INF5120
”Modellbasert Systemutvikling”
”Modelbased System development”

Lecture: 27.03.2017

Arne-Jørgen Berre

arneb@ifi.uio.no or Arne.J.Berre@sintef.no

Content

- ThingML – part 2
- Service Modeling
- SoaML introduction
- UML 2.0 Collaboration models
- SoaML – Service Architecture
- UML 2.0 Composite models
- SoaML – Port/connector models

Course parts (16 lectures) - 2017

- January (1-3) (Introduction to Modeling, Business Architecture and the Smart Building project):
- 1-16/1: Introduction to INF5120
- 2-23/1: Modeling structure and behaviour (UML and UML 2.0 and metamodeling) - (establish Oblig groups)
- 3-30/1: WebRatio for Web Apps/Portals and Mobile Apps – and Entity/Class modeling – (Getting started with WebRatio)

- February (4-7) (Modeling of User Interfaces, Flows and Data model diagrams, Apps/Web Portals - IFML/Client-Side):
- 4-6/2: Business Model Canvas, Value Proposition, Lean Canvas and Essence
- 5-13/2: IFML – Interaction Flow Modeling Language, WebRatio advanced – for Web and Apps
- 6-20/2: BPMN process, UML Activ.Diagrams, Workflow and Orchestration modelling value networks
- 7-27/2: Modeling principles – Quality in Models
- 27/2: Oblig 1: Smart Building – Business Architecture and App/Portal with IFML WebRatio UI for Smart Building

- March (8-11) (Modeling of IoT/CPS/Cloud, Services and Big Data – UML SM/SD/Collab, ThingML Server-Side):
- 8-6/3: Basis for DSL and ThingML -> UML State Machines and Sequence Diagrams
- 9-13/3: ThingML DSL - UML Composite structures, State Machines and Sequence Diagrams II
- 10-20/3: Guest lecture, "Experience with Modelling", Anton Landmark, SINTEF
- 11-27/3: ThingML part 2 and UML Service Modeling, Architectural models, SoaML. Role modeling and UML Collaboration diagrams

- April/May (12-14) (MDE – Creating Your own Domain Specific Language):
- 12-3/4: Model driven engineering – Metamodels, DSL, UML Profiles, EMF, Sirius Editors – intro to Oblig 3

- EASTER – 10/4 og 17/4
- 20/4: Oblig 2: Smart Building – Individual and group delivery - Internet of Things control with ThingML – Raspberry Pi, Wireless sensors (temperature, humidity), actuators (power control)

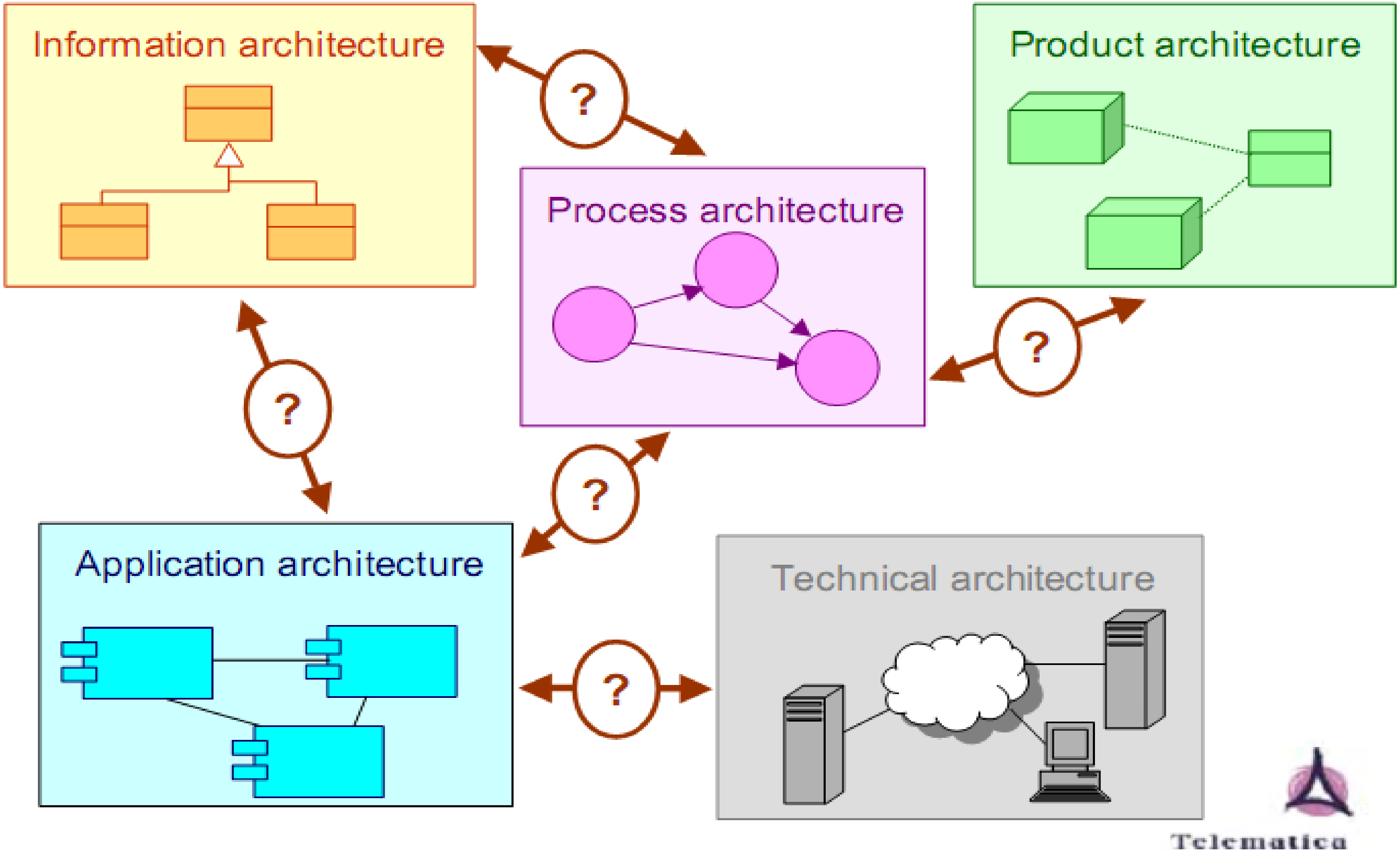
- 13-24/4: MDE transformations, Non Functional requirements – Discussion of Oblig2 and 3
- 1. Mai – Official holiday
- 4/5: Oblig 3 - Your own Domain Specific Language – (ArchiMate) (Delivery – Thursday May 4th)
- 14-8/5: SmartBuilding – Integrating App with Server side and Architmate editor (Discussion of Oblig 3)

- May (15-17): (Bringing it together)
- 15-15/5: Summary of the course – Final demonstrations
- 16-22/5: Previous exams – group collaborations (No lecture)
- 17-29/5: Conclusions, Preparations for the Exam by old exams
- June (Exam)
- 13/6: Exam (4 hours), June 13th, 0900-1300

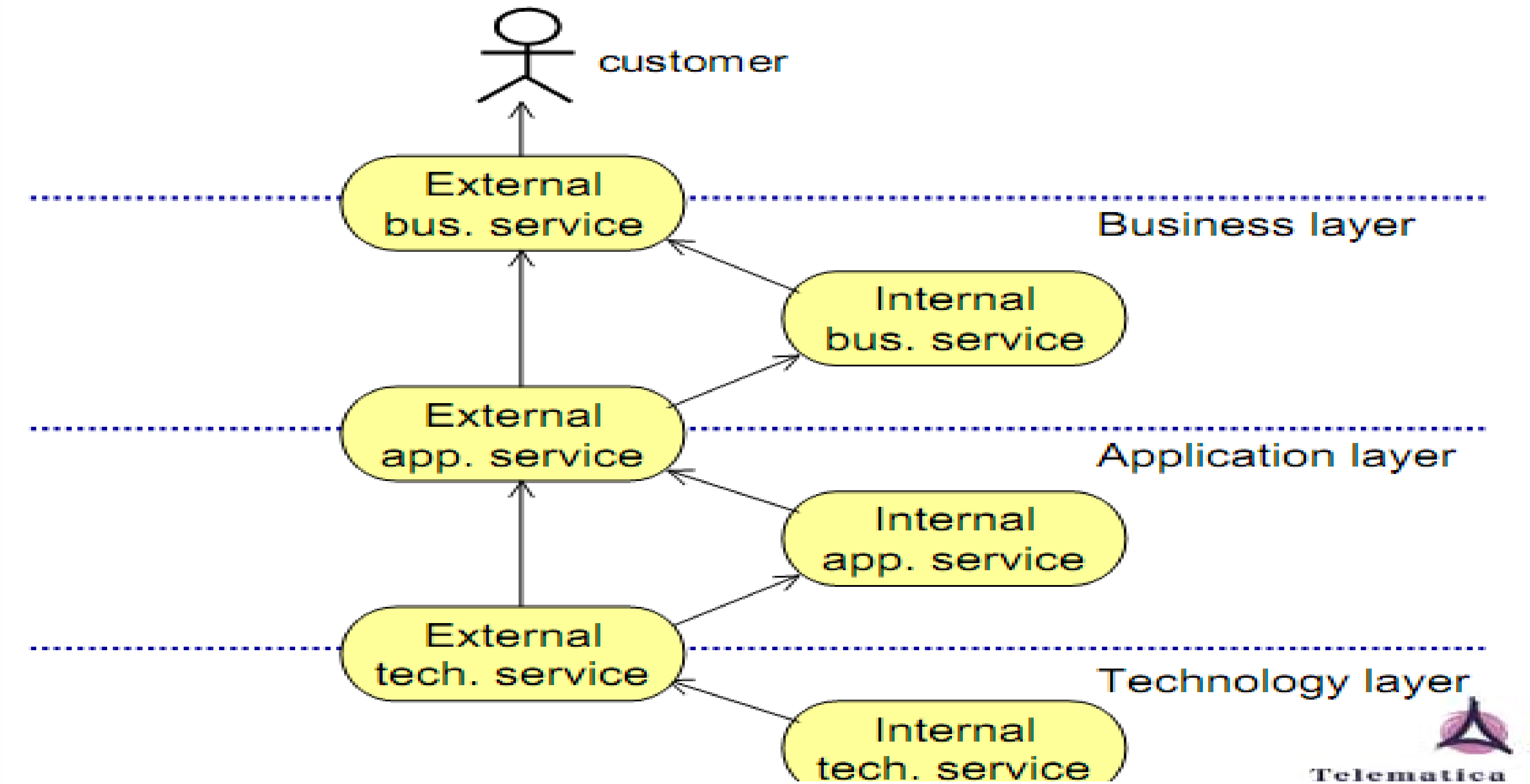
ArchiMate

Authors : eSchoolink Group - ITNLU

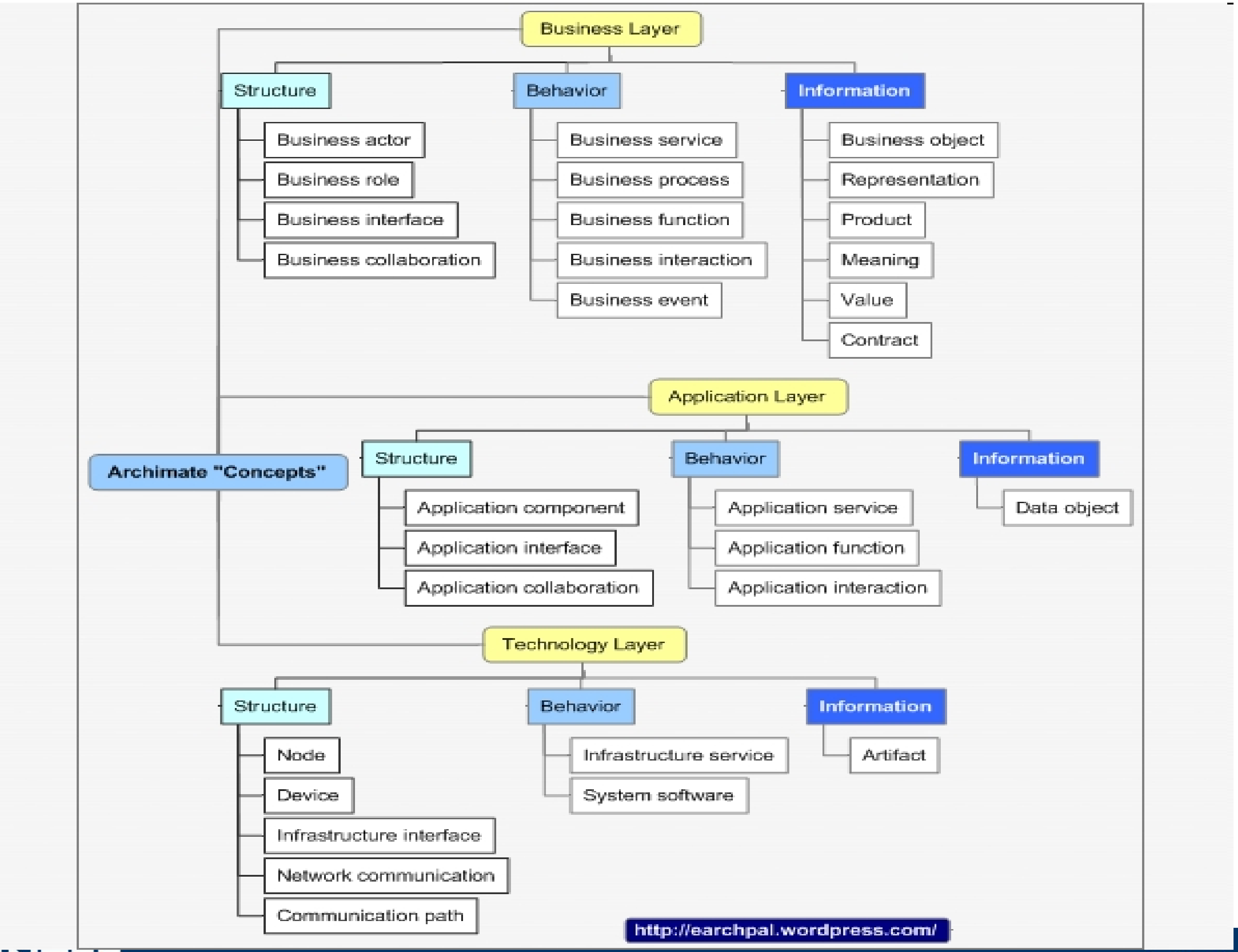
Enterprise Architecture: Describing *Coherence*



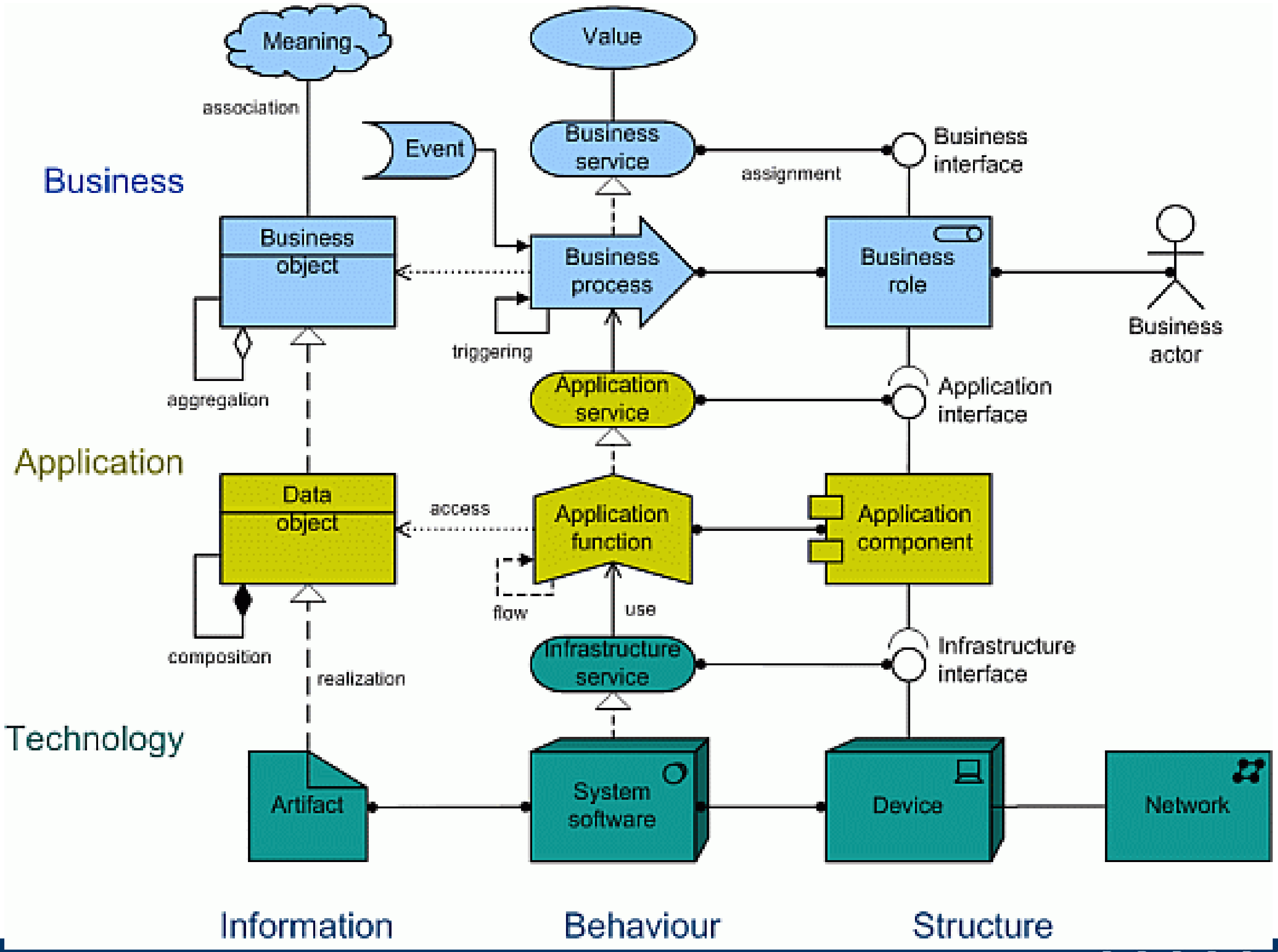
Layers

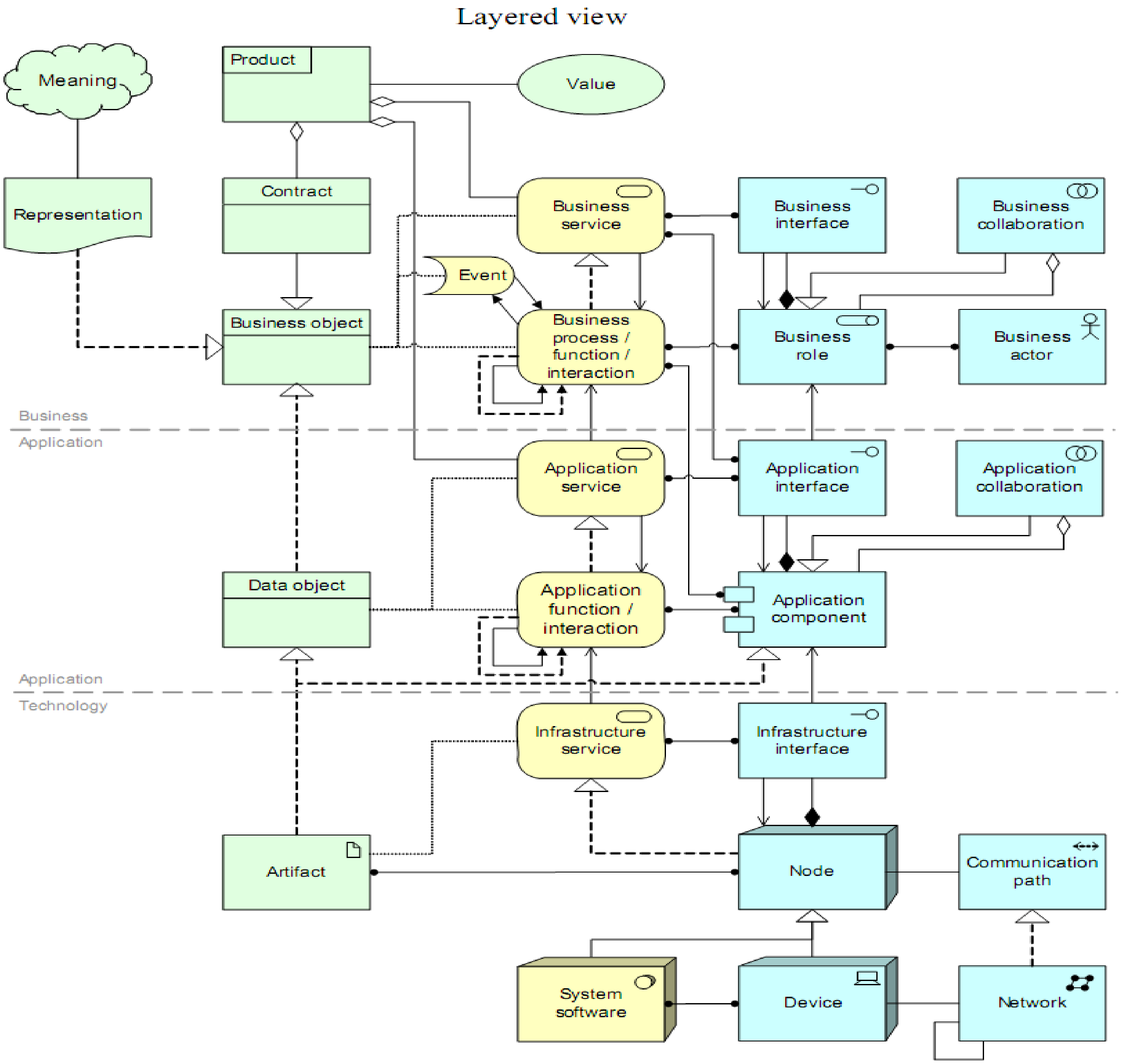

customer
External bus. service
Internal bus. service
Business layer
External app. service
Internal app. service
Application layer
External tech. service
Internal tech. service
Technology layer
Telematica

Layers , domains



Layers , domains





Overview of the ArchiMate concepts and main relationships

ArchiMate vs UML

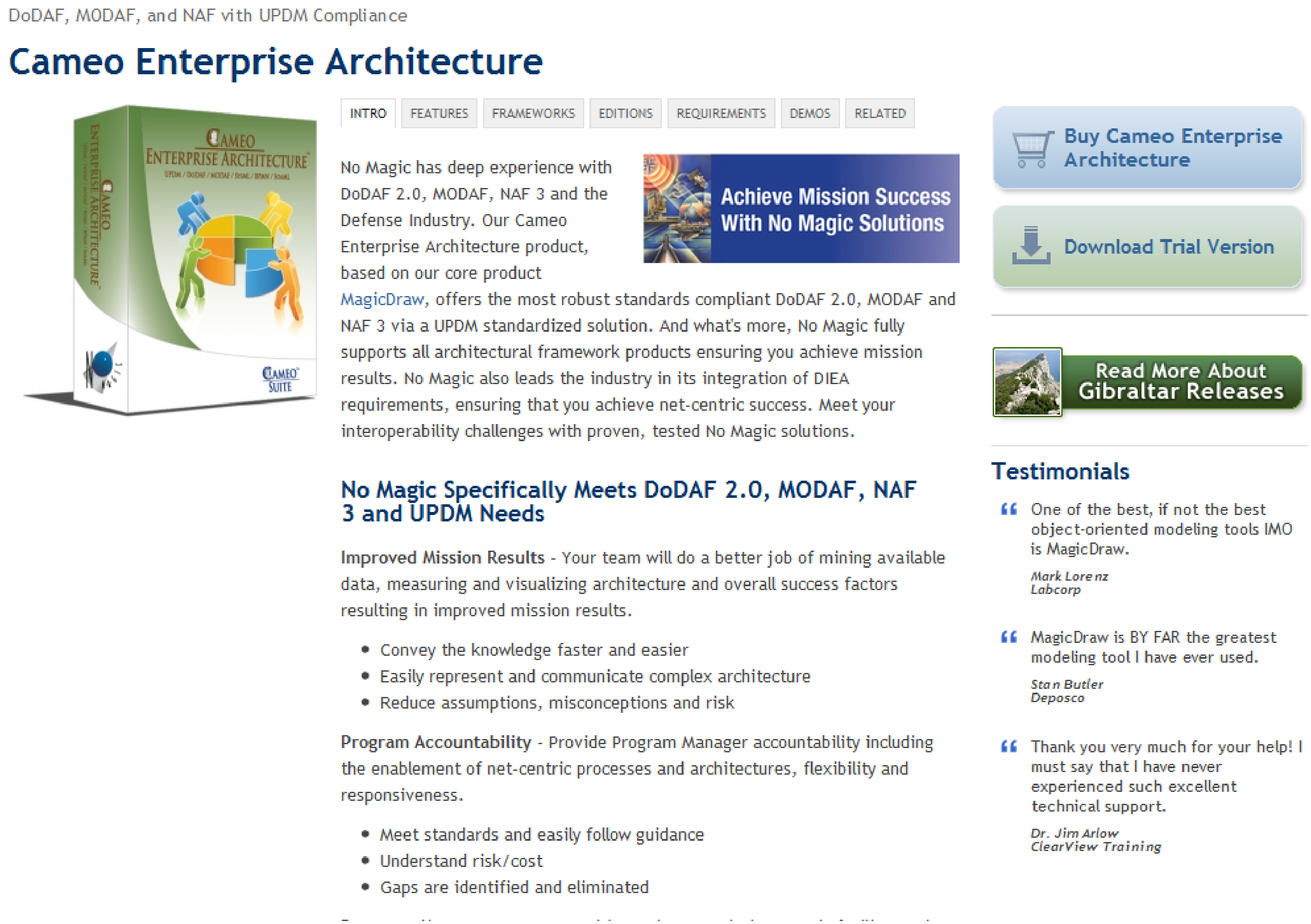
ArchiMate

- ArchiMate was created to model the architecture of an enterprise (all of the systems in an organization).
- ArchiMate models the business, information system (application and data), and technology architectures of the environment, including how these architectures are inter-related.

UML

- **UML still functions best as a way to document the architecture of a single system**
- **UML provides 13 diagram types, providing flexibility to describe many different types of systems.**

MagicDraw
Cameo Enterprise Architecture



DoDAF, MODAF, and NAF vith UPDM Compliance

Cameo Enterprise Architecture

INTRO
FEATURES
FRAMEWORKS
EDITIONS
REQUIREMENTS
DEMOS
RELATED
CAMEO ENTERPRISE ARCHITECTURE
CAMEO SUITE
Achieve Mission Success With No Magic Solutions
Buy Cameo Enterprise Architecture
Download Trial Version
Read More About Gibraltar Releases

No Magic has deep experience with DoDAF 2.0, MODAF, NAF 3 and the Defense Industry. Our Cameo Enterprise Architecture product, based on our core product MagicDraw, offers the most robust standards compliant DoDAF 2.0, MODAF and NAF 3 via a UPDM standardized solution. And what's more, No Magic fully supports all architectural framework products ensuring you achieve mission results. No Magic also leads the industry in its integration of DIEA requirements, ensuring that you achieve net-centric success. Meet your interoperability challenges with proven, tested No Magic solutions.

No Magic Specifically Meets DoDAF 2.0, MODAF, NAF 3 and UPDM Needs

Improved Mission Results - Your team will do a better job of mining available data, measuring and visualizing architecture and overall success factors resulting in improved mission results.

- Convey the knowledge faster and easier
- Easily represent and communicate complex architecture
- Reduce assumptions, misconceptions and risk

Program Accountability - Provide Program Manager accountability including the enablement of net-centric processes and architectures, flexibility and responsiveness.

- Meet standards and easily follow guidance
- Understand risk/cost
- Gaps are identified and eliminated

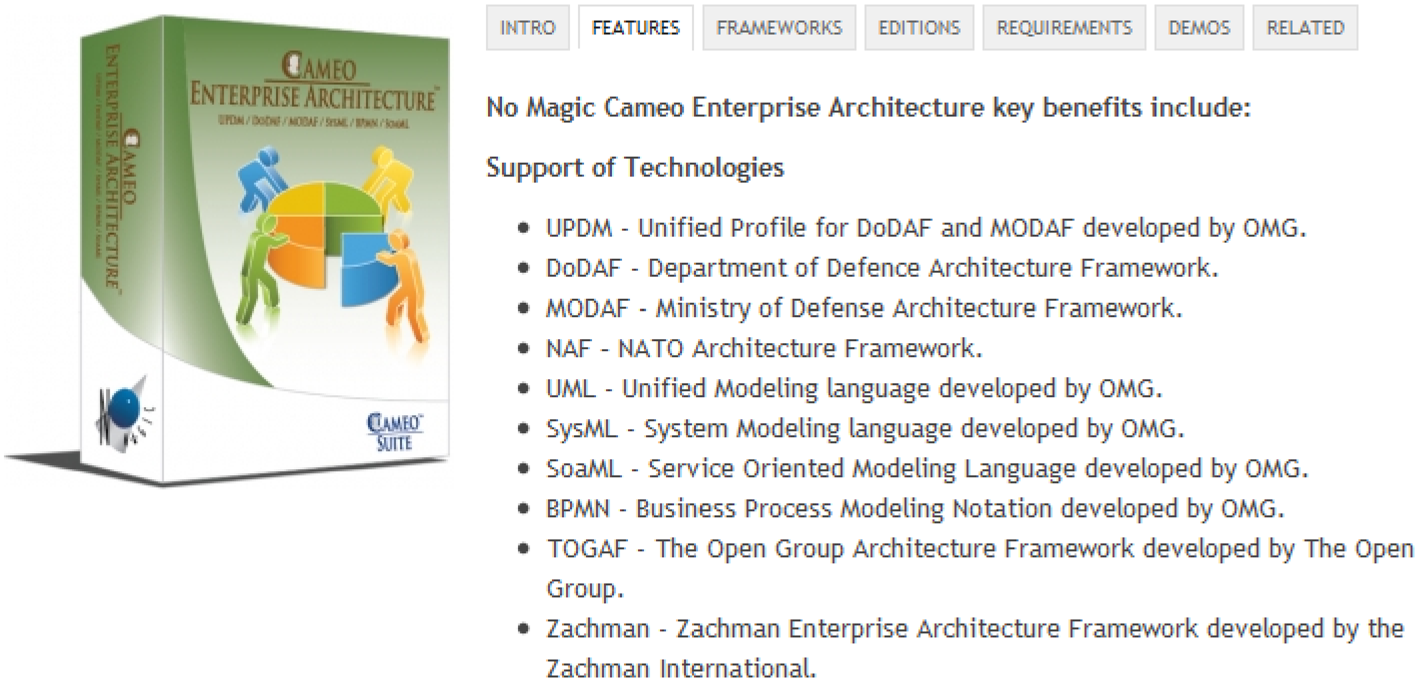
Testimonials

> One of the best, if not the best object-oriented modeling tools IMO is MagicDraw.
>
> *Mark Lorenz*
> *Labcorp*

> MagicDraw is BY FAR the greatest modeling tool I have ever used.
>
> *Stan Butler*
> *Deposco*

> Thank you very much for your help! I must say that I have never experienced such excellent technical support.
>
> *Dr. Jim Arlow*
> *ClearView Training*

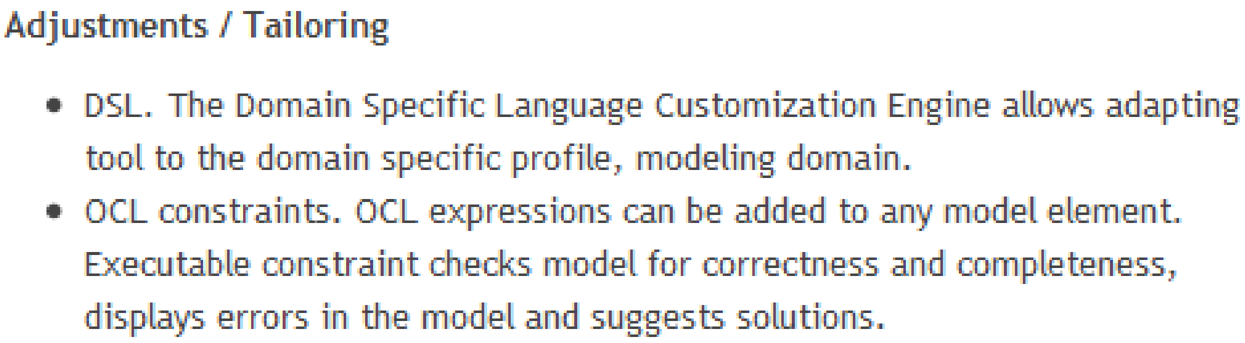
Cameo Enterprise Architecture



No Magic Cameo Enterprise Architecture key benefits include:

Support of Technologies

- UPDM - Unified Profile for DoDAF and MODAF developed by OMG.
- DoDAF - Department of Defence Architecture Framework.
- MODAF - Ministry of Defense Architecture Framework.
- NAF - NATO Architecture Framework.
- UML - Unified Modeling language developed by OMG.
- SysML - System Modeling language developed by OMG.
- SoaML - Service Oriented Modeling Language developed by OMG.
- BPMN - Business Process Modeling Notation developed by OMG.
- TOGAF - The Open Group Architecture Framework developed by The Open Group.
- Zachman - Zachman Enterprise Architecture Framework developed by the Zachman International.



Adjustments / Tailoring

- DSL. The Domain Specific Language Customization Engine allows adapting tool to the domain specific profile, modeling domain.
- OCL constraints. OCL expressions can be added to any model element. Executable constraint checks model for correctness and completeness, displays errors in the model and suggests solutions.

Service Modeling and Service Design

- UML 2.0 Components with Ports, SoaML (and SysML)

- Service views in UPDM, (DODAF/MODAF/NAF)
- SESAR ISRM connected to AIRM
- GRA UML connected to NIEM
- ISO 19119 connected to ISO 19103, and RM/ODP

SoaML Introduction

See SoaML standard document on course web page / Dropbox

SoaML history

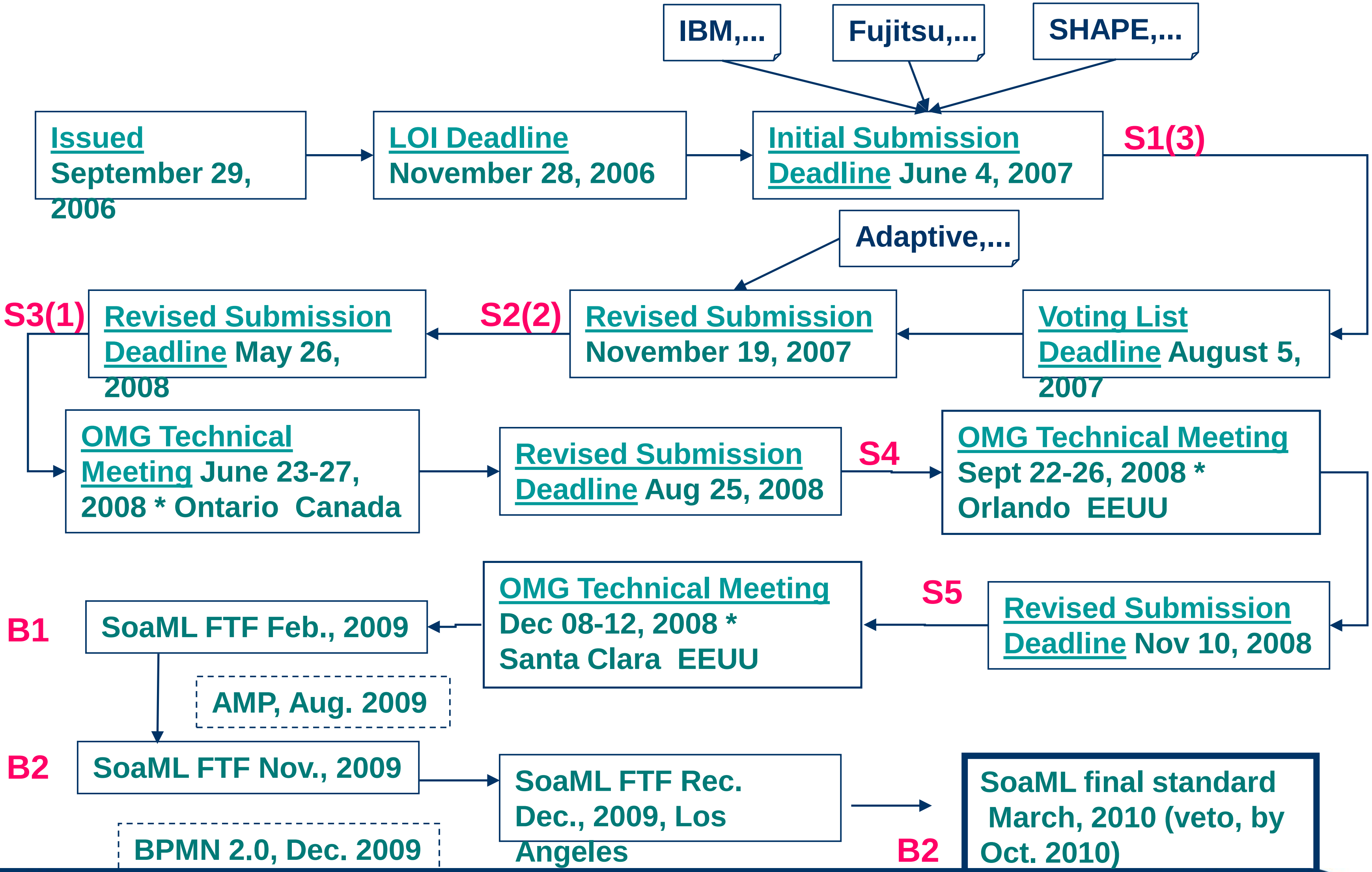
2006, September	OMG RFP
2007, June	3 initial submissions
2008 & 2009	Merge process
2009, December	SoaML 1.0 finished
2010, March	SoaML 1.0 adopted by OMG
2011, December	SoaML 1.0 formal standard by OMG

- FTF chairs: Arne J. Berre, SINTEF and Jim Amsden, IBM

- http://www.soaml.org

UPMS SoaML Timeline

Sx – Submission version x
Bx – Beta version x



IBM,...

Fujitsu,...

SHAPE,...

Issued September 29, 2006

LOI Deadline November 28, 2006

Initial Submission Deadline June 4, 2007

S1(3)

Voting List Deadline August 5, 2007

Adaptive,...

Revised Submission November 19, 2007

S2(2)

Revised Submission Deadline May 26, 2008

S3(1)

OMG Technical Meeting June 23-27, 2008 * Ontario Canada

Revised Submission Deadline Aug 25, 2008

S4

OMG Technical Meeting Sept 22-26, 2008 * Orlando EEUU

Revised Submission Deadline Nov 10, 2008

S5

OMG Technical Meeting Dec 08-12, 2008 * Santa Clara EEUU

B1

SoaML FTF Feb., 2009

AMP, Aug. 2009

B2

SoaML FTF Nov., 2009

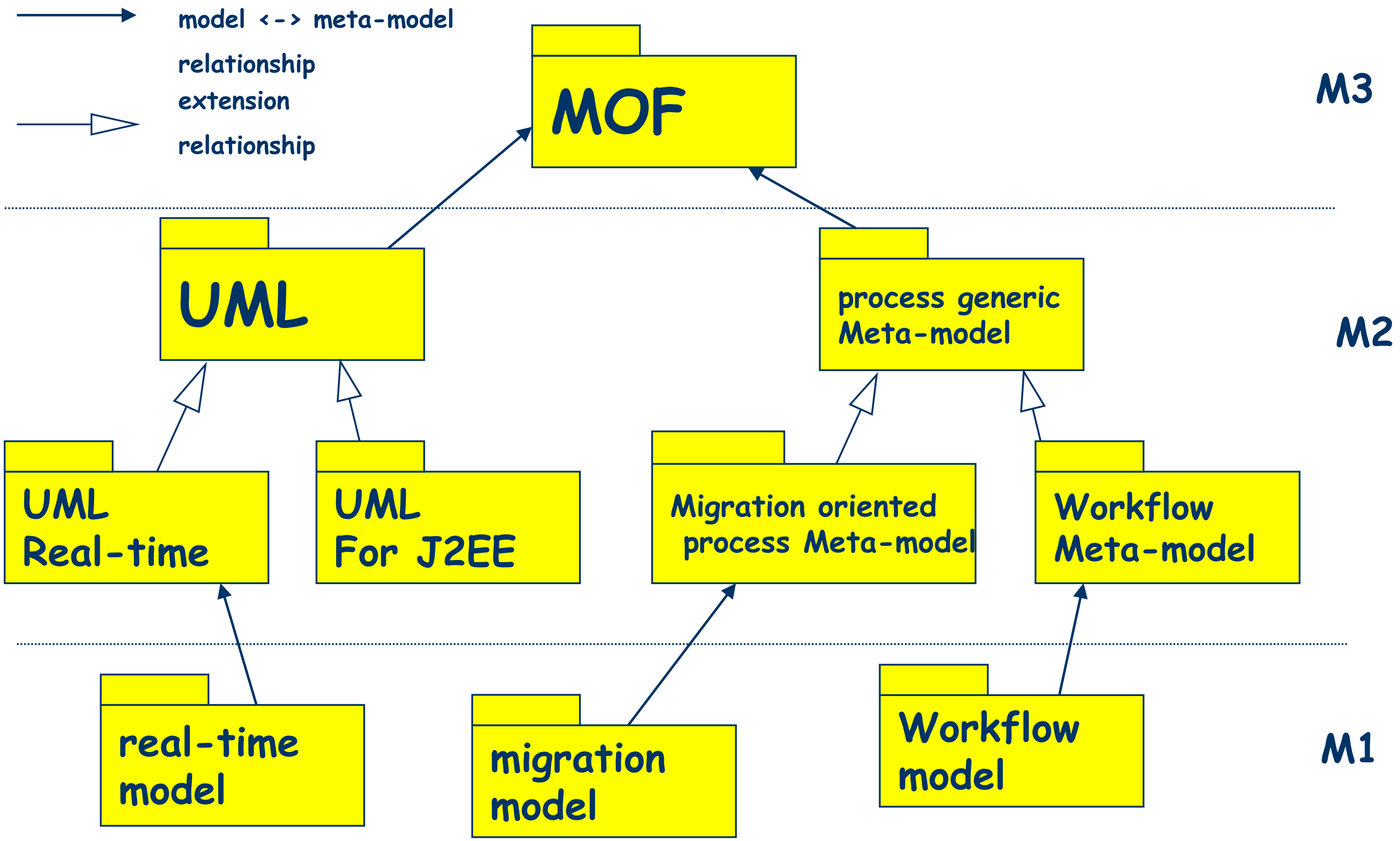
BPMN 2.0, Dec. 2009

SoaML FTF Rec. Dec., 2009, Los Angeles

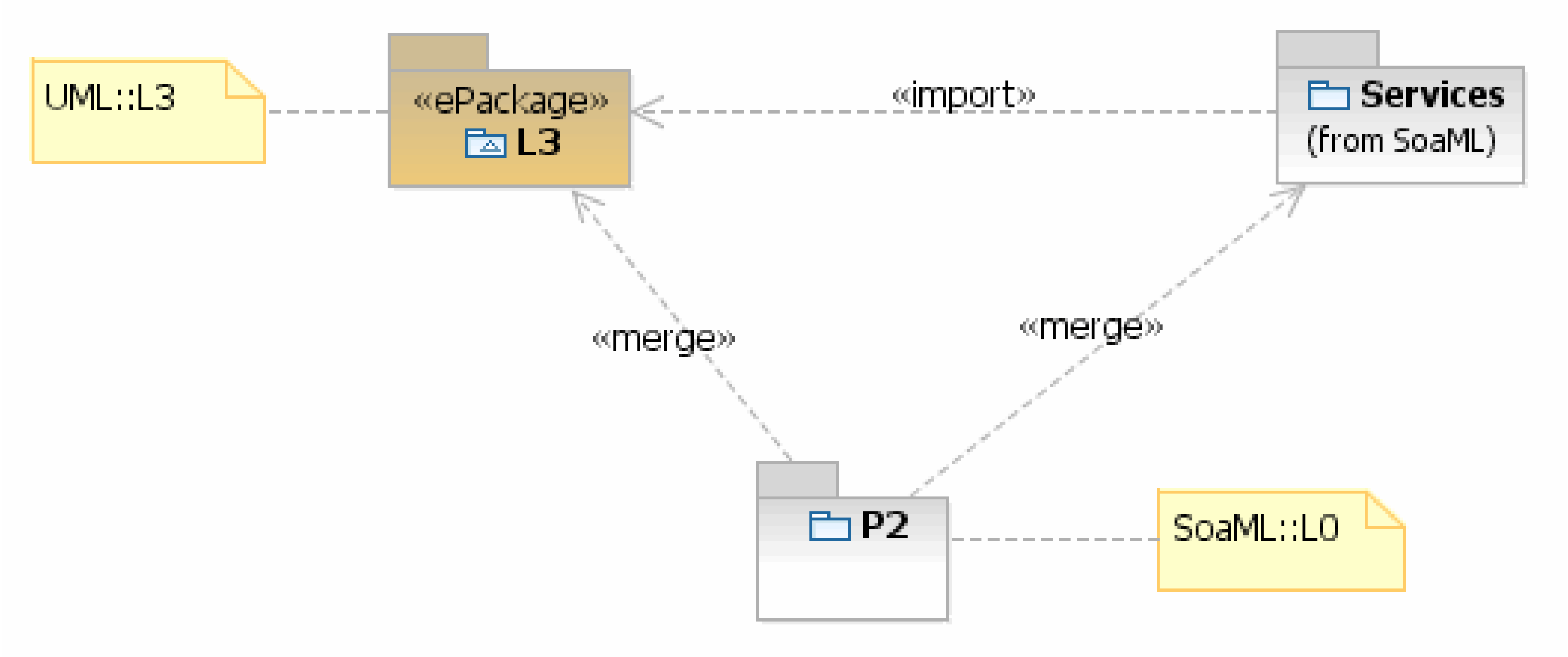
B2

SoaML final standard March, 2010 (veto, by Oct. 2010)

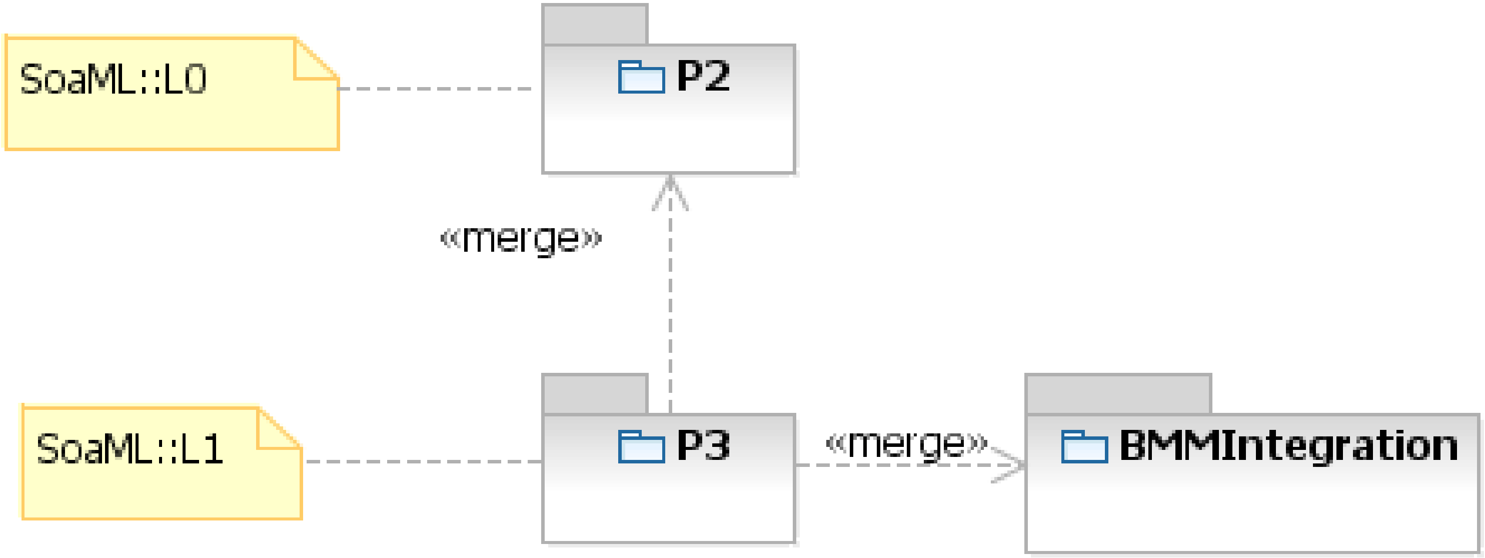
Metamodels and profiles



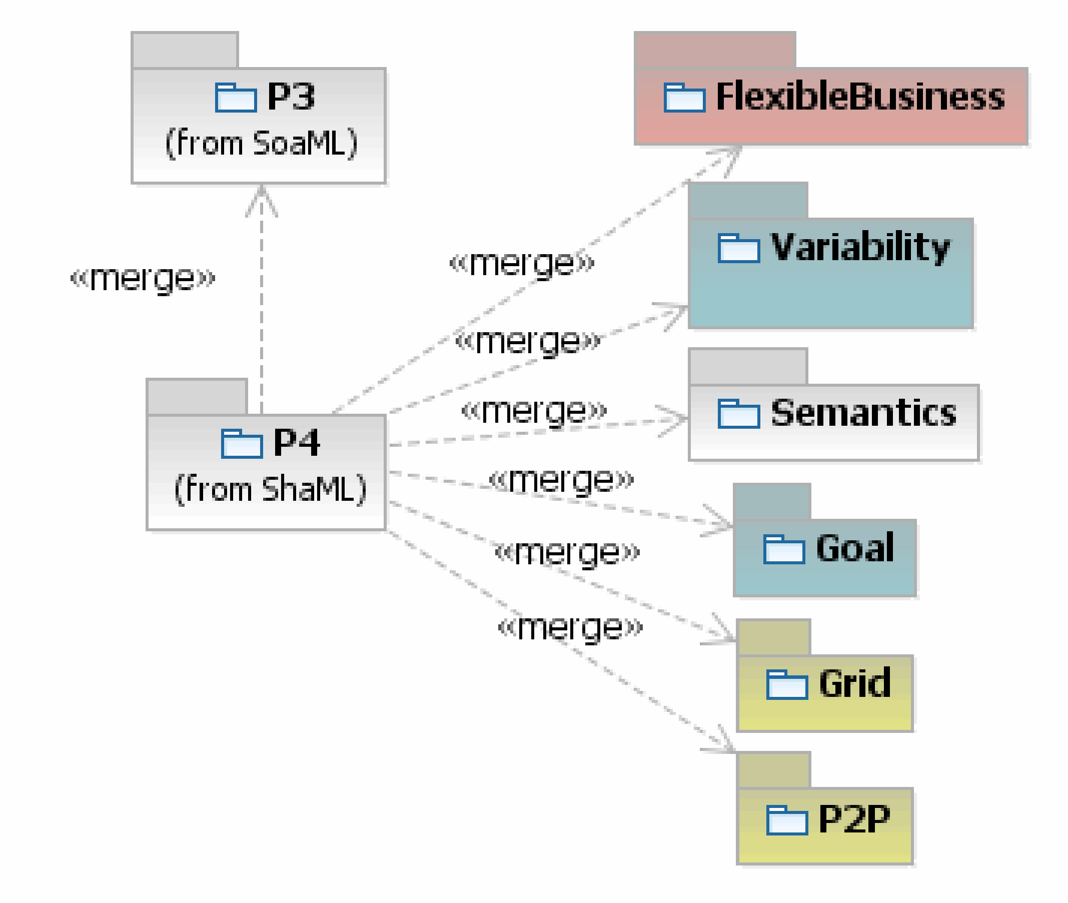
UML/SoaML Metamodel approach – P2



UML/SoaML Metamodel approach – P3



SoaML/ShaML Metamodel approach –P4



SoaML references

- OMG Web site
  - SoaML Wiki: http://www.SoaML.org
  - Specification: http://www.omgwiki.org/SoaML/doku.php?id=specification

UML tools with SoaML

- MagicDraw, NoMagic
- Enterprise Architect, Sparq
- Modelio, Softeam
- RSA/RSM, IBM
- …

SoaML – Goals

- **Intuitive and complete** support for modelling services in UML
- Support for **bi-directional asynchronous services** between multiple parties
- Support for **Services Architectures** where parties provide and use multiple services.
- Support for **services defined to contain other services**
- Easily mapped to and made **part of a business process specification**
- **Compatibility with UML, BPDM and BPMN** for business processes
- Direct mapping to web services
- **Top-down, bottom up or meet-in-the-middle modelling**
- **Design by contract** or **dynamic adaptation** of services
- To specify and relate the **service capability and its contract**
- **No changes to UML**

SoaML – Scope

- Extensions to UML2.1 to support the following new modeling capabilities:
  - 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 2.0.

- SoaML focuses on the basic service modelling concepts
  - A foundation for further extensions both related to integration with other OMG metamodels like BPMN 2.0, SBVR, OSM, ODM and others.

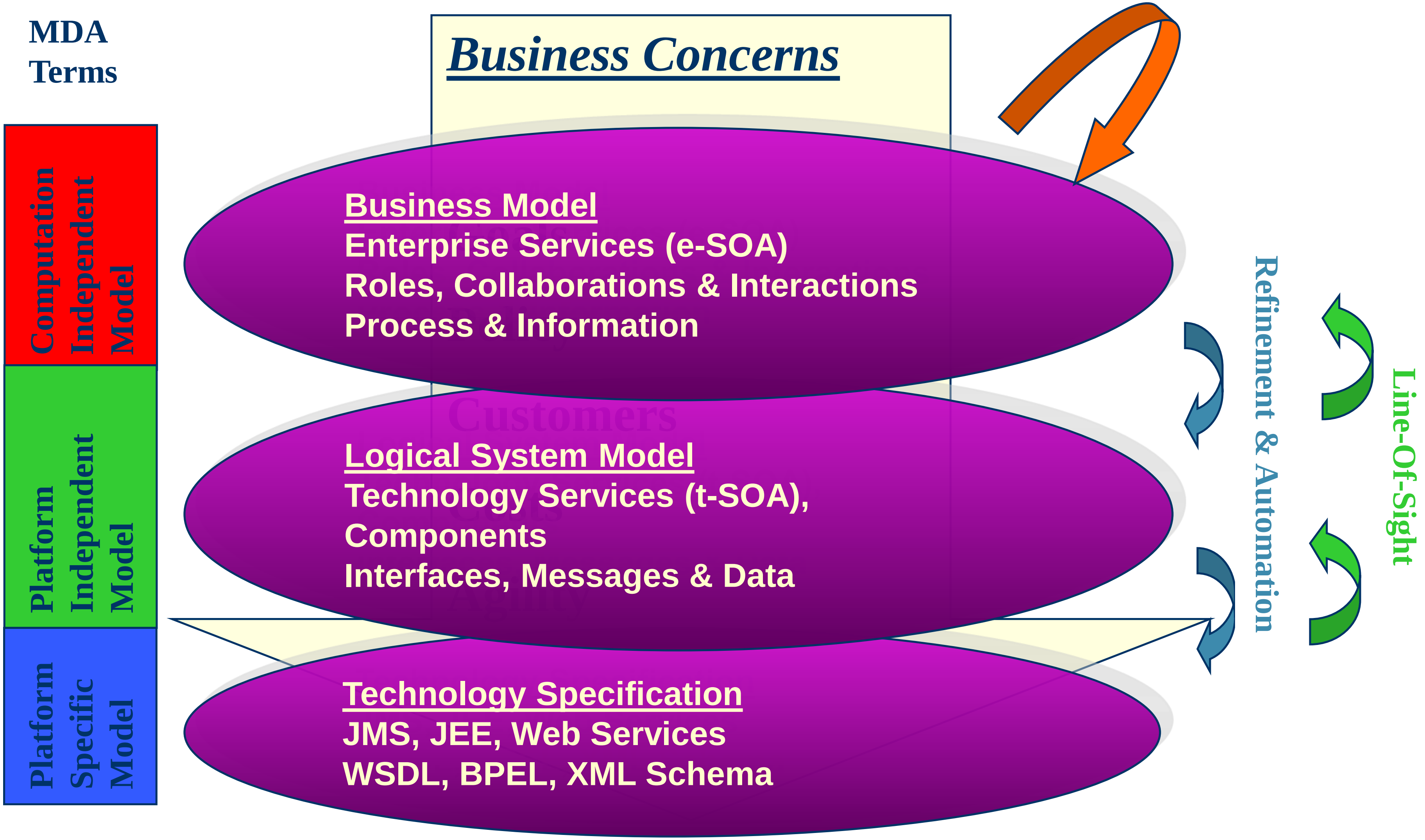
- SoaML is NOT a methodology

Definition of service in SoaML

- ***”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, then, 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).

SOA in Model Driven Architecture (MDA)



MDA Terms

Computation Independent Model

Platform Independent Model

Platform Specific Model

Business Concerns

Business Model
Enterprise Services (e-SOA)
Roles, Collaborations & Interactions
Process & Information

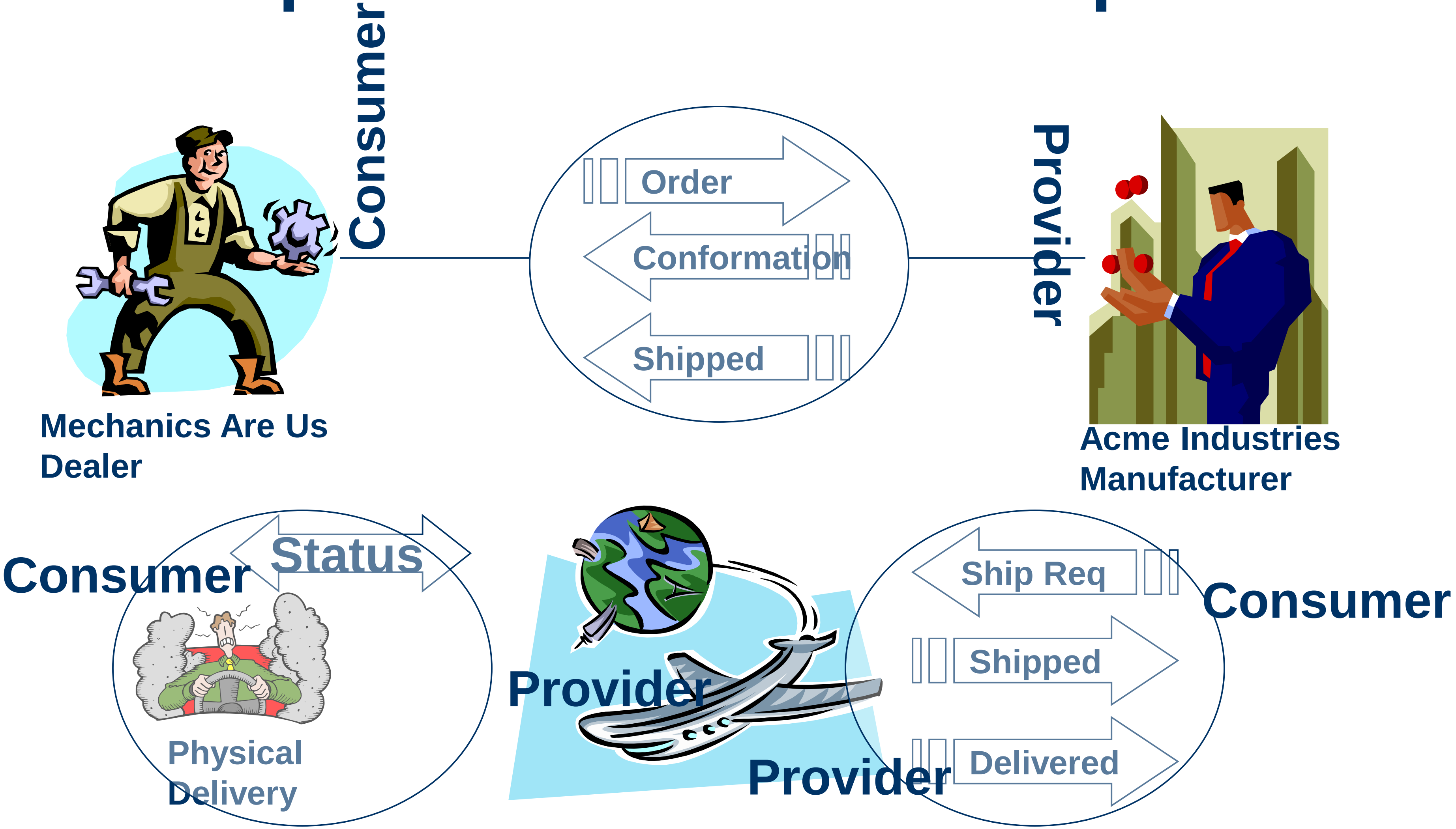
Logical System Model
Technology Services (t-SOA),
Components
Interfaces, Messages & Data

Technology Specification
JMS, JEE, Web Services
WSDL, BPEL, XML Schema

SoaML – Key concepts

- Services architecture – specification of community
  - Participants – role
  - Service contracts – collaboration (provide and consume)
- Service contract – specification of service
  - Role – Provider and consumer
  - Interfaces
  - Choreography (protocol, behaviour)
- Service interface – bi-directional service
- Simple interface – one-directional service
- Message Type – data exchanged between services

Marketplace Services – Example



Consumer

Order

Conformation

Shipped

Provider

Mechanics Are Us
Dealer

Acme Industries
Manufacturer

Consumer

Status

Physical
Delivery

Provider

Provider

Ship Req

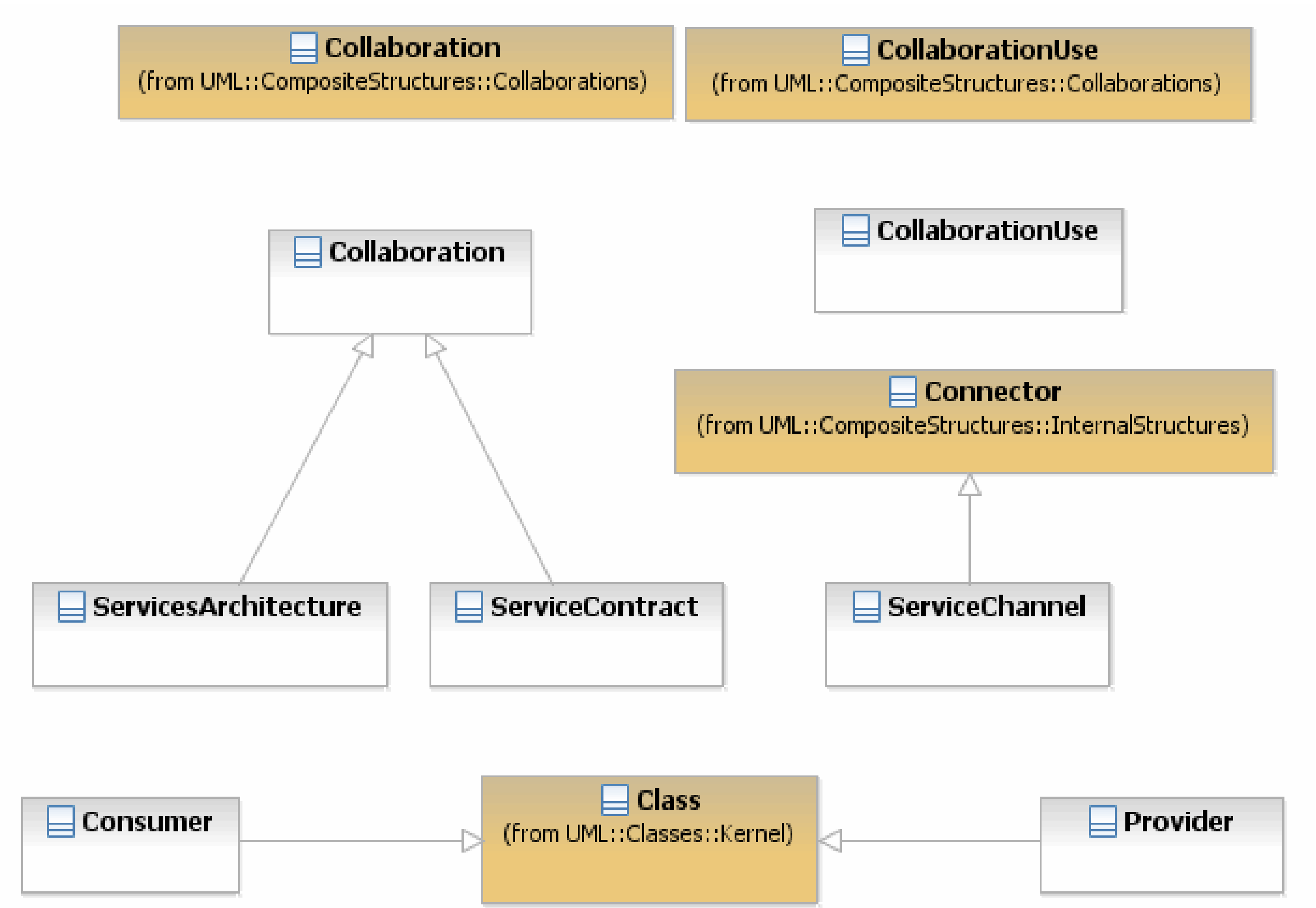
Shipped

Delivered

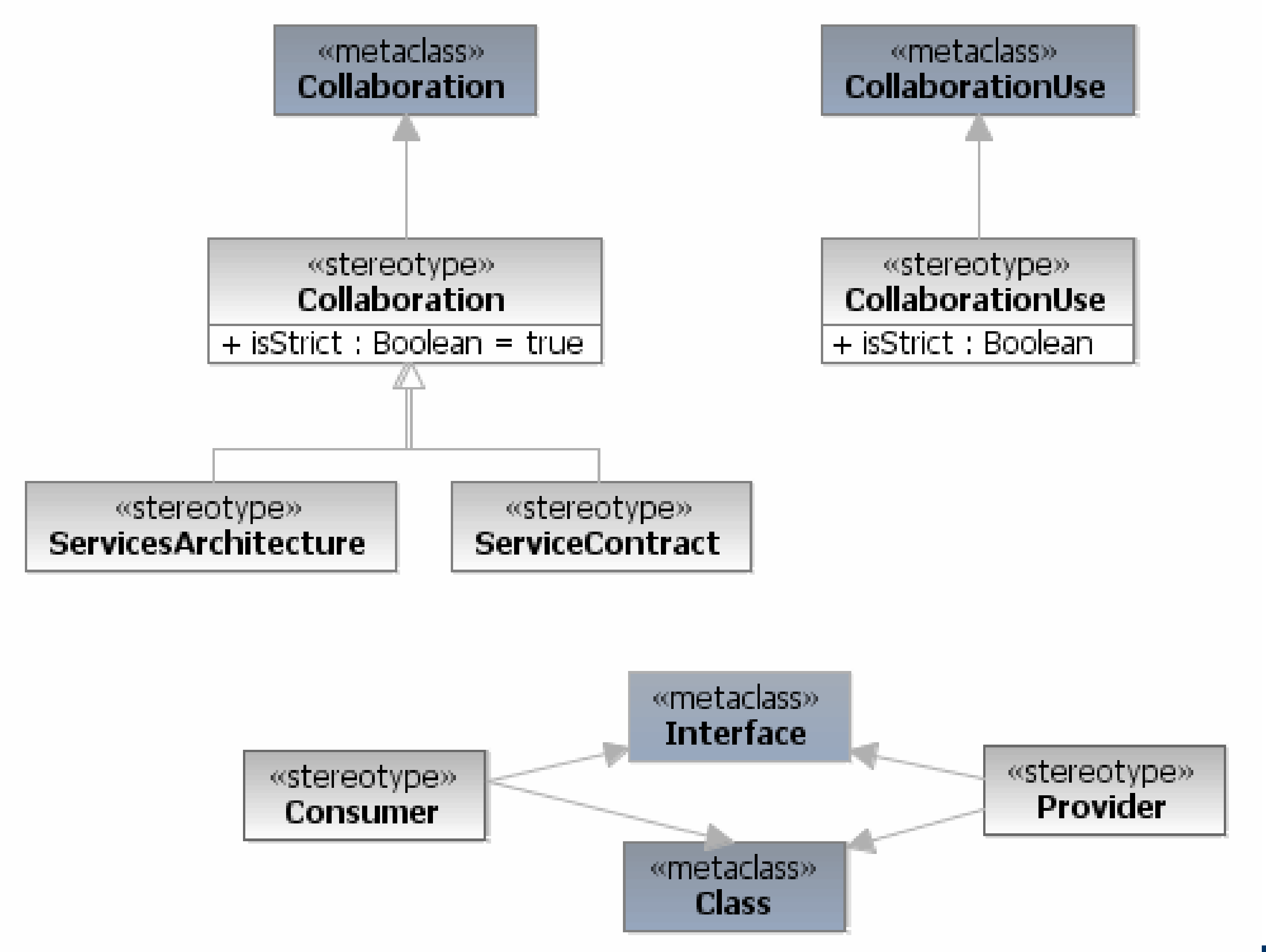
Consumer

GetItThere Freight Shipper

ServiceContracts and ServiceArchitectures Metamodel

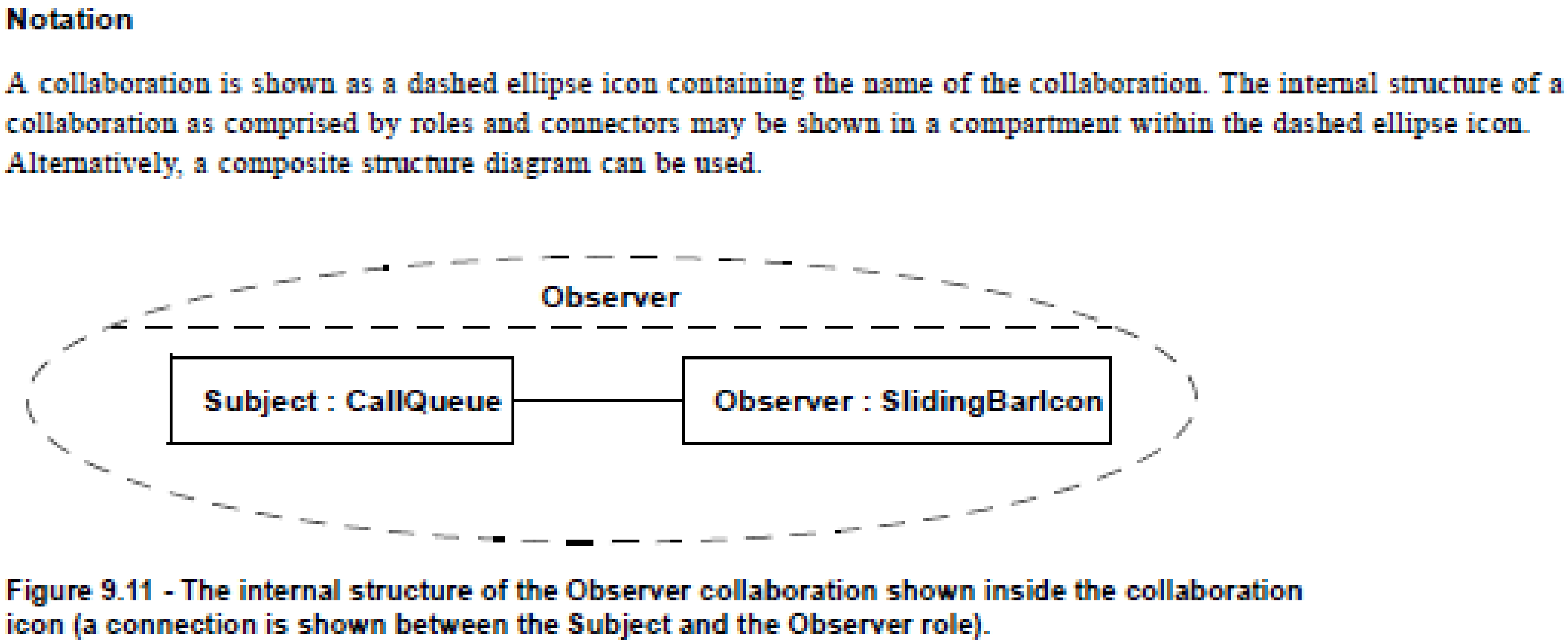

Collaboration
(from UML::CompositeStructures::Collaborations)
CollaborationUse
(from UML::CompositeStructures::Collaborations)
Collaboration
CollaborationUse
Connector
(from UML::CompositeStructures::InternalStructures)
ServicesArchitecture
ServiceContract
ServiceChannel
Consumer
Class
(from UML::Classes::Kernel)
Provider

ServiceContracts and ServiceArchitectures Profile



UML 2.0 Collaboration diagrams and SoaML

Collaboration

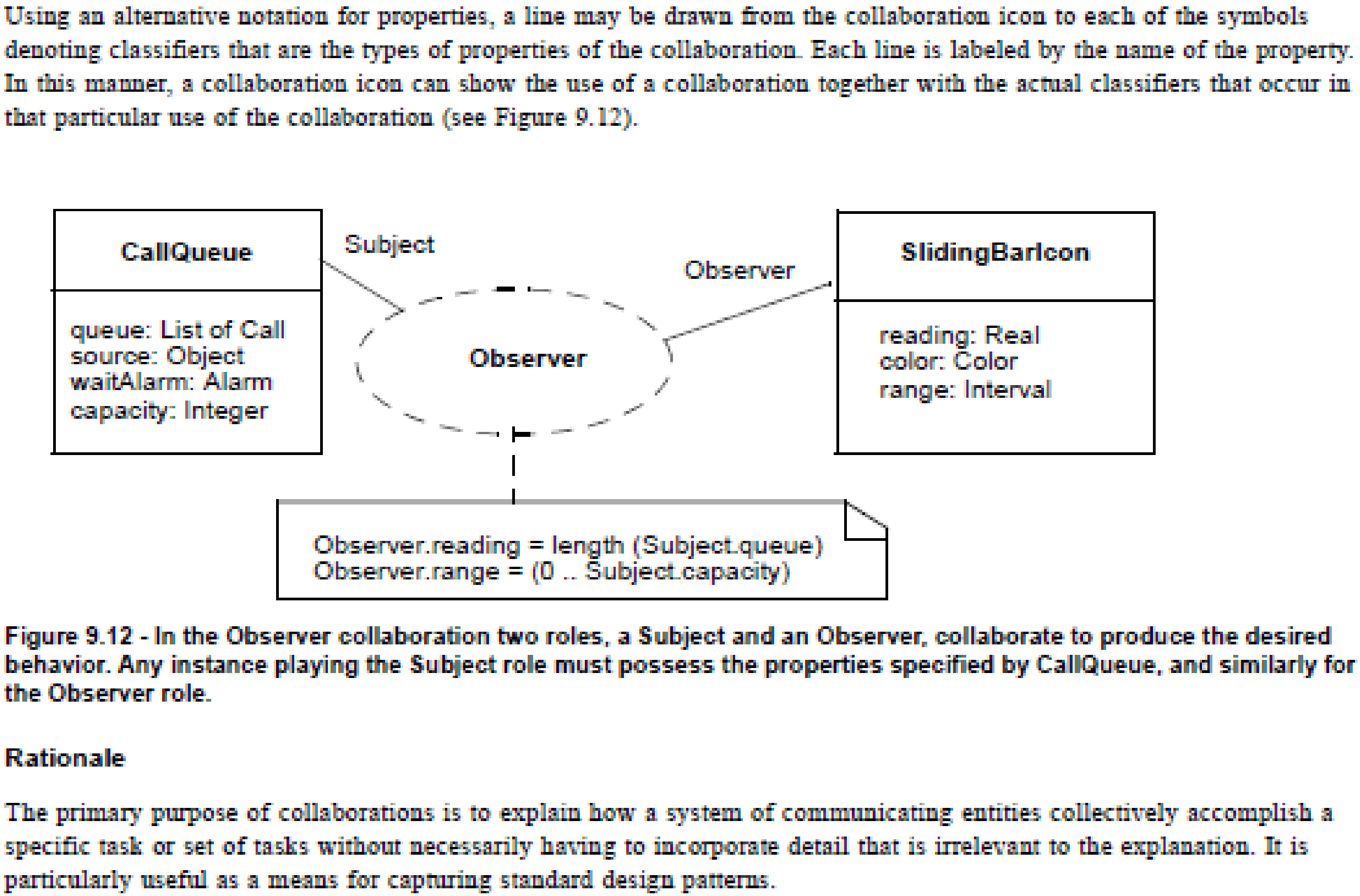


Notation

A collaboration is shown as a dashed ellipse icon containing the name of the collaboration. The internal structure of a collaboration as comprised by roles and connectors may be shown in a compartment within the dashed ellipse icon. Alternatively, a composite structure diagram can be used.

Figure 9.11 - The internal structure of the Observer collaboration shown inside the collaboration icon (a connection is shown between the Subject and the Observer role).

Collaboration



Using an alternative notation for properties, a line may be drawn from the collaboration icon to each of the symbols denoting classifiers that are the types of properties of the collaboration. Each line is labeled by the name of the property. In this manner, a collaboration icon can show the use of a collaboration together with the actual classifiers that occur in that particular use of the collaboration (see Figure 9.12).

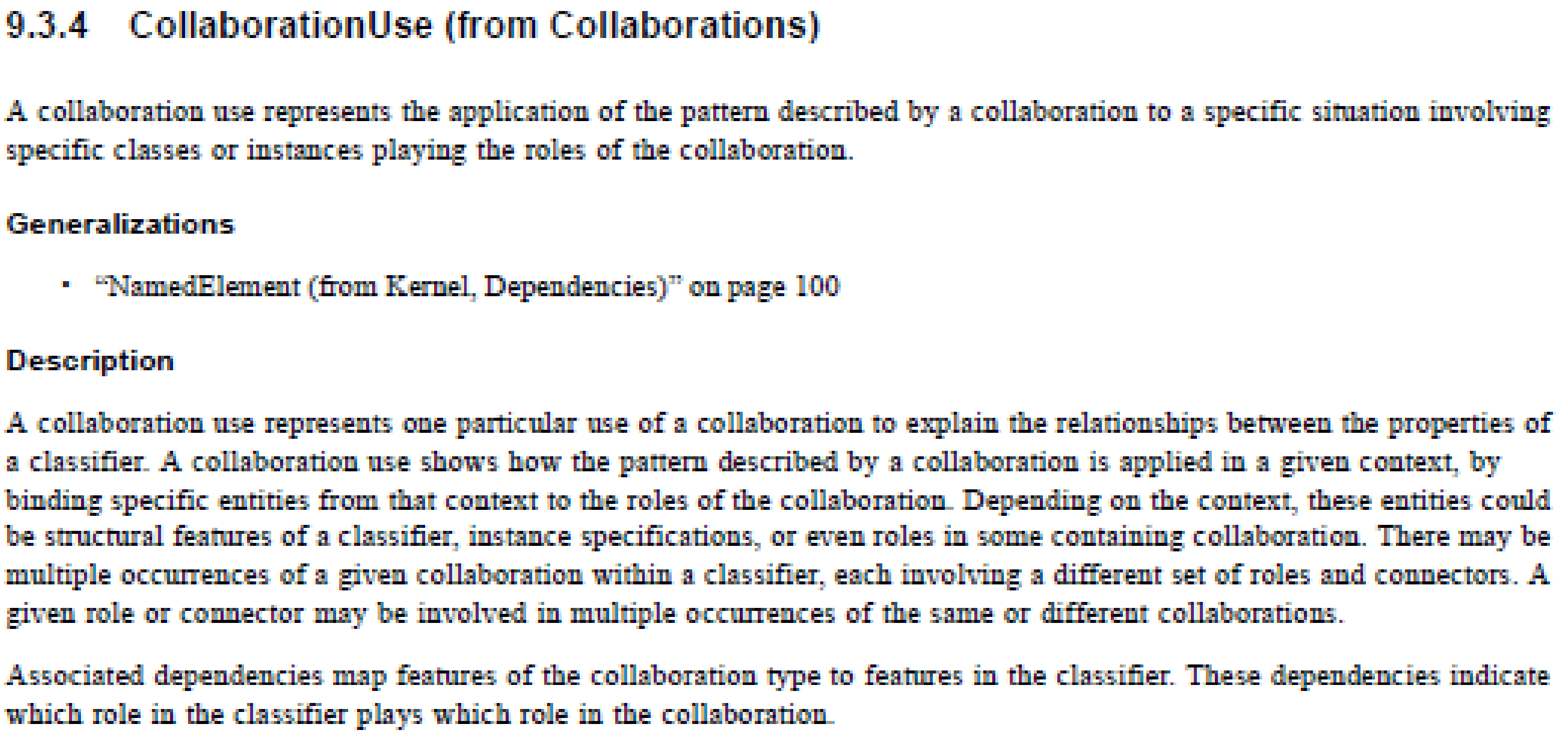
Observer.reading = length (Subject.queue)
Observer.range = (0 .. Subject.capacity)

Figure 9.12 - In the Observer collaboration two roles, a Subject and an Observer, collaborate to produce the desired behavior. Any instance playing the Subject role must possess the properties specified by CallQueue, and similarly for the Observer role.

Rationale

The primary purpose of collaborations is to explain how a system of communicating entities collectively accomplish a specific task or set of tasks without necessarily having to incorporate detail that is irrelevant to the explanation. It is particularly useful as a means for capturing standard design patterns.

CollaborationUse



9.3.4 CollaborationUse (from Collaborations)

A collaboration use represents the application of the pattern described by a collaboration to a specific situation involving specific classes or instances playing the roles of the collaboration.

Generalizations

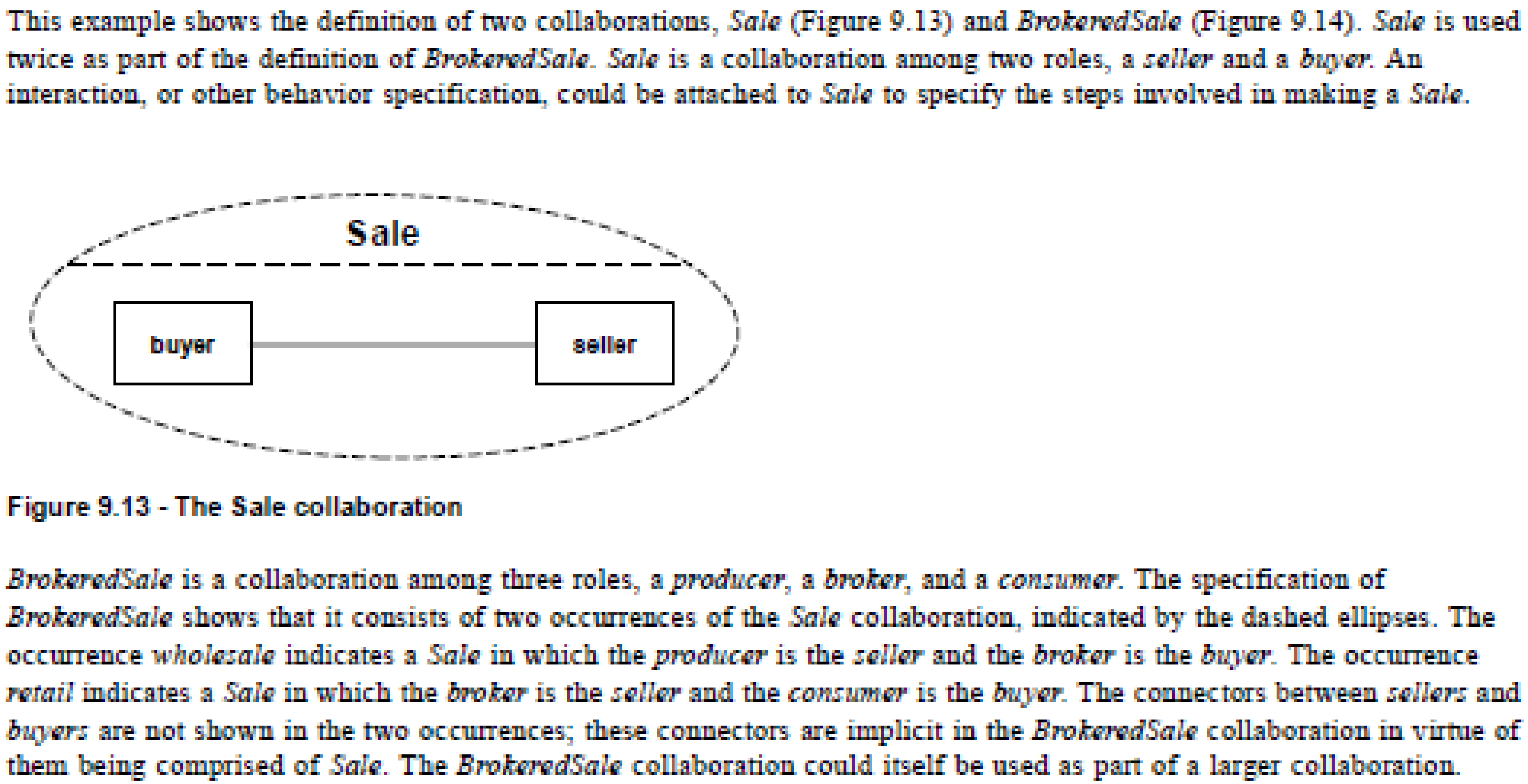
- "NamedElement (from Kernel, Dependencies)" on page 100

Description

A collaboration use represents one particular use of a collaboration to explain the relationships between the properties of a classifier. A collaboration use shows how the pattern described by a collaboration is applied in a given context, by binding specific entities from that context to the roles of the collaboration. Depending on the context, these entities could be structural features of a classifier, instance specifications, or even roles in some containing collaboration. There may be multiple occurrences of a given collaboration within a classifier, each involving a different set of roles and connectors. A given role or connector may be involved in multiple occurrences of the same or different collaborations.

Associated dependencies map features of the collaboration type to features in the classifier. These dependencies indicate which role in the classifier plays which role in the collaboration.

CollaborationUse



This example shows the definition of two collaborations, *Sale* (Figure 9.13) and *BrokeredSale* (Figure 9.14). *Sale* is used twice as part of the definition of *BrokeredSale*. *Sale* is a collaboration among two roles, a *seller* and a *buyer*. An interaction, or other behavior specification, could be attached to *Sale* to specify the steps involved in making a *Sale*.

Sale
buyer
seller

Figure 9.13 - The Sale collaboration

BrokeredSale is a collaboration among three roles, a *producer*, a *broker*, and a *consumer*. The specification of *BrokeredSale* shows that it consists of two occurrences of the *Sale* collaboration, indicated by the dashed ellipses. The occurrence *wholesale* indicates a *Sale* in which the *producer* is the *seller* and the *broker* is the *buyer*. The occurrence *retail* indicates a *Sale* in which the *broker* is the *seller* and the *consumer* is the *buyer*. The connectors between *sellers* and *buyers* are not shown in the two occurrences; these connectors are implicit in the *BrokeredSale* collaboration in virtue of them being comprised of *Sale*. The *BrokeredSale* collaboration could itself be used as part of a larger collaboration.

CollaborationUse

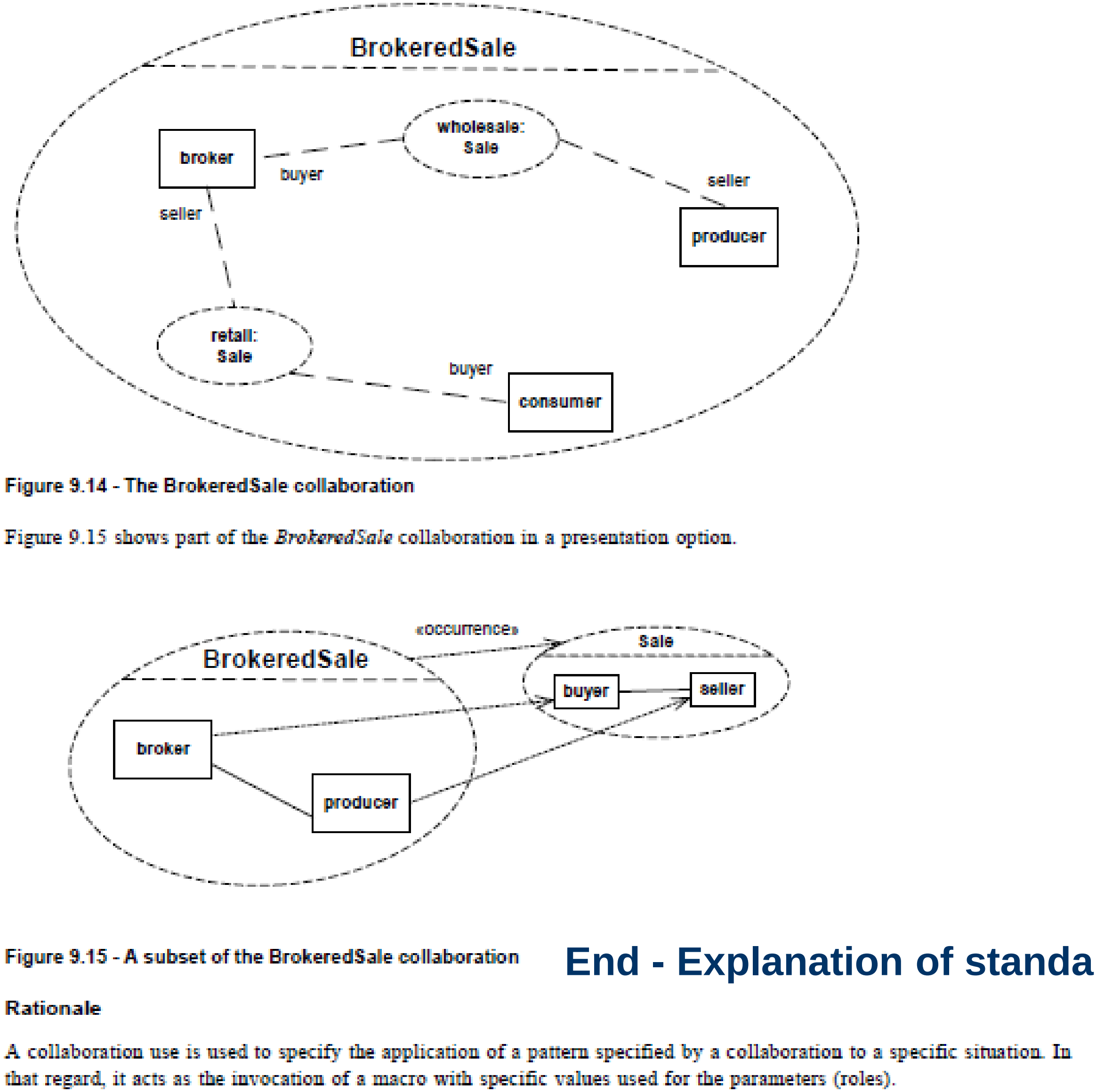

BrokeredSale
wholesale: Sale
broker
buyer
seller
seller
producer
retail: Sale
buyer
consumer

Figure 9.14 - The BrokeredSale collaboration

Figure 9.15 shows part of the *BrokeredSale* collaboration in a presentation option.

«occurrence»
BrokeredSale
Sale
buyer
seller
broker
producer

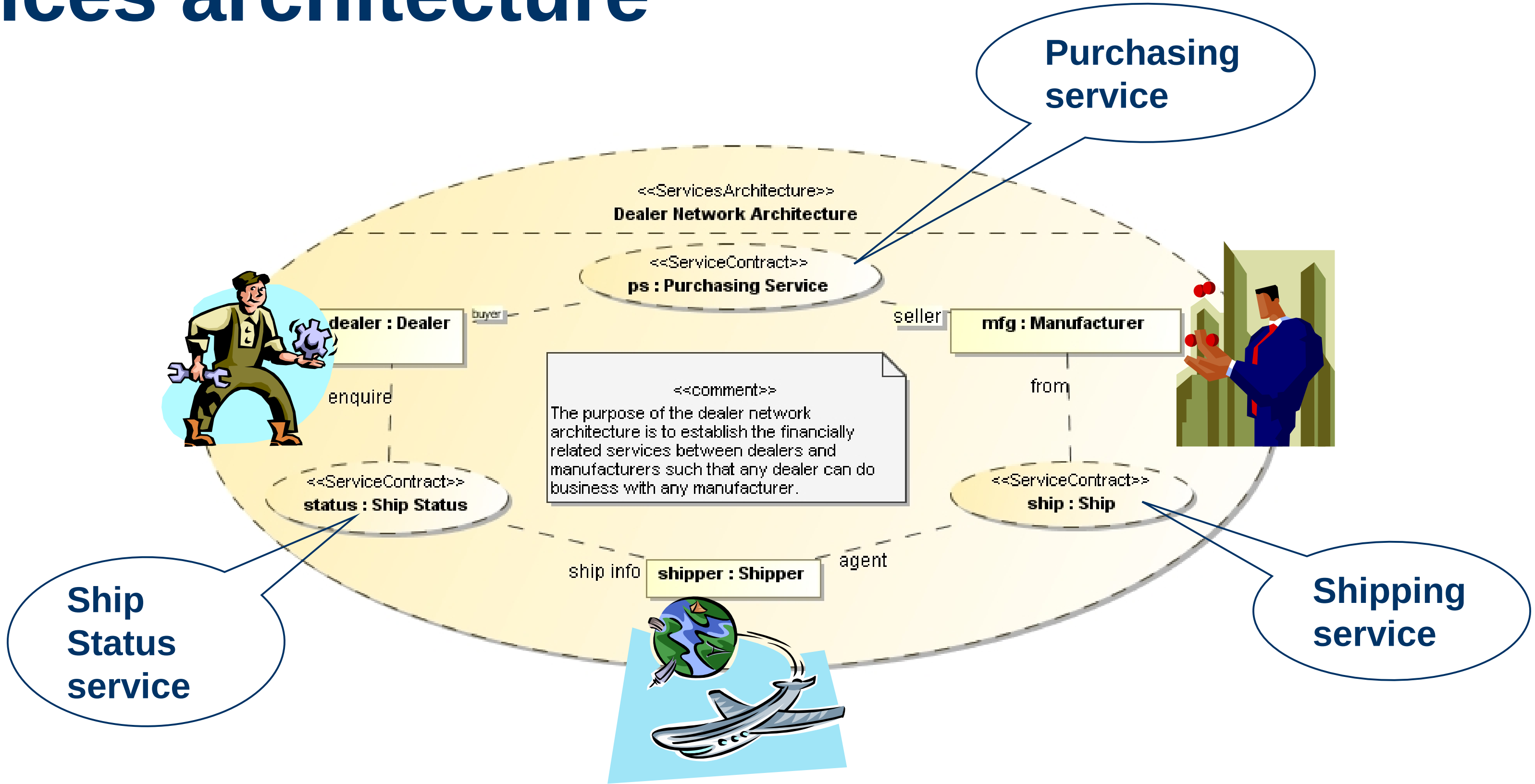
Figure 9.15 - A subset of the BrokeredSale collaboration

Rationale

A collaboration use is used to specify the application of a pattern specified by a collaboration to a specific situation. In that regard, it acts as the invocation of a macro with specific values used for the parameters (roles).

End - Explanation of standard UML 2.3

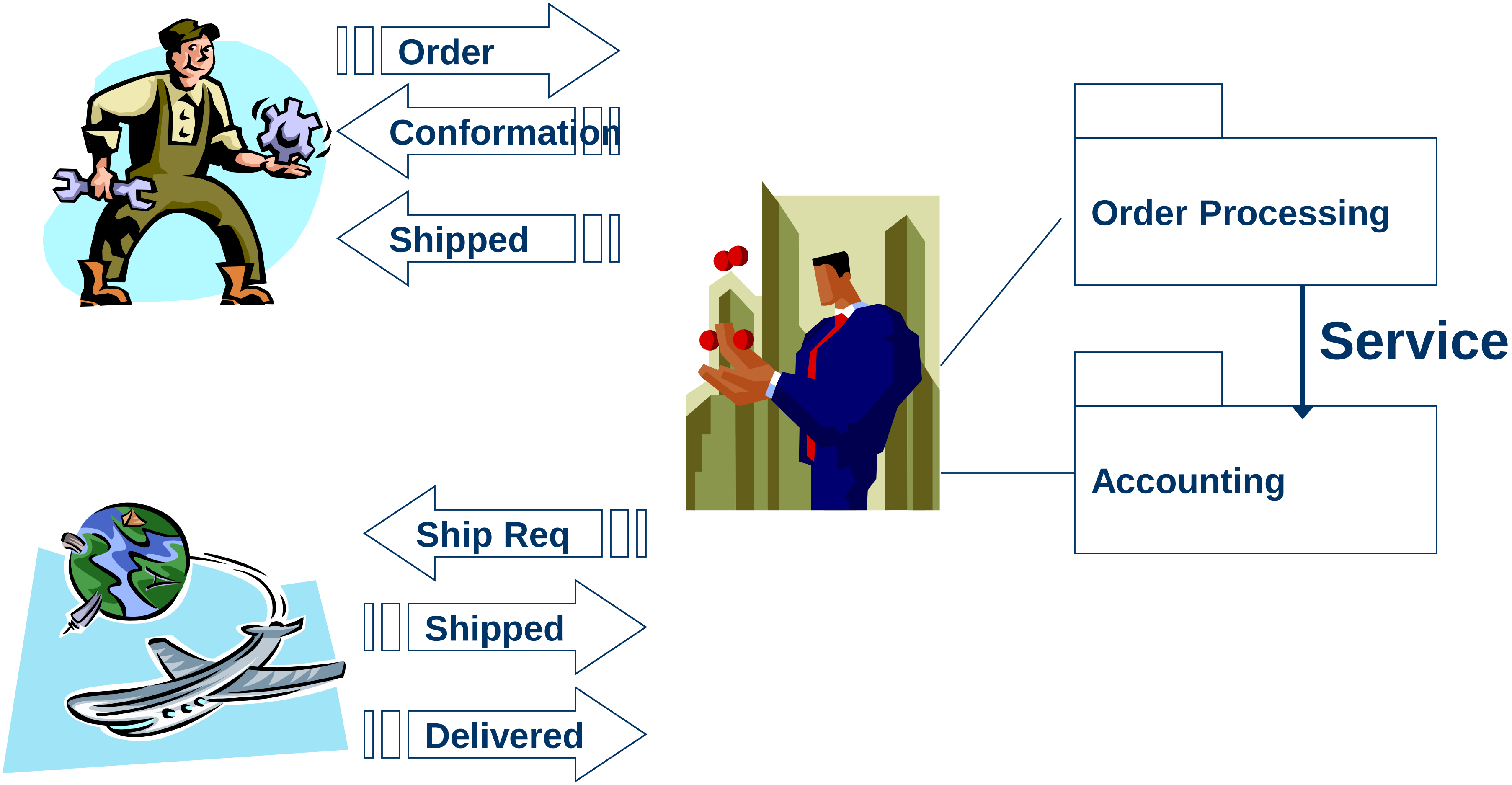
Services architecture



<<comment>>
The purpose of the dealer network architecture is to establish the financially related services between dealers and manufacturers such that any dealer can do business with any manufacturer.

- A ServicesArchitecture (or SOA):
  - is a network of participant roles *providing* and *consuming services* to fulfil a purpose.
  - defines the requirements for the types of participants and service realizations that fulfil those roles.
  - It is defined using a UML Collaboration.

Inside the Manufacturer



Order

Conformation

Shipped

Order Processing

Service

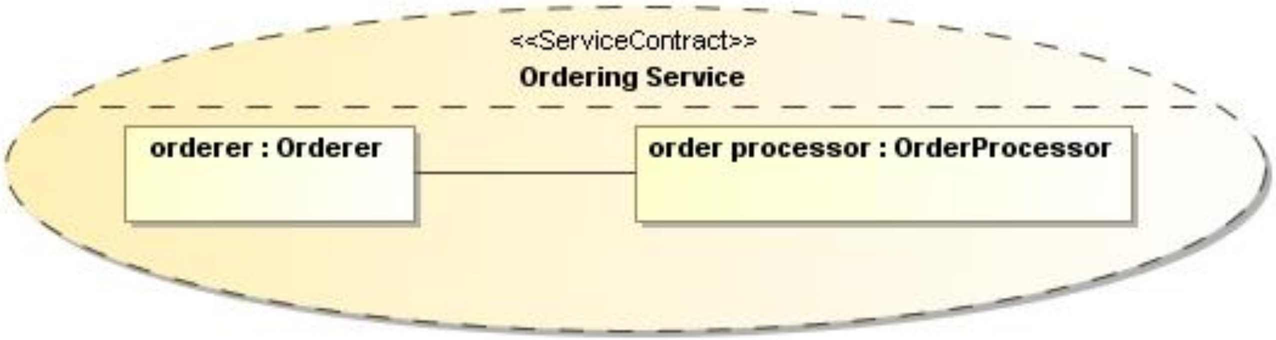
Accounting

Ship Req

Shipped

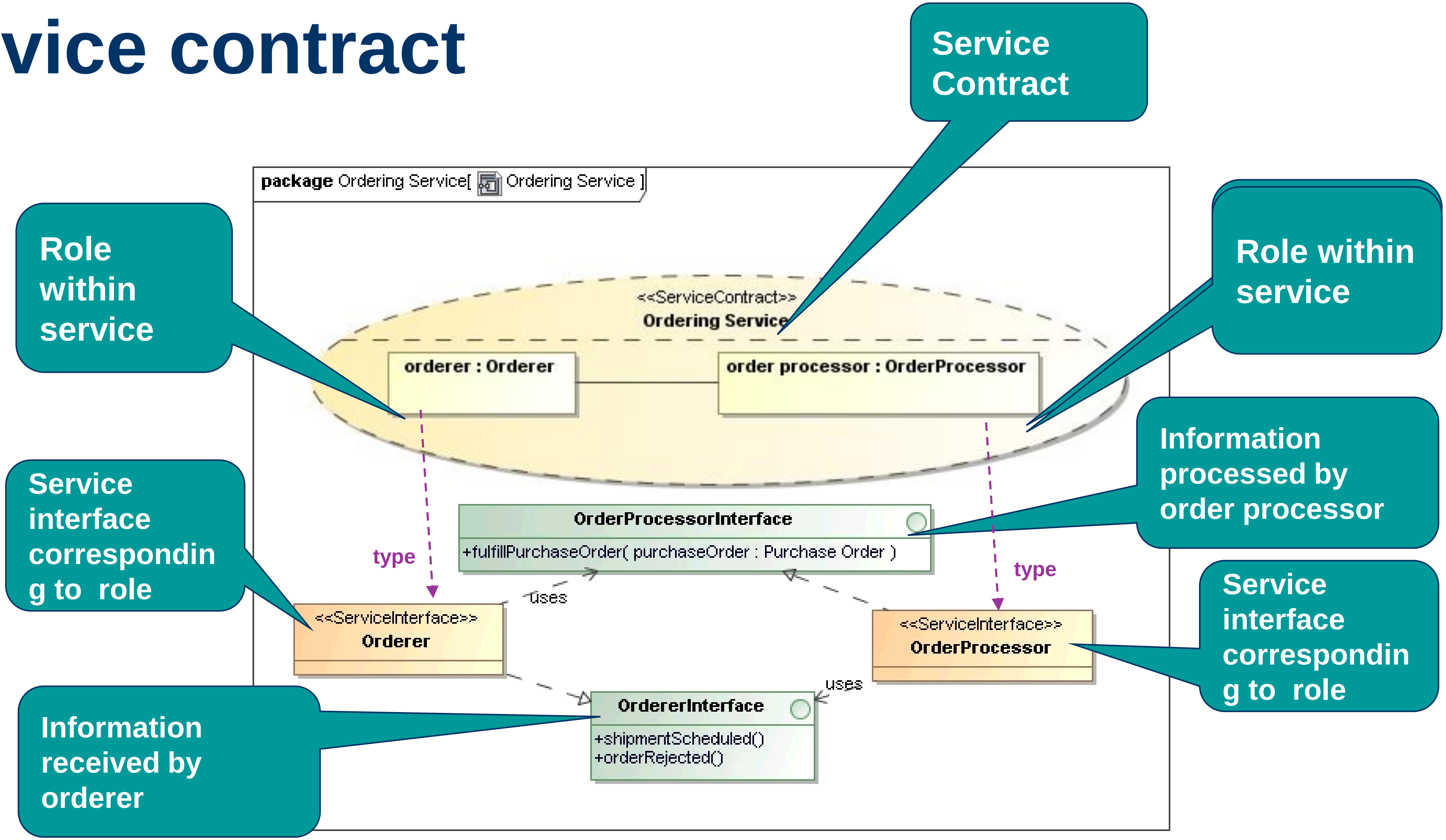
Delivered

Service contract



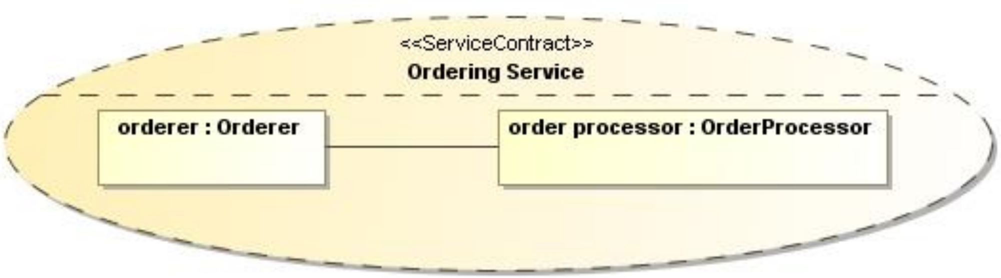
- A ServiceContract:
  - Fully specifies the service (terms, conditions, interfaces, choreography, etc.)
  - is binding on *both* the providers and consumers of that service.
  - is defined using a UML collaboration that is focused on the interactions involved in providing a service.
- A participant plays a role in the larger scope of a ServicesArchitecture and also plays a role as the provider or user of services specified by ServiceContracts.

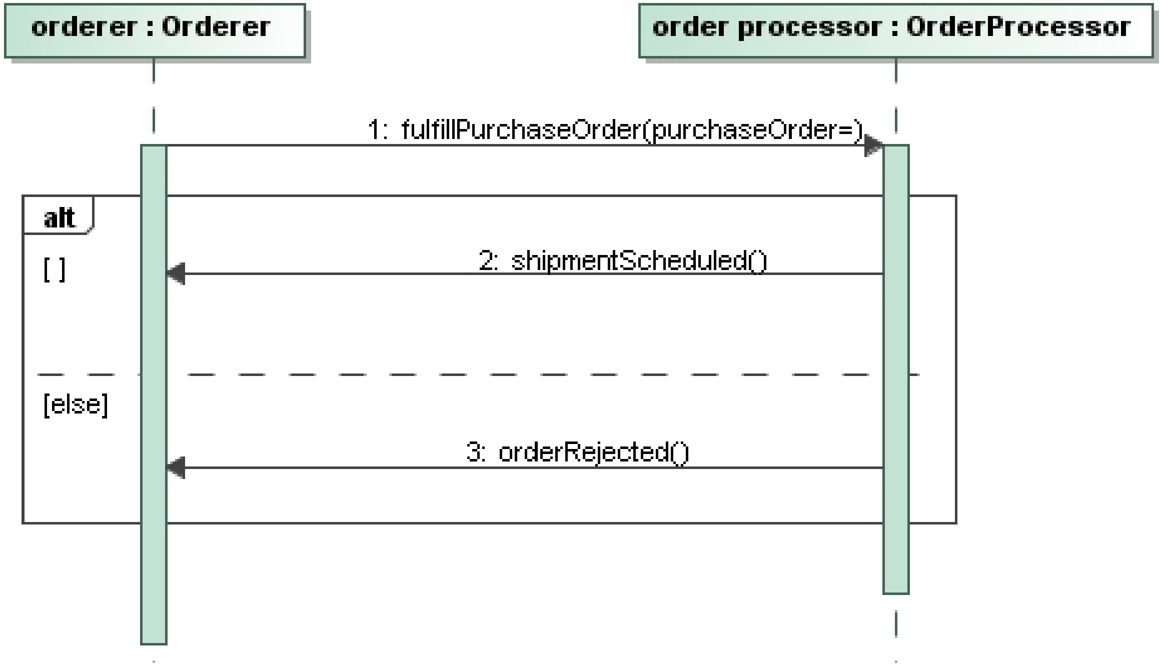
rvice contract


Service Contract
package Ordering Service[Ordering Service]
Role within service
Role within service
<<ServiceContract>>
Ordering Service
orderer : Orderer
order processor : OrderProcessor
Information processed by order processor
Service interface correspondin g to role
OrderProcessorInterface
+fulfillPurchaseOrder(purchaseOrder : Purchase Order)
type
type
uses
Service interface correspondin g to role
<<ServiceInterface>>
Orderer
<<ServiceInterface>>
OrderProcessor
uses
OrdererInterface
+shipmentScheduled()
+orderRejected()
Information received by orderer

- The service contract specifies the details of the service – what information, assets and responsibilities are exchanged and under what rules.

Simple protocol choreography for Ordering service contract

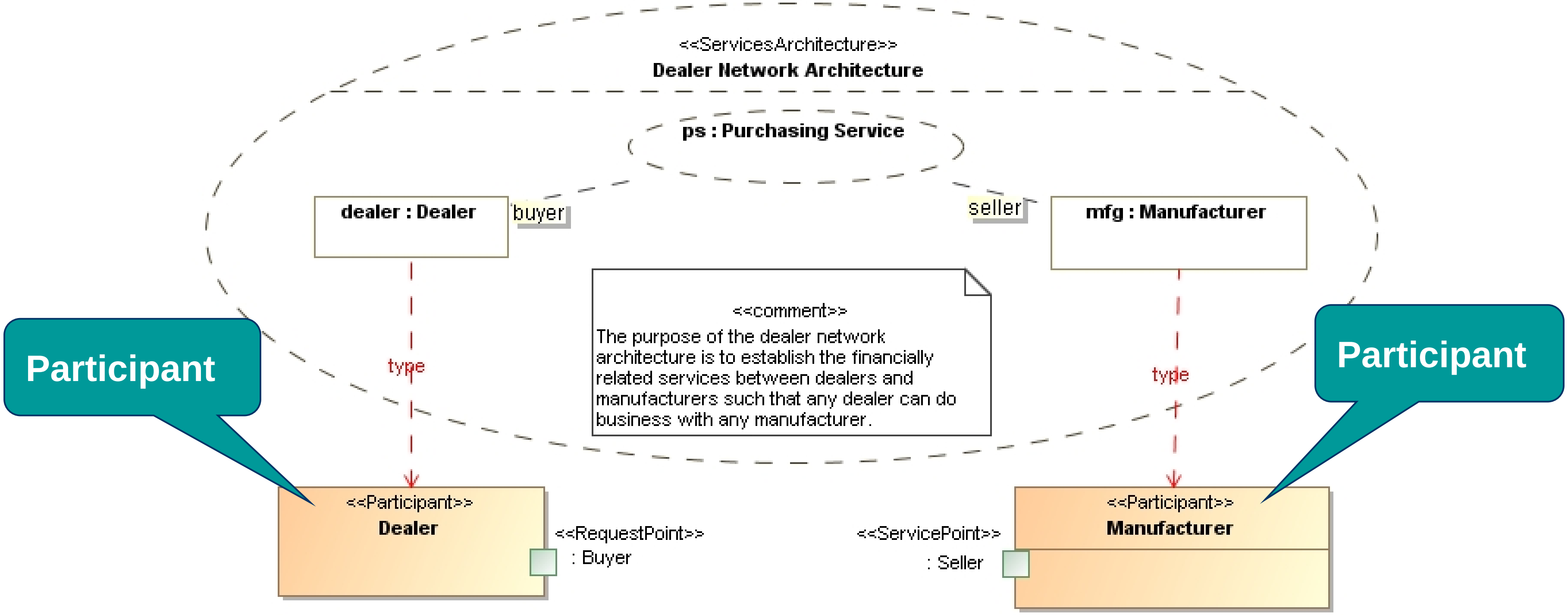




orderer : Orderer
order processor : OrderProcessor
1: fulfillPurchaseOrder(purchaseOrder=)
alt
[ ]
2: shipmentScheduled()
[else]
3: orderRejected()

Behaviour i ms or State
Machine

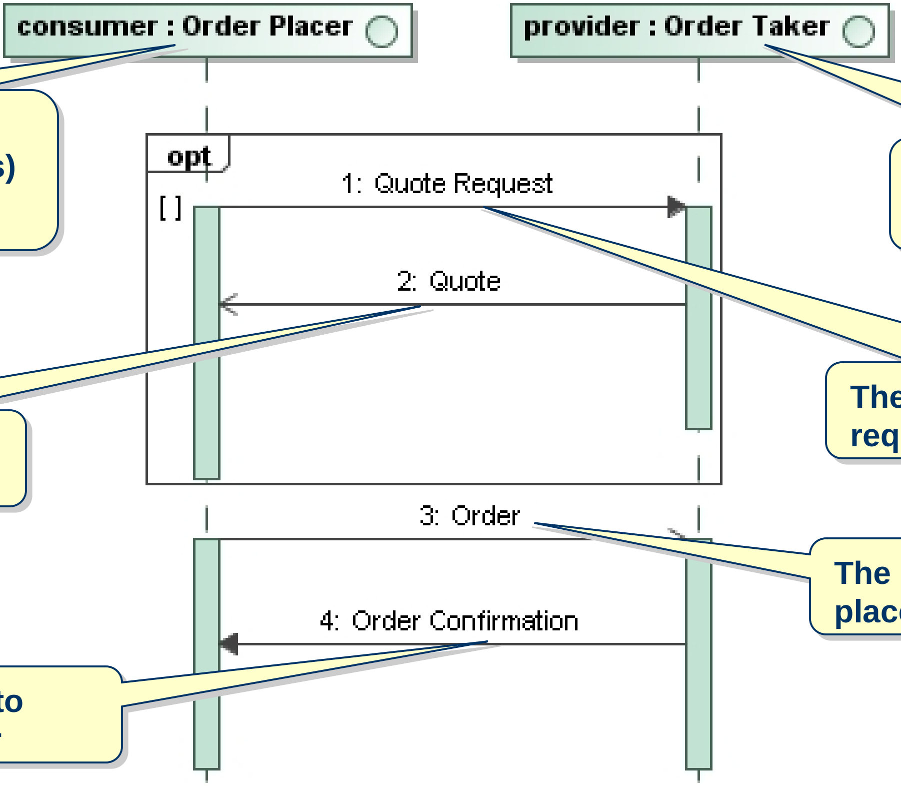
Participants



<<comment>>
The purpose of the dealer network architecture is to establish the financially related services between dealers and manufacturers such that any dealer can do business with any manufacturer.

- Participants:
  - represent logical or real people or organizational units that participate in services architectures and/or business processes.
  - provide and use services, defining their external contract

Service Choreography for “Place Order”



The role of the consumer (a participant that places orders) and the consumers interface

The role of the provider (an order taker) and their interface

The optional interaction to request a quote

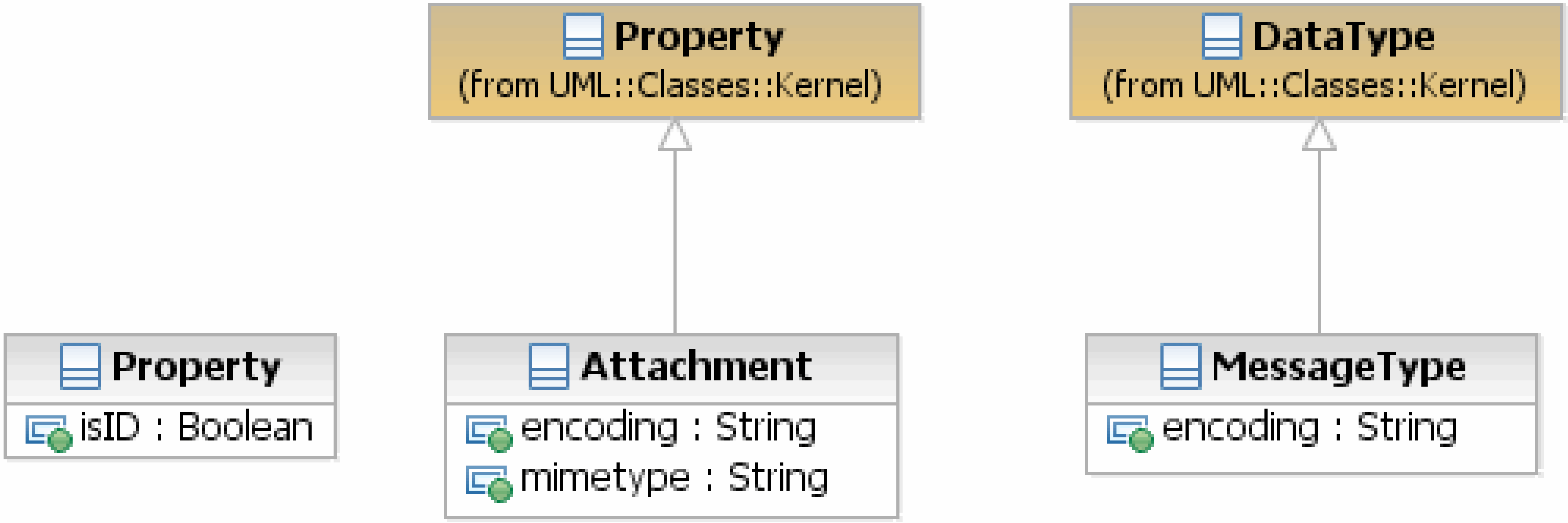
The optional interaction to return the quote

The required interaction to place an order

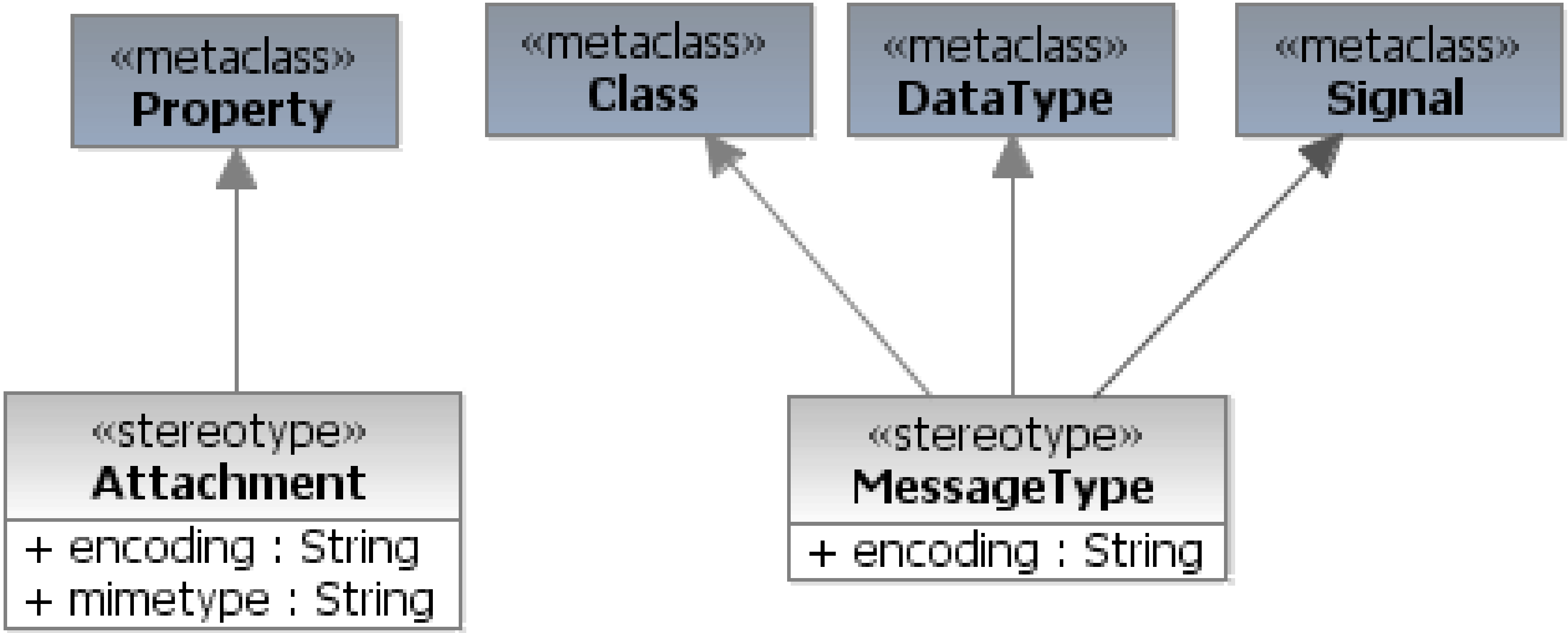
The required interaction to accept or reject the order

A more detailed look at the same service. Note that this models a fully asynchronous SOA – like most business interactions, the document message types are detailed on the next page.

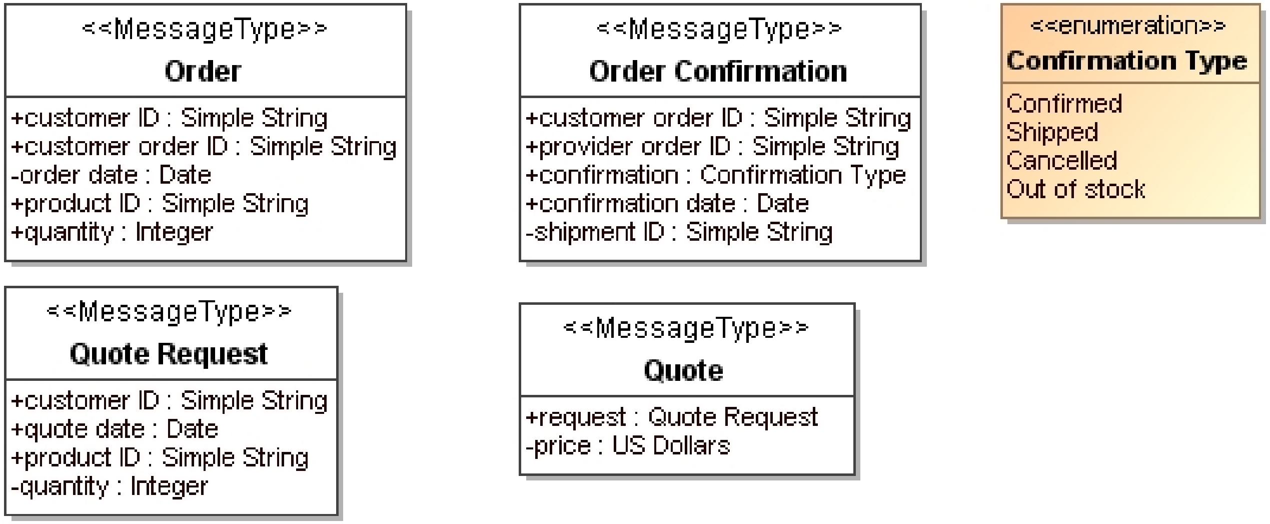
Service Data Metamodel



Service Data Profile



Message Detail for Place Order



<<MessageType>>
Order
+customer ID : Simple String
+customer order ID : Simple String
-order date : Date
+product ID : Simple String
+quantity : Integer

<<MessageType>>
Order Confirmation
+customer order ID : Simple String
+provider order ID : Simple String
+confirmation : Confirmation Type
+confirmation date : Date
-shipment ID : Simple String

<<enumeration>>
Confirmation Type
Confirmed
Shipped
Cancelled
Out of stock

<<MessageType>>
Quote Request
+customer ID : Simple String
+quote date : Date
+product ID : Simple String
-quantity : Integer

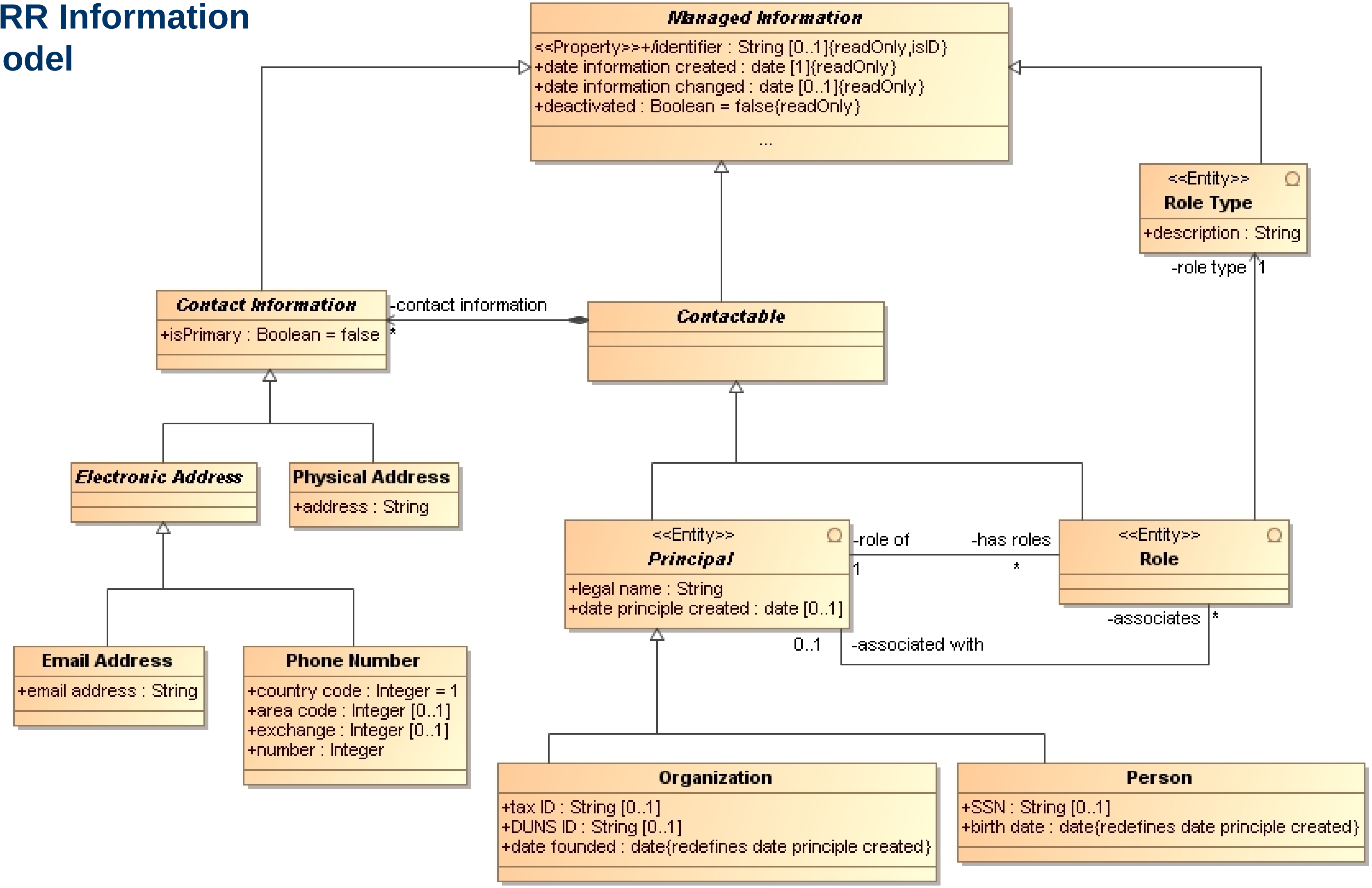
<<MessageType>>
Quote
+request : Quote Request
-price : US Dollars

This is the detail for the message types that correspond to the interactions for the place order service.

Note that at the technology level this can produce XML schema for the messages.

Example Information Model

CRR Information Model



Managed Information

<<Property>>+/identifier : String [0..1]{readOnly,isID}
+date information created : date [1]{readOnly}
+date information changed : date [0..1]{readOnly}
+deactivated : Boolean = false{readOnly}

...

<<Entity>>
Role Type

+description : String

-role type 1

Contact Information

+isPrimary : Boolean = false

-contact information *

Contactable

Electronic Address

Physical Address

+address : String

<<Entity>>
Principal

+legal name : String
+date principle created : date [0..1]

-role of 1

-has roles *

<<Entity>>
Role

-associates *

0..1 -associated with

Email Address

+email address : String

Phone Number

+country code : Integer = 1
+area code : Integer [0..1]
+exchange : Integer [0..1]
+number : Integer

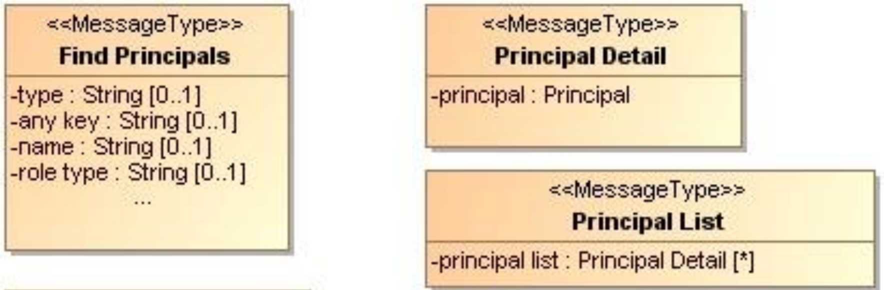
Organization

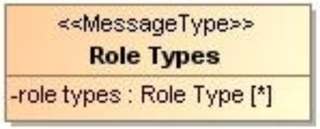
+tax ID : String [0..1]
+DUNS ID : String [0..1]
+date founded : date{redefines date principle created}

Person

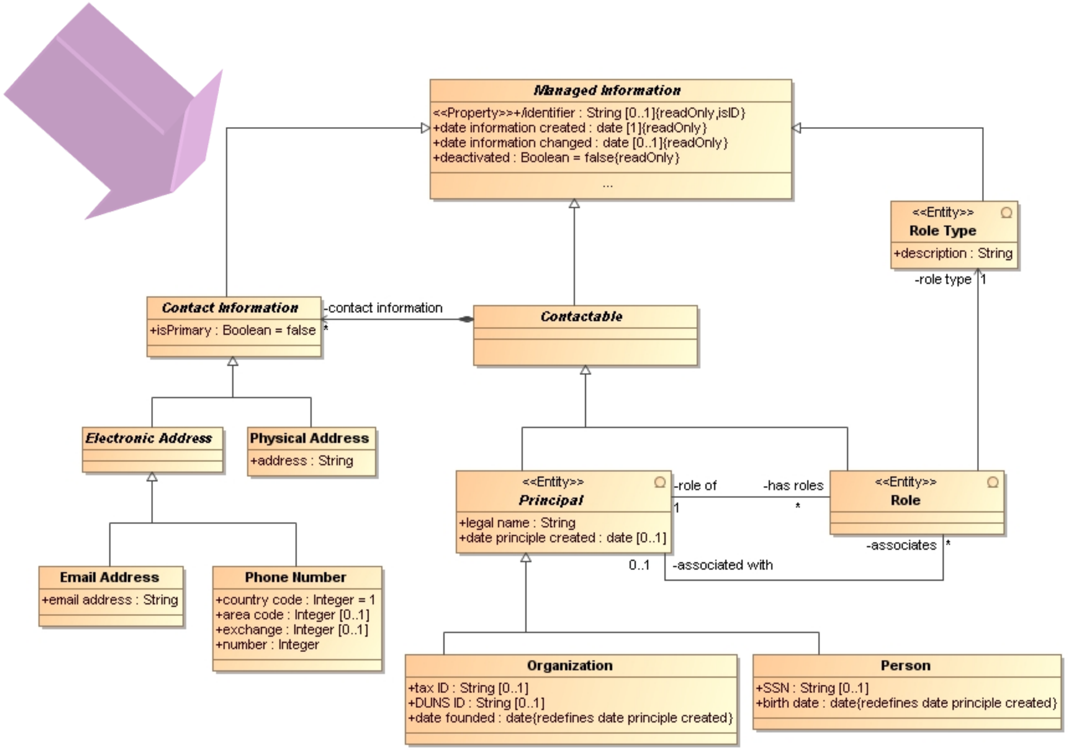
+SSN : String [0..1]
+birth date : date{redefines date principle created}

Linking messages to business information



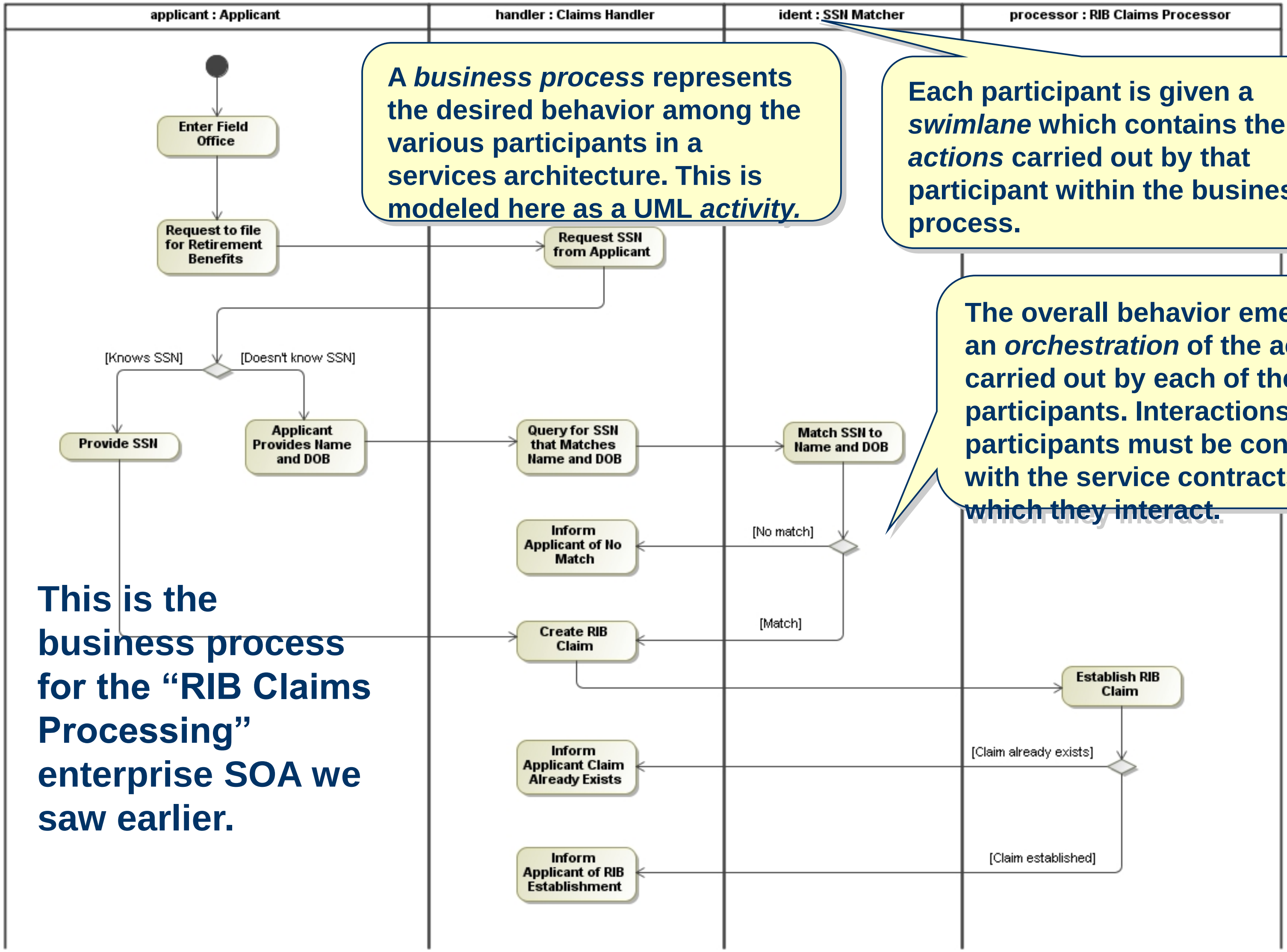


SOA Messages can reference and include parts of the logical information model – forming a connection between SOA and enterprise data



Managed Information
<<Property>>+/identifier : String [0..1]{readOnly,isID}
+date information created : date [1]{readOnly}
+date information changed : date [0..1]{readOnly}
+deactivated : Boolean = false{readOnly}
...
<<Entity>>
Role Type
+description : String
-role type
1
Contact Information
+isPrimary : Boolean = false
-contact information
*
Contactable
Electronic Address
Physical Address
+address : String
Email Address
+email address : String
Phone Number
+country code : Integer = 1
+area code : Integer [0..1]
+exchange : Integer [0..1]
+number : Integer
<<Entity>>
Principal
+legal name : String
+date principle created : date [0..1]
-role of
1
-has roles
*
<<Entity>>
Role
-associates
*
0..1
-associated with
Organization
+tax ID : String [0..1]
+DUNS ID : String [0..1]
+date founded : date{redefines date principle created}
Person
+SSN : String [0..1]
+birth date : date{redefines date principle created}

Linking the Business Process


applicant : Applicant
handler : Claims Handler
ident : SSN Matcher
processor : RIB Claims Processor
Enter Field Office
Request to file for Retirement Benefits
Request SSN from Applicant
[Knows SSN]
[Doesn't know SSN]
Provide SSN
Applicant Provides Name and DOB
Query for SSN that Matches Name and DOB
Match SSN to Name and DOB
[No match]
Inform Applicant of No Match
[Match]
Create RIB Claim
Establish RIB Claim
[Claim already exists]
Inform Applicant Claim Already Exists
[Claim established]
Inform Applicant of RIB Establishment

A *business process* represents the desired behavior among the various participants in a services architecture. This is modeled here as a UML *activity.*

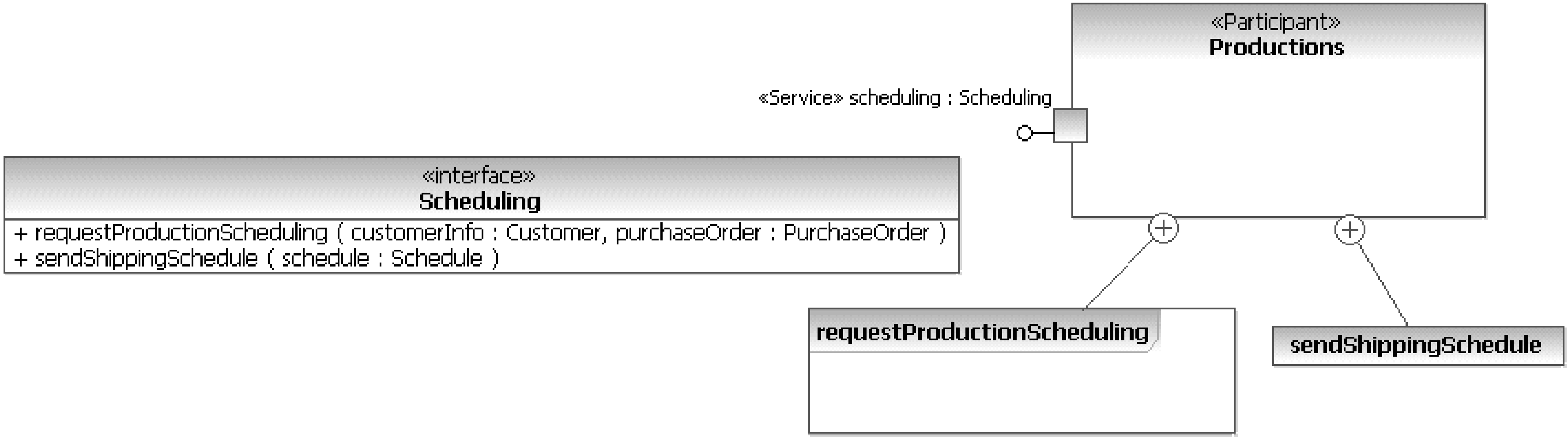
Each participant is given a *swimlane* which contains the *actions* carried out by that participant within the business process.

The overall behavior emerges as an *orchestration* of the actions carried out by each of the participants. Interactions with participants must be consistent with the service contracts by which they interact.

This is the business process for the “RIB Claims Processing” enterprise SOA we saw earlier.

UML 2.0 Composite diagrams and SoaML

Service ports and service participants



«Participant»
Productions

«Service» scheduling : Scheduling

«interface»
Scheduling

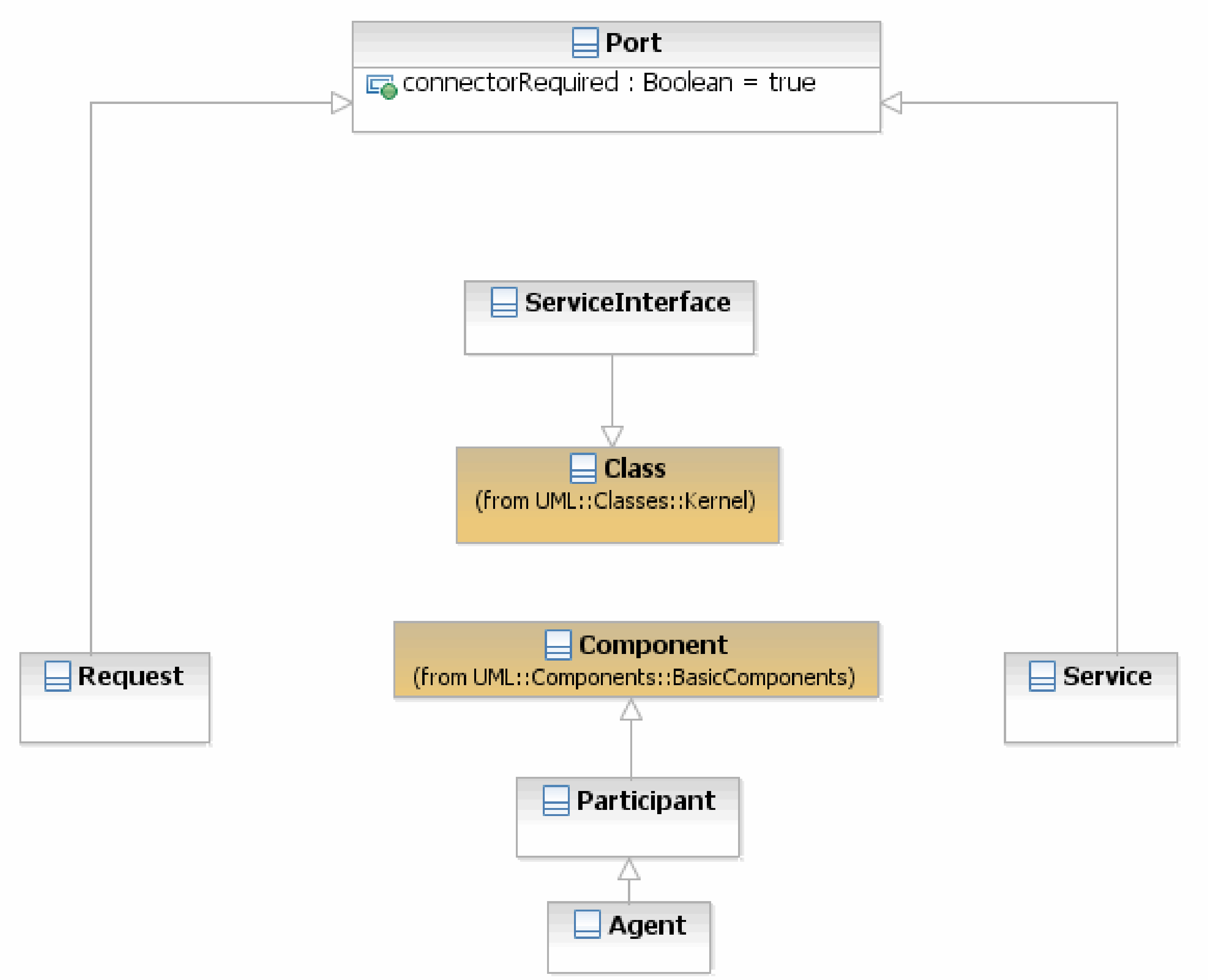
+ requestProductionScheduling (customerInfo : Customer, purchaseOrder : PurchaseOrder)
+ sendShippingSchedule (schedule : Schedule)

requestProductionScheduling

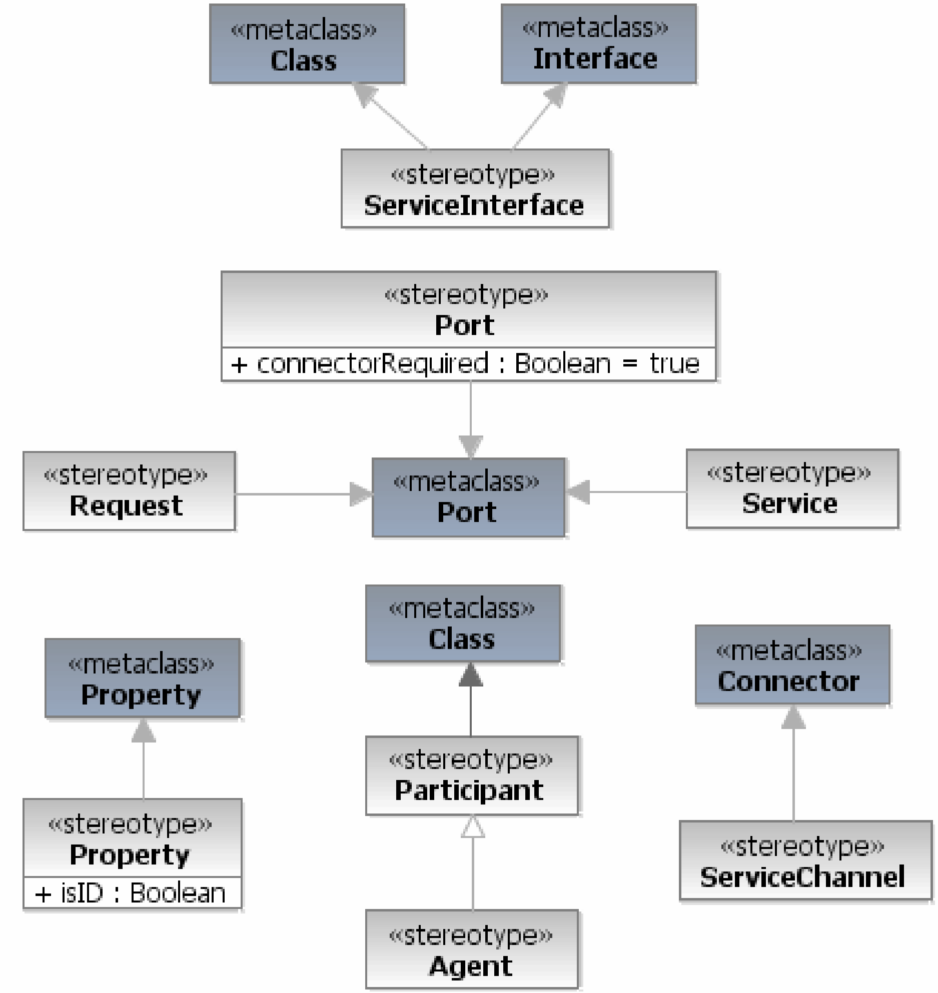
sendShippingSchedule

- A Service port:
  - is the offer of a service by one participant to others using well defined terms, conditions and interfaces
  - defines the connection point through which a Participant offers its capabilities and provides a service to clients.
  - It is defined using a UML Port on a Participant, and stereotyped as a <<Service>>
- A Service port is a mechanism by which a provider Participant makes available services that meet the needs of consumer requests as defined by ServiceInterfaces, Interfaces and ServiceContracts.

ServiceInterfaces and Participants Metamodel


Port
connectorRequired : Boolean = true
ServiceInterface
Class
(from UML::Classes::Kernel)
Request
Component
(from UML::Components::BasicComponents)
Service
Participant
Agent

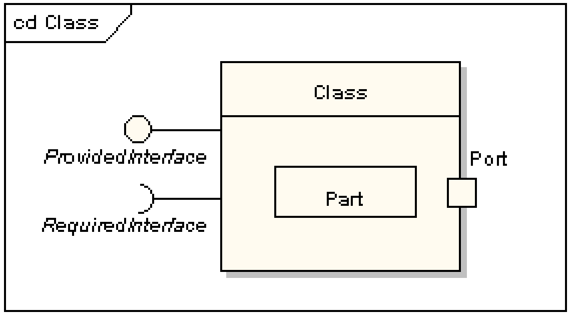
ServiceInterfaces and Participants Profile



UML Composite Diagrams

- Composite Diagrams
  A composite structure diagram is a diagram that shows the internal structure of a classifier, including its interaction points to other parts of the system. It shows the configuration and relationship of parts, that together, perform the behavior of the containing classifier.

  classes can be displayed as composite elements exposing interfaces and containing ports and parts.



Start - Explanation of standard UML 2.3

Part

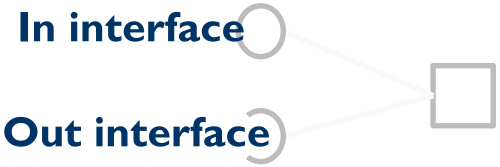
- A part is an element that represents a set of one or more instances which are owned by a containing classifier instance. So for example, if a diagram instance owned a set of graphical elements, then the graphical elements could be represented as parts; if it were useful to do so, to model some kind of relationship between them. Note that a part can be removed from its parent before the parent is deleted, so that the part isn't deleted at the same time.

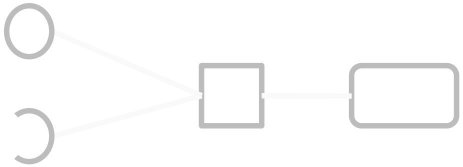
  A part is shown as an unadorned rectangle contained within the body of a class or component element.

Ports

- A port is attached to an active class.
- The port has:
  - A name.
  - An interface specifying the signals that can be received.
  - An interface specifying the signals that can be sent.

- Two types of ports:
  - Connected to internal communication channels (by default).
  - Connected to the state machine for the class instance (a behaviour port).



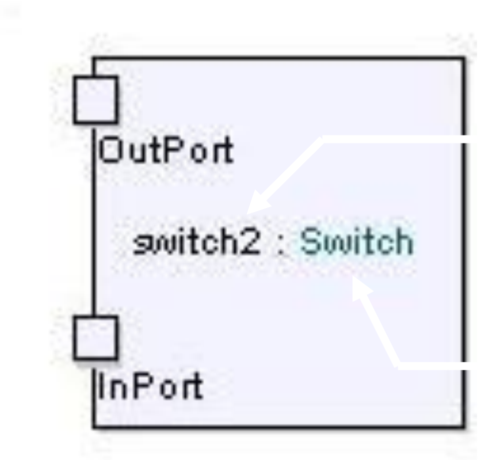


A behaviour port

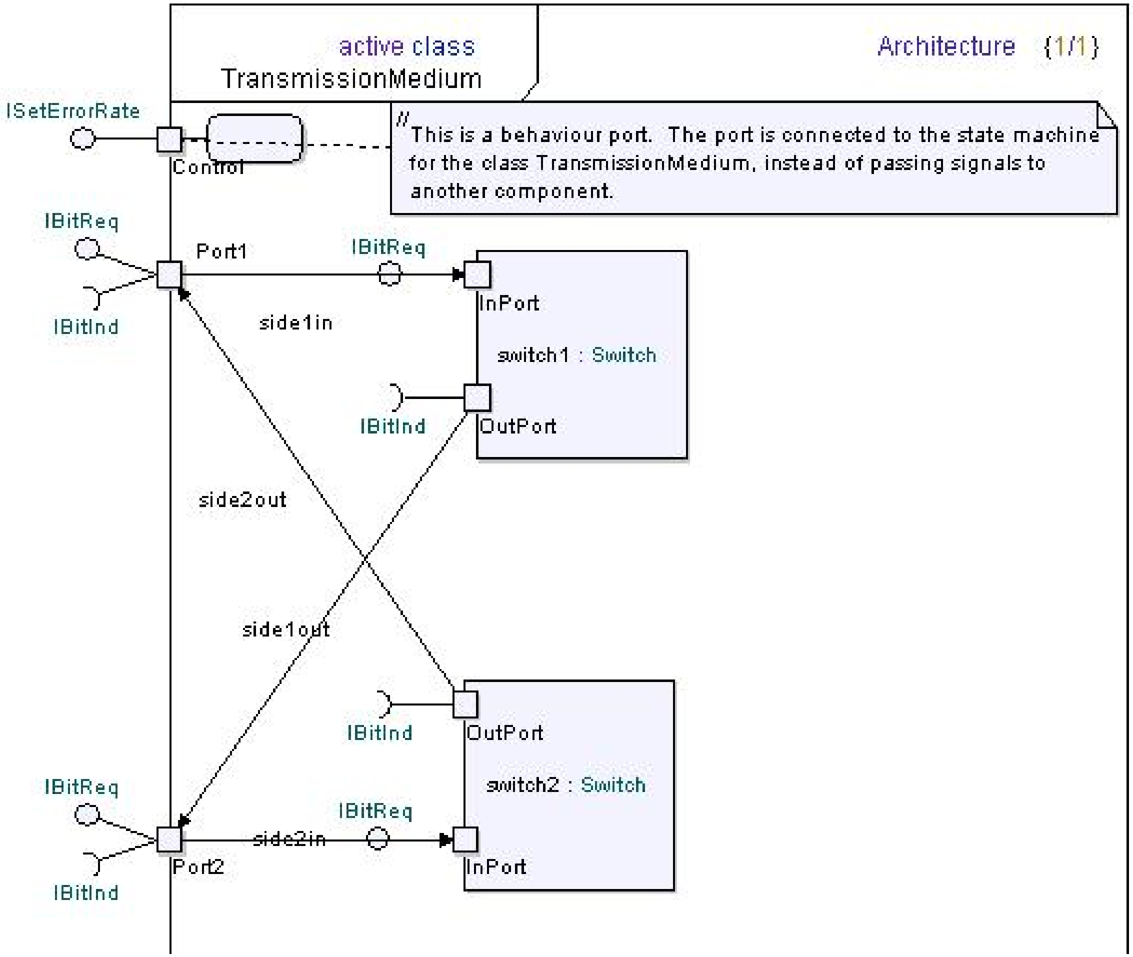
Composite Structure

- A composite structure diagram shows the relationship among internal components of a class, in terms of communication paths.
- The class may have one or more communications ports through which signals can be sent or received.
- The ports are connected either to:
  - Internal components
    - Channels connect the ports of the class to the ports of the internal components.
    - Channels can be unidirectional (one direction only) or bidirectional (both directions).
  - The state machine behaviour of the class (a behaviour port).

Object instance references



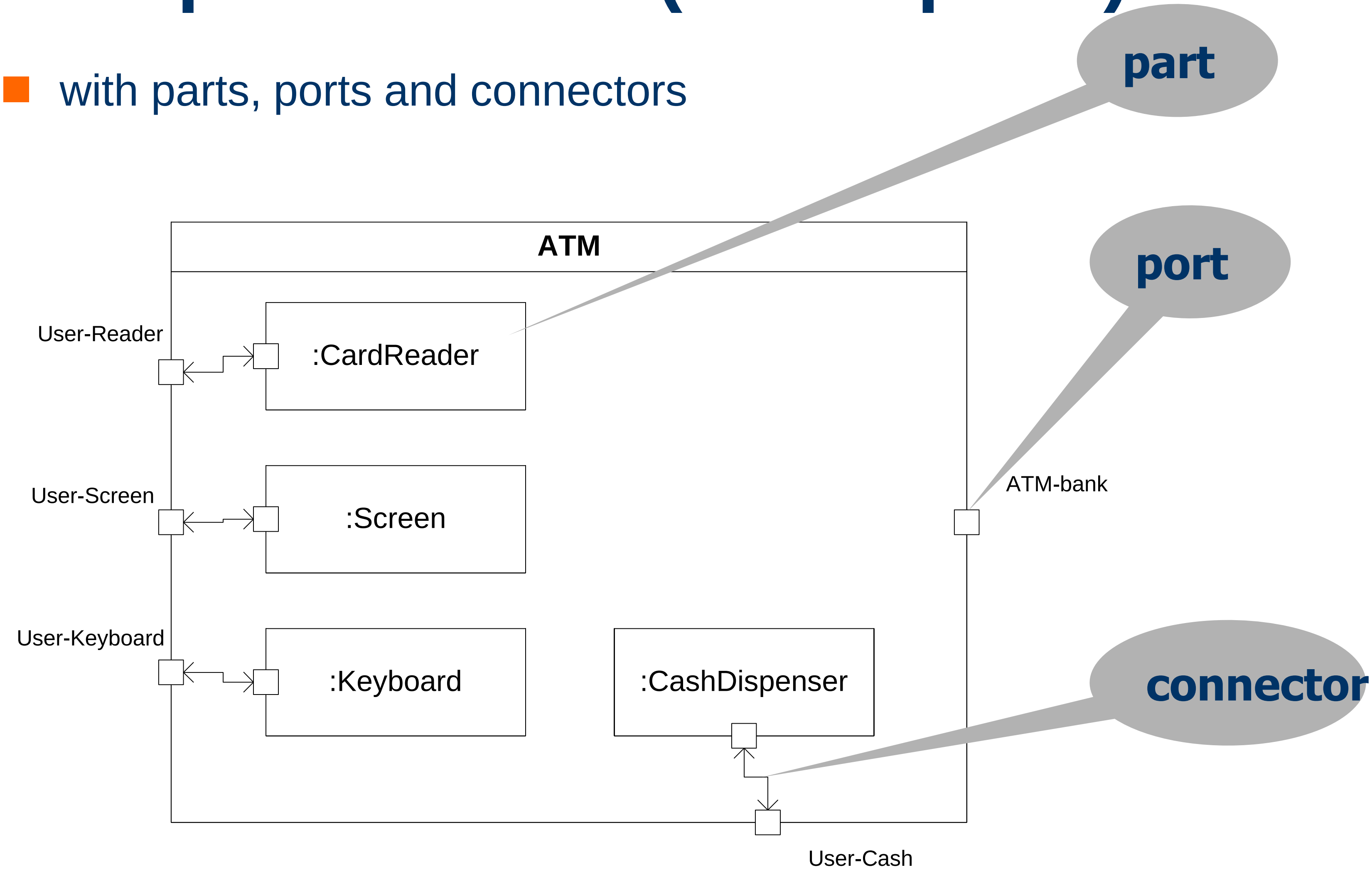
Composite Structure



// This is a behaviour port. The port is connected to the state machine for the class TransmissionMedium, instead of passing signals to another component.

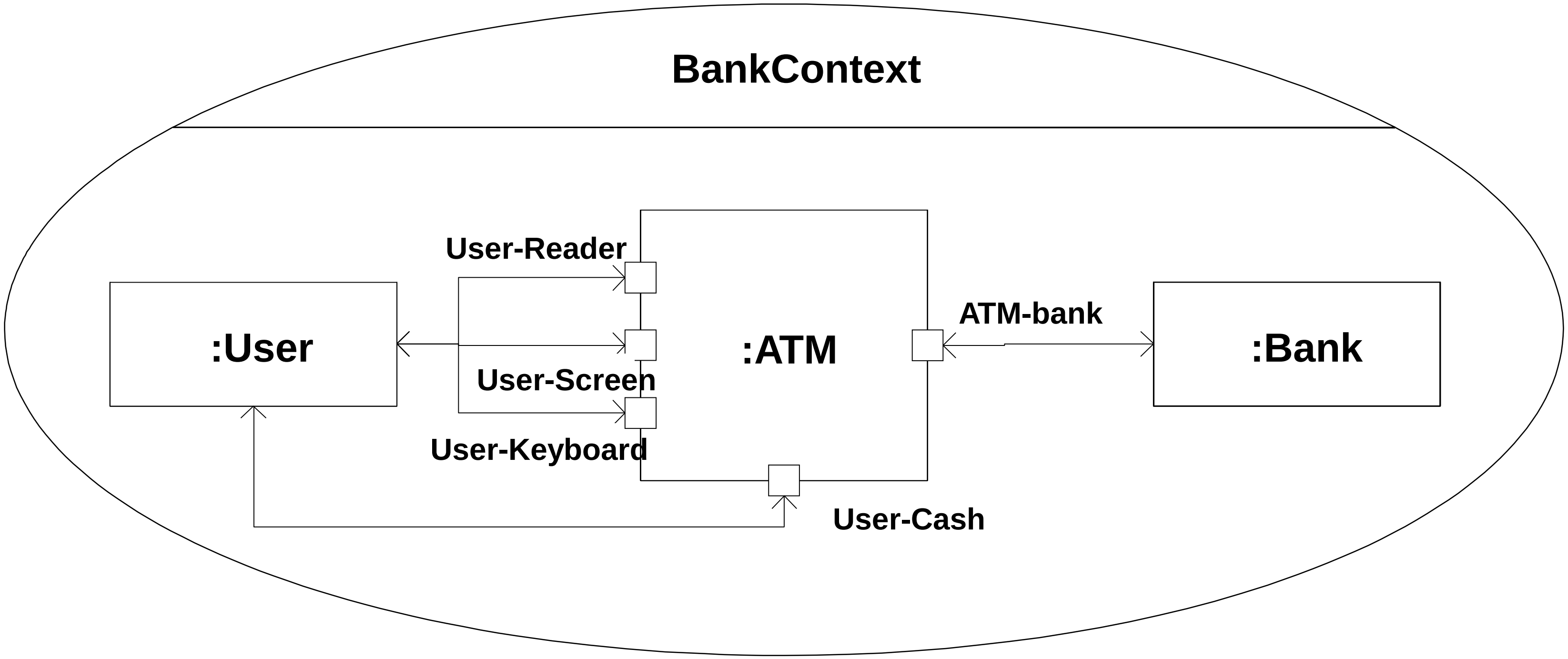
Composite class (incomplete)

- with parts, ports and connectors



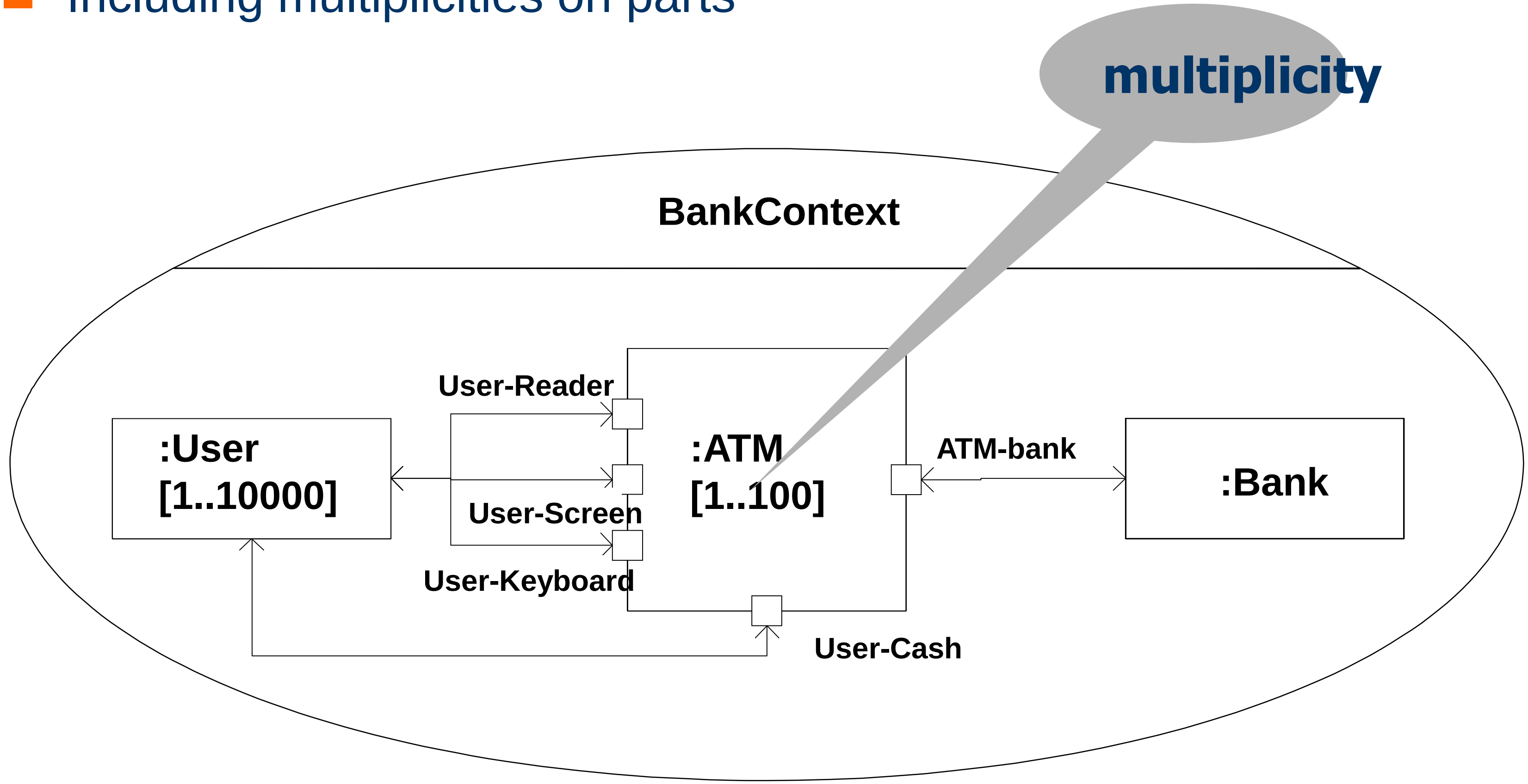
Context Model in UML2.0 - I

- composite structure as part of a Collaboration



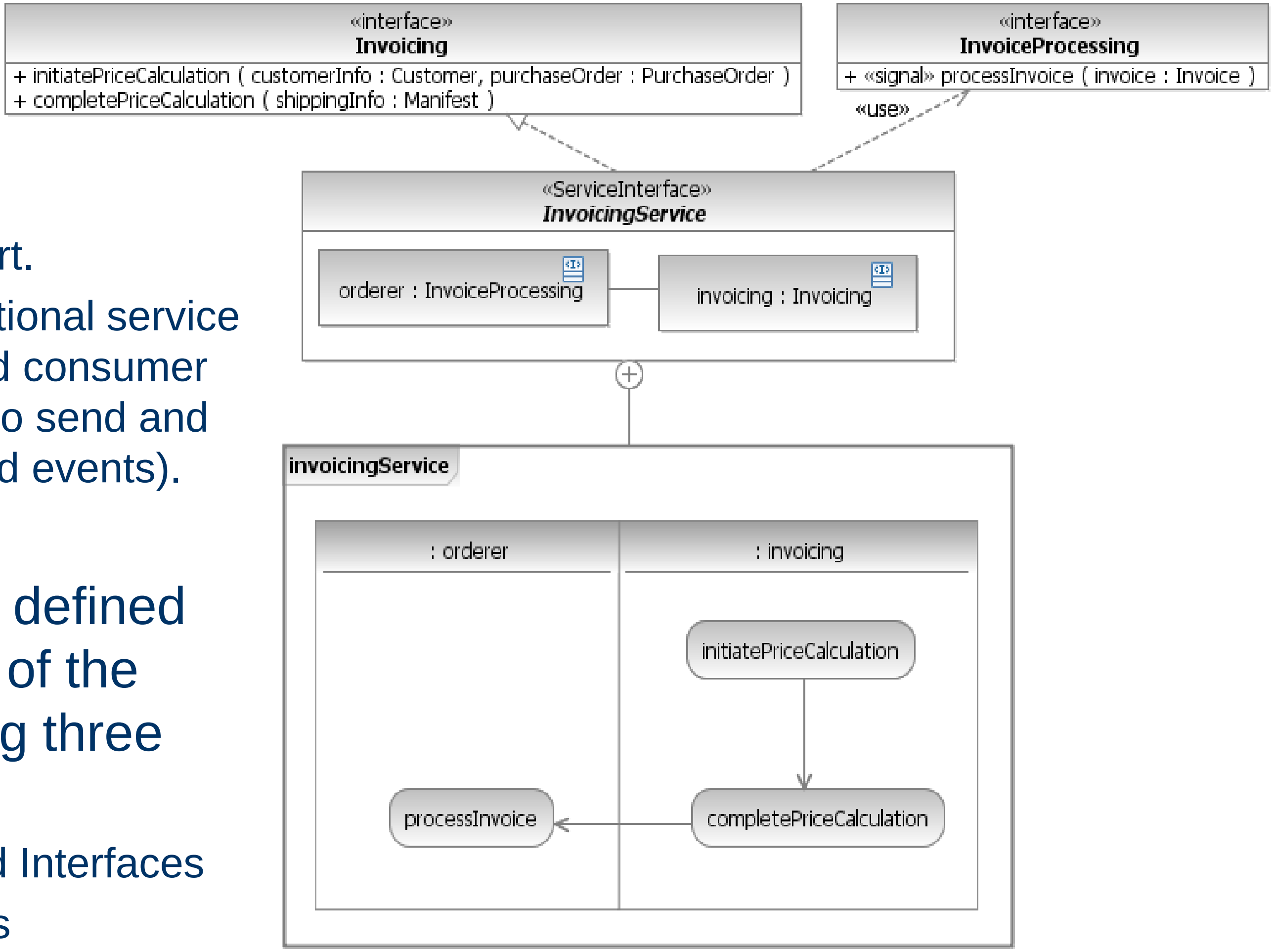
Context Model in UML2.0 - II

- Including multiplicities on parts



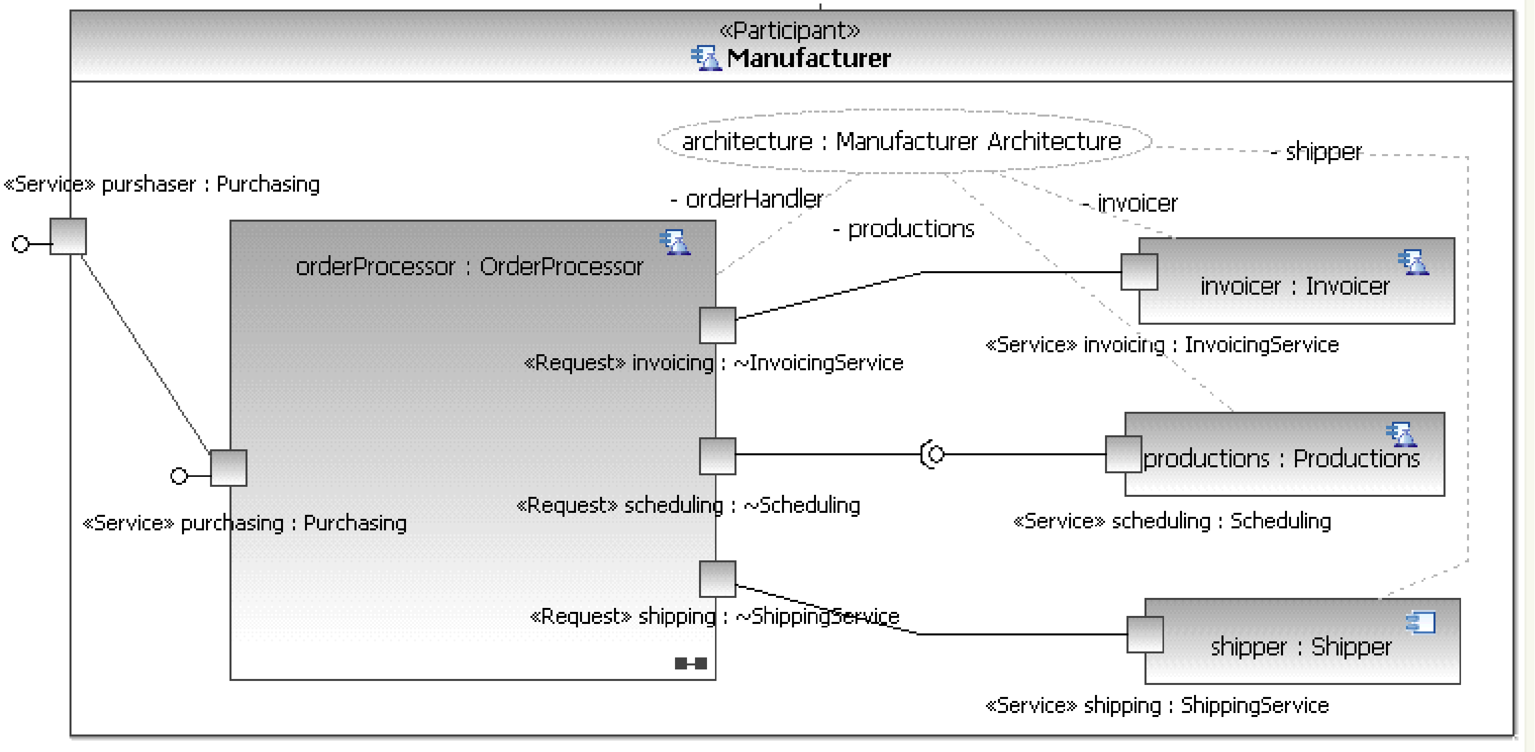
End - Explanation of standard UML 2.3

Service interface


«interface»
Invoicing
+ initiatePriceCalculation (customerInfo : Customer, purchaseOrder : PurchaseOrder)
+ completePriceCalculation (shippingInfo : Manifest)
«interface»
InvoiceProcessing
+ «signal» processInvoice (invoice : Invoice)
«use»
«ServiceInterface»
InvoicingService
orderer : InvoiceProcessing
invoicing : Invoicing
invoicingService
: orderer
: invoicing
initiatePriceCalculation
processInvoice
completePriceCalculation

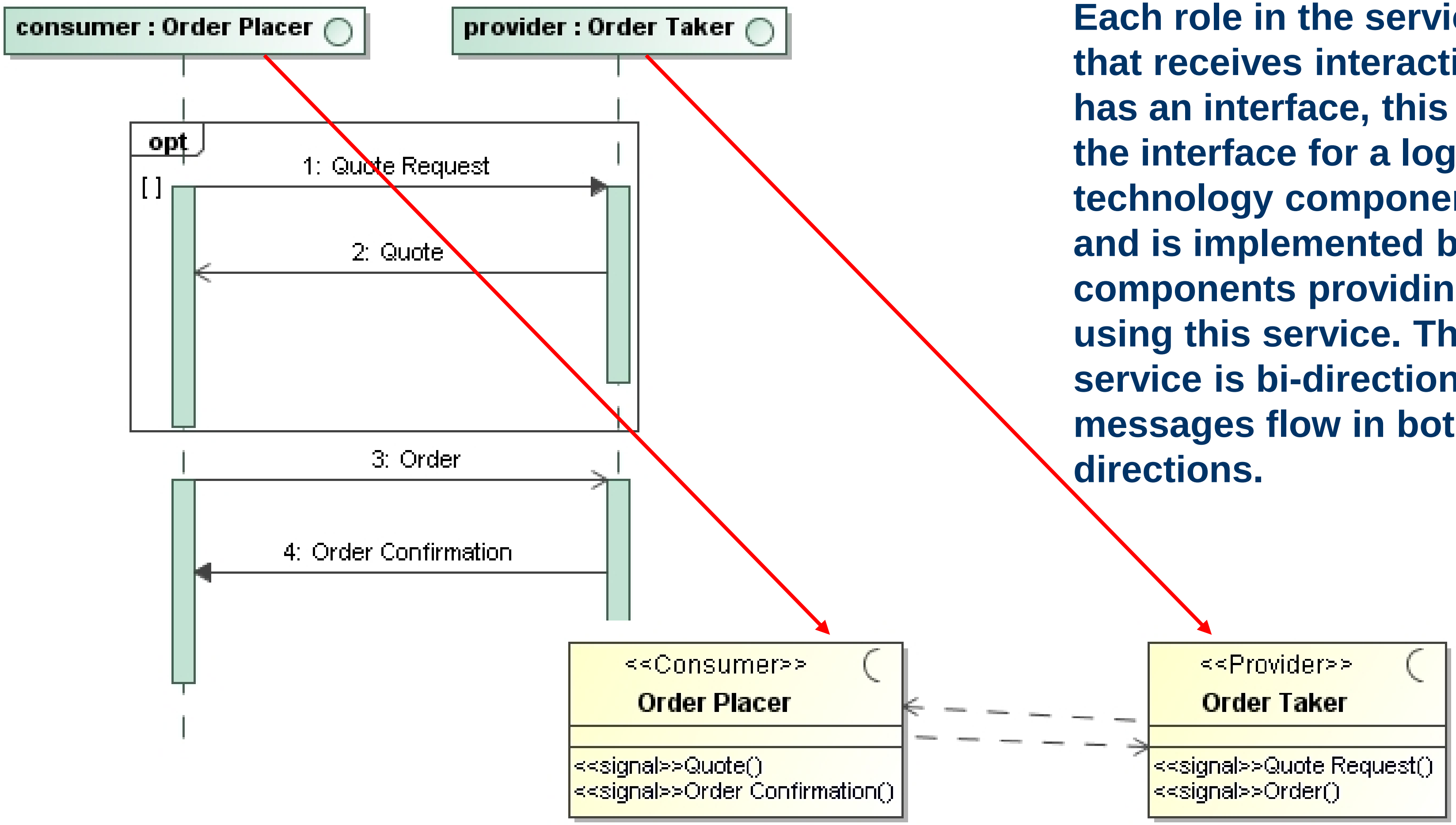
- A ServiceInterface:
  - can type a service port.
  - can specify a bi-directional service (both the provider and consumer have responsibilities to send and receive messages and events).
- A ServiceInterface is defined from the perspective of the service provider using three primary sections:
  - provided and required Interfaces
  - ServiceInterface class
  - protocol Behavior.

Participant with service and request ports


«Participant»
Manufacturer
architecture : Manufacturer Architecture
- shipper
«Service» purshaser : Purchasing
- orderHandler
- productions
- invoicer
orderProcessor : OrderProcessor
invoicer : Invoicer
«Request» invoicing : ~InvoicingService
«Service» invoicing : InvoicingService
productions : Productions
«Service» purchasing : Purchasing
«Request» scheduling : ~Scheduling
«Service» scheduling : Scheduling
«Request» shipping : ~ShippingService
shipper : Shipper
«Service» shipping : ShippingService

- A Service Port is typed by a ServiceInterface
- A Request port is typed by a conjugate ServiceInterface (defines the use of a service rather than its provision). This will allow us to connect service providers and consumers in a Participant.
- *Can be transformed to the appropriate interface/implementation code.*

Interfaces for Participants

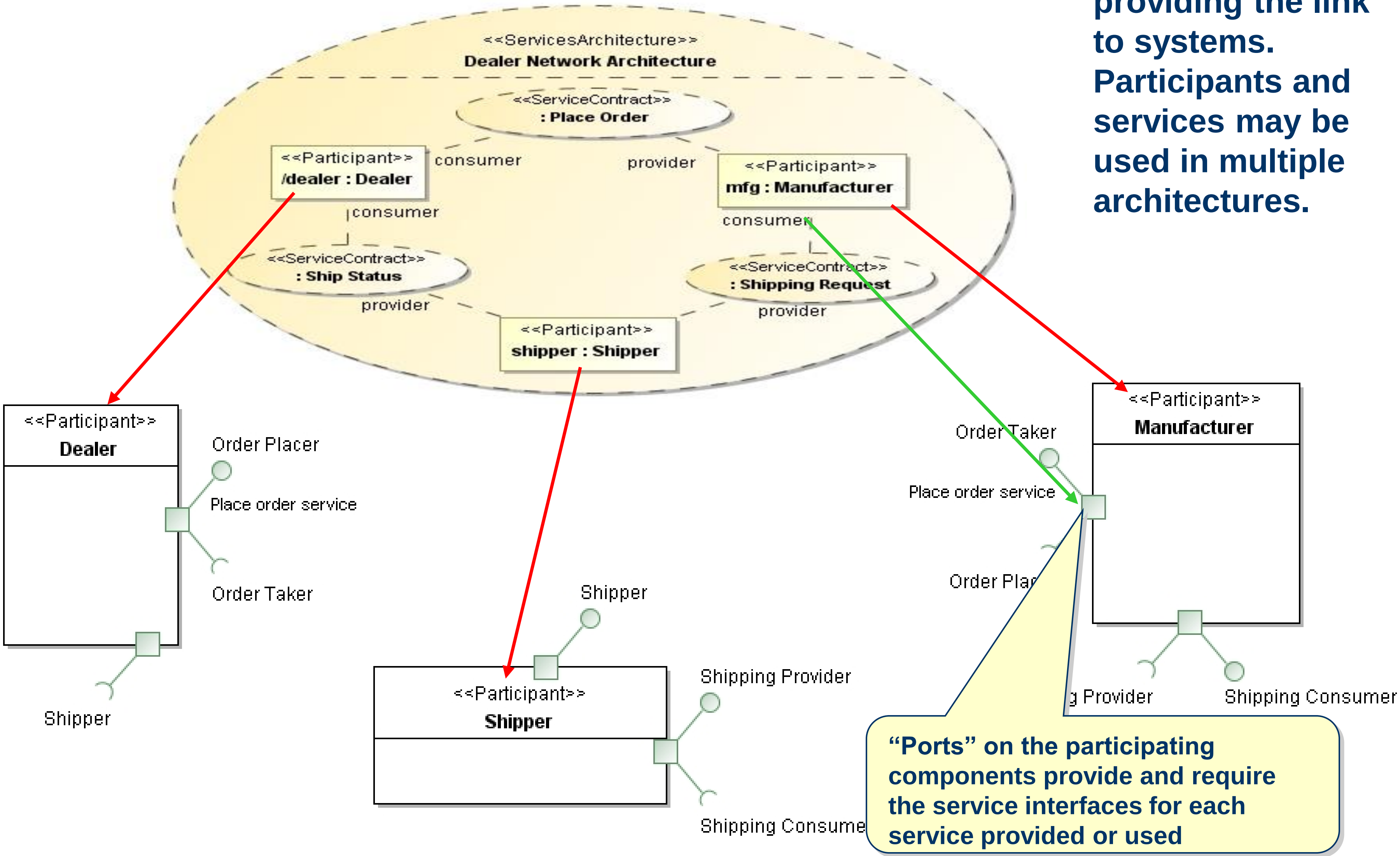


Each role in the service that receives interactions has an interface, this is the interface for a logical technology component and is implemented by components providing or using this service. This service is bi-directional - messages flow in both directions.

Interfaces will correspond with parts of WSDL in a web services mapping of SoaML

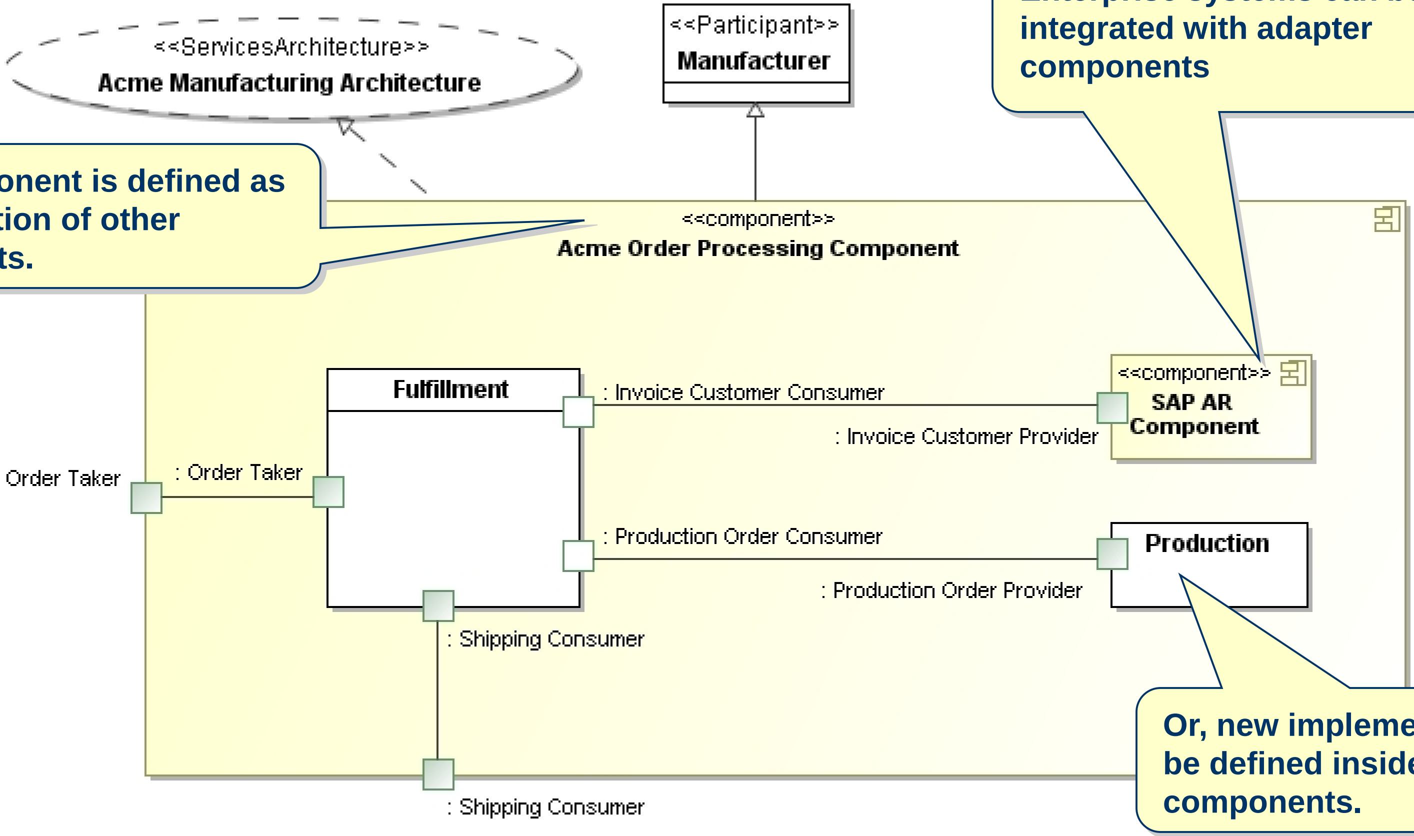
Logical System Components

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



"Ports" on the participating components provide and require the service interfaces for each service provided or used

Composite Application Components



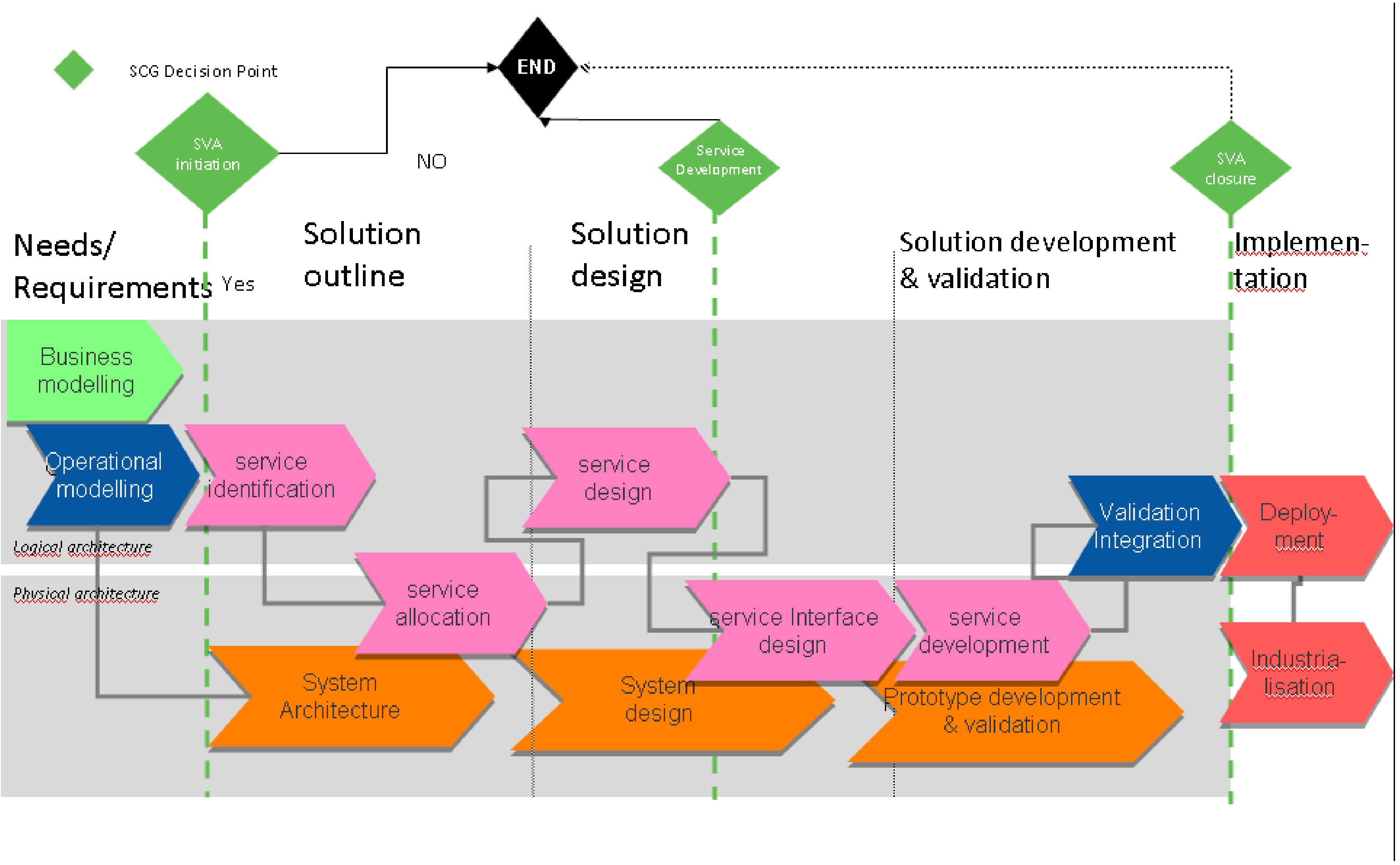
This component is defined as a composition of other components.

Enterprise systems can be integrated with adapter components

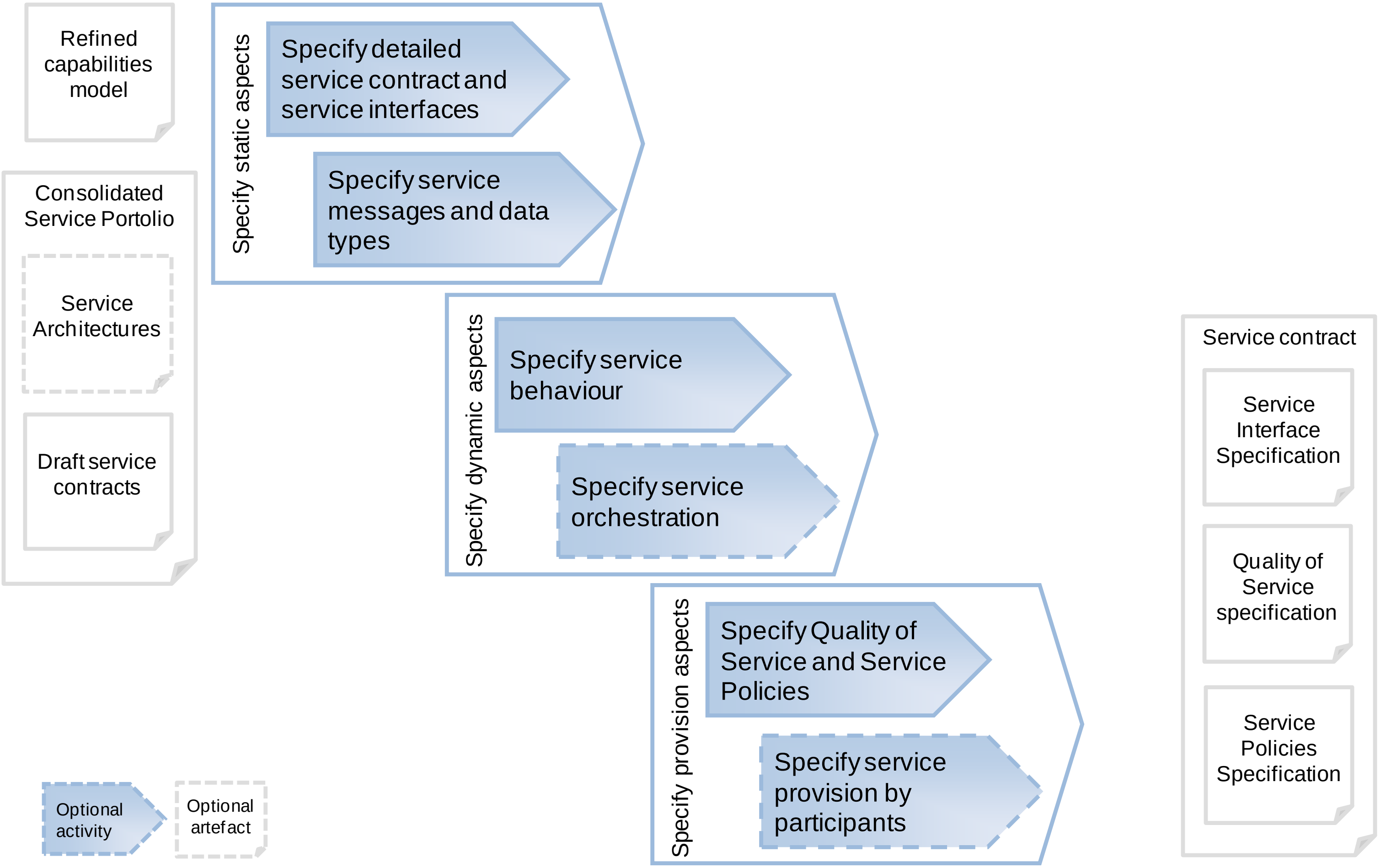
Or, new implementation can be defined inside of components.

Components can be assembled from other components by linking their services. This corresponds to the architecture for Acme.

Service architecting process



Service Design



Refined capabilities model

Consolidated Service Portolio

Service Architectures

Draft service contracts

Specify static aspects

Specify detailed service contract and service interfaces

Specify service messages and data types

Specify dynamic aspects

Specify service behaviour

Specify service orchestration

Specify provision aspects

Specify Quality of Service and Service Policies

Specify service provision by participants

Service contract

Service Interface Specification

Quality of Service specification

Service Policies Specification

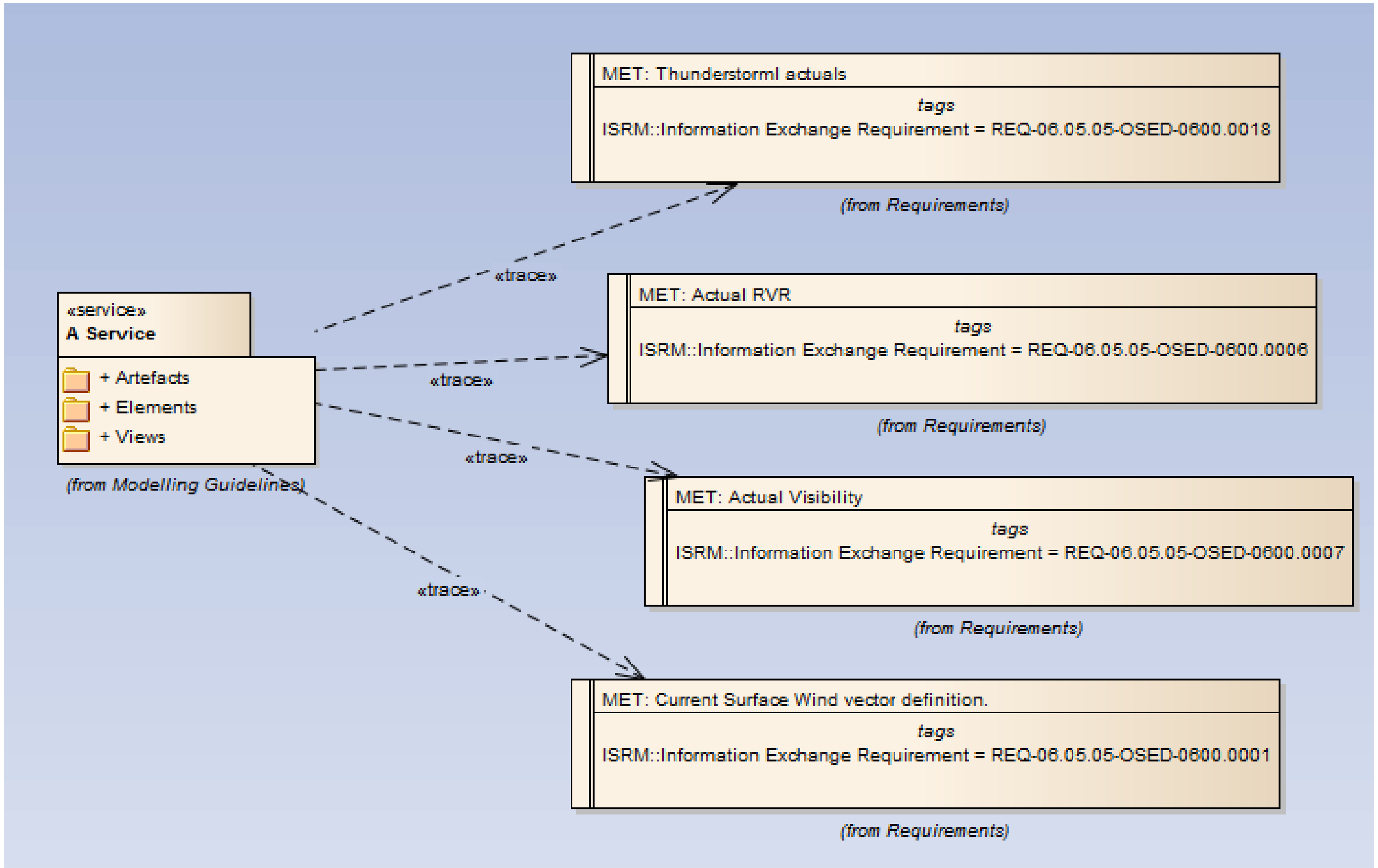
Optional activity

Optional artefact

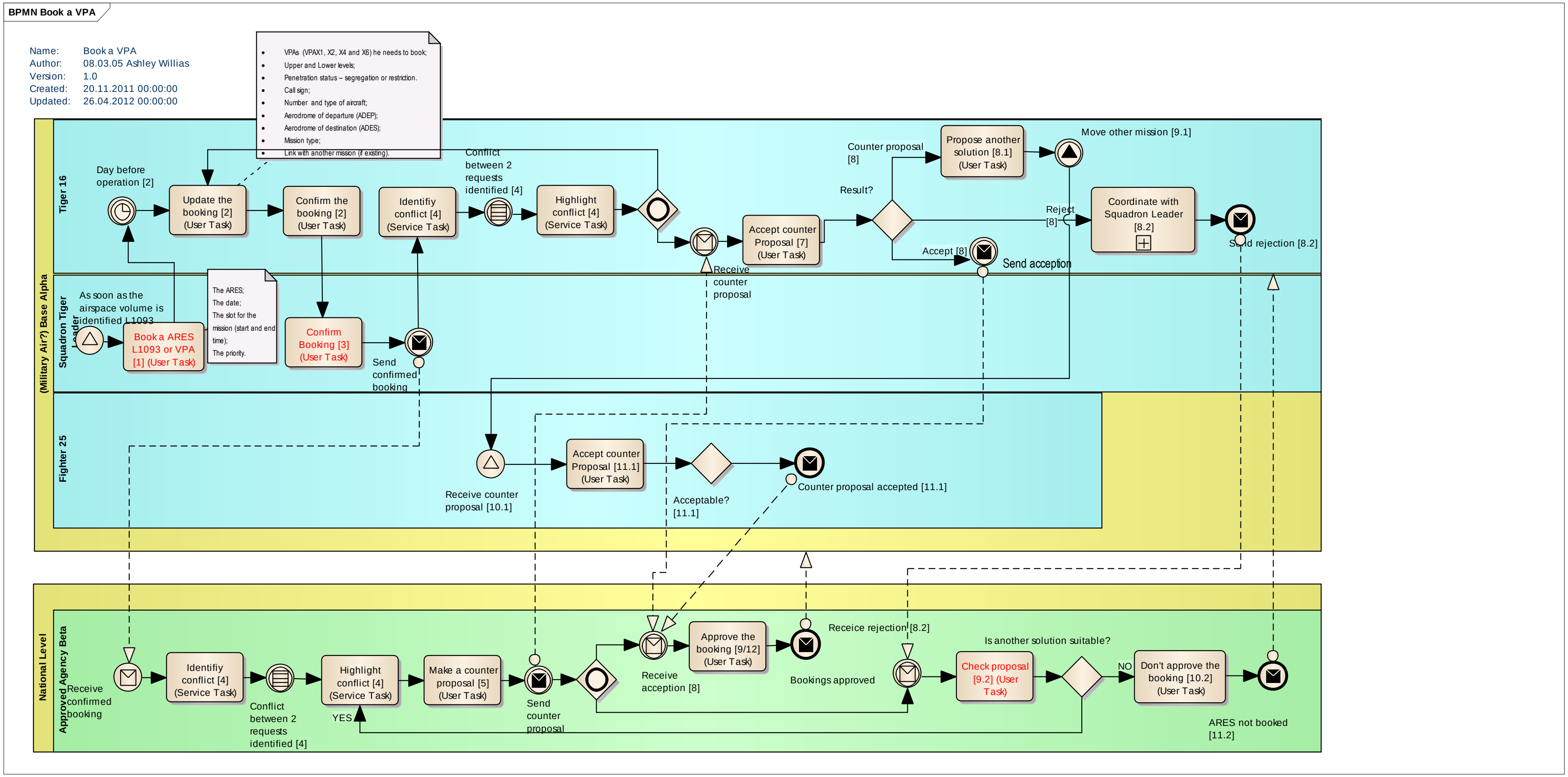
Service Interface Specification characteristics

	Design Abstraction Level	Interoperability Level	Service Provision/ Consumption architecture design	Service Data Message Schema	Communication Technology
Service Design	HIGH LEVEL	LOGICAL	HIGH LEVEL	LOGICAL	N/A
Physical Service Interface Design	DETAILED / TECHNOLOGY-ORIENTED	PHYSICAL	DETAILED	PHYSICAL	IDENTIFIED

Service to requirements mapping


MET: Thunderstorml actuals
tags
ISRM::Information Exchange Requirement = REQ-06.05.05-OSED-0600.0018
(from Requirements)
«trace»
«service»
A Service
+ Artefacts
+ Elements
+ Views
(from Modelling Guidelines)
MET: Actual RVR
tags
ISRM::Information Exchange Requirement = REQ-06.05.05-OSED-0600.0006
(from Requirements)
«trace»
«trace»
MET: Actual Visibility
tags
ISRM::Information Exchange Requirement = REQ-06.05.05-OSED-0600.0007
(from Requirements)
«trace»
MET: Current Surface Wind vector definition.
tags
ISRM::Information Exchange Requirement = REQ-06.05.05-OSED-0600.0001
(from Requirements)

Service to BPMN operational process


BPMN Book a VPA
Name: Book a VPA
Author: 08.03.05 Ashley Willias
Version: 1.0
Created: 20.11.2011 00:00:00
Updated: 26.04.2012 00:00:00
VPAs (VPAX1, X2, X4 and X6) he needs to book;
Upper and Lower levels;
Penetration status – segregation or restriction.
Call sign;
Number and type of aircraft;
Aerodrome of departure (ADEP);
Aerodrome of destination (ADES);
Mission type;
Link with another mission (if existing).
(Military Air?) Base Alpha
Tiger 16
Day before operation [2]
Update the booking [2] (User Task)
Confirm the booking [2] (User Task)
Identifiy conflict [4] (Service Task)
Conflict between 2 requests identified [4]
Highlight conflict [4] (Service Task)
Accept counter Proposal [7] (User Task)
Receive counter proposal
Result?
Counter proposal [8]
Propose another solution [8.1] (User Task)
Move other mission [9.1]
Reject [8]
Coordinate with Squadron Leader [8.2]
Send rejection [8.2]
Accept [8]
Send acception
Squadron Tiger leader
As soon as the airspace volume is identified L1093
Book a ARES L1093 or VPA [1] (User Task)
The ARES;
The date;
The slot for the mission (start and end time);
The priority.
Confirm Booking [3] (User Task)
Send confirmed booking
Fighter 25
Receive counter proposal [10.1]
Accept counter Proposal [11.1] (User Task)
Acceptable? [11.1]
Counter proposal accepted [11.1]
National Level
Approved Agency Beta
Receive confirmed booking
Identifiy conflict [4] (Service Task)
Conflict between 2 requests identified [4]
Highlight conflict [4] (Service Task)
YES
Make a counter proposal [5] (User Task)
Send counter proposal
Receive acception [8]
Approve the booking [9/12] (User Task)
Bookings approved
Receice rejection [8.2]
Is another solution suitable?
Check proposal [9.2] (User Task)
NO
Don't approve the booking [10.2] (User Task)
ARES not booked [11.2]