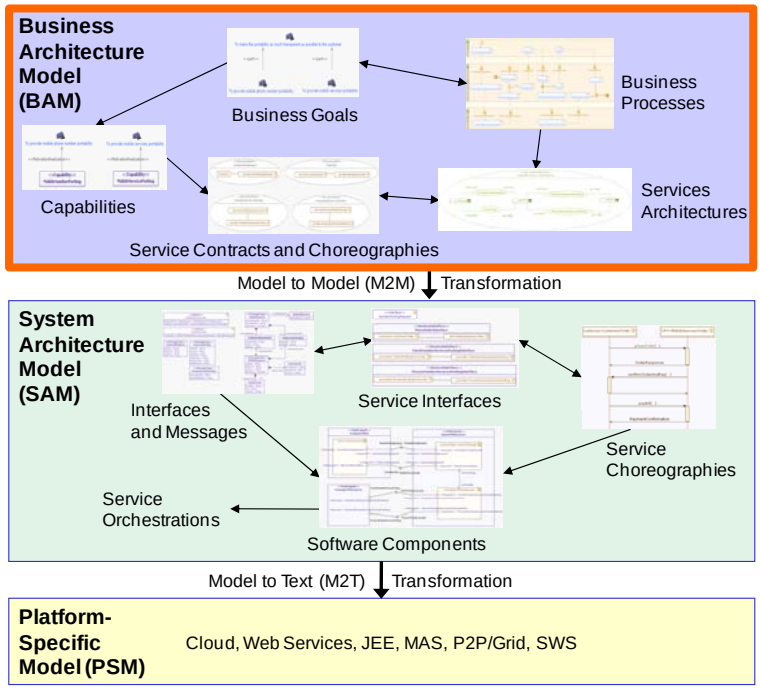

- Represents the business perspective of an SOA.
- Captures business requirements and identifies services:
  - Business goals
  - Business processes
  - Capabilities
- Specifies the business community and its services:
  - **Services architectures** of the business community.
  - **Service contracts** between the business entities participating in the community.

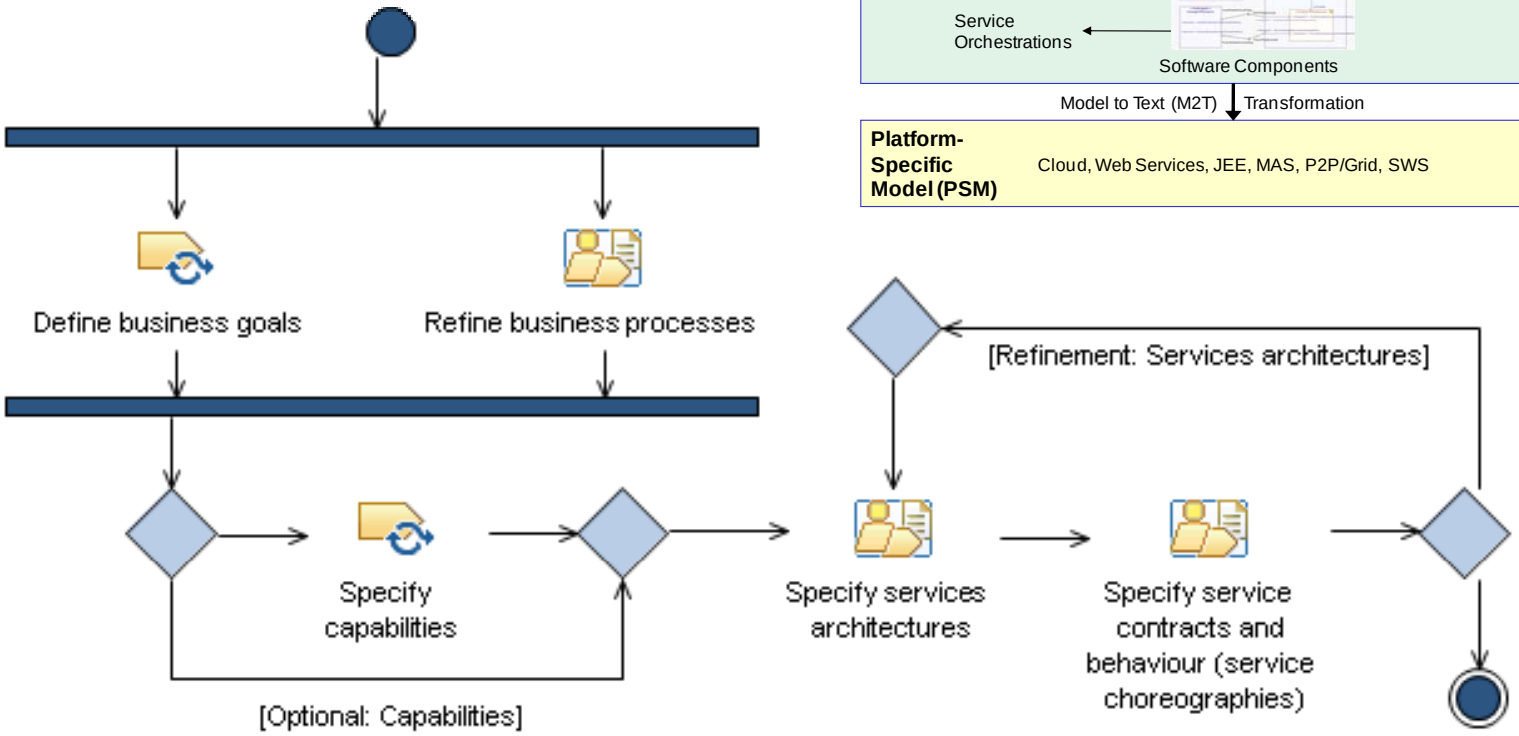
# Case study: Travel agency
## (Discount Voyage)

- A Travel Agency has some contact with Partner Agencies which provide reservation for Flights, Hotels, Cars, etc.
- A Client can interact with the Travel Agency to:
  - Reserve a Travel:
  - Cancel a Travel
  - Pay the Travel
- The Travel Agency needs to be in contact with a Visa payment center in order to be paid by the Client, and pay the Partner Agencies.



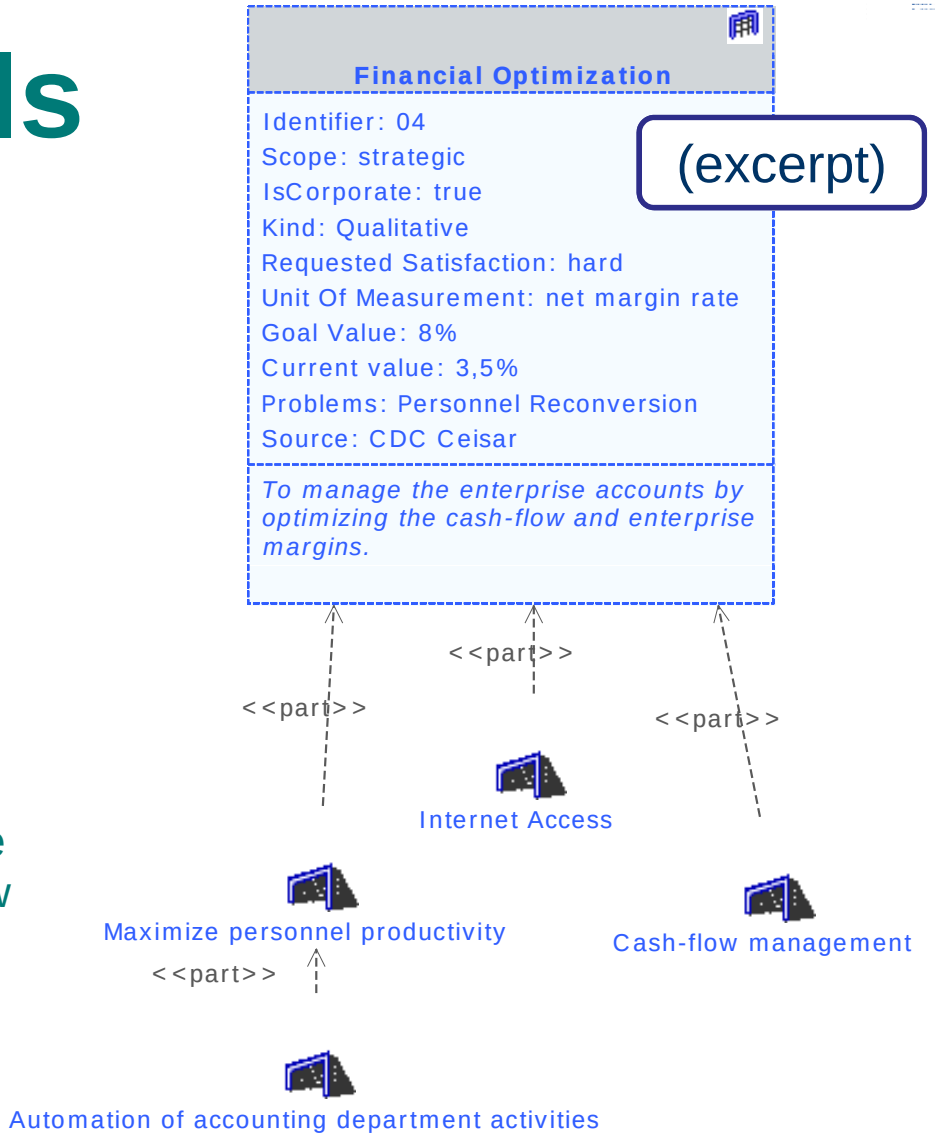
# Business Architecture Modelling



# Define business goals

- Business Motivation Model (BMM) identifies and defines:
  - factors that motivate the business plan
  - elements of a business plan
- Modelling steps:
  - Identify goals
  - Specify goals and their relationships
  - Perform goal analysis linking the goals to existing or potential new business processes

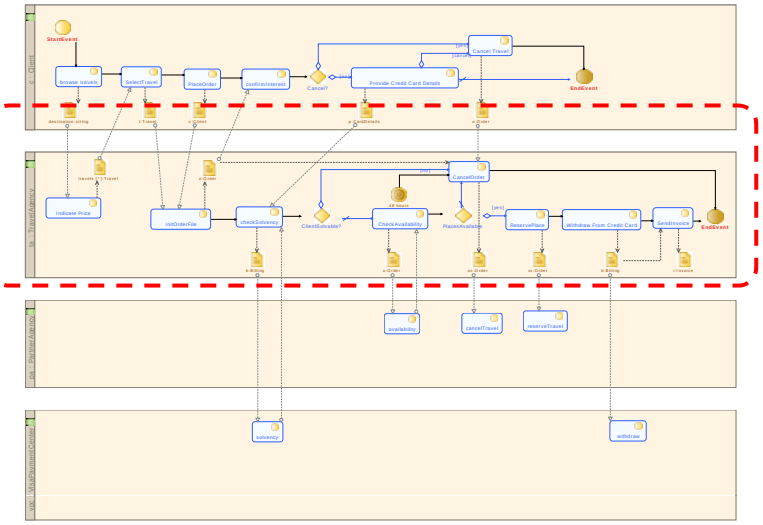


# Refine business processes (1/2)

- Identify relevant business processes:
  - Focus on business processes that enable the business goals to be met.
  - Public and collaborative business processes between different business organizations.
    - Map to **community-level services architectures** in SoaML
  - Private business processes within a business organization.
    - Map to **participant-level services architectures** in SoaML.
- Refine the business processes:
  - Identify business entities and model them as **pools** or **swimlanes**.
    - Map to **participants** in SoaML
  - Focus on tasks that describe the interaction points between the business entities.
    - Map to **service contracts** in SoaML
  - Identify the control and data flows between these tasks.
    - Map to **message types** in SoaML

# Refine business processes (2/2)

Case study example

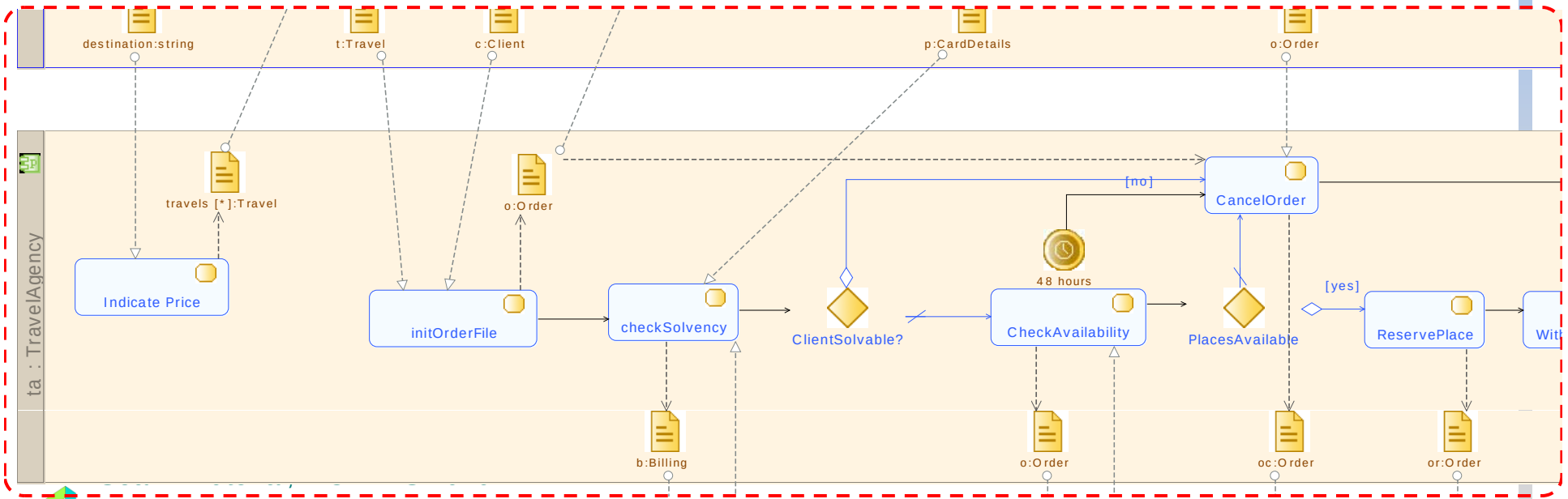


Client

TravelAgency

PartnerAgency

VisaPaymentCenter



# Specify capabilities [optional]

- A capability identifies a cohesive set of functions or resources provided by one or more participants.
  - Capabilities are used to identify candidate services.
  - Starting point for large business architectures.
- Modelling techniques:
  - Goal-service modelling
    - Identifies capabilities needed to enable business goals.
  - Domain decomposition
    - Uses activities in business processes and other descriptions of business functions to identify needed capabilities.
  - Existing asset analysis
    - Mines capabilities from existing applications.

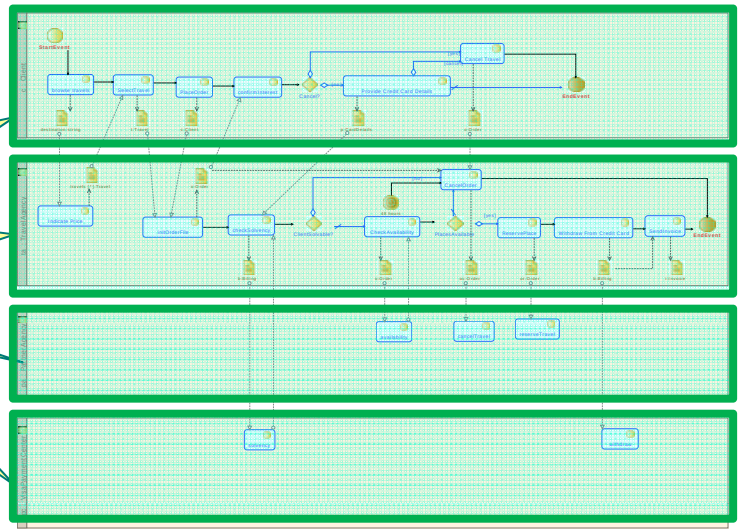
# Specify services architectures (1/3)

- Services architectures
  - UML collaborations stereotyped «ServicesArchitecture».
  - Identified from the BPMN processes.
- Participants
  - UML classes stereotyped «Participant».
  - Identified from pools, participants and lanes in the BPMN processes.
- Service contracts.
  - UML collaborations stereotyped «ServiceContract».
  - Identified from possible interactions between participants in the BPMN processes.

# Specify services architectures (2/3)

Case study example

Each Pool can be mapped to a Participant

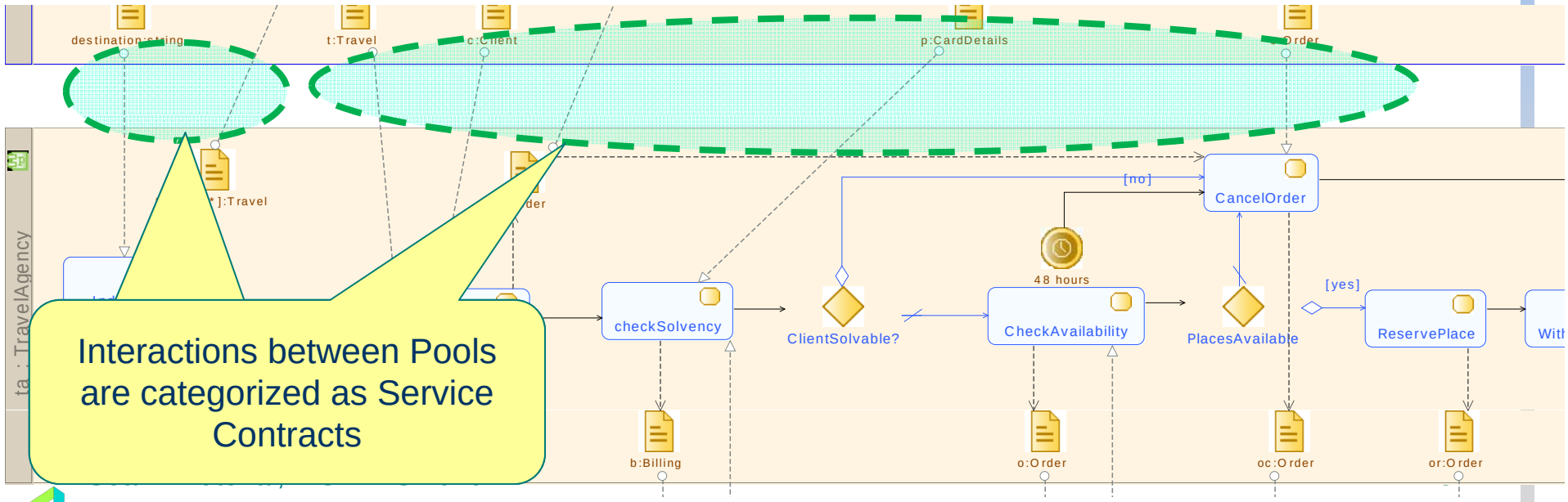


Client

TravelAgency

PartnerAgency

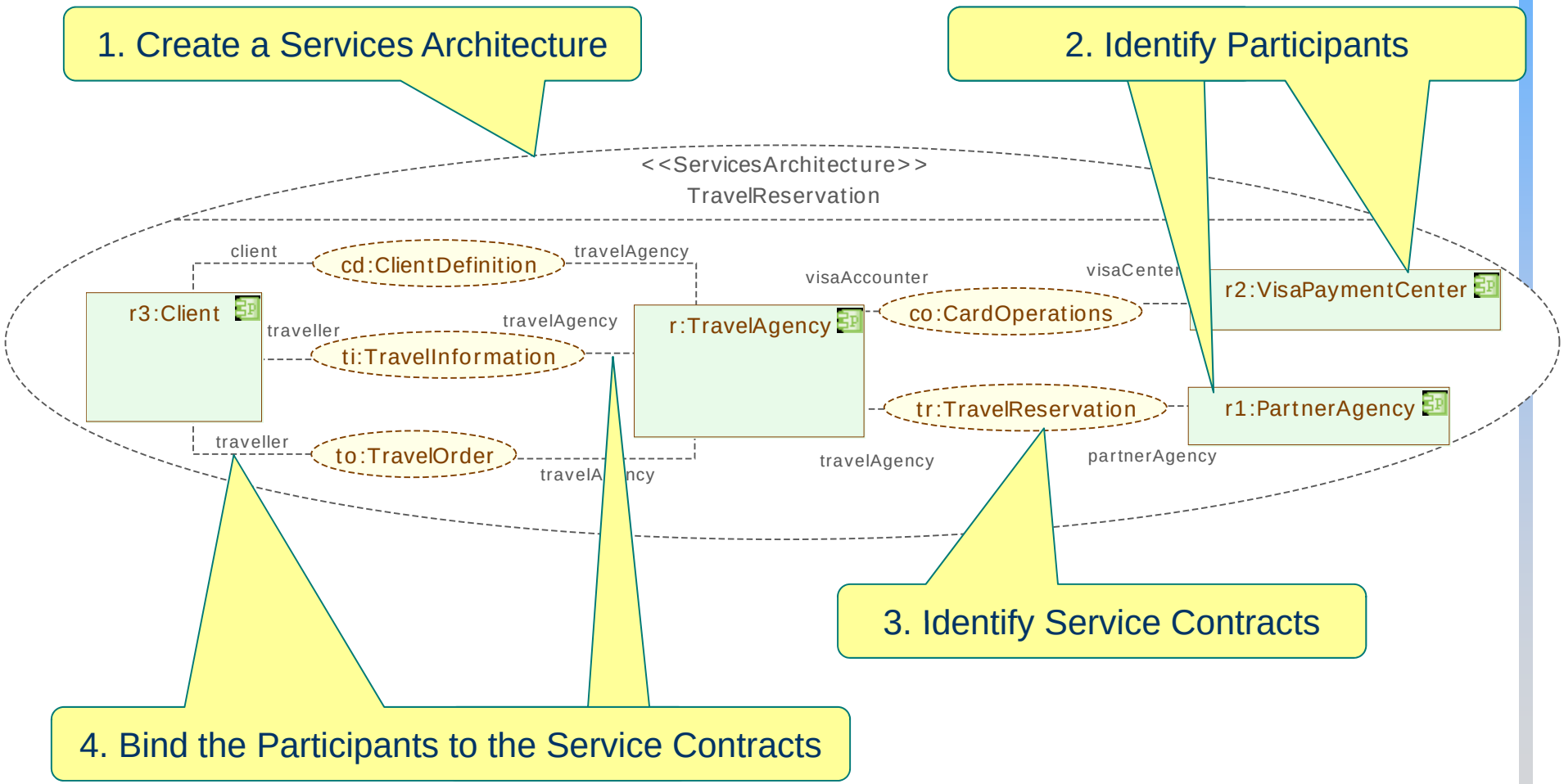
VisaPaymentCenter




Interactions between Pools are categorized as Service Contracts

# Specify services architectures (3/3)

Case study example

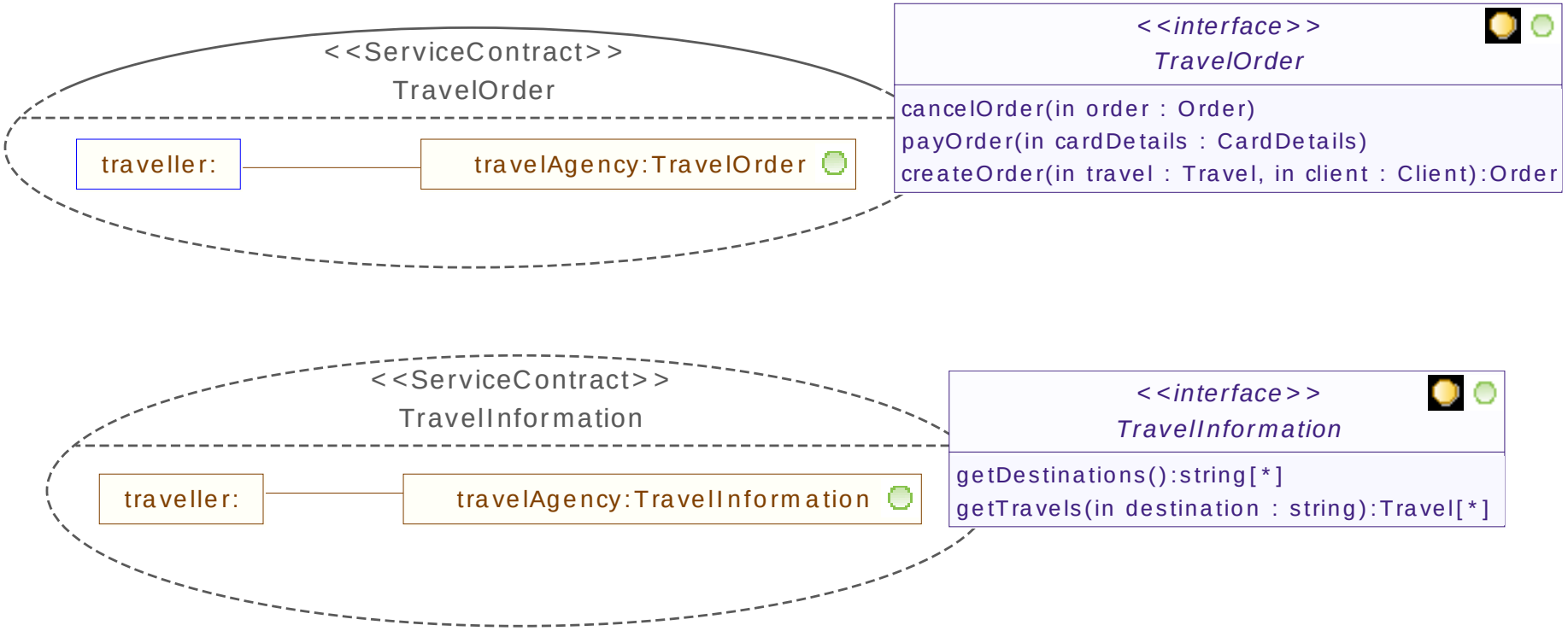


# Specify service contracts (1/2)

- Analyse the BPMN diagrams to identify service contracts.
- This is a design-choice as there is no single construct in BPMN that resembles a service contract.
- Certain pattern of objects can reveal a service contract, e.g.
  - two single tasks follow one after another across a pool or lane and are
  - connected with a sequence flow and associated with a data object.
- Specify the provider and consumer roles and interfaces
  - operations and messages at the business level

# Specify service contracts (2/2)

Case study example

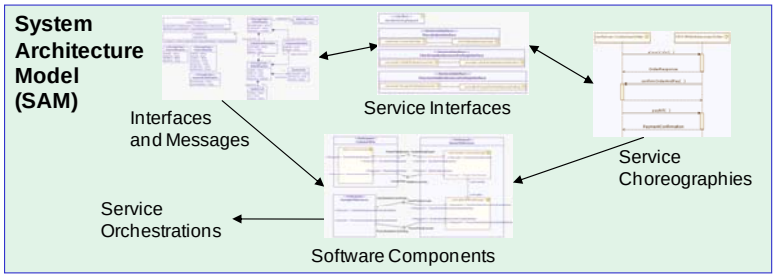


# Specify service choreographies (service contract behaviour) [optional]

- The service contract behaviour specifies the **choreography**:
  - what is transmitted and
  - when it is transmitted between participants
  - to enact a service exchange.

- We recommend to model the behaviour of any complex service contract in order to get a better understanding of the interaction between the roles.
  - use any UML behaviour or e.g. BPMN
  - message sequence between the provider and consumer interfaces

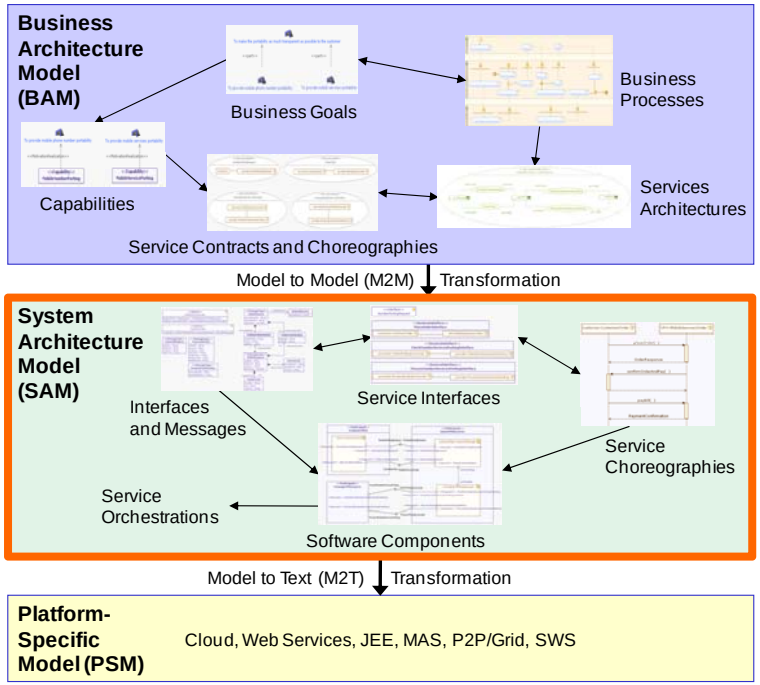
# System Architecture Model (SAM)

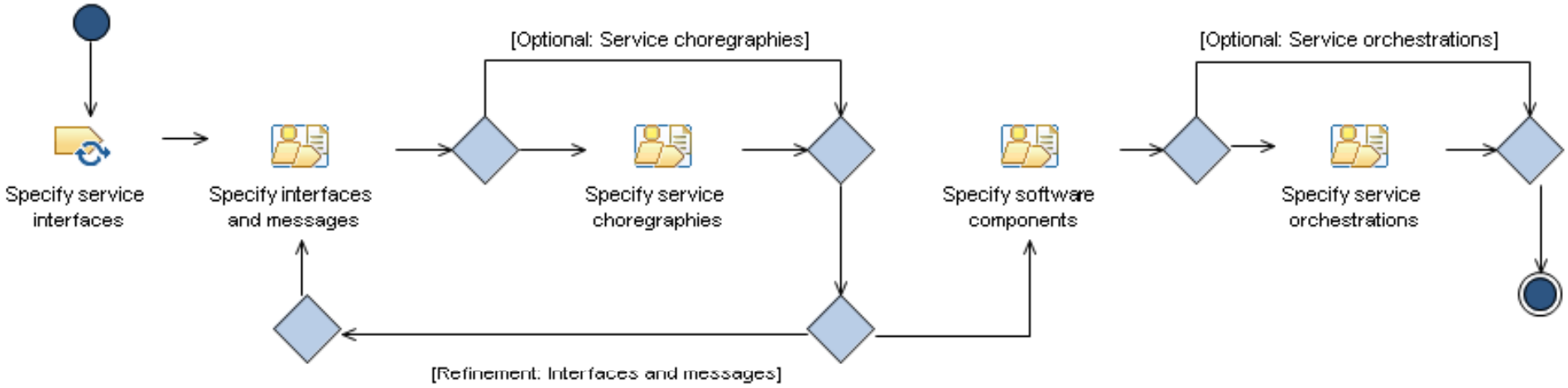



- Represents the IT perspective of an SOA
  - It partitions the system into software components and interfaces.
- Structural modelling to specify components, their interfaces and dependencies:
  - Service interfaces
  - Interfaces and messages
  - Software components
- Behavioural modelling to specify component interactions and protocols:
  - Service choreographies
  - Service orchestrations

# System Architecture Modelling

- The System Architecture Modelling will refine the models of the Business Architecture Modelling.



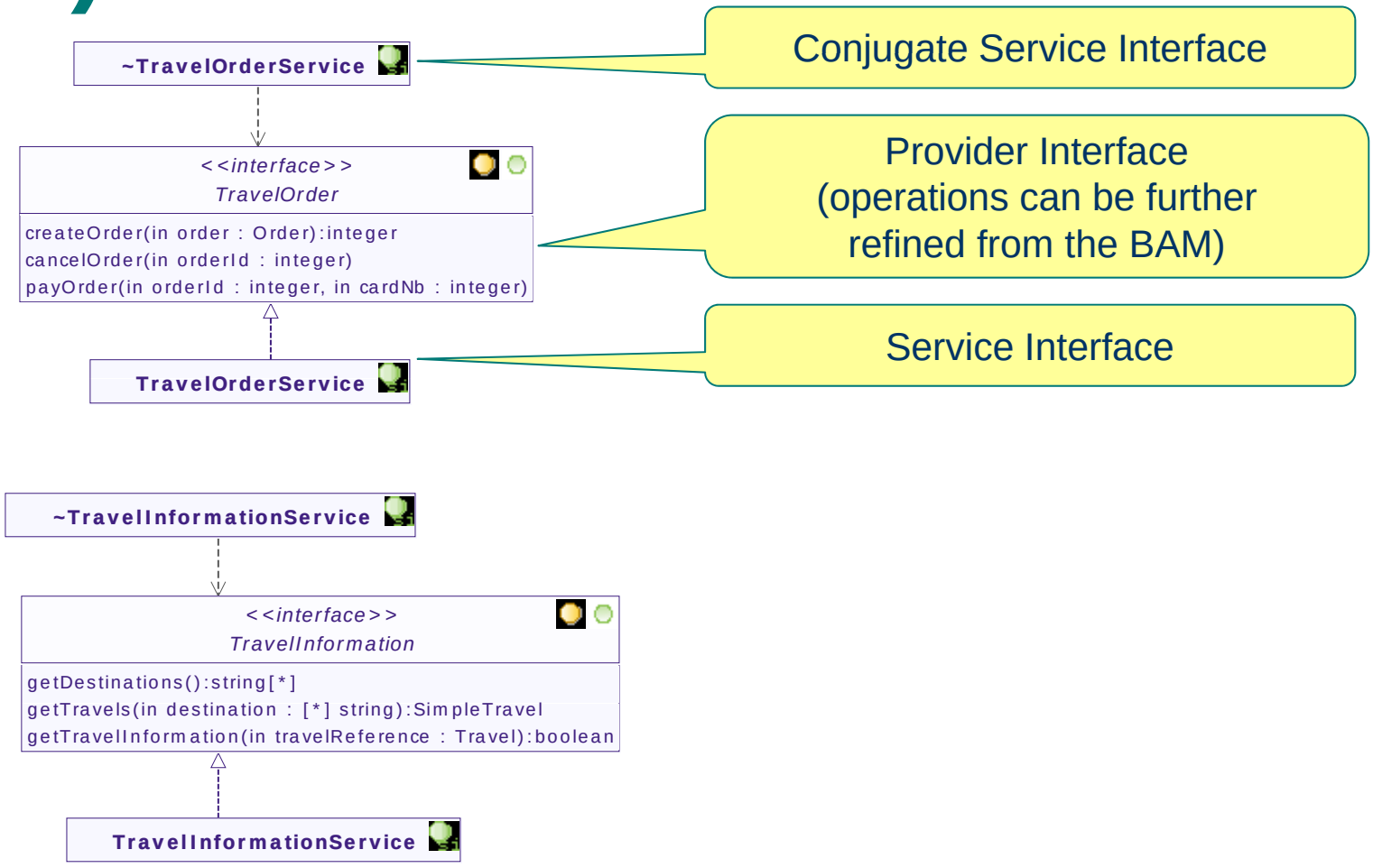



# Specify service interfaces (1/2)

- Define **service interface** as UML class stereotyped «ServiceInterface».
  - One-to-one mapping between service contracts and service interfaces.
- Define **provided interface** as UML interface stereotyped «Provider».
- Define **consumer interface** as UML Interface stereotyped «Consumer».
  - The consumer interface is specified only when callbacks are needed.
- Link the **provided** and **consumer interfaces** to the **service interface**.
  - Create an **interface realization** between the provider interface and the service interface.
  - Create a **usage link** between the consumer interface and the service interface.
- Create and link a **conjugate service interface** as UML class stereotyped «ServiceInterface».
  - The name starts with a '**~**', and represents the counter-part of a service interface.
  - It **uses** the **provider interface**
  - It **realizes** the **consumer interface**.

# Specify service interfaces (2/2)

Case study example

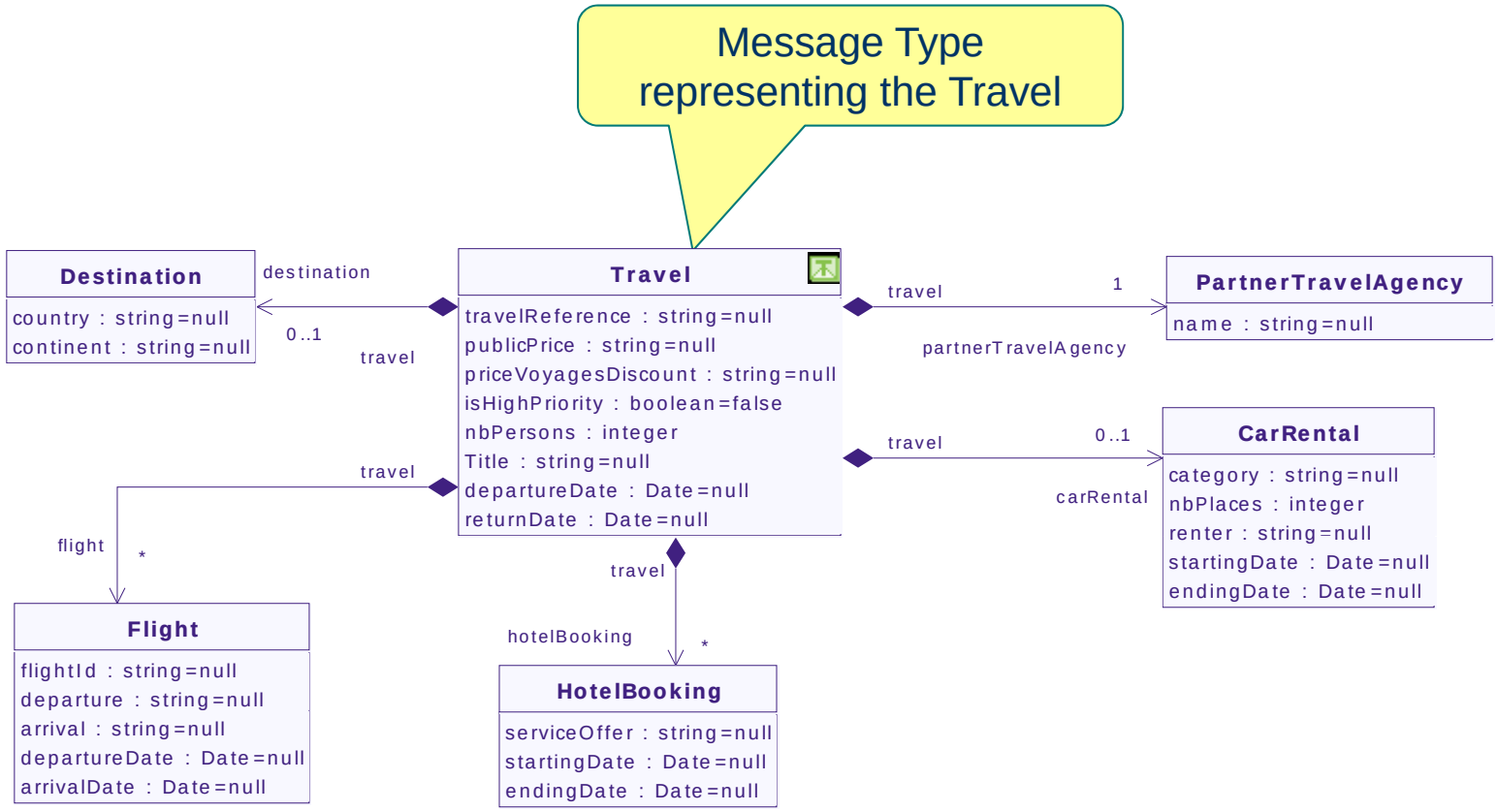


# Specify interfaces and messages (1/2)

- Define **message types** as UML classes stereotyped «MessageType».
- The specification of message types is closely linked to the specification of operations and callbacks in the interfaces.
  - Data passed into, and/or returned from the invocation of an operation or event signal defined in an interface.
  - For an **asynchronous** document-centric approach you will typically only specify one **input parameter**.
  - For a **synchronous** document-centric approach you will typically only specify one **input parameter** and one **response parameter**.
- Message types may have properties that can be either modelled as UML properties or associated UML classes.

# Specify interfaces and messages (2/2)

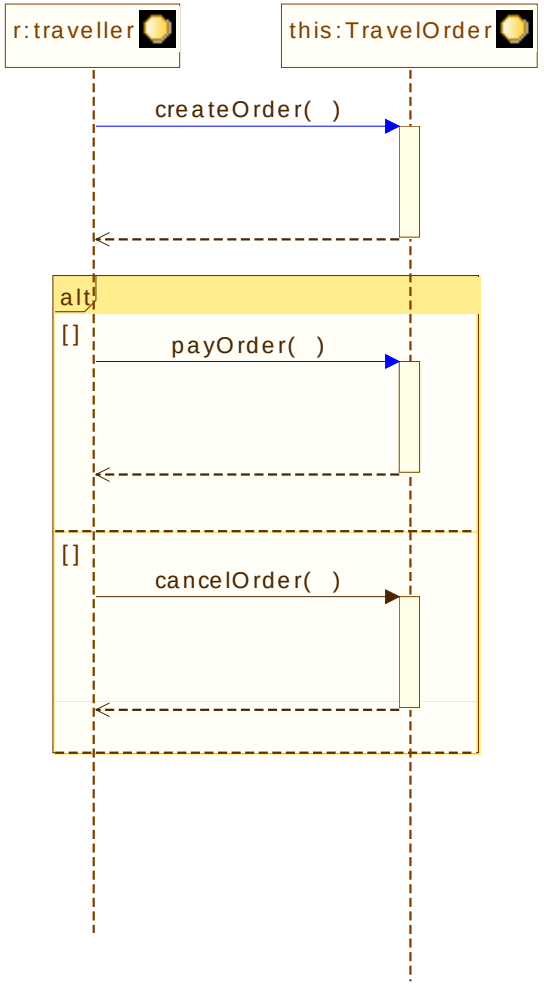
Case study example



# Specify service choreographies [optional]

Case study example

- The **behaviour** of a **service interface**.
  - Useful for understanding complex services.
  - Refinement of the service contract behaviour of the BAM.
- Expresses the expected interactions between consumers and providers of services.
- It can be specified as any UML behaviour.
  - Activity, interaction or state machine.
  - Message sequence between the provider and consumer interfaces.

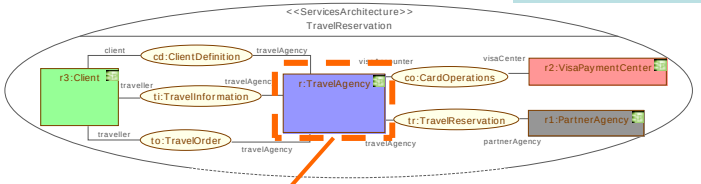


# Specify software components (1/2)

- Refine the participants of the services architecture by possibly decomposing them into several components.
  - Use SoaML participants for logical system components.
  - Use UML components for components that represent internal design-time software components.
- Create the ports of the participants/components
  - **Service port** will be typed by a **service interface** (service provider).
  - **Request port** will be typed by a **conjugate service interface** (service consumer).
- Link the ports through their provided/required interfaces
  - The types of the **linked ports** must be such that a conjugate service interface is **matched** by a service interface.

# Specify software components (2/2)

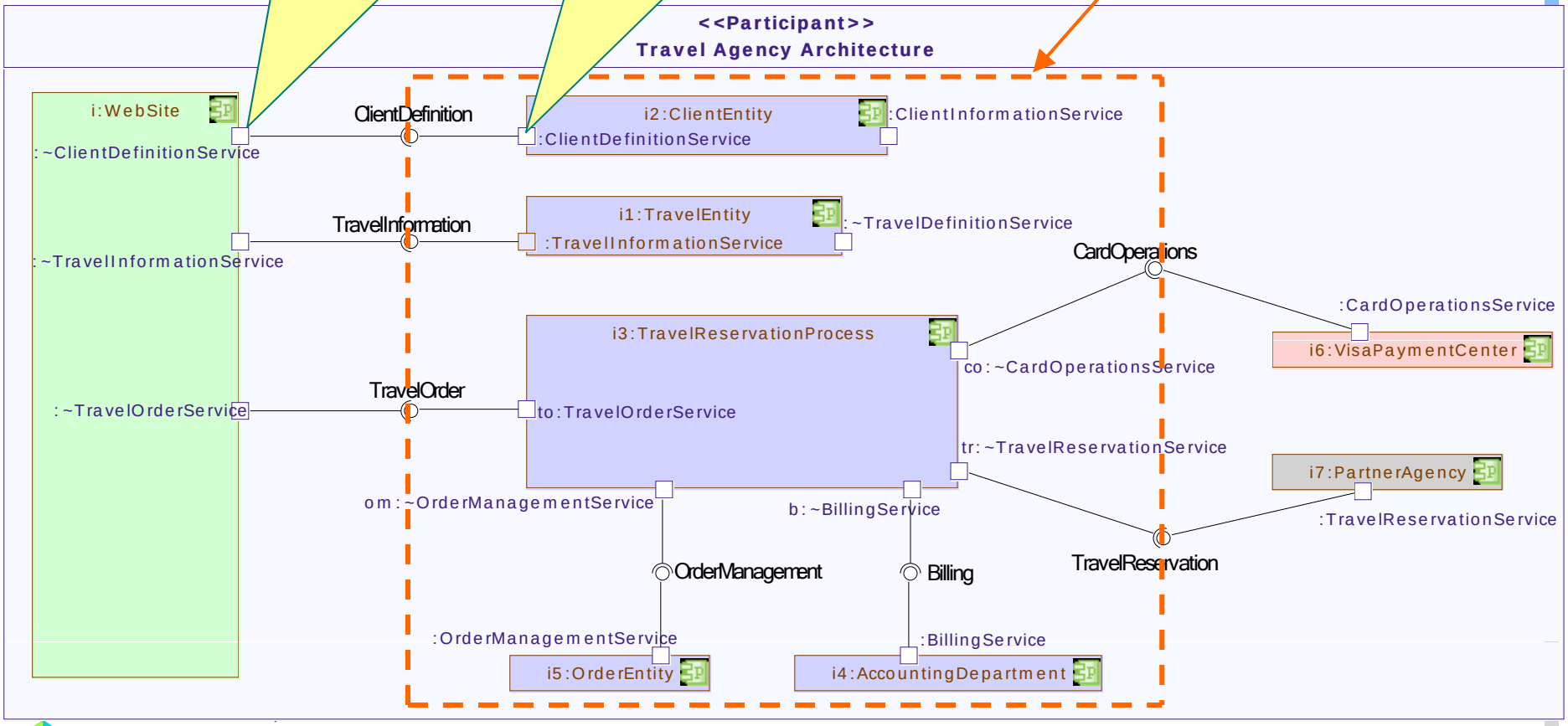
Case study example




<<Request>> port, typed by a Conjugate Service Interface

<<Service>> port, typed by a Service Interface

Participants are refined



# Specify service orchestrations [optional]

- **Orchestration of services:**
  - Refinement of the BPMN processes from the BAM
  - Seen from the perspective of **one specific participant architecture**.
- **Orchestration is specified with activity or BPMN diagram:**
  - Must be contained in the corresponding participant architecture.
  - Each activity represents a call to an operation of an interface.
  - Model the control flow between the activities.


# Thanks for your attention