Service oriented architecture Modeling Language (SoaML)

Tutorial @MOPAS2010, Athens, Greece

17th June 2010

Arne-Jørgen Berre, Dumitru Roman,
Brian Elvesæter, Cyril Carrez

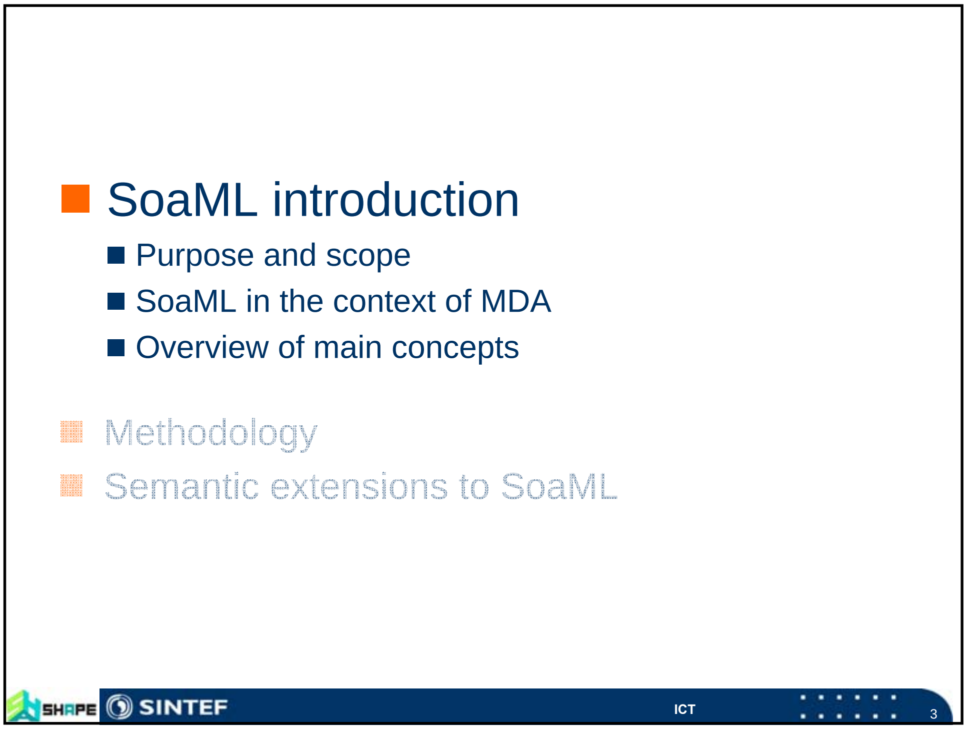
SINTEF ICT

Agenda

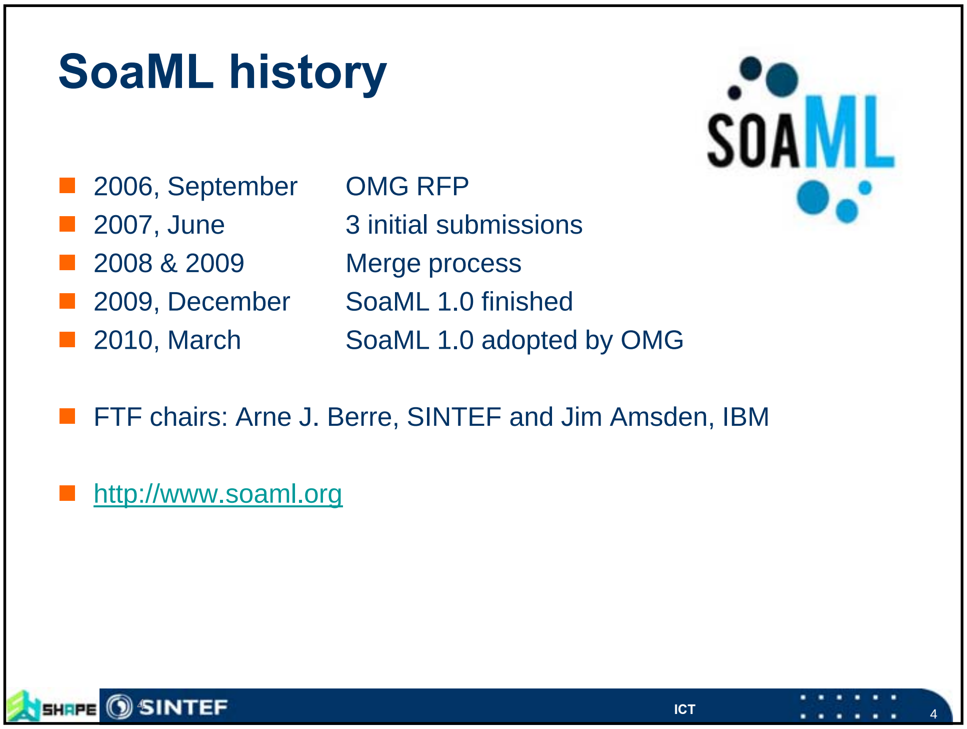
- SoaML introduction
  - Purpose and scope
  - SoaML in the context of MDA
  - Overview of main concepts
    - ServicesArchitecture, ServiceContract, ServiceInterface, ServiceCapability
- Methodology
  - Methodology overview
  - Walkthrough of methodology around a Travel Agency
- Break
- Semantic extensions to SoaML
  - Semantic Web Service: Intro, WSMO/WSML
  - VTA use case and examples
  - Modelling SWS in SoaML

Examples

Dealer Network (SoaML spec)
Travel Agency (SHAPE tutorial) http://www.modeliosoft.fr/tutorials-fr/model-realise-soa-application-fr.html



- SoaML introduction
  - Purpose and scope
  - SoaML in the context of MDA
  - Overview of main concepts
- Methodology
- Semantic extensions to SoaML

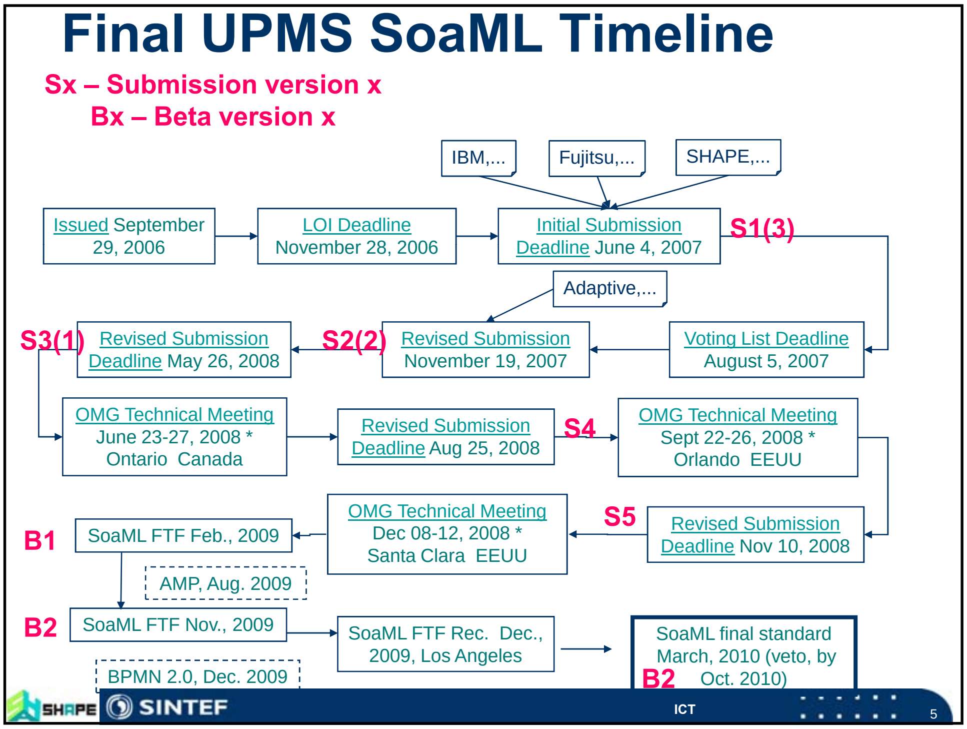


SoaML history

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

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

- http://www.soaml.org



Final 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)

Adaptive,...

Voting List Deadline August 5, 2007

S2(2) Revised Submission November 19, 2007

S3(1) Revised Submission Deadline May 26, 2008

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

S5 Revised Submission Deadline Nov 10, 2008

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

B1 SoaML FTF Feb., 2009

AMP, Aug. 2009

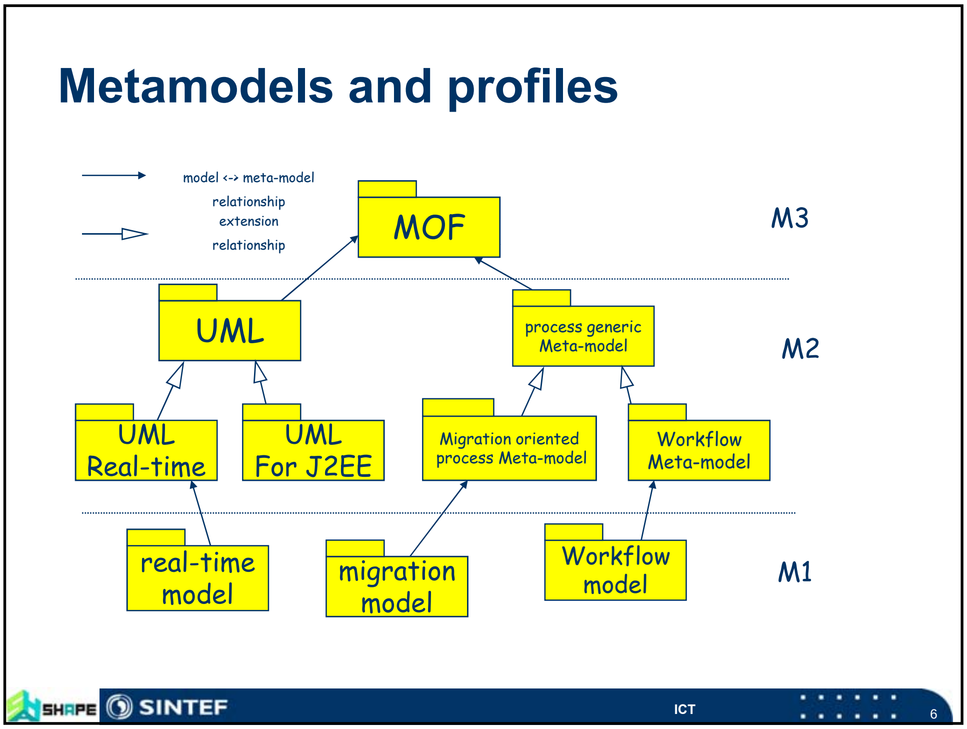
B2 SoaML FTF Nov., 2009

SoaML FTF Rec. Dec., 2009, Los Angeles

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

B2

BPMN 2.0, Dec. 2009



Metamodels and profiles

model <-> meta-model relationship

extension relationship

M3

MOF

M2

UML

process generic Meta-model

UML Real-time

UML For J2EE

Migration oriented process Meta-model

Workflow Meta-model

M1

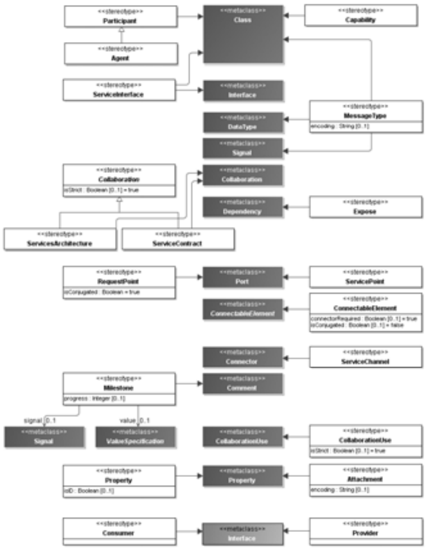
real-time model

migration model

Workflow model

The SoaML Profile

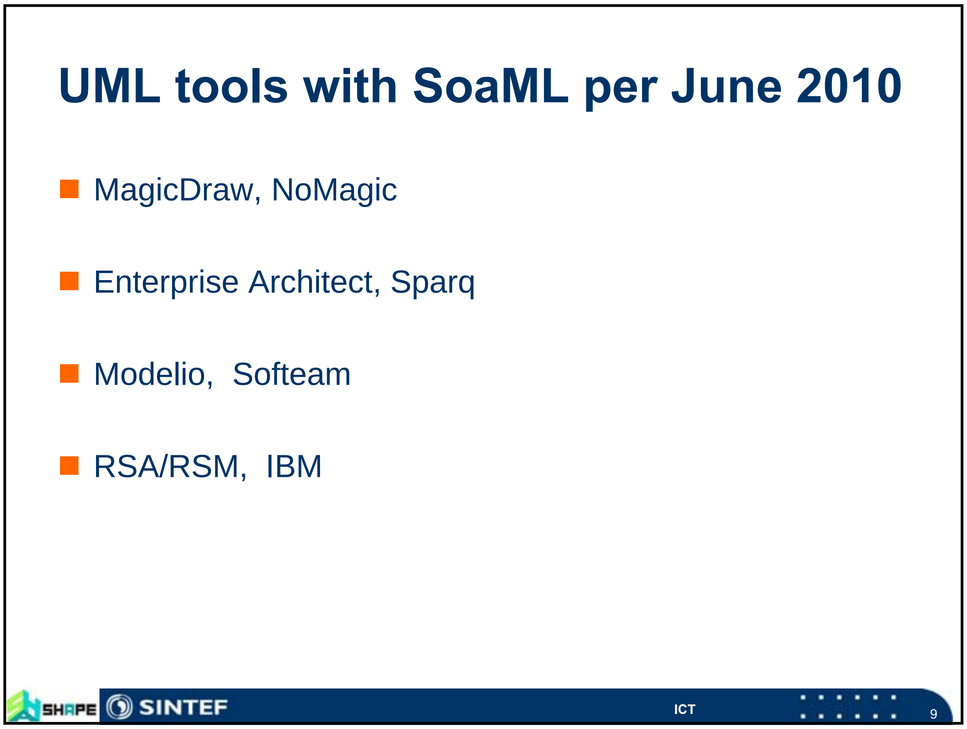
- SoaML is defined as a small set of UML stereotypes.
- These specialize a UML tool for use with SoaML.
- Standard UML can be used as well, as part of a SoaML model.
- Some tools provide enhanced SoaML support.


<<stereotype>> Participant
<<metaclass>> Class
<<stereotype>> Capability
<<stereotype>> Agent
<<stereotype>> ServiceInterface
<<metaclass>> Interface
<<stereotype>> MessageType
encoding : String [0..1]
<<metaclass>> DataType
<<metaclass>> Signal
<<stereotype>> Collaboration
isStrict : Boolean [0..1] = true
<<metaclass>> Collaboration
<<metaclass>> Dependency
<<stereotype>> Expose
<<stereotype>> ServicesArchitecture
<<stereotype>> ServiceContract
<<stereotype>> RequestPoint
isConjugated : Boolean = true
<<metaclass>> Port
<<stereotype>> ServicePoint
<<metaclass>> ConnectableElement
<<stereotype>> ConnectableElement
connectorRequired : Boolean [0..1] = true
isConjugated : Boolean [0..1] = false
<<metaclass>> Connector
<<stereotype>> ServiceChannel
<<stereotype>> Milestone
progress : Integer [0..1]
<<metaclass>> Comment
signal 0..1
value 0..1
<<metaclass>> Signal
<<metaclass>> ValueSpecification
<<metaclass>> CollaborationUse
<<stereotype>> CollaborationUse
isStrict : Boolean [0..1] = true
<<stereotype>> Property
isID : Boolean [0..1]
<<metaclass>> Property
<<stereotype>> Attachment
encoding : String [0..1]
<<stereotype>> Consumer
<<metaclass>> Interface
<<stereotype>> Provider

Current SoaML Support

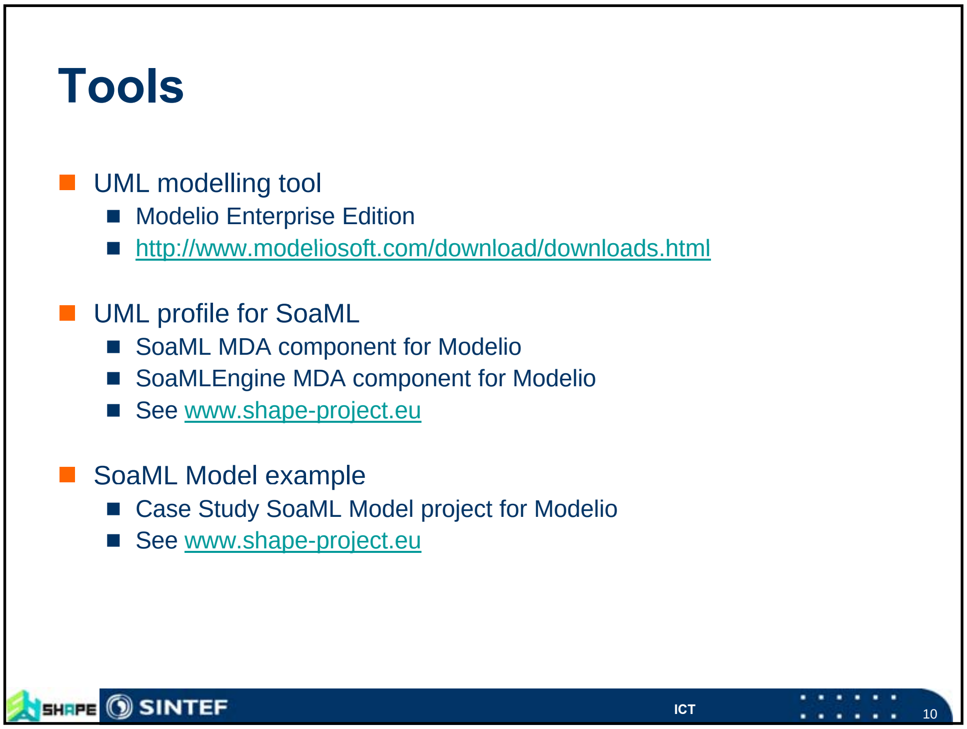
- OMG Web site
  - SoaML Wiki: http://www.SoaML.org
  - Specification: http://www.omgwiki.org/SoaML/doku.php?id=specification
- Known SoaML Tooling
  - Cameo SOA+ (NoMagic) UML with SoaML Modeling and Provisioning
  - ModelPro (ModelDriven.org) Open Source MDA provisioning for SoaML
  - Enterprise Architect (Sparx) SoaML Profile for UML tool
  - Objecteering (Softeam) SoaML Profile for UML Tool
  - RSA (IBM) UML tool with SoaML & code generation [Not yet released]

May 2009



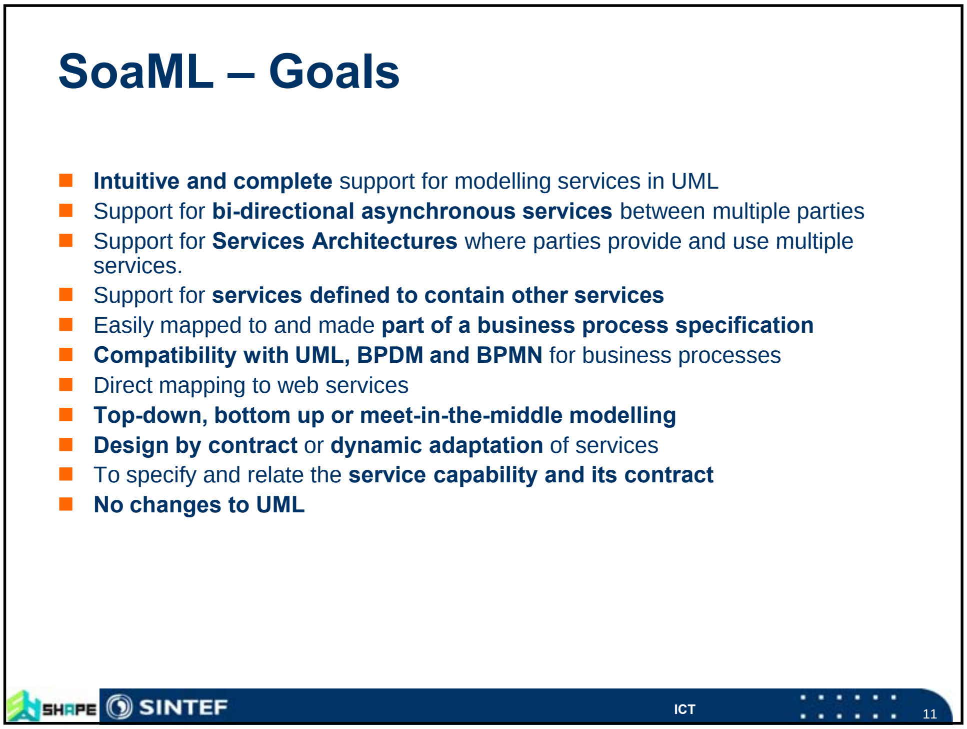
UML tools with SoaML per June 2010

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



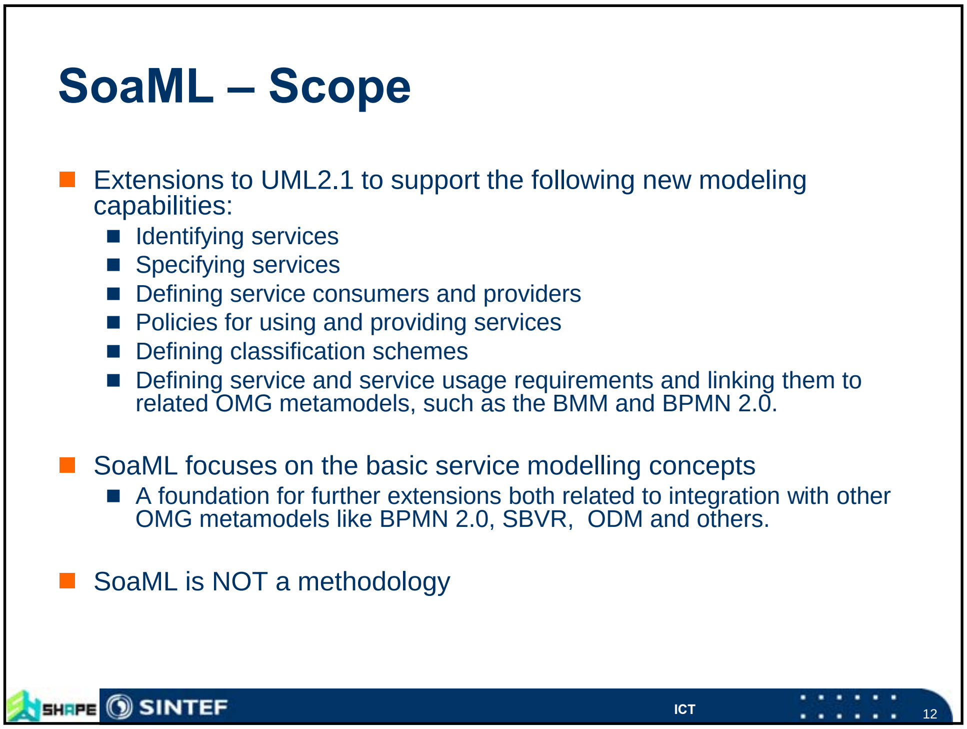
Tools

- UML modelling tool
  - Modelio Enterprise Edition
  - http://www.modeliosoft.com/download/downloads.html
- UML profile for SoaML
  - SoaML MDA component for Modelio
  - SoaMLEngine MDA component for Modelio
  - See www.shape-project.eu
- SoaML Model example
  - Case Study SoaML Model project for Modelio
  - See www.shape-project.eu



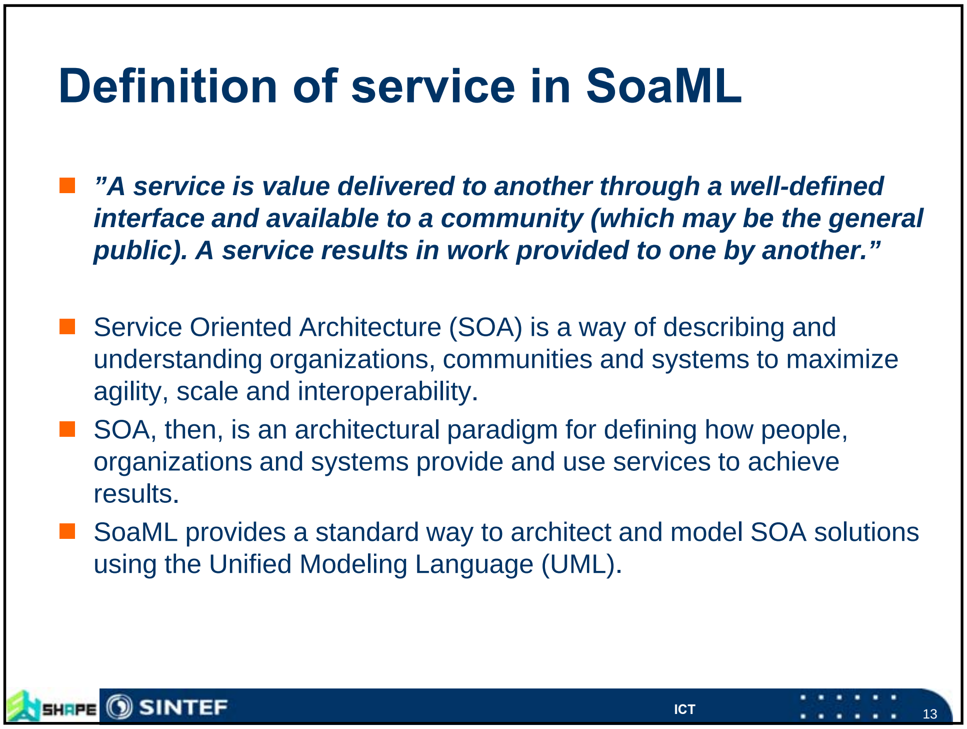
SoaML – Goals

- **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

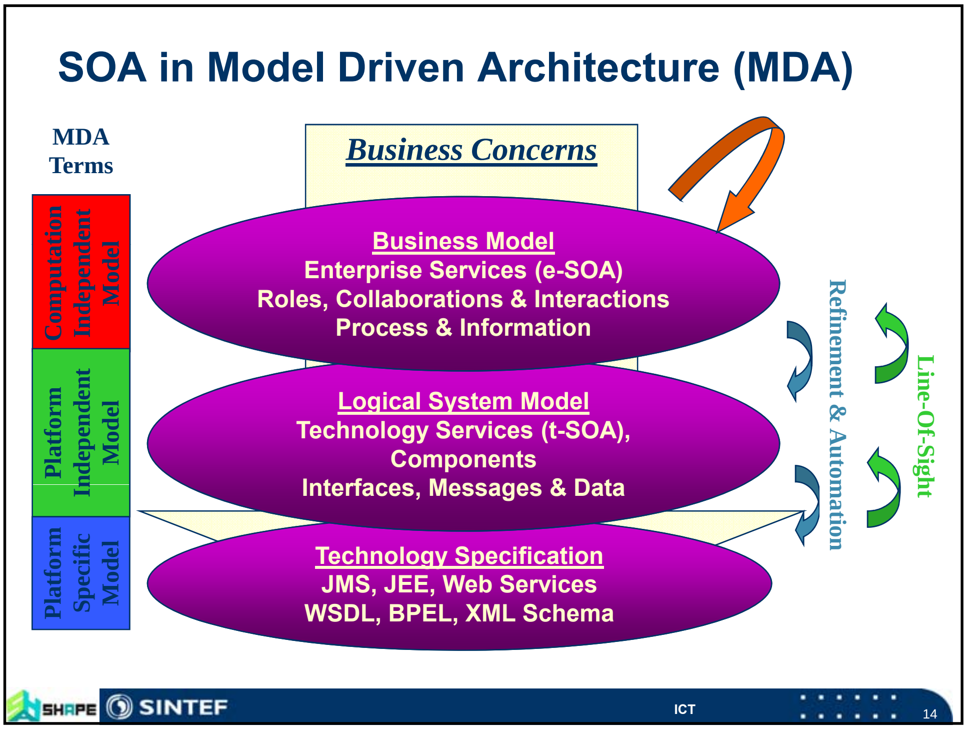
- Extensions to UML2.1 to support the following new modeling capabilities:
  - Identifying services
  - Specifying services
  - Defining service consumers and providers
  - Policies for using and providing services
  - Defining classification schemes
  - Defining service and service usage requirements 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, 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).

SHAPE SINTEF ICT 13



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

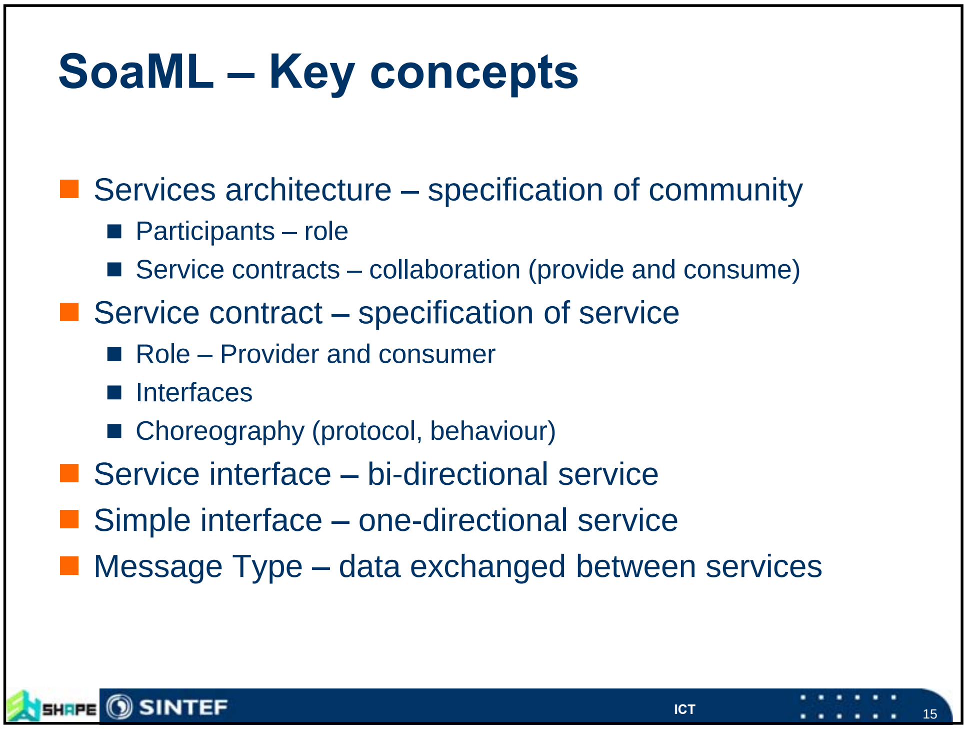
Logical System Model
Technology Services (t-SOA),
Components
Interfaces, Messages & Data

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

Refinement & Automation

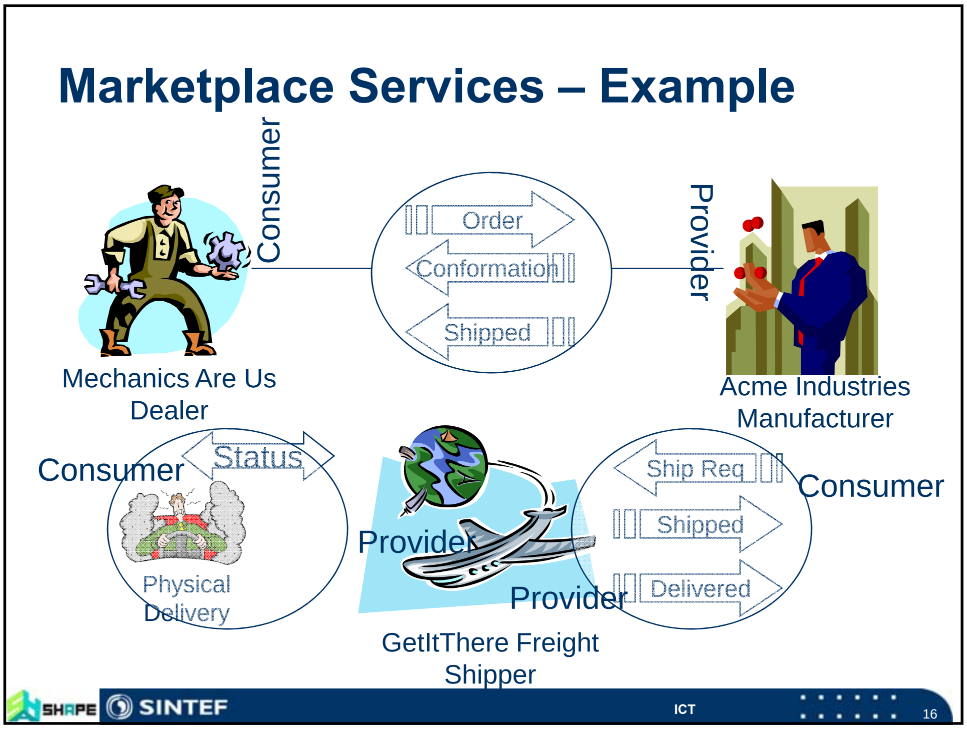
Line-Of-Sight

SHAPE SINTEF ICT 14



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

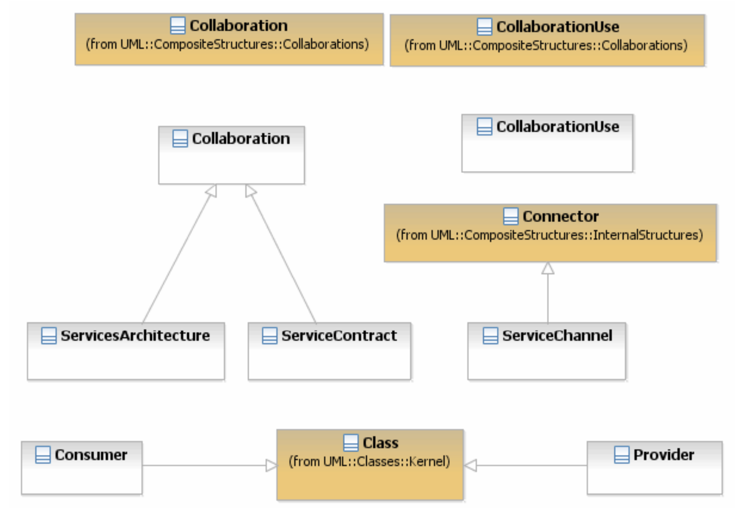
Shipped

Delivered

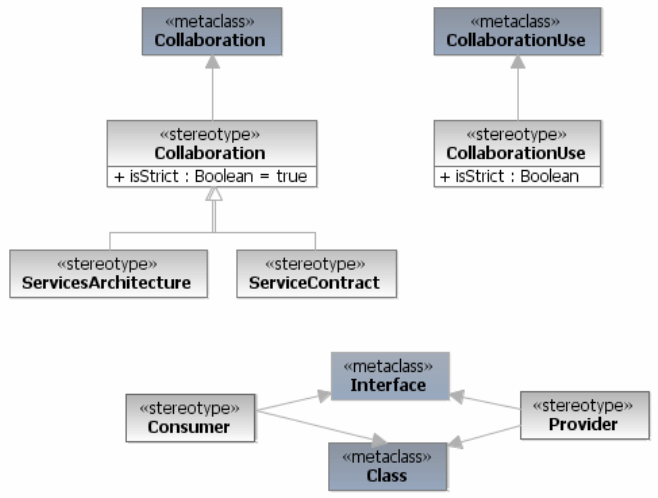
Consumer

GetItThere Freight
Shipper

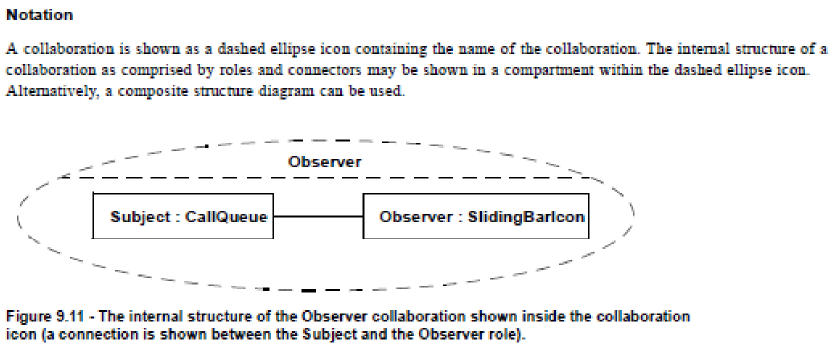
ServiceContracts and ServiceArchitectures Metamodel


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



Collaboration



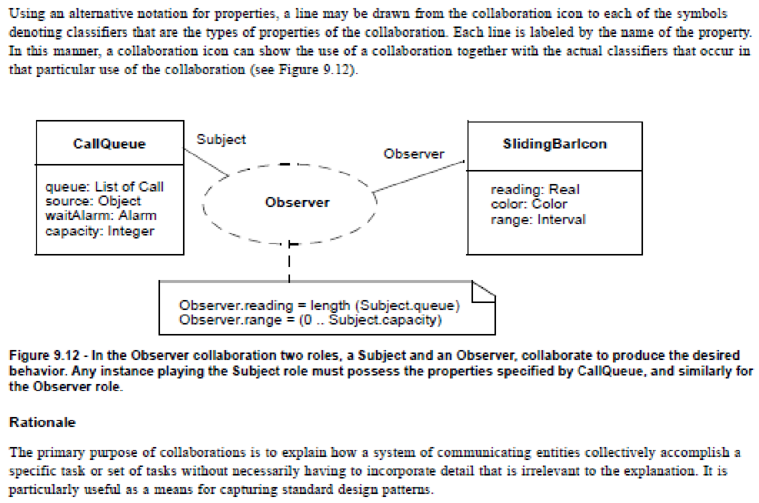
Notation

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).

Start - Explanation of standard UML 2.3

Collaboration



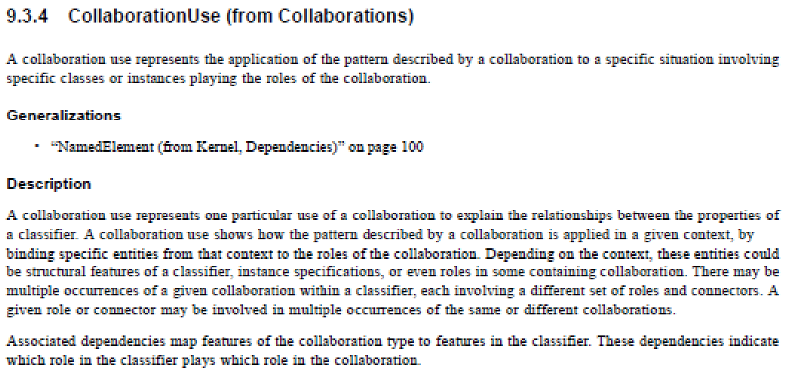
Using an alternative notation for properties, a line may be drawn from the collaboration icon to each of the symbols denoting classifiers that are the types of properties of the collaboration. Each line is labeled by the name of the property. In this manner, a collaboration icon can show the use of a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

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

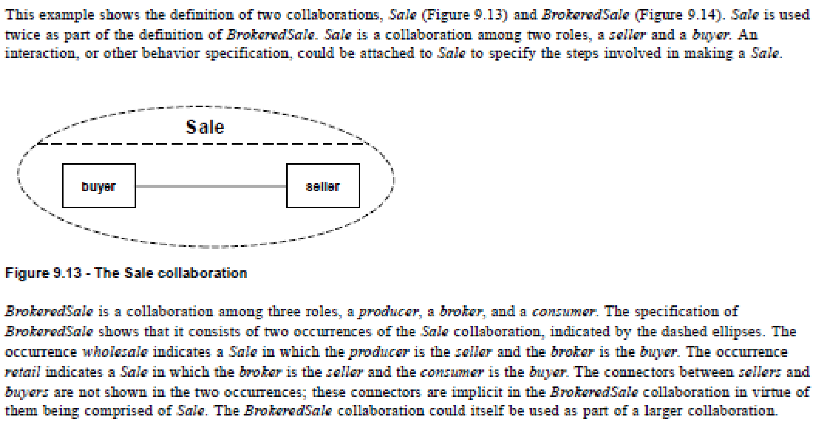
- "NamedElement (from Kernel, Dependencies)" on page 100

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

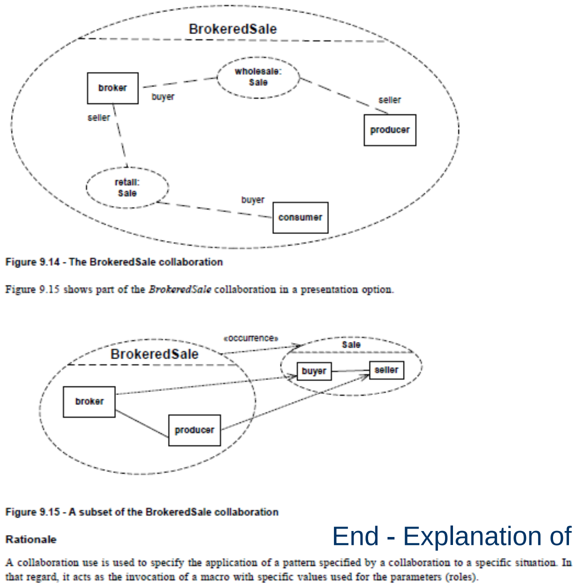

BrokeredSale
broker
buyer
wholesale:
Sale
seller
producer
seller
retail:
Sale
buyer
consumer

Figure 9.14 - The BrokeredSale collaboration

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

«occurrence»
Sale
BrokeredSale
buyer
seller
broker
producer

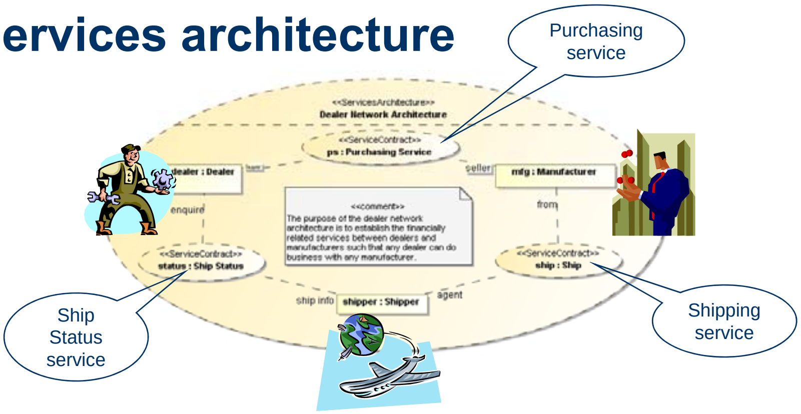
Figure 9.15 - A subset of the BrokeredSale collaboration

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).

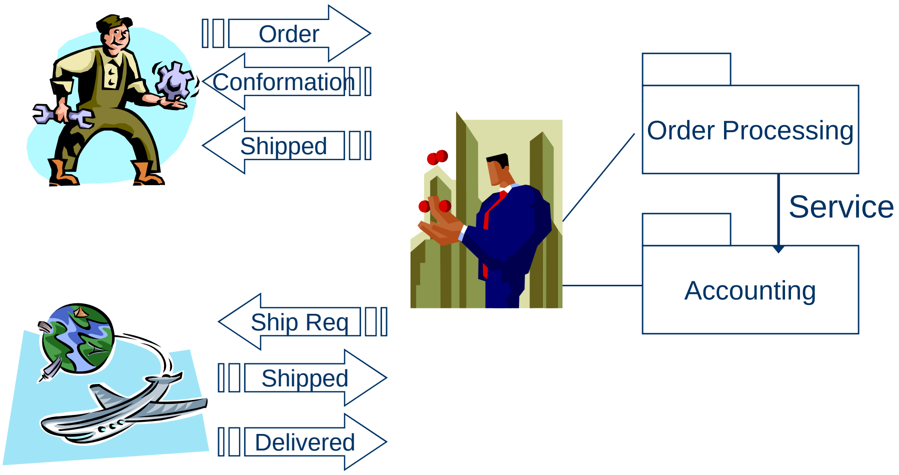
End - Explanation of standard UML 2.3

Services architecture



- 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

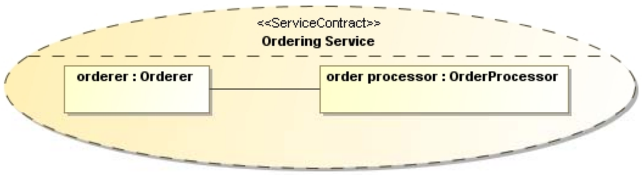
Accounting

Ship Req

Shipped

Delivered

Service contract



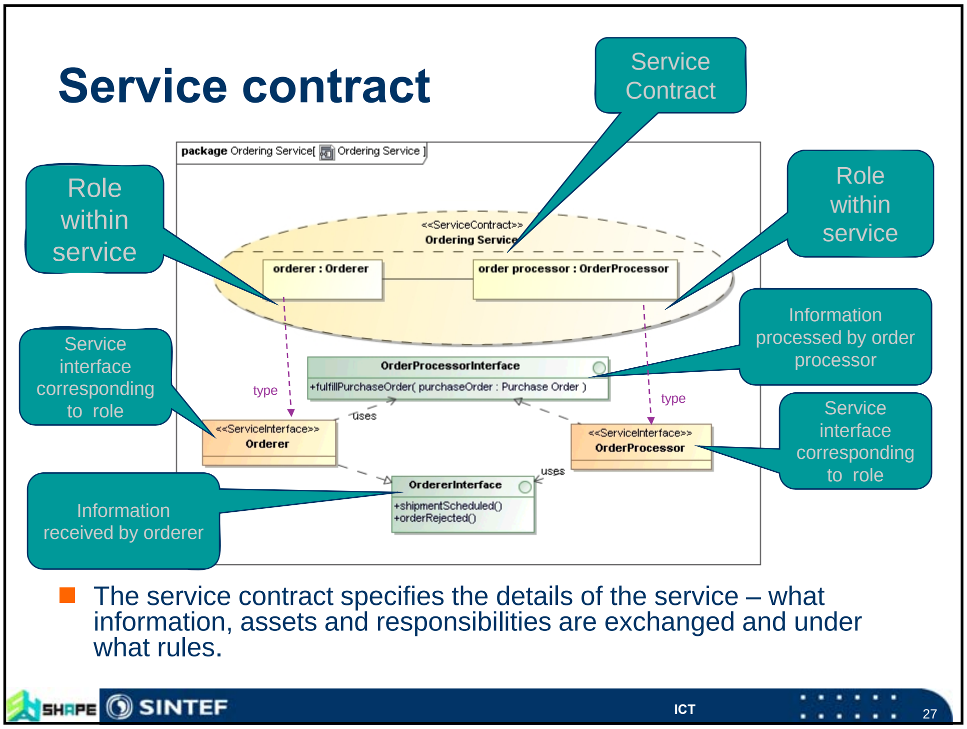
<<ServiceContract>>

Ordering Service

orderer : Orderer

order processor : OrderProcessor

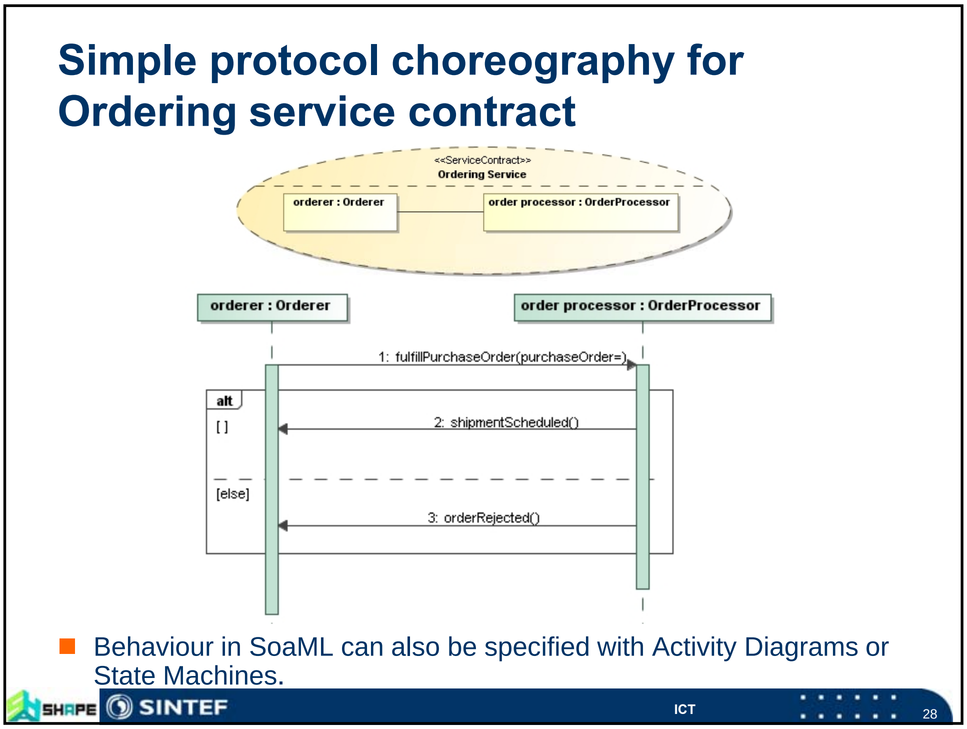
- A ServiceContract:
  - Fully specifies the service (terms, conditions, interfaces, choreography, etc.)
  - is binding on *both* the providers and consumers of that service.
  - is defined using a UML collaboration that is focused on the interactions involved in providing a service.
- A participant 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.



Service contract

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

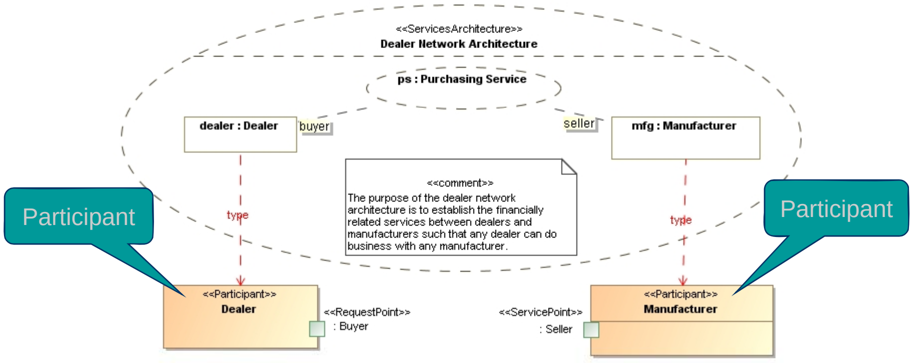
- The service contract specifies the details of the service – what information, assets and responsibilities are exchanged and under what rules.



Simple protocol choreography for Ordering service contract

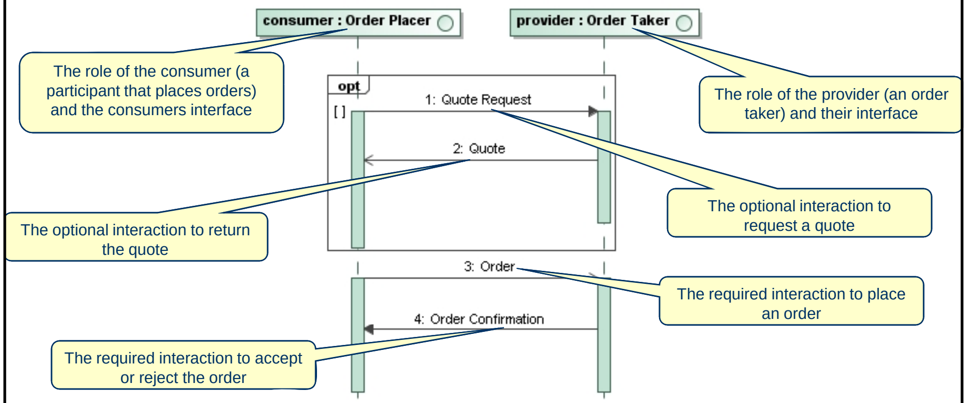
- Behaviour in SoaML can also be specified with Activity Diagrams or State Machines.

Participants



- 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

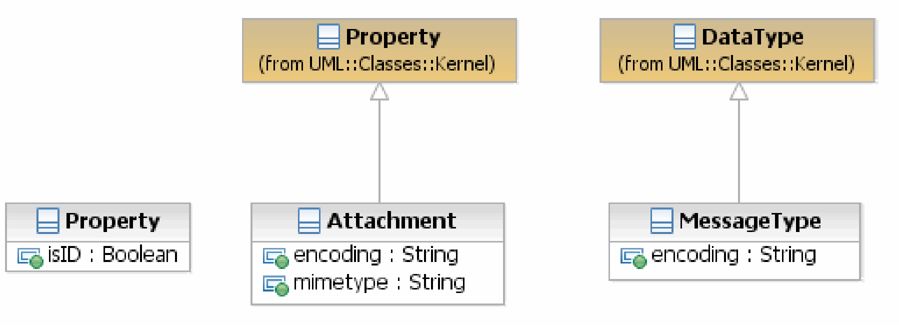
Service Choreography for “Place Order”



consumer : Order Placer
provider : Order Taker
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
opt
[ ]
1: Quote Request
2: Quote
The optional interaction to request a quote
The optional interaction to return the quote
3: Order
The required interaction to place an order
4: Order Confirmation
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



Property
(from UML::Classes::Kernel)

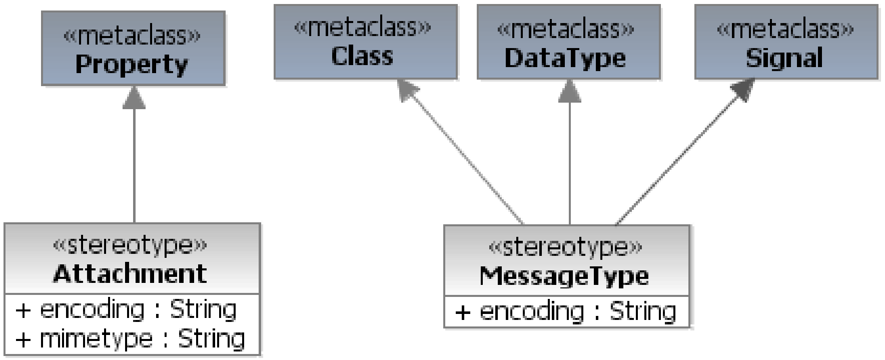
DataType
(from UML::Classes::Kernel)

Property
isID : Boolean

Attachment
encoding : String
mimetype : String

MessageType
encoding : String

Service Data Profile



«metaclass»
Property

«metaclass»
Class

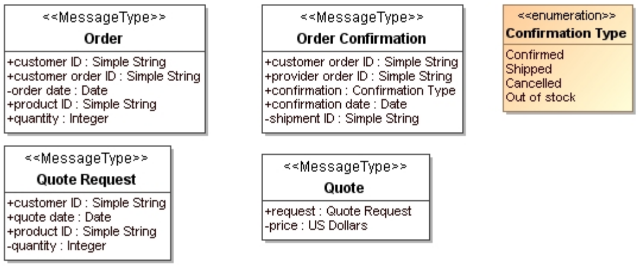
«metaclass»
DataType

«metaclass»
Signal

«stereotype»
Attachment
+ encoding : String
+ mimetype : String

«stereotype»
MessageType
+ encoding : String

Message Detail for Place Order

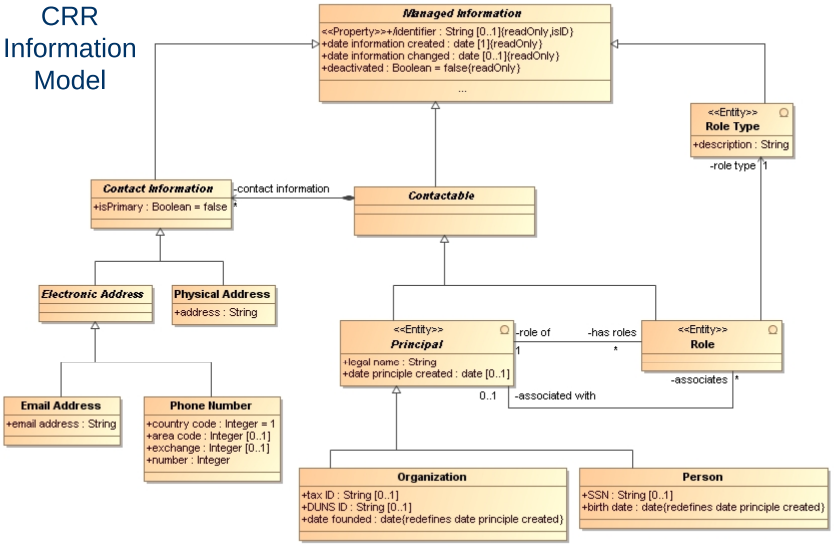

<<MessageType>>
Order
+customer ID : Simple String
+customer order ID : Simple String
-order date : Date
+product ID : Simple String
+quantity : Integer
<<MessageType>>
Order Confirmation
+customer order ID : Simple String
+provider order ID : Simple String
+confirmation : Confirmation Type
+confirmation 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}

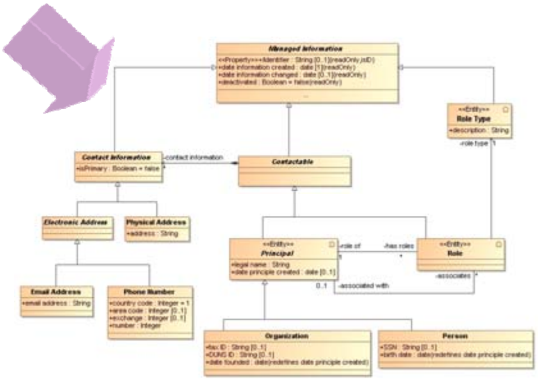
Linking messages to business information

<<MessageType>>
Find Principals
-type : String [0..1]
-any key : String [0..1]
-name : String [0..1]
-role type : String [0..1]
...

<<MessageType>>
Principal Detail
-principal : Principal

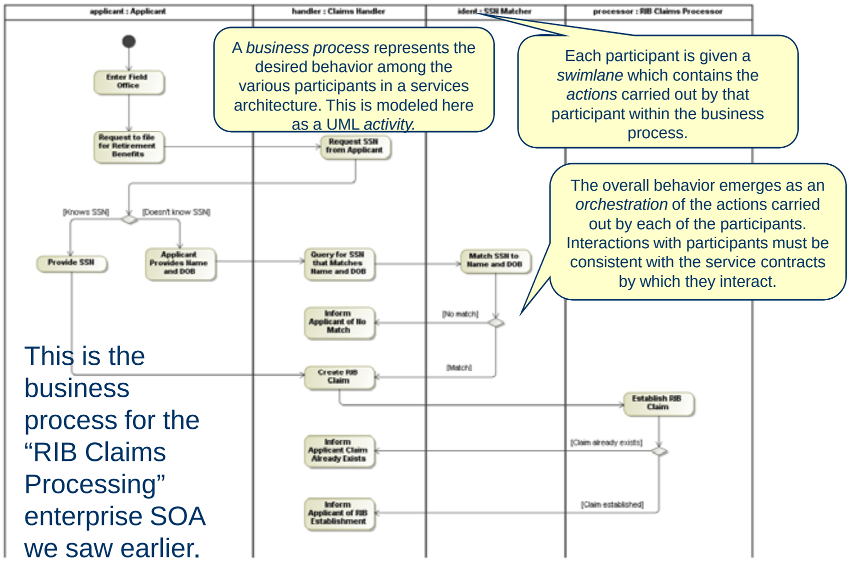
<<MessageType>>
Principal List
-principal list : Principal Detail [*]

<<MessageType>>
Role Types
-role types : Role Type [*]


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
-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}

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

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
Inform Applicant of No Match
[No match]
Create RIB Claim
[Match]
Establish RIB Claim
Inform Applicant Claim Already Exists
[Claim already exists]
Inform Applicant of RIB Establishment
[Claim established]

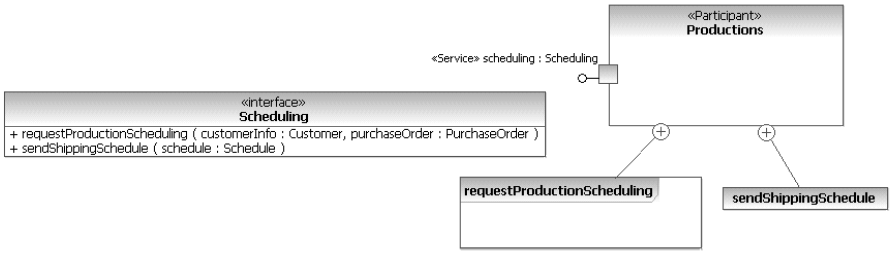
A *business process* represents the desired behavior among the various participants in a services architecture. This is modeled here as a UML *activity.*

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.

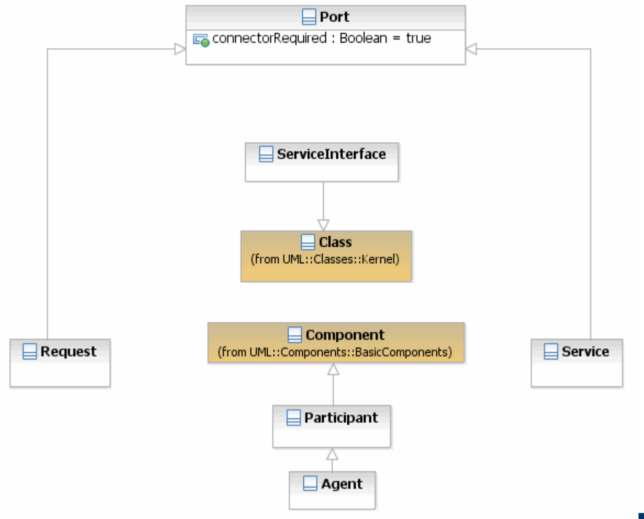
This is the business process for the “RIB Claims Processing” enterprise SOA we saw earlier.

Service ports and service participants

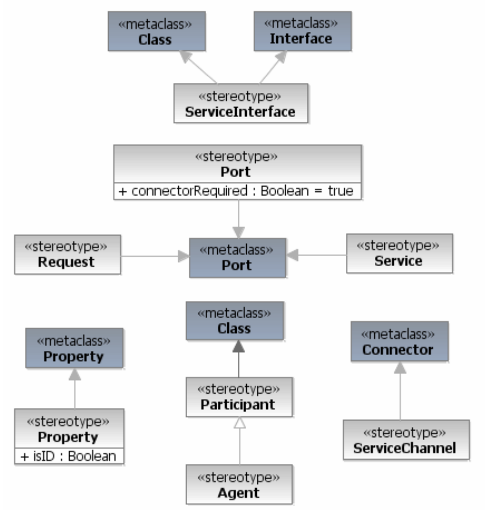

«Participant»
Productions
«Service» scheduling : Scheduling
«interface»
Scheduling
+ requestProductionScheduling (customerInfo : Customer, purchaseOrder : PurchaseOrder)
+ sendShippingSchedule (schedule : Schedule)
requestProductionScheduling
sendShippingSchedule

- 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)
Component
(from UML::Components::BasicComponents)
Request
Service
Participant
Agent

ServiceInterfaces and Participants Profile

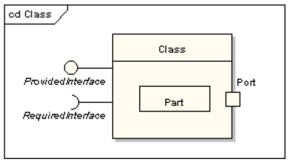


UML Composite Diagrams

- 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

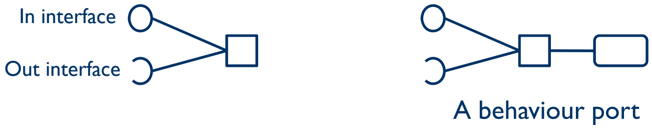
- A part is an element that represents a set of one or more instances which are owned by a containing classifier instance. So for example, if a diagram instance owned a set of graphical elements, then the graphical elements could be represented as parts; if it were useful to do so, to model some kind 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.

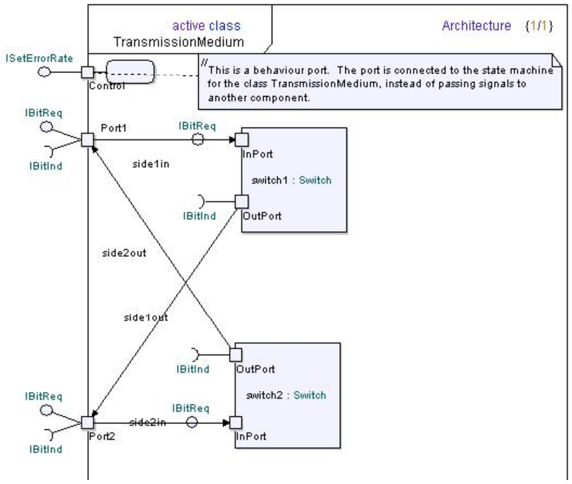
- Two types of ports:
  - Connected to internal communication channels (by default).
  - Connected to the state machine for the class instance (a behaviour port).



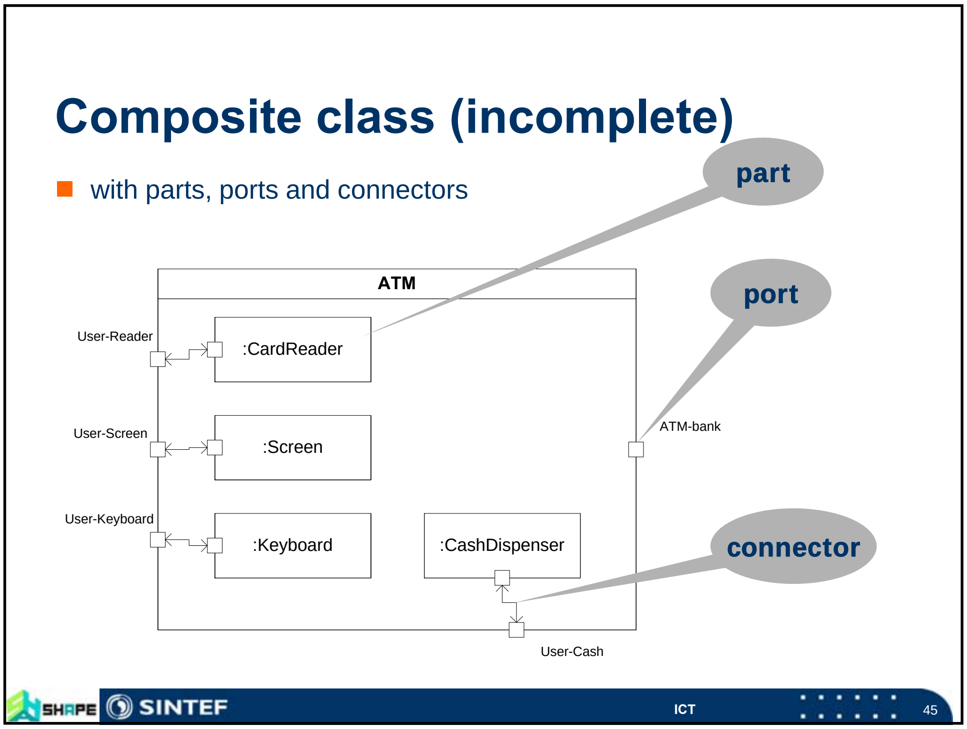
Composite Structure

- 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).

Composite Structure

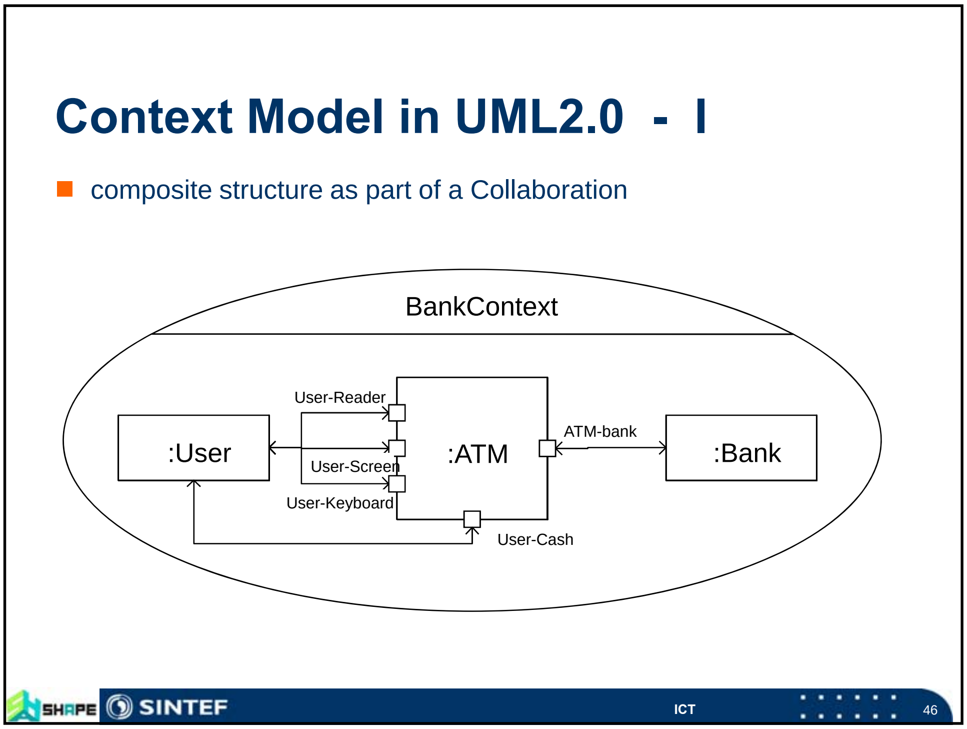


// This is a behaviour port. The port is connected to the state machine for the class TransmissionMedium, instead of passing signals to another component.



Composite class (incomplete)

- with parts, ports and connectors

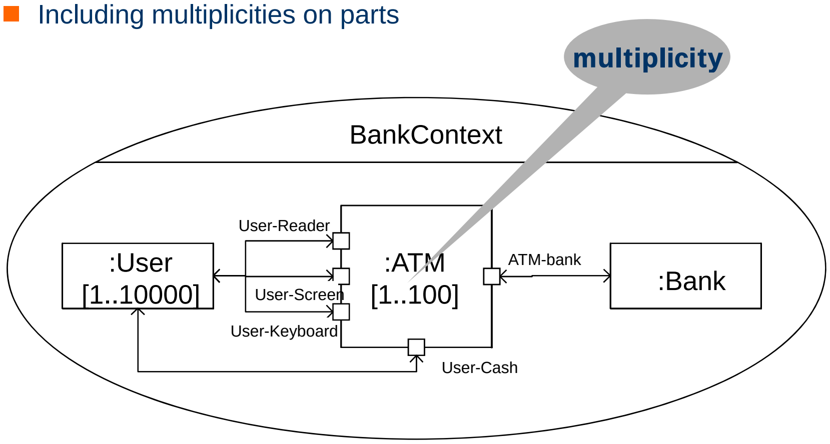


Context Model in UML2.0 - I

- composite structure as part of a Collaboration

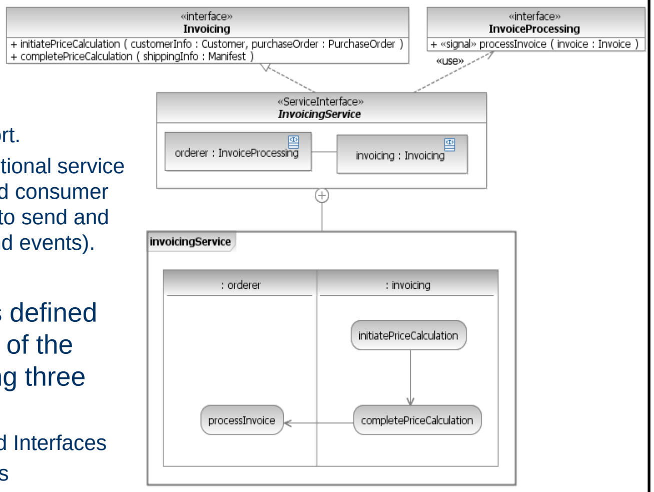
Context Model in UML2.0 - II

- Including multiplicities on parts



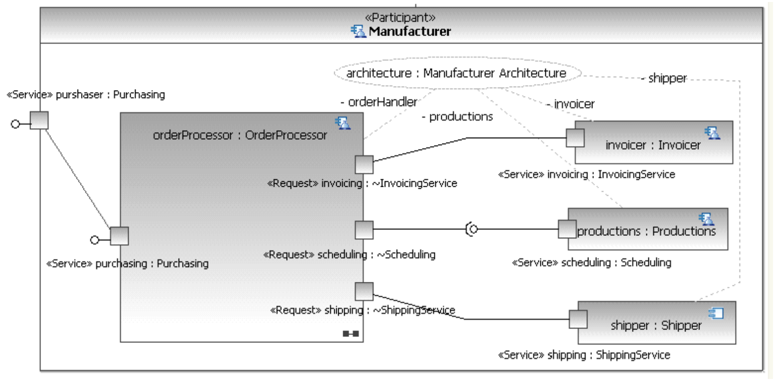
End - Explanation of standard UML 2.3

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

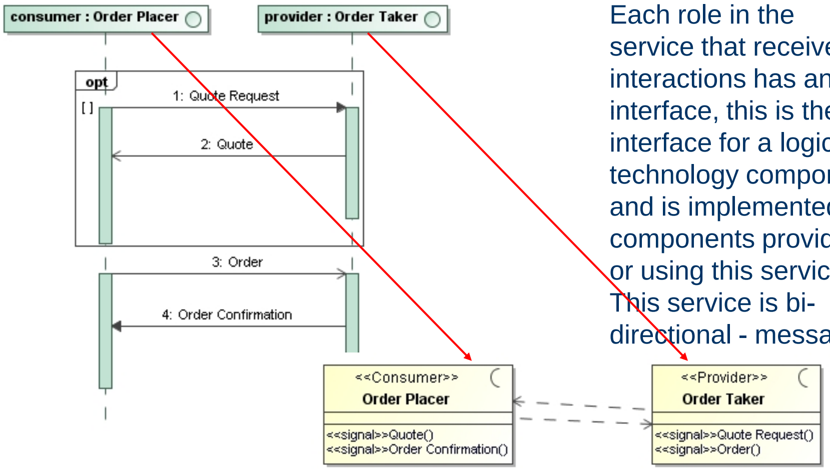
- A ServiceInterface:
  - can type a service port.
  - can specify a bi-directional service (both the provider and consumer have responsibilities to send and receive messages and events).
- A ServiceInterface is defined from the perspective of the service provider using three primary sections:
  - provided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

Participant with service and request ports


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

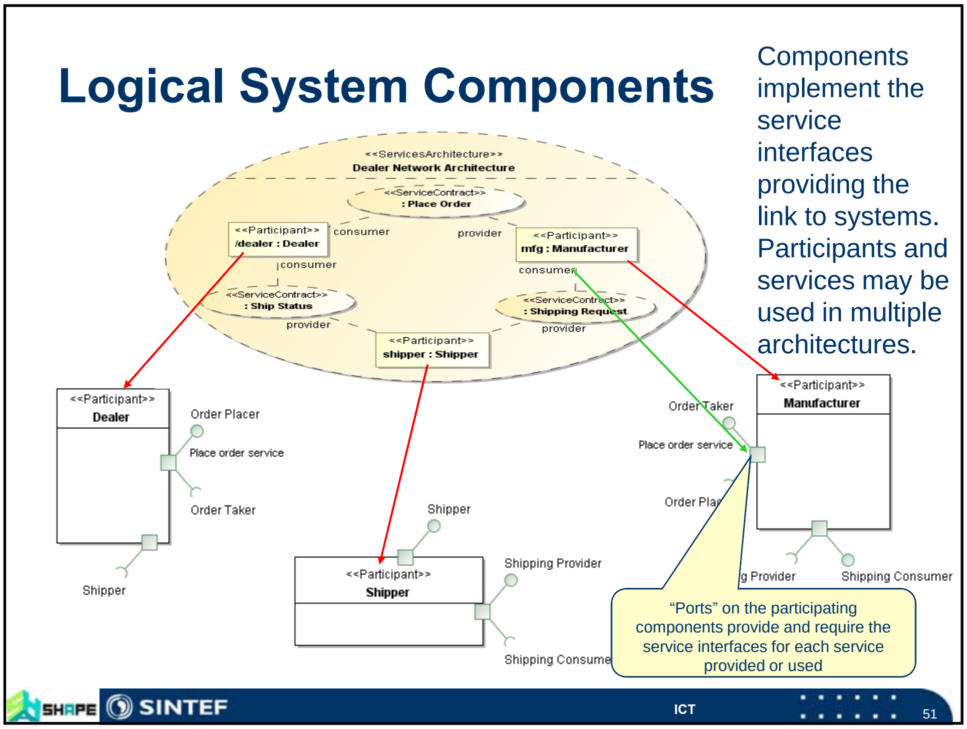
- A Service Port is typed by a ServiceInterface
- A Request port is typed by a conjugate ServiceInterface (defines the use of a service rather than its provision). This will allow us to connect service providers and consumers in a Participant.
- *Can be transformed to the appropriate interface/implementation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

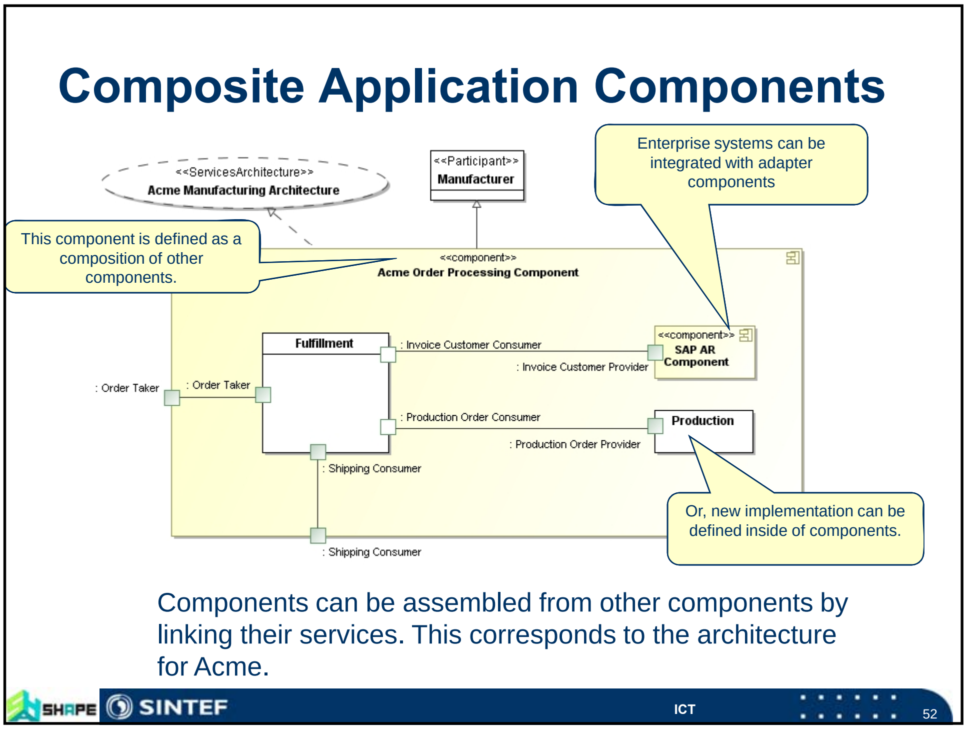
Interfaces will correspond with parts of WSDL in a web services mapping of SoaML



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



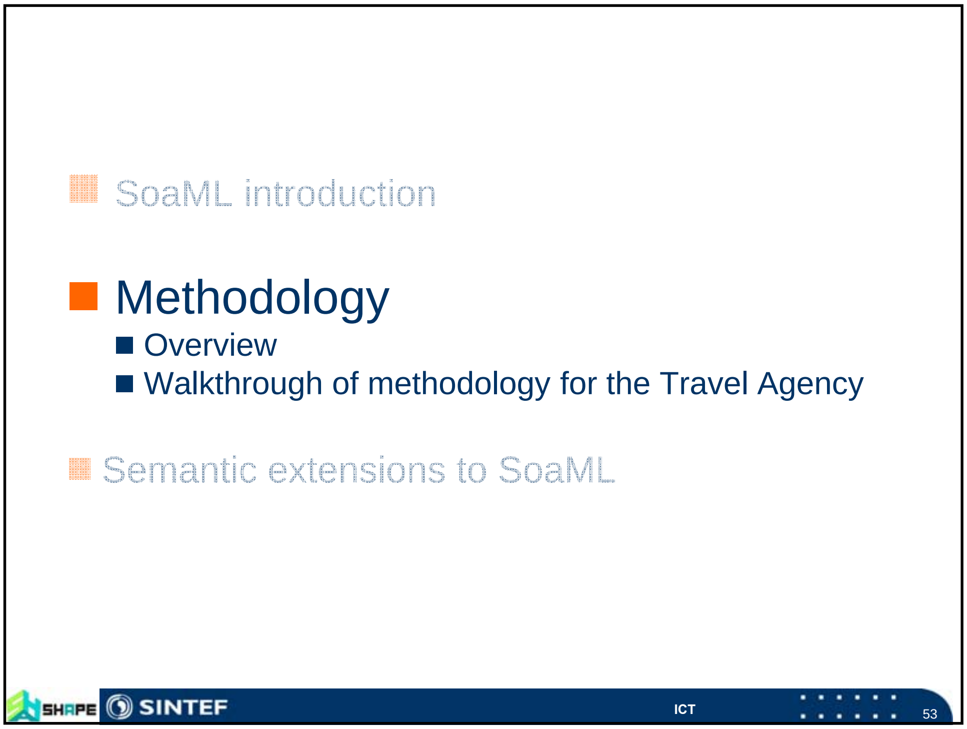
Composite Application Components

This component is defined as a composition of other components.

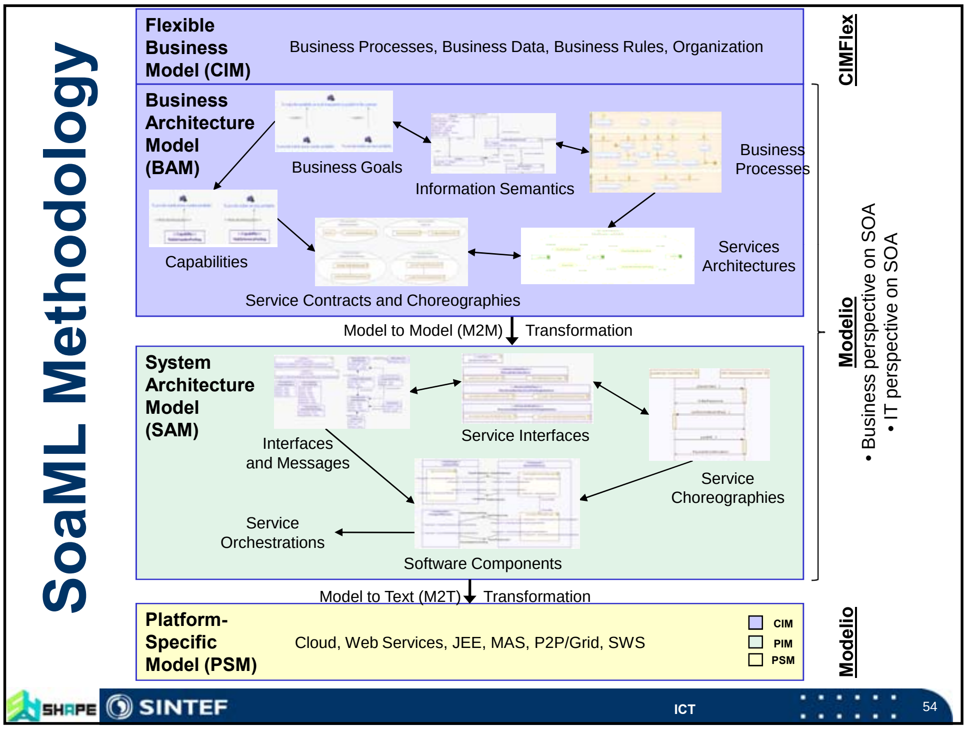
Enterprise systems can be integrated with adapter components

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.



- SoaML introduction
- Methodology
  - Overview
  - Walkthrough of methodology for the Travel Agency
- Semantic extensions to SoaML



SoaML Methodology

Flexible Business Model (CIM) Business Processes, Business Data, Business Rules, Organization

Business Architecture Model (BAM)

Business Goals

Information Semantics

Business Processes

Capabilities

Service Contracts and Choreographies

Services Architectures

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

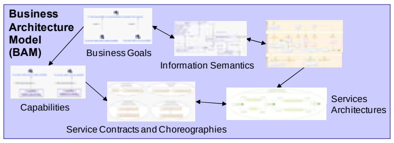
□ CIM
□ PIM
□ PSM

CIMFlex

Modelio
- Business perspective on SOA
- IT perspective on SOA

Modelio

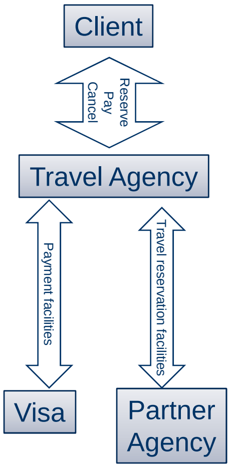
Business Architecture Model (BAM)

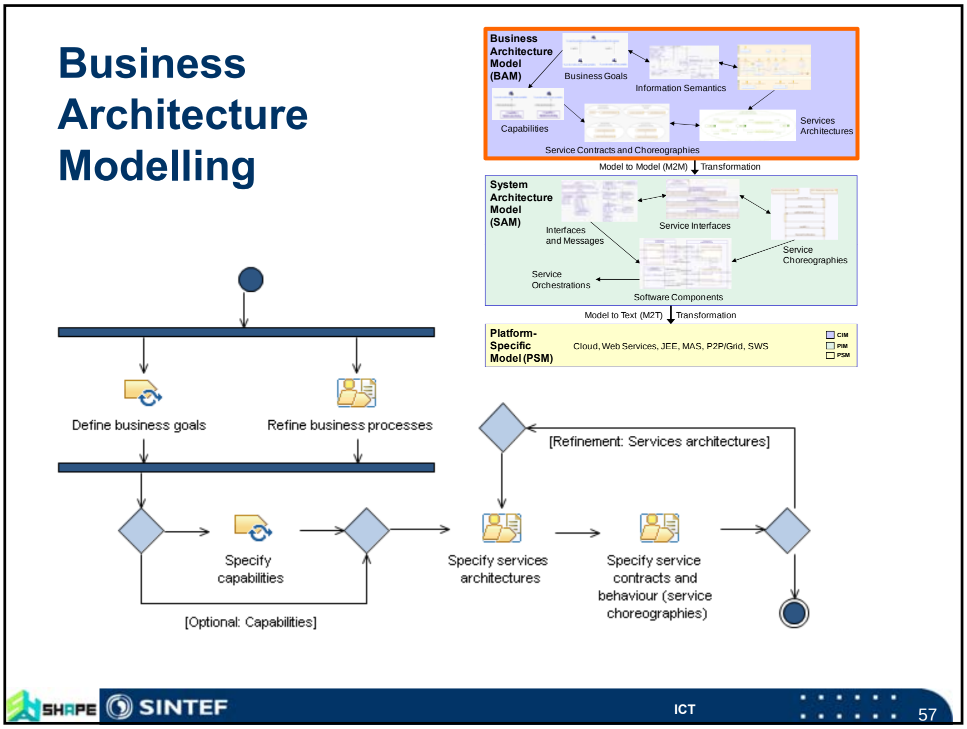

Business Architecture Model (BAM)
Business Goals
Information Semantics
Capabilities
Services Architectures
Service Contracts and Choreographies

- The BAM expresses the business operations and the environment which the service-oriented architecture is to support
  - Represents the business perspective of an SOA
- The BAM captures business requirements and identifies services:
  - Business goals
  - Information semantics (terminologies, ontologies)
  - Business processes with associated organisation roles and information elements
  - Capabilities
- The BAM further describes:
  - The ***services architecture*** of the business community
  - The ***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:
    - Flight, Flight+Hotel, Flight+Hotel+Car, etc.
  - Cancel a Travel
  - Pay the Travel
- The Travel Agency need to be in contact with a Visa payment center in order to be paid by the Client, and pay back the Partner Agencies.





Business Architecture Modelling

Business Architecture Model (BAM)

Business Goals

Information Semantics

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

Define business goals

Refine business processes

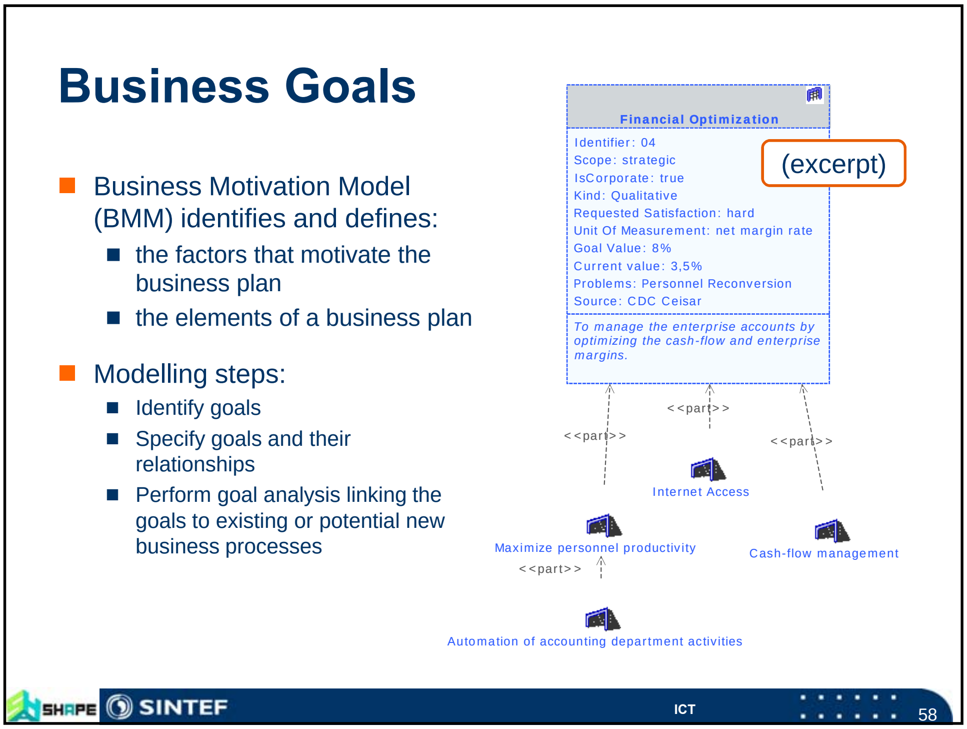
Specify capabilities

[Optional: Capabilities]

Specify services architectures

Specify service contracts and behaviour (service choreographies)

[Refinement: Services architectures]



Business Goals

- Business Motivation Model (BMM) identifies and defines:
  - the factors that motivate the business plan
  - the 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

Financial Optimization

Identifier: 04
Scope: strategic
IsCorporate: true
Kind: Qualitative
Requested Satisfaction: hard
Unit Of Measurement: net margin rate
Goal Value: 8%
Current value: 3,5%
Problems: Personnel Reconversion
Source: CDC Ceisar

To manage the enterprise accounts by optimizing the cash-flow and enterprise margins.

(excerpt)

<<part>>

<<part>>

<<part>>

Internet Access

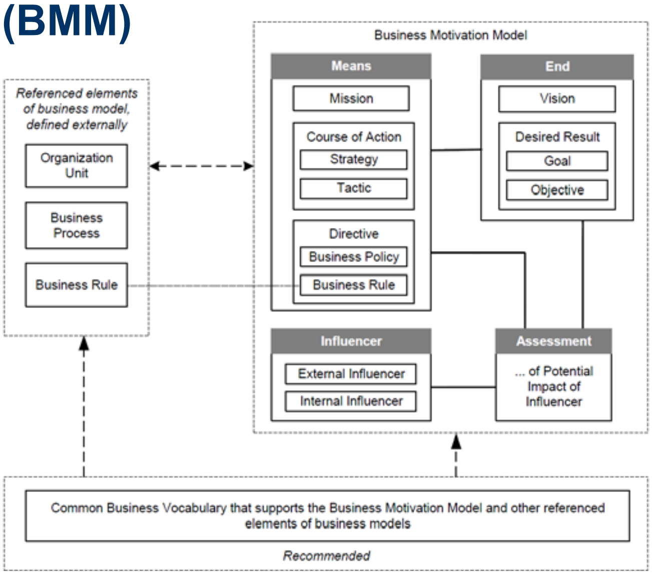
Maximize personnel productivity

Cash-flow management

<<part>>

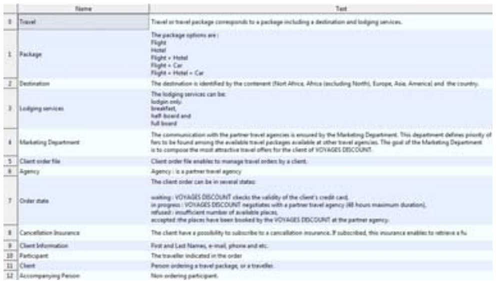
Automation of accounting department activities

Business Motivation Model (BMM)

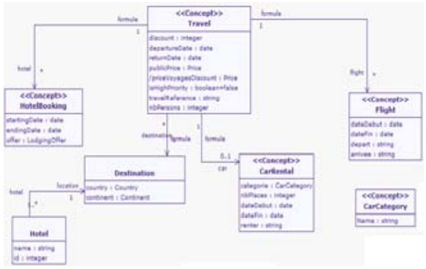

Business Motivation Model
Means
Mission
Course of Action
Strategy
Tactic
Directive
Business Policy
Business Rule
End
Vision
Desired Result
Goal
Objective
Influencer
External Influencer
Internal Influencer
Assessment
... of Potential Impact of Influencer
Referenced elements of business model, defined externally
Organization Unit
Business Process
Business Rule
Common Business Vocabulary that supports the Business Motivation Model and other referenced elements of business models
Recommended

Information Semantics

- Captures core terminology in dictionaries and explicitly specifies the relations between terms.
- Modelling steps:
  - Capture terms in dictionaries
  - Model explicitly terms and the relations between them (potentially using ontologies)



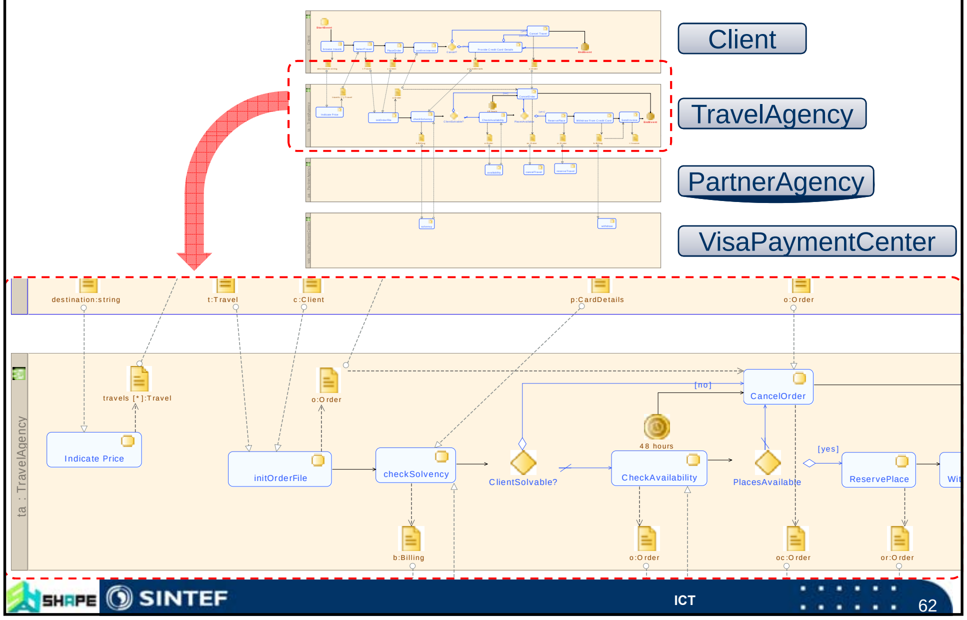
	Name	Text
0	Travel	Travel or travel package corresponds to a package including a destination and lodging services.
1	Package	The package options are : Flight Hotel Flight + Hotel Flight + Car Flight + Hotel + Car
2	Destination	The destination is identified by the continent (Nort Africa, Africa (excluding North), Europe, Asia, America) and the country.
3	Lodging services	The lodging services can be: lodgin only, breakfast, half-board and full board
4	Marketing Department	The communication with the partner travel agencies is ensured by the Marketing Department. This department defines priority offers to be found among the available travel packages available at other travel agencies. The goal of the Marketing Department is to compose the most attractive travel offers for the client of VOYAGES DISCOUNT.
5	Client order file	Client order file enables to manage travel orders by a client.
6	Agency	Agency : is a partner travel agency
7	Order state	The client order can be in several states: waiting : VOYAGES DISCOUNT checks the validity of the client's credit card, in progress : VOYAGES DISCOUNT negotiates with a partner travel agency (48 hours maximum duration), refused : insufficient number of available places, accepted :the places have been booked by the VOYAGES DISCOUNT at the partner agency.
8	Cancellation Insurance	The client have a possibility to subscribe to a cancellation insurance. If subscribed, this insurance enables to retrieve a fu
9	Client Information	First and Last Names, e-mail, phone and etc.
10	Participant	The traveller indicated in the order
11	Client	Person ordering a travel package, or a traveller.
12	Accompanying Person	Non ordering participant.

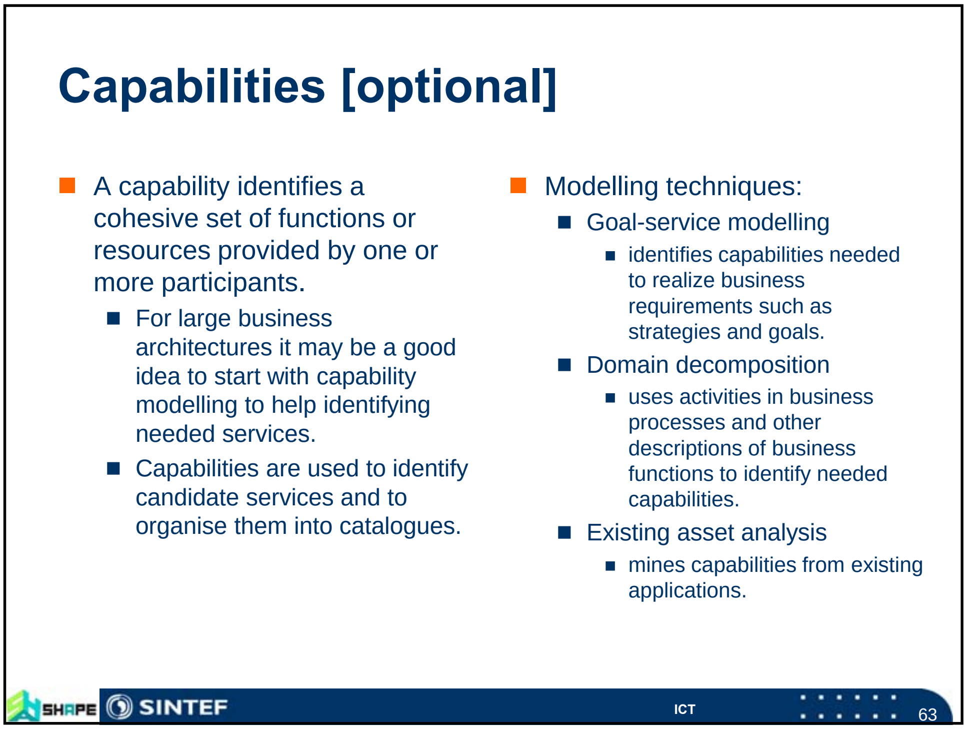


Business Processes (1/2)

- BPMN models on CIM level must be refined for the SOA to be defined.
  - Focus on the business processes that enable the business goals to be met.
- Identify relevant business processes:
  - Public and collaborative business processes that involve interactions and potential usage of software services between different business organizations.
    - Map to ***community-level services architectures*** in SoaML
  - Private business processes for the business organizations.
    - 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 the 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

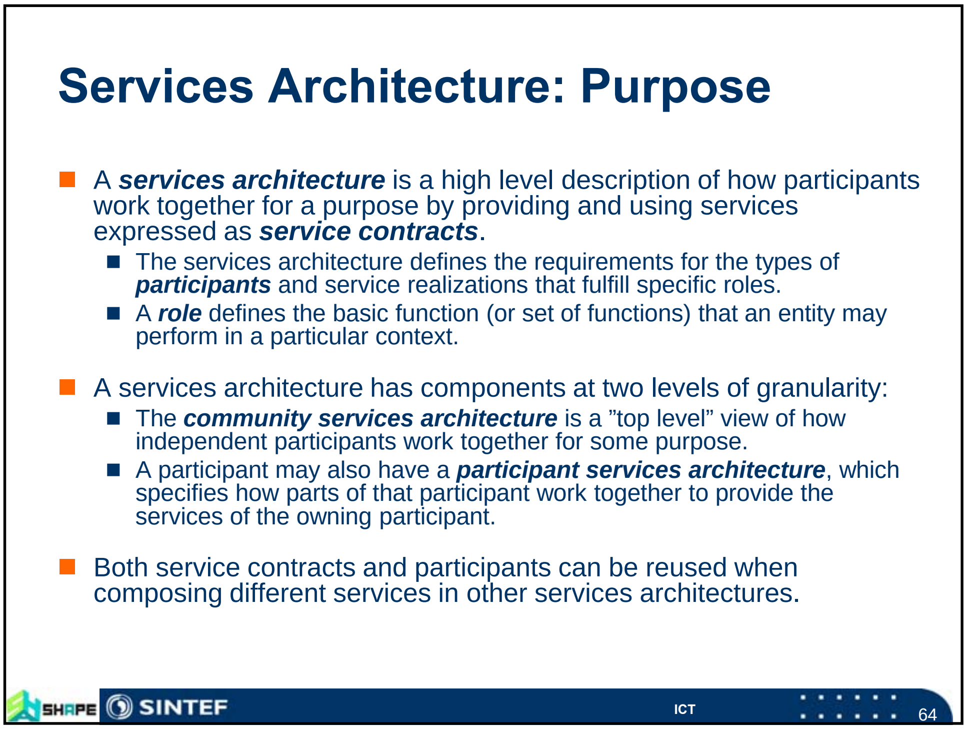
Business Processes (2/2)





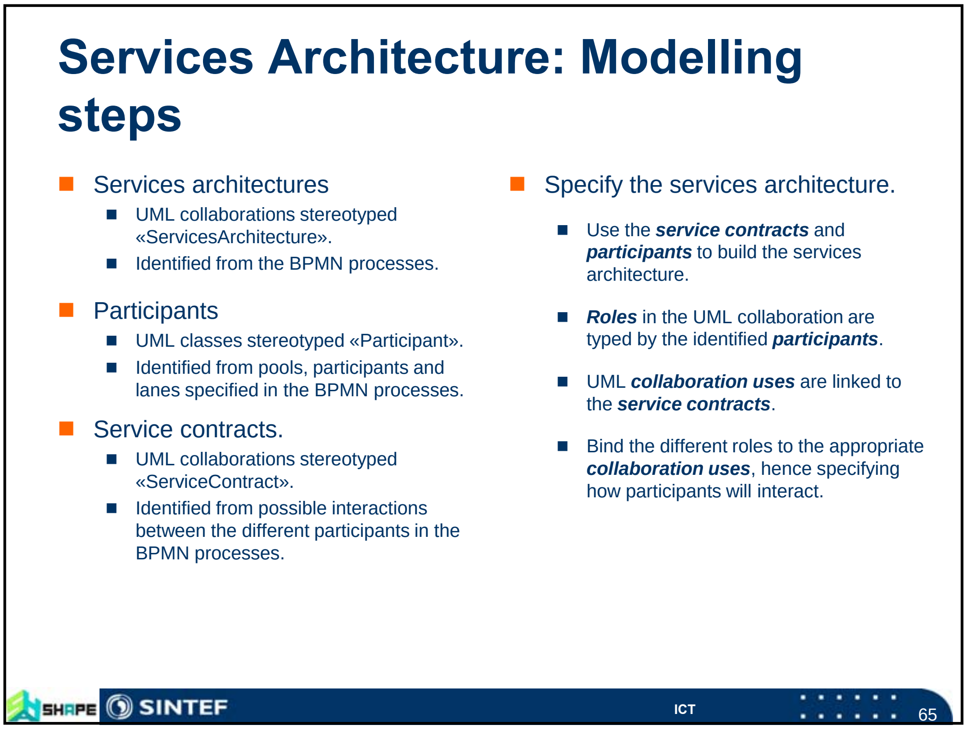
Capabilities [optional]

- A capability identifies a cohesive set of functions or resources provided by one or more participants.
  - For large business architectures it may be a good idea to start with capability modelling to help identifying needed services.
  - Capabilities are used to identify candidate services and to organise them into catalogues.
- Modelling techniques:
  - Goal-service modelling
    - identifies capabilities needed to realize business requirements such as strategies and 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.



Services Architecture: Purpose

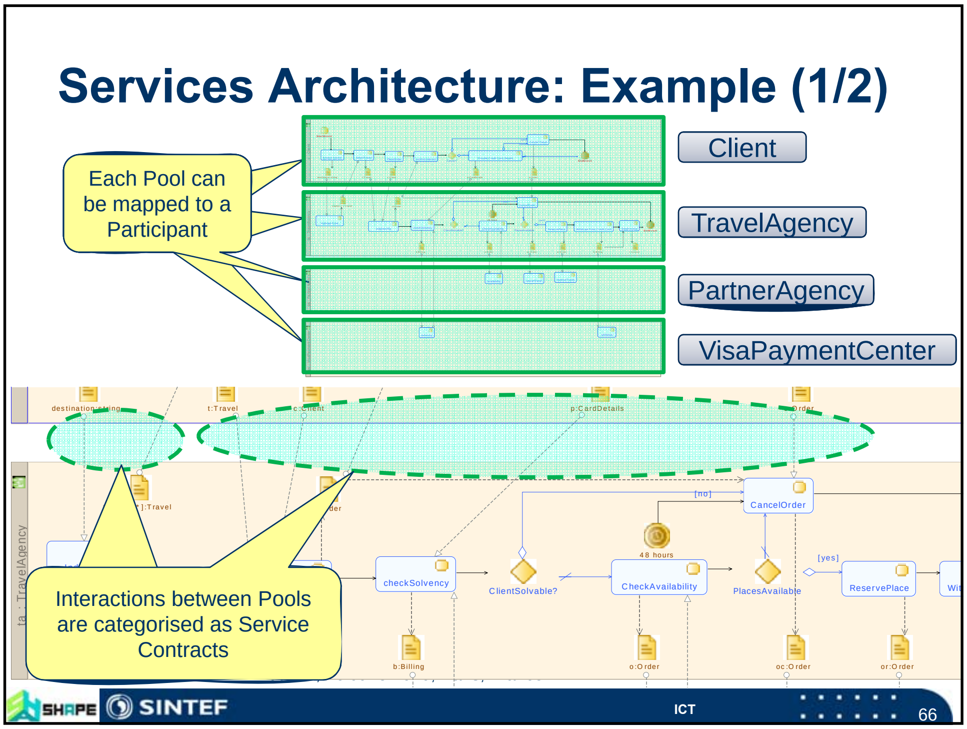
- A ***services architecture*** is a high level description of how participants work together for a purpose by providing and using services expressed as ***service contracts***.
  - The services architecture defines the requirements for the types of ***participants*** and service realizations that fulfill specific roles.
  - A ***role*** defines the basic function (or set of functions) that an entity may perform in a particular context.
- A services architecture has components at two levels of granularity:
  - The ***community services architecture*** is a ”top level” view of how independent participants work together for some purpose.
  - A participant may also have a ***participant services architecture***, which specifies how parts of that participant work together to provide the services of the owning participant.
- Both service contracts and participants can be reused when composing different services in other services architectures.



Services Architecture: Modelling steps

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

- Specify the services architecture.
  - Use the ***service contracts*** and ***participants*** to build the services architecture.
  - ***Roles*** in the UML collaboration are typed by the identified ***participants***.
  - UML ***collaboration uses*** are linked to the ***service contracts***.
  - Bind the different roles to the appropriate ***collaboration uses***, hence specifying how participants will interact.



Services Architecture: Example (1/2)

Each Pool can be mapped to a Participant

Client

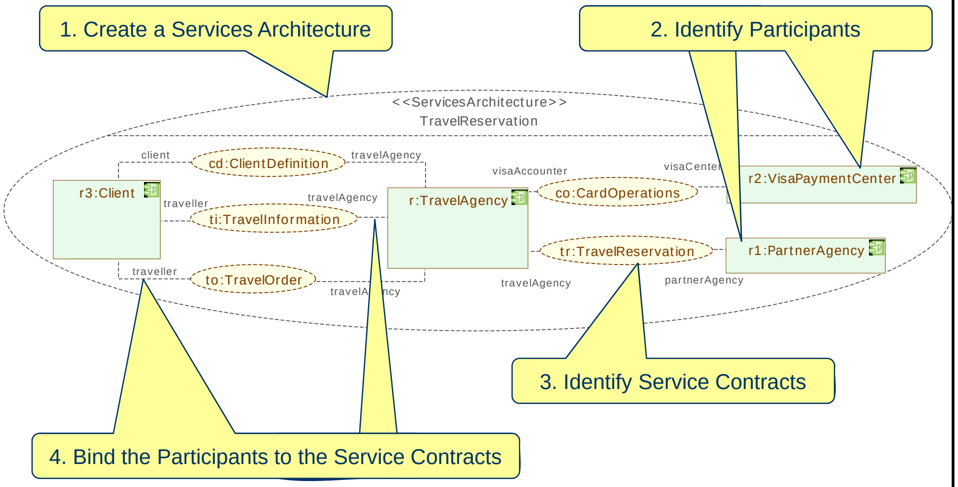
TravelAgency

PartnerAgency

VisaPaymentCenter

Interactions between Pools are categorised as Service Contracts

Services Architecture: Example (2/2)



1. Create a Services Architecture

2. Identify Participants

3. Identify Service Contracts

4. Bind the Participants to the Service Contracts

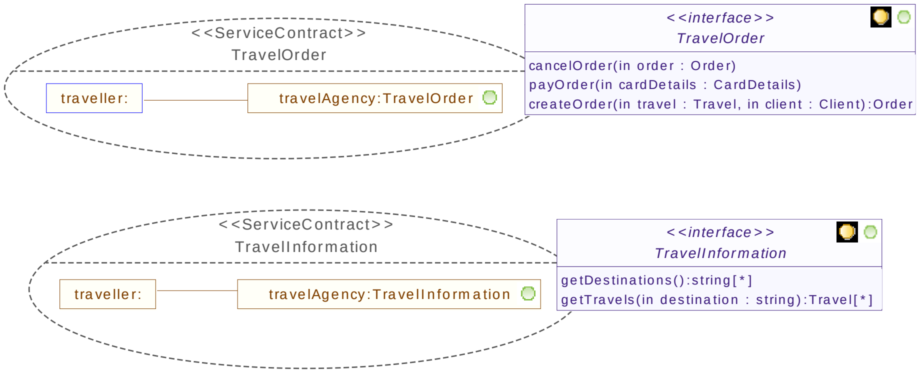
Service Contract: Purpose

- A service contract defines service specifications that define
  - the roles each participant plays in the service (such as provider and consumer) and
  - the interfaces they implement to play that role in that service.
- The service contract represents
  - an agreement between the involved participants for how
  - the service is to be provided and consumed.
- This agreement includes
  - the interfaces,
  - choreography and
  - any other terms and conditions.
- Service contracts are part of one or more services architectures that define
  - how a set of participants provide and use services
  - for a particular business purpose or process.

Service Contract: Modelling steps

- Identify the service contracts
  - Analyse the BPMN diagrams to identify service contracts.
  - This is a design-choice as there is no single construct in the BPMN that resembles a service contract.
- Certain pattern of objects can reveal service contracts for instance
  - when 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.
- Identify the roles.
  - Identify the ***providers*** and ***consumers*** of the service contract.
  - They are specified as ***roles*** typed by an ***interface***.
  - The roles are connected by a ***service channel***.
- Specify the interfaces at business level
  - Specify the business operations used in the interface.

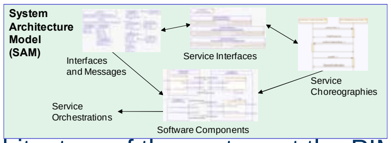
Service Contract: Example


<<ServiceContract>>
TravelOrder
traveller:
travelAgency:TravelOrder
<<interface>>
TravelOrder
cancelOrder(in order : Order)
payOrder(in cardDetails : CardDetails)
createOrder(in travel : Travel, in client : Client):Order
<<ServiceContract>>
TravelInformation
traveller:
travelAgency:TravelInformation
<<interface>>
TravelInformation
getDestinations():string[*]
getTravels(in destination : string):Travel[*]

Service Contract Behaviour: Service Choreography [optional]

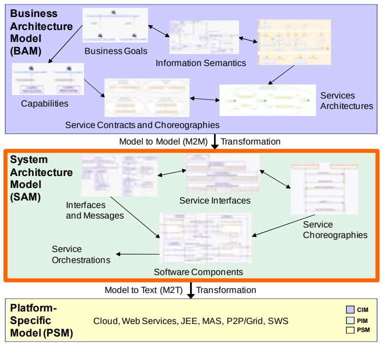
- The service contract behaviour specifies the behaviour of a service contract, i.e., the service choreography:
  - A choreography is a specification of what is transmitted and when it is transmitted between participants to enact a service exchange.
  - It expresses the expected business interactions between the consumers and providers of services.
- We recommend to model the behaviour of any complex service contract in order to get a better understanding of the interaction between the roles.
  - It can be specified as any UML behaviour or e.g. BPMN
  - Define the message sequence between the provider and consumer interfaces.

System Architecture Model (SAM)

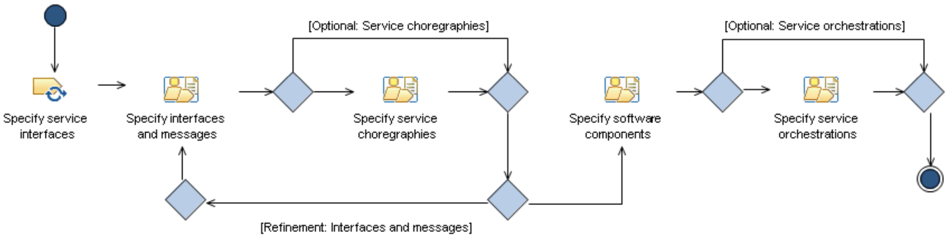


- The SAM describes the overall architecture of the system at the PIM level.
  - Represents the IT perspective of an SOA
  - It partitions the system into software components and interfaces.
- A structural model describes the components, their dependencies, and their interfaces:
  - Service interfaces
  - Interfaces and messages
  - Software components
- A dynamic model describes the component interactions and protocols:
  - Service choreographies
  - Service orchestrations

System Architecture Modelling



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



[Optional: Service choregraphies]
[Optional: Service orchestrations]
Specify service interfaces
Specify interfaces and messages
Specify service choregraphies
Specify software components
Specify service orchestrations
[Refinement: Interfaces and messages]

Service Interface: Purpose

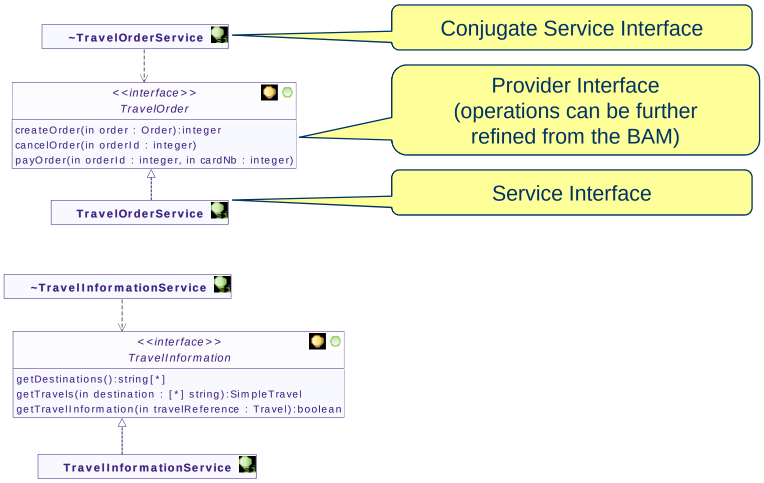
- A ***service interface*** defines the interfaces and responsibilities of a participant to provide or consume a service.
  - ***Service interfaces*** refine the ***service contracts*** specified in the BAM
- Used to type the ***port*** of a ***participant*** (i.e. software component).
- Service interfaces are reusable protocol definitions for different participants providing or consuming the same service.

- ***Simple interface*** based approach
  - focuses the attention on a one-way interaction provided by a participant on a port represented as a «Provider» UML interface
- ***Service interface*** based approach:
  - allows for bidirectional services, with ”callbacks” from the provider to the consumer
  - may include specific protocols, commands and information exchange

Service Interface: Modelling steps

- Define the ***service interfaces***.
  - UML interface stereotyped «ServiceInterface».
  - One-to-one mapping between service contracts and service interfaces.
- Define the ***provided interface***.
  - UML interface stereotyped «Provider».
- Define the ***consumer interface***.
  - UML Interface with «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***.
  - Create a conjugate service interface as a «ServiceInterface» that ***uses*** the ***provider interface*** and ***realizes*** the ***consumer interface***.
  - The name of the conjugage Interface starts with '~', and represent the counter-part of a service interface.

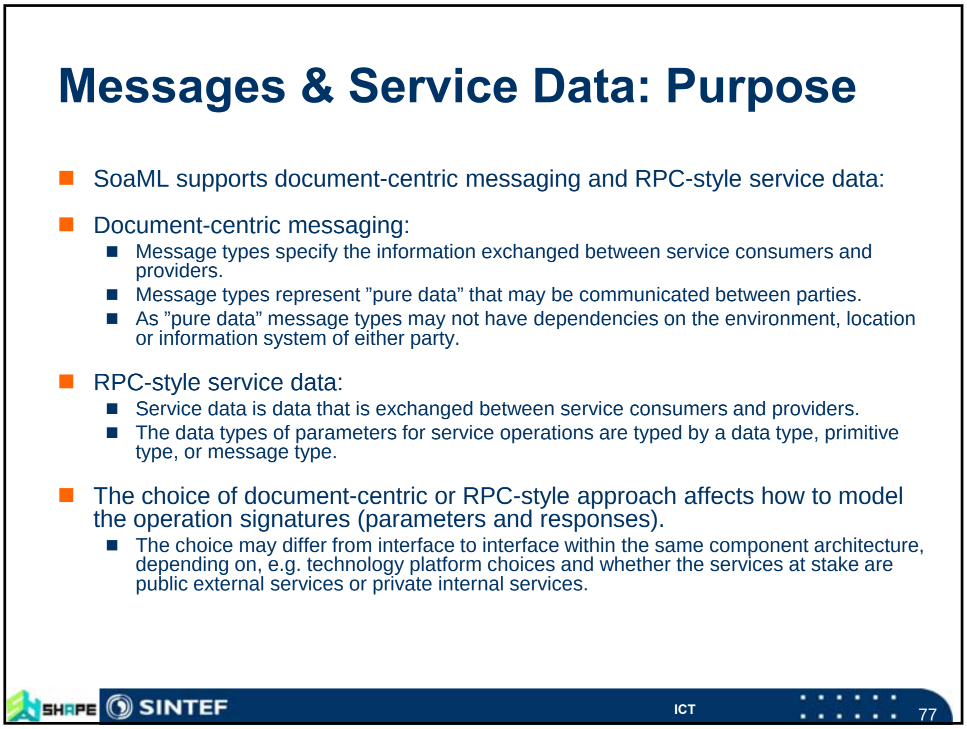
Service Interface: Example


~TravelOrderService
<<interface>>
TravelOrder
createOrder(in order : Order):integer
cancelOrder(in orderId : integer)
payOrder(in orderId : integer, in cardNb : integer)
TravelOrderService
~TravelInformationService
<<interface>>
TravelInformation
getDestinations():string[*]
getTravels(in destination : [*] string):SimpleTravel
getTravelInformation(in travelReference : Travel):boolean
TravelInformationService

Conjugate Service Interface

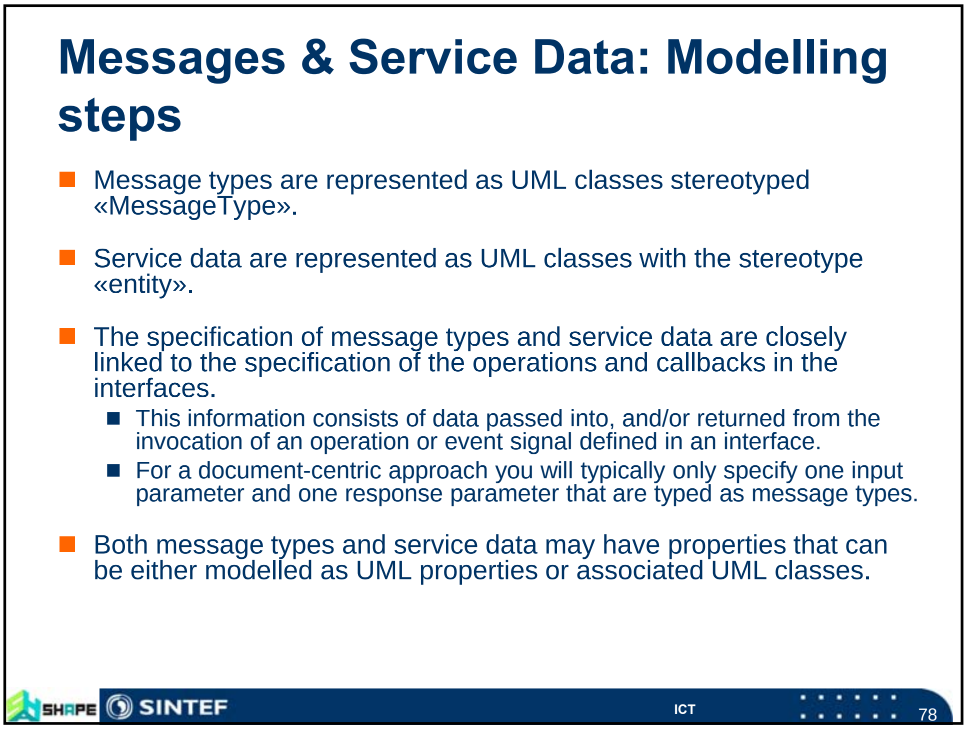
Provider Interface (operations can be further refined from the BAM)

Service Interface



Messages & Service Data: Purpose

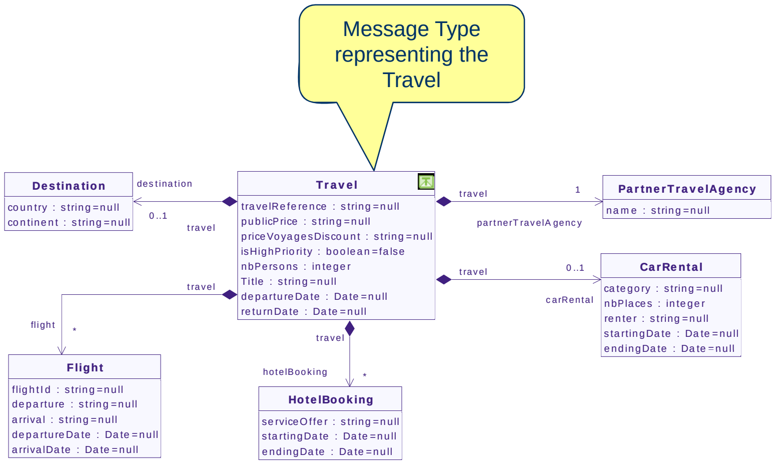
- SoaML supports document-centric messaging and RPC-style service data:
- Document-centric messaging:
  - Message types specify the information exchanged between service consumers and providers.
  - Message types represent ”pure data” that may be communicated between parties.
  - As ”pure data” message types may not have dependencies on the environment, location or information system of either party.
- RPC-style service data:
  - Service data is data that is exchanged between service consumers and providers.
  - The data types of parameters for service operations are typed by a data type, primitive type, or message type.
- The choice of document-centric or RPC-style approach affects how to model the operation signatures (parameters and responses).
  - The choice may differ from interface to interface within the same component architecture, depending on, e.g. technology platform choices and whether the services at stake are public external services or private internal services.



Messages & Service Data: Modelling steps

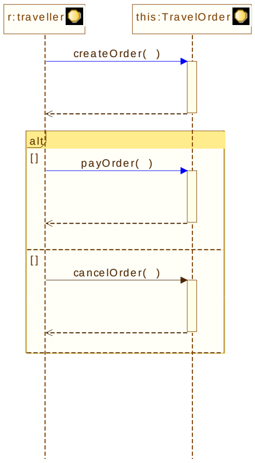
- Message types are represented as UML classes stereotyped «MessageType».
- Service data are represented as UML classes with the stereotype «entity».
- The specification of message types and service data are closely linked to the specification of the operations and callbacks in the interfaces.
  - This information consists of data passed into, and/or returned from the invocation of an operation or event signal defined in an interface.
  - For a document-centric approach you will typically only specify one input parameter and one response parameter that are typed as message types.
- Both message types and service data may have properties that can be either modelled as UML properties or associated UML classes.

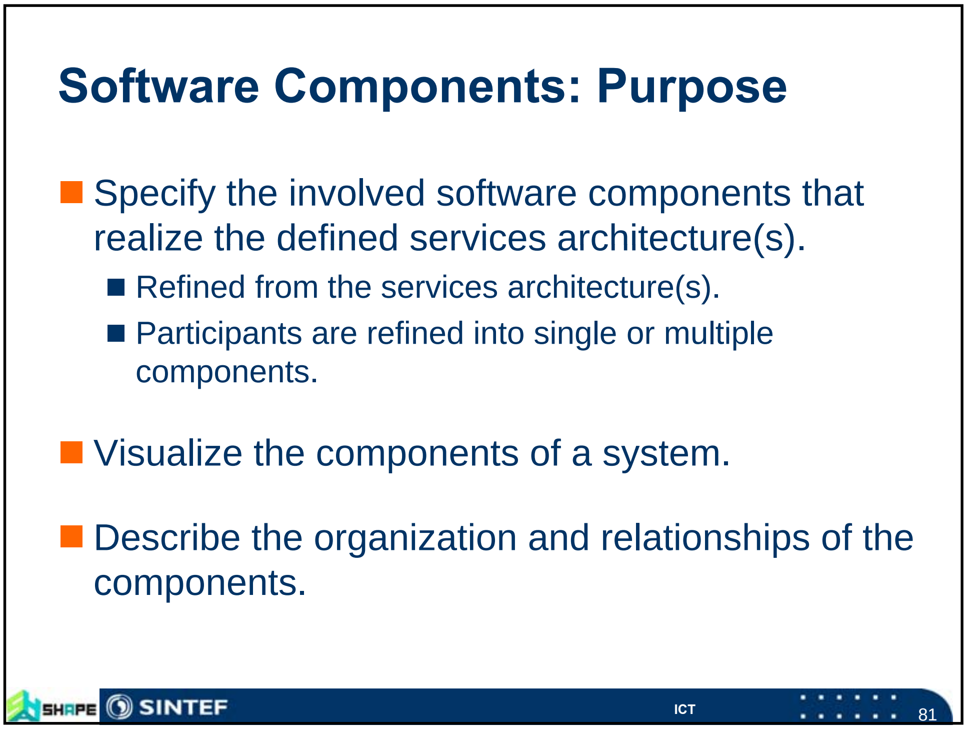
Message & Service Data: Example


Message Type representing the Travel

Service Choreographies [optional]

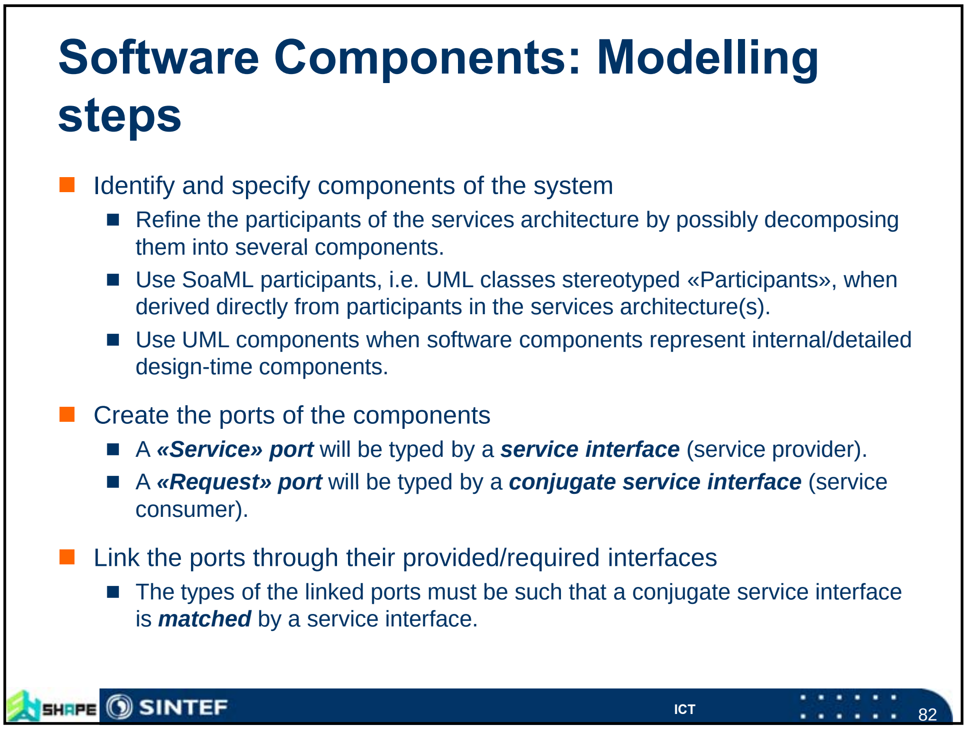
- Specifies the ***behaviour*** of a ***service interface***.
  - Usefull for complex services.
  - Refinement of the service contract behaviour of the BAM.
- Expresses the expected interactions between consumers and providers of services.
- It can be specified as any UML behaviour.
- Modelling steps:
  - Create an activity, interaction or state machine.
  - Define the message sequence between the provider and consumer interfaces.





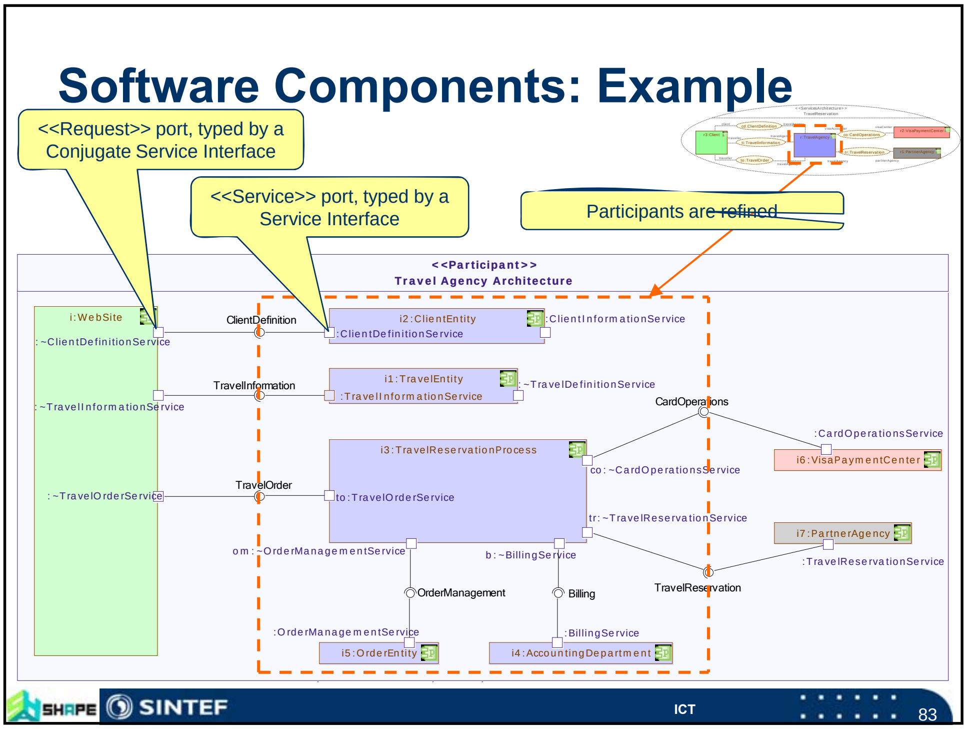
Software Components: Purpose

- Specify the involved software components that realize the defined services architecture(s).
  - Refined from the services architecture(s).
  - Participants are refined into single or multiple components.
- Visualize the components of a system.
- Describe the organization and relationships of the components.



Software Components: Modelling steps

- Identify and specify components of the system
  - Refine the participants of the services architecture by possibly decomposing them into several components.
  - Use SoaML participants, i.e. UML classes stereotyped «Participants», when derived directly from participants in the services architecture(s).
  - Use UML components when software components represent internal/detailed design-time components.
- Create the ports of the components
  - A ***«Service» port*** will be typed by a ***service interface*** (service provider).
  - A ***«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.

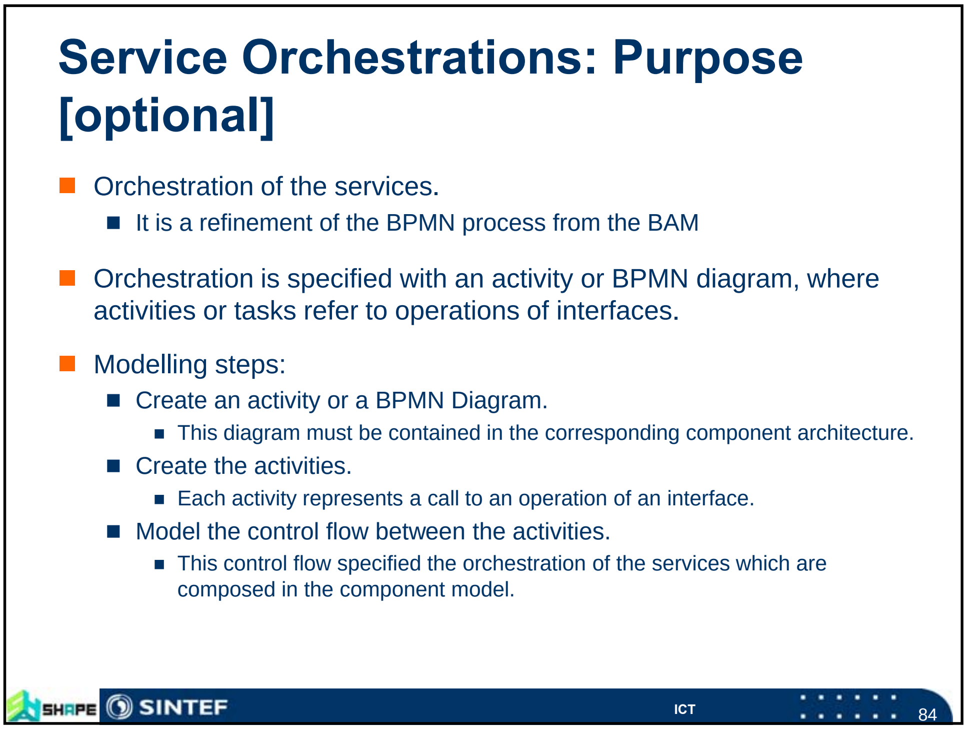


Software Components: Example

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

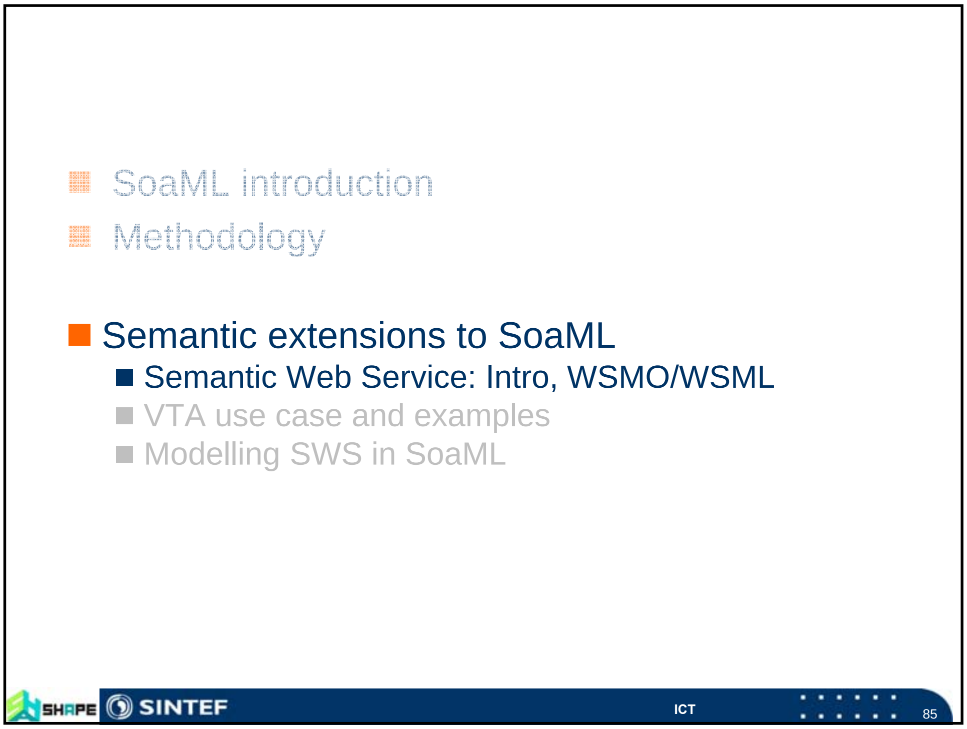
<<Service>> port, typed by a Service Interface

Participants are refined

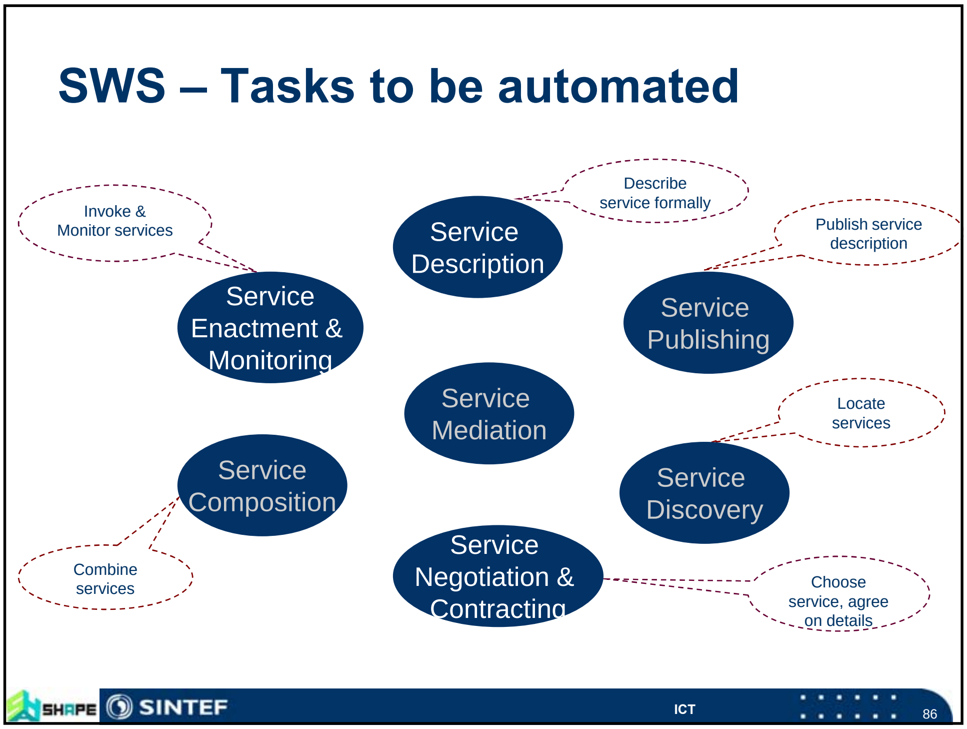


Service Orchestrations: Purpose [optional]

- Orchestration of the services.
  - It is a refinement of the BPMN process from the BAM
- Orchestration is specified with an activity or BPMN diagram, where activities or tasks refer to operations of interfaces.
- Modelling steps:
  - Create an activity or a BPMN Diagram.
    - This diagram must be contained in the corresponding component architecture.
  - Create the activities.
    - Each activity represents a call to an operation of an interface.
  - Model the control flow between the activities.
    - This control flow specified the orchestration of the services which are composed in the component model.



- SoaML introduction
- Methodology
- Semantic extensions to SoaML
  - Semantic Web Service: Intro, WSMO/WSML
  - VTA use case and examples
  - Modelling SWS in SoaML



SWS – Tasks to be automated

Service Description – Describe service formally

Service Publishing – Publish service description

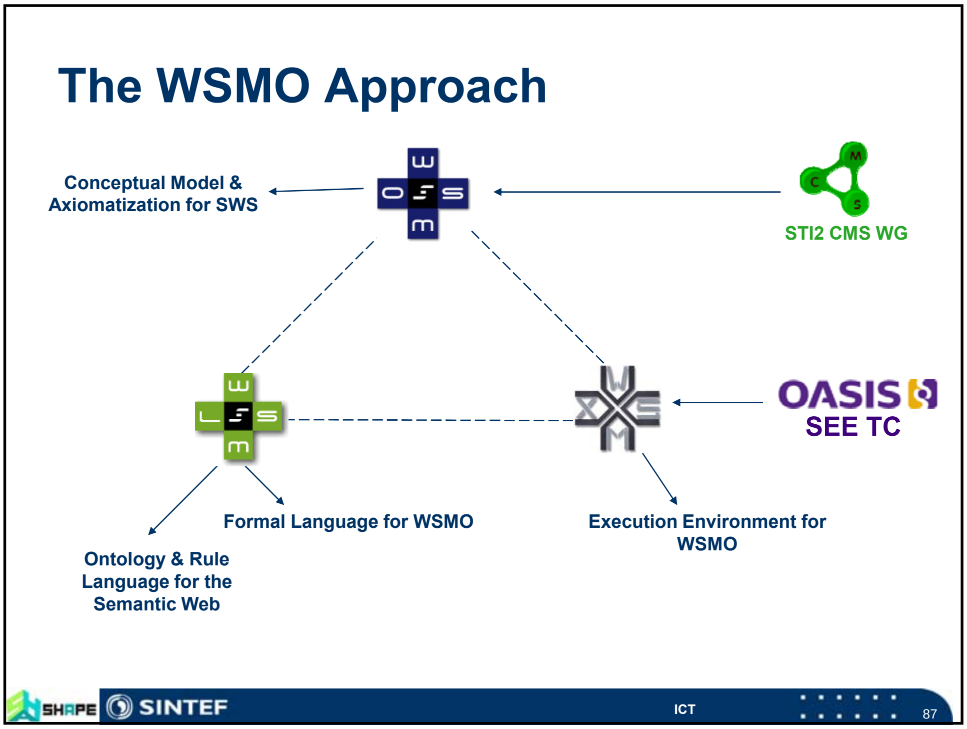
Service Discovery – Locate services

Service Negotiation & Contracting – Choose service, agree on details

Service Composition – Combine services

Service Enactment & Monitoring – Invoke & Monitor services

Service Mediation



The WSMO Approach

Conceptual Model & Axiomatization for SWS

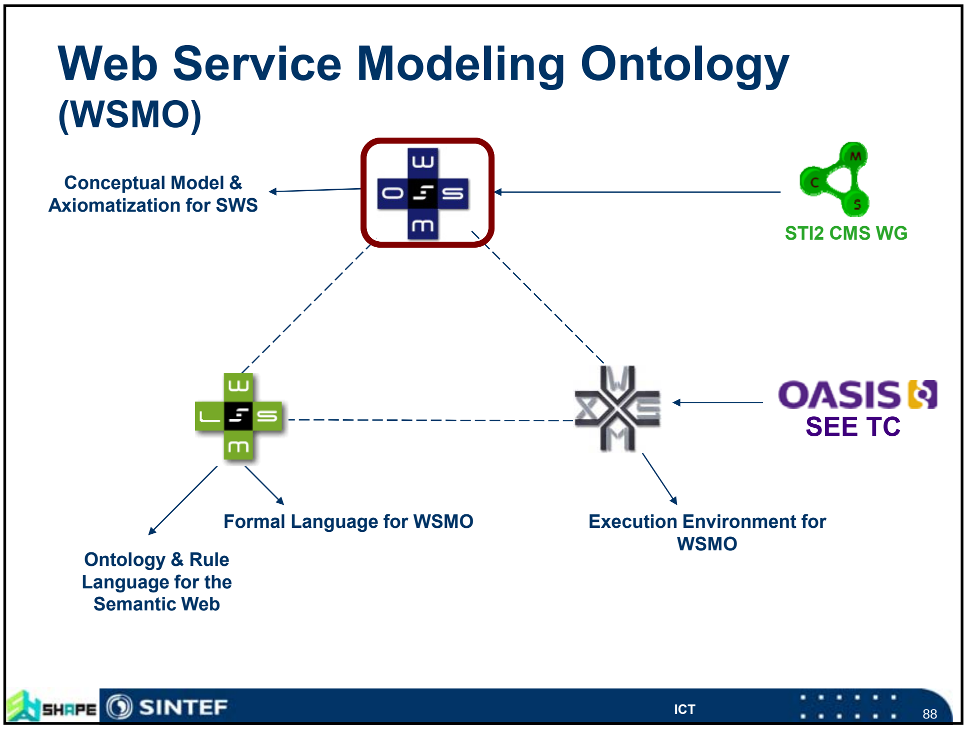
STI2 CMS WG

OASIS SEE TC

Formal Language for WSMO

Execution Environment for WSMO

Ontology & Rule Language for the Semantic Web



Web Service Modeling Ontology (WSMO)

Conceptual Model & Axiomatization for SWS

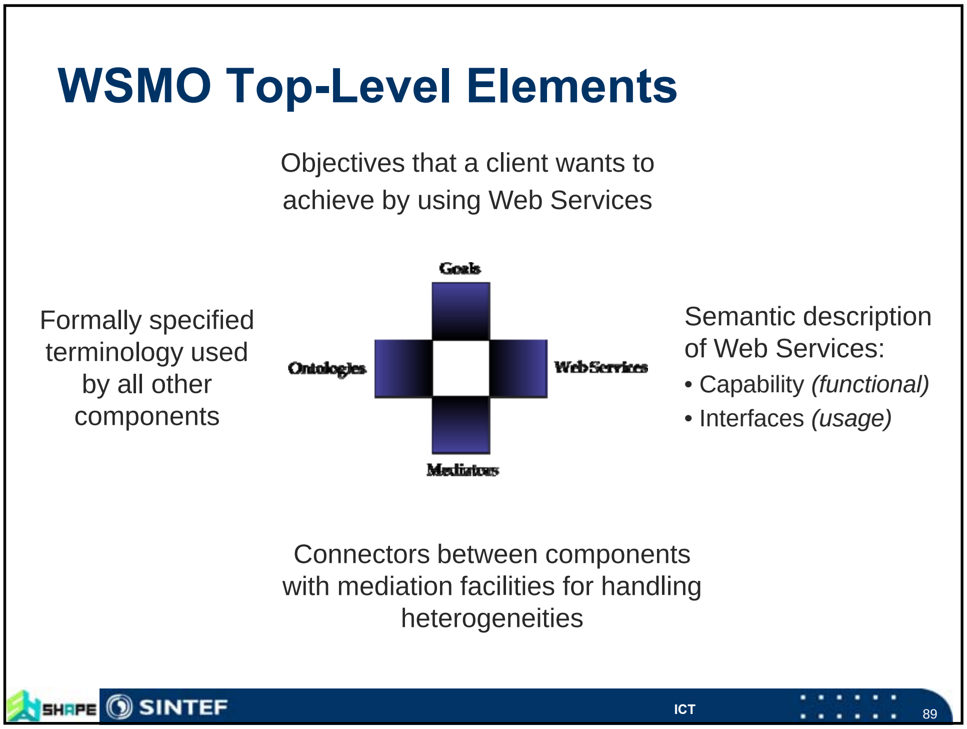
STI2 CMS WG

OASIS SEE TC

Formal Language for WSMO

Execution Environment for WSMO

Ontology & Rule Language for the Semantic Web



WSMO Top-Level Elements

Objectives that a client wants to achieve by using Web Services

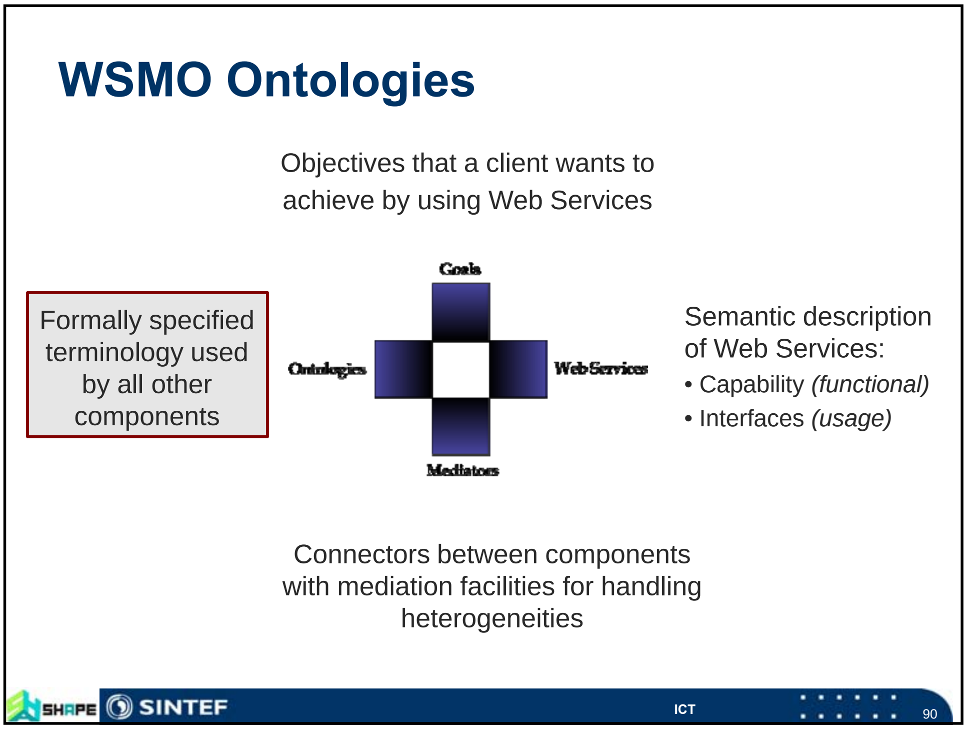
Formally specified terminology used by all other components

Goals
Ontologies
Web Services
Mediators

Semantic description of Web Services:

- Capability *(functional)*
- Interfaces *(usage)*

Connectors between components with mediation facilities for handling heterogeneities



WSMO Ontologies

Objectives that a client wants to achieve by using Web Services

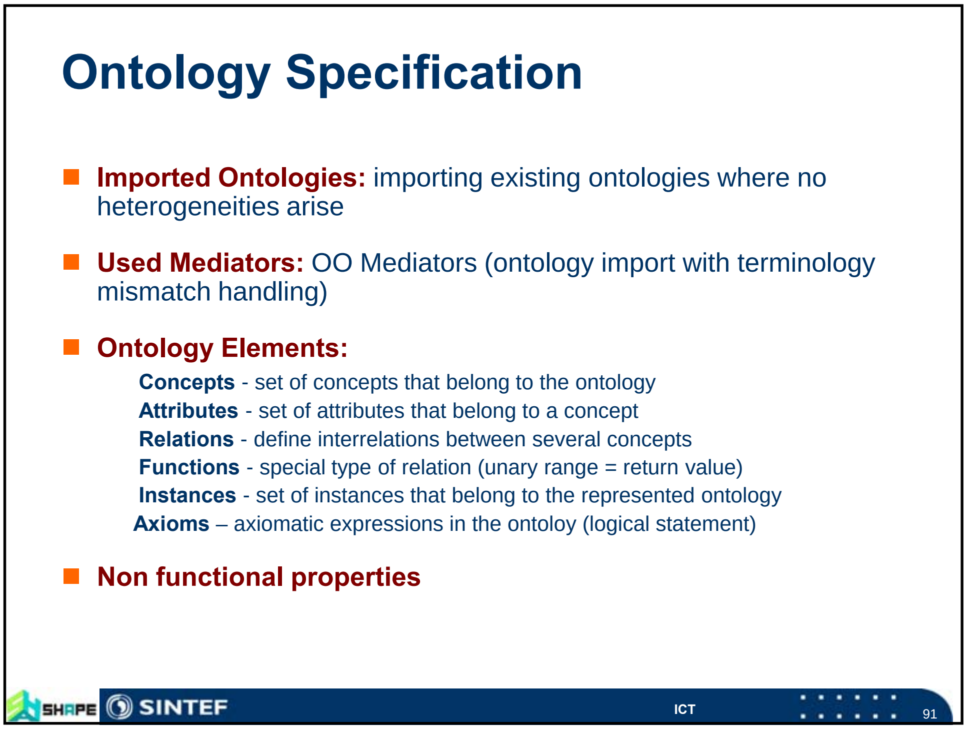
Formally specified terminology used by all other components

Goals
Ontologies
Web Services
Mediators

Semantic description of Web Services:

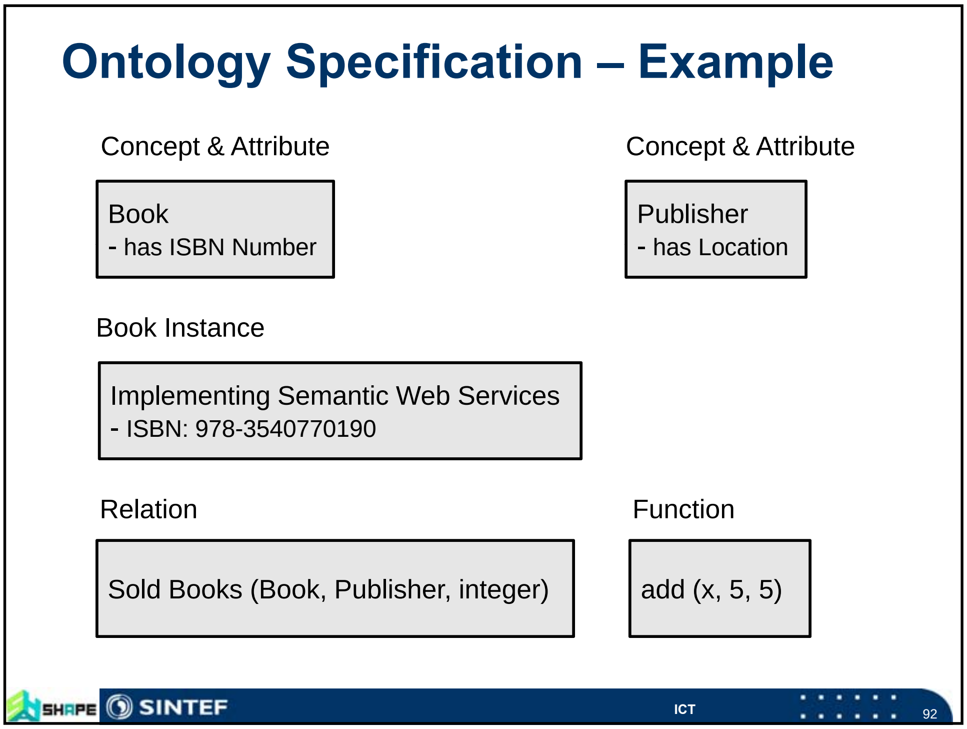
- Capability *(functional)*
- Interfaces *(usage)*

Connectors between components with mediation facilities for handling heterogeneities



Ontology Specification

- **Imported Ontologies:** importing existing ontologies where no heterogeneities arise
- **Used Mediators:** OO Mediators (ontology import with terminology mismatch handling)
- **Ontology Elements:**
  - **Concepts** - set of concepts that belong to the ontology
  - **Attributes** - set of attributes that belong to a concept
  - **Relations** - define interrelations between several concepts
  - **Functions** - special type of relation (unary range = return value)
  - **Instances** - set of instances that belong to the represented ontology
  - **Axioms** – axiomatic expressions in the ontoloy (logical statement)
- **Non functional properties**



Ontology Specification – Example

Concept & Attribute

Book
- has ISBN Number

Concept & Attribute

Publisher
- has Location

Book Instance

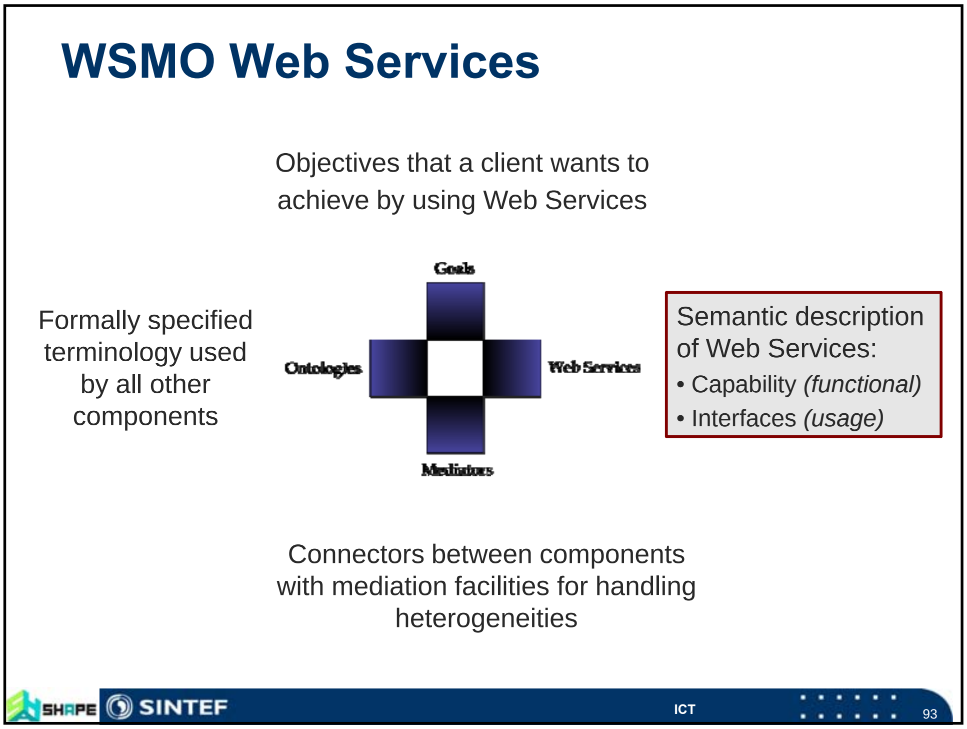
Implementing Semantic Web Services
- ISBN: 978-3540770190

Relation

Sold Books (Book, Publisher, integer)

Function

add (x, 5, 5)



WSMO Web Services

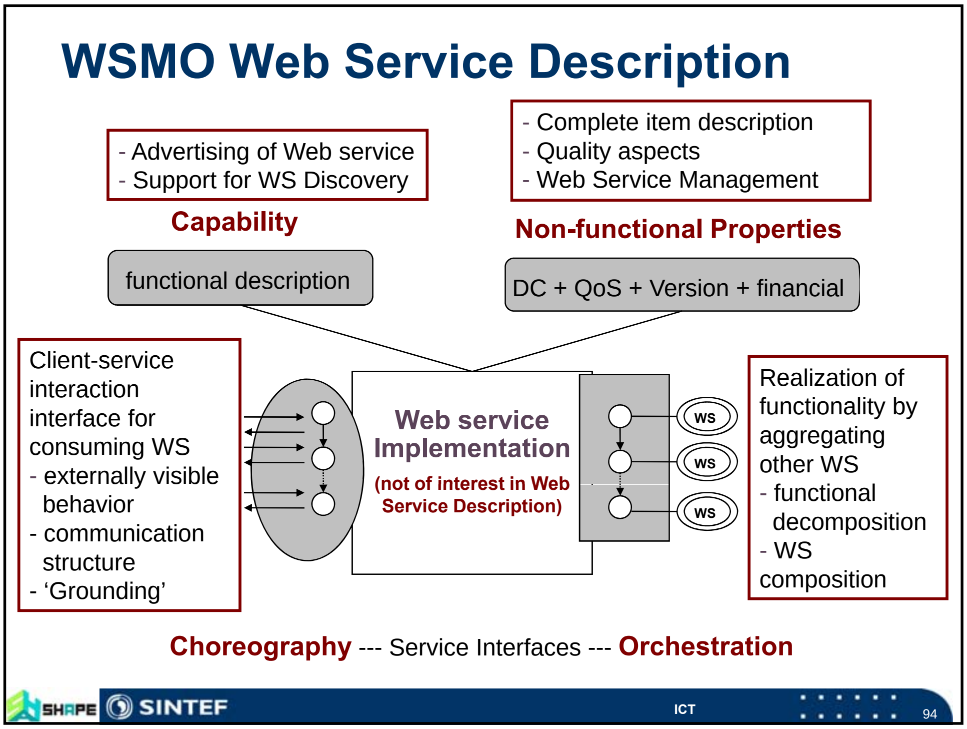
Objectives that a client wants to achieve by using Web Services

Formally specified terminology used by all other components

Semantic description of Web Services:

- Capability *(functional)*
- Interfaces *(usage)*

Connectors between components with mediation facilities for handling heterogeneities



WSMO Web Service Description

- Advertising of Web service
- Support for WS Discovery

Capability

functional description

- Complete item description
- Quality aspects
- Web Service Management

Non-functional Properties

DC + QoS + Version + financial

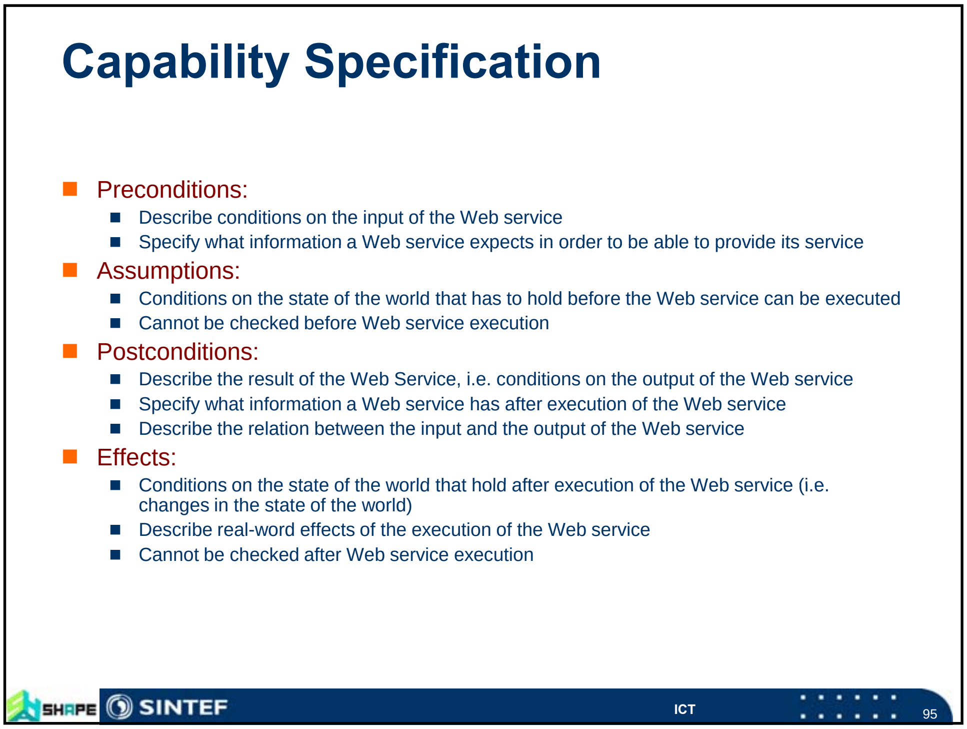
Client-service interaction interface for consuming WS
- externally visible behavior
- communication structure
- ‘Grounding’

Web service Implementation

(not of interest in Web Service Description)

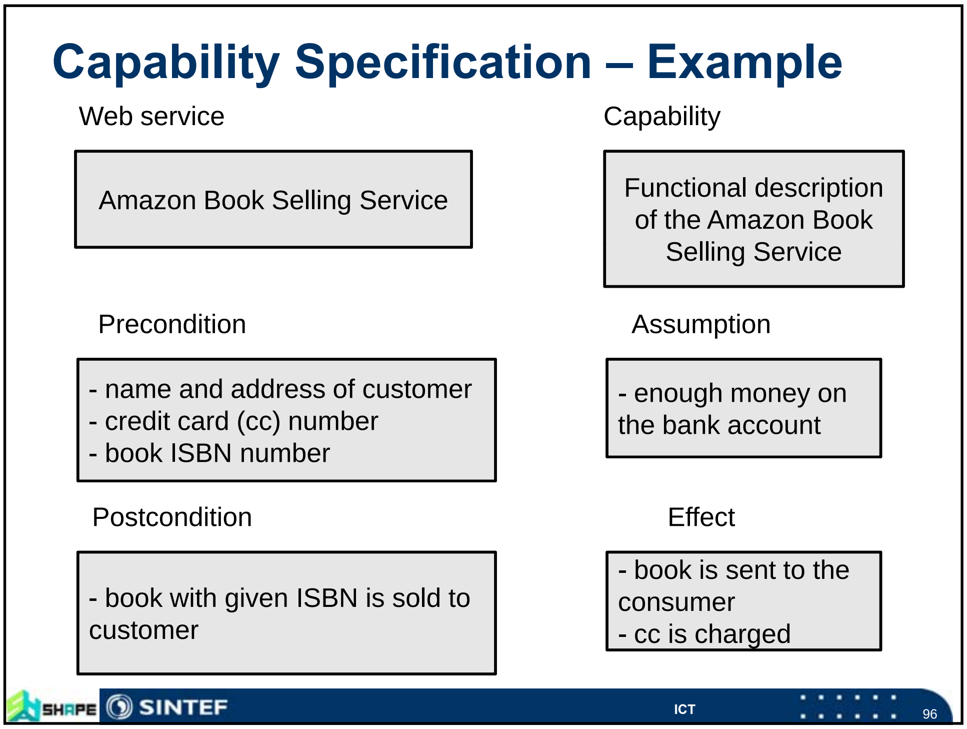
Realization of functionality by aggregating other WS
- functional decomposition
- WS composition

Choreography --- Service Interfaces --- **Orchestration**



Capability Specification

- Preconditions:
  - Describe conditions on the input of the Web service
  - Specify what information a Web service expects in order to be able to provide its service
- Assumptions:
  - Conditions on the state of the world that has to hold before the Web service can be executed
  - Cannot be checked before Web service execution
- Postconditions:
  - Describe the result of the Web Service, i.e. conditions on the output of the Web service
  - Specify what information a Web service has after execution of the Web service
  - Describe the relation between the input and the output of the Web service
- Effects:
  - Conditions on the state of the world that hold after execution of the Web service (i.e. changes in the state of the world)
  - Describe real-word effects of the execution of the Web service
  - Cannot be checked after Web service execution



Capability Specification – Example

Web service

Amazon Book Selling Service

Capability

Functional description of the Amazon Book Selling Service

Precondition

- name and address of customer
- credit card (cc) number
- book ISBN number

Assumption

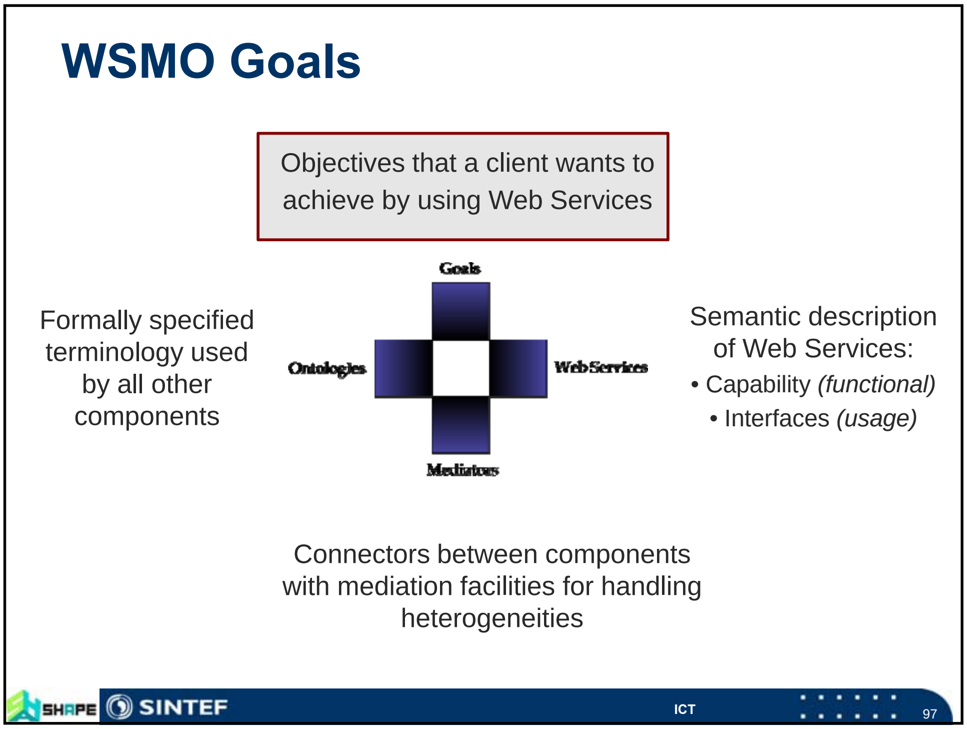
- enough money on the bank account

Postcondition

- book with given ISBN is sold to customer

Effect

- book is sent to the consumer
- cc is charged



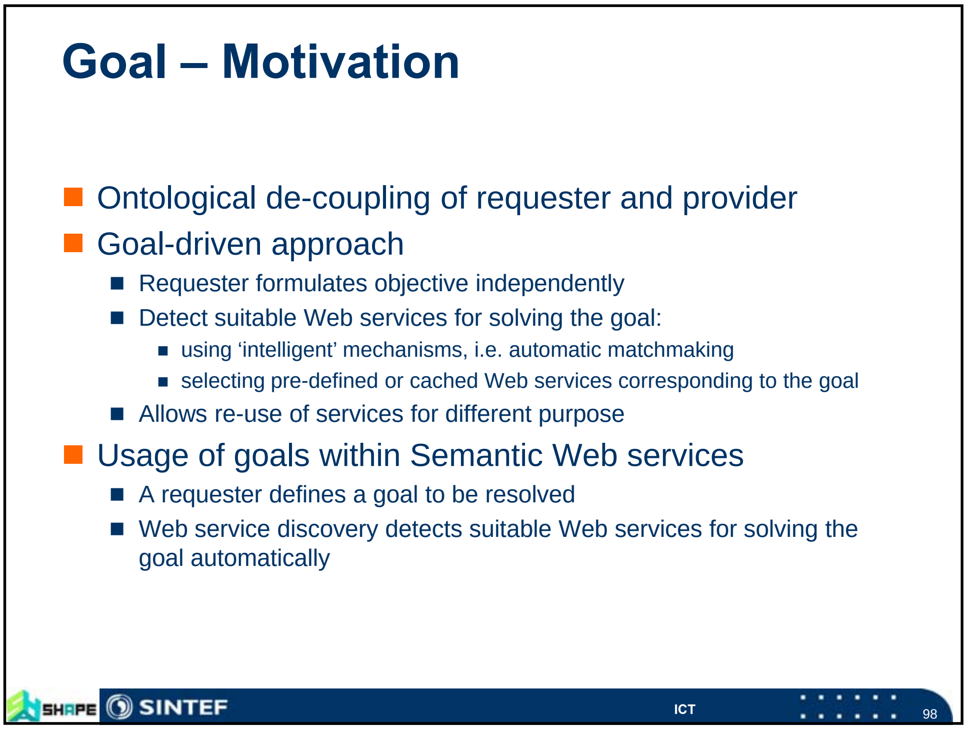
WSMO Goals

Objectives that a client wants to achieve by using Web Services

Formally specified terminology used by all other components

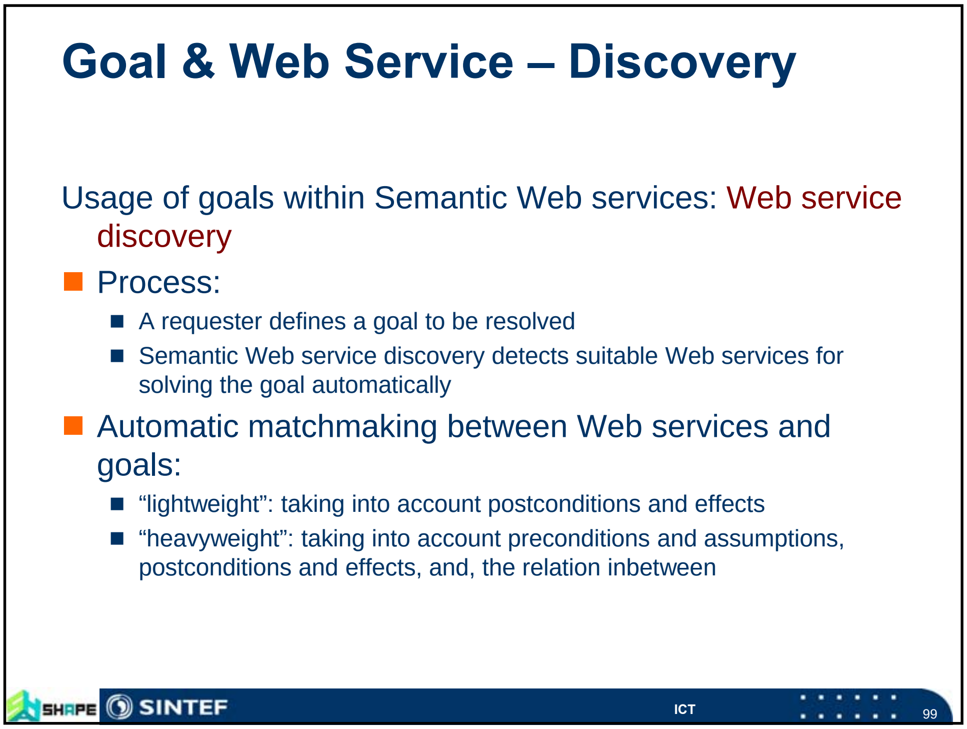
Semantic description of Web Services:
• Capability *(functional)*
• Interfaces *(usage)*

Connectors between components with mediation facilities for handling heterogeneities



Goal – Motivation

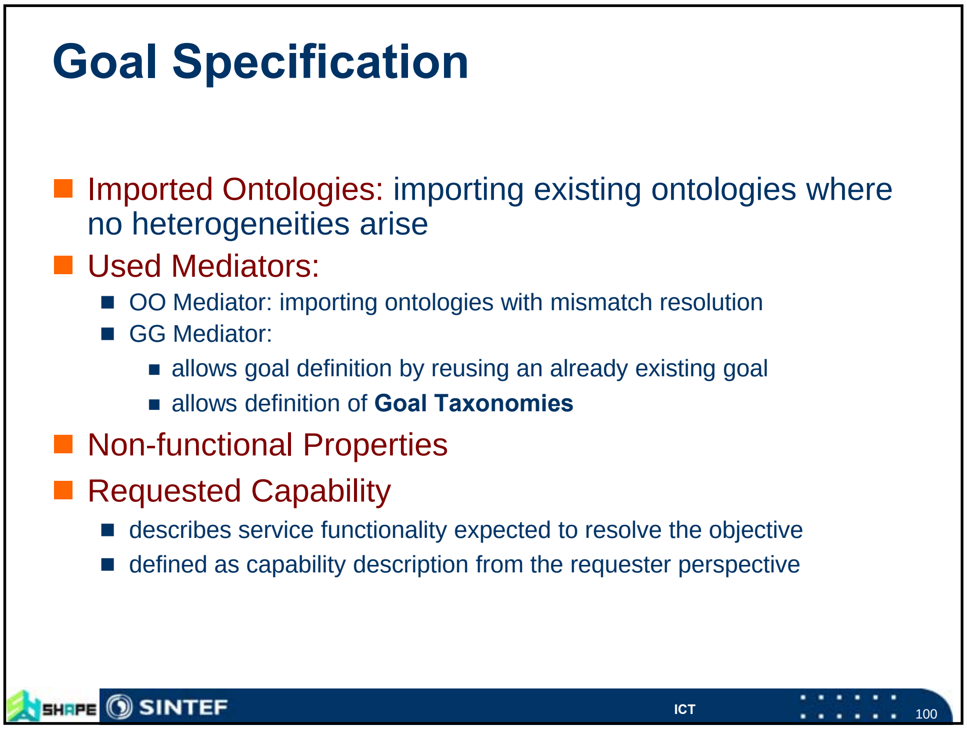
- Ontological de-coupling of requester and provider
- Goal-driven approach
  - Requester formulates objective independently
  - Detect suitable Web services for solving the goal:
    - using ‘intelligent’ mechanisms, i.e. automatic matchmaking
    - selecting pre-defined or cached Web services corresponding to the goal
  - Allows re-use of services for different purpose
- Usage of goals within Semantic Web services
  - A requester defines a goal to be resolved
  - Web service discovery detects suitable Web services for solving the goal automatically



Goal & Web Service – Discovery

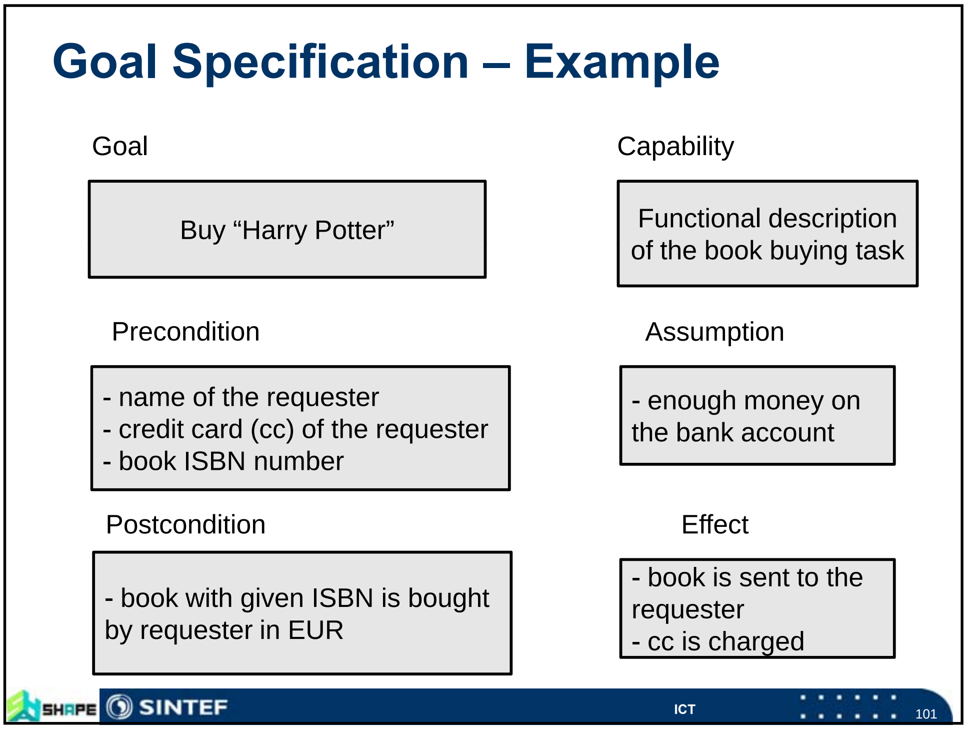
Usage of goals within Semantic Web services: Web service discovery

- Process:
  - A requester defines a goal to be resolved
  - Semantic Web service discovery detects suitable Web services for solving the goal automatically
- Automatic matchmaking between Web services and goals:
  - “lightweight”: taking into account postconditions and effects
  - “heavyweight”: taking into account preconditions and assumptions, postconditions and effects, and, the relation inbetween



Goal Specification

- Imported Ontologies: importing existing ontologies where no heterogeneities arise
- Used Mediators:
  - OO Mediator: importing ontologies with mismatch resolution
  - GG Mediator:
    - allows goal definition by reusing an already existing goal
    - allows definition of **Goal Taxonomies**
- Non-functional Properties
- Requested Capability
  - describes service functionality expected to resolve the objective
  - defined as capability description from the requester perspective



Goal Specification – Example

Goal

Buy “Harry Potter”

Capability

Functional description of the book buying task

Precondition

- name of the requester
- credit card (cc) of the requester
- book ISBN number

Assumption

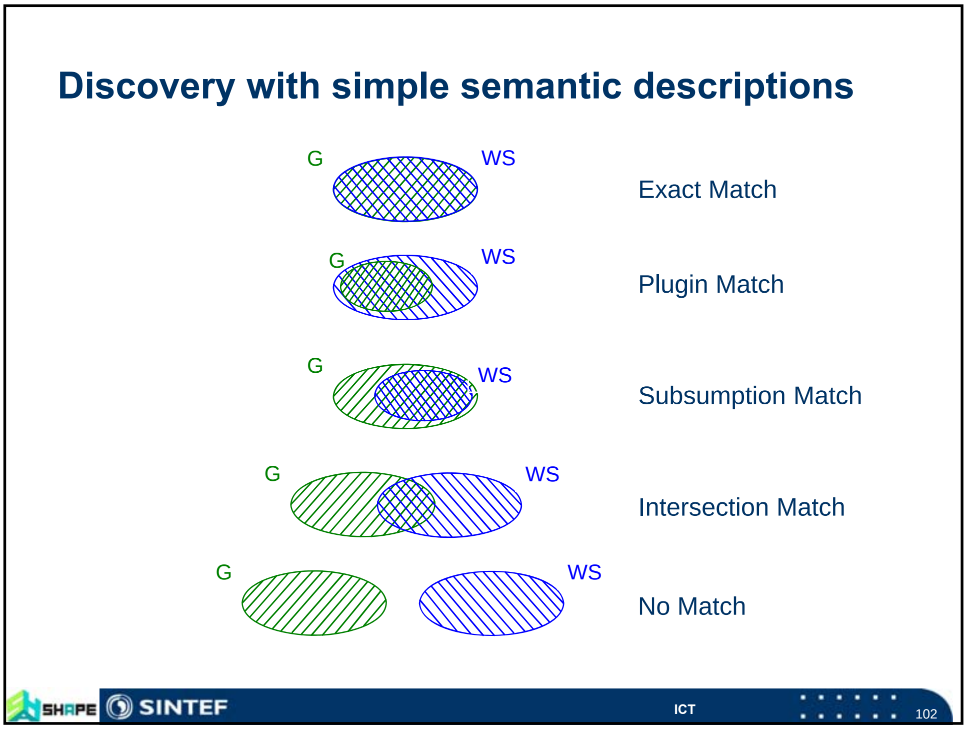
- enough money on the bank account

Postcondition

- book with given ISBN is bought by requester in EUR

Effect

- book is sent to the requester
- cc is charged



Discovery with simple semantic descriptions

Exact Match

Plugin Match

Subsumption Match

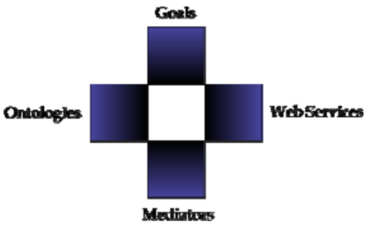
Intersection Match

No Match

WSMO Mediators

Objectives that a client wants to achieve by using Web Services

Formally specified terminology used by all other components



Semantic description of Web Services:

- Capability *(functional)*
- Interfaces *(usage)*

Connectors between components with mediation facilities for handling heterogeneities

Mediation – Motivation (1)

- Heterogeneity …
  - concerning mismatches on structural / semantic / conceptual / functional / level
  - occur between different components that shall interoperate
  - unavoidable in distributed & open environments like the Internet
- Concept of Mediation (Wiederhold, 94):
  - mediators as components that resolve mismatches
  - declarative approach:
    - semantic description of resources
    - 'intelligent' mechanisms that resolve mismatches independent of content
  - mediation cannot be fully automated (integration decision)

Mediation – Motivation (2)

- Data Level:
  - mediate heterogeneous data sources, more specifically solving the problem of ontology integration
    - mapping
    - merging
    - alignment
  - transformation between languages / formalisms
- Functional Level:
  - mediate mismatches between Web service/Goal and Web Service/Goals functionalities
- Process/Protocol Level:
  - mediate heterogeneous Business Processes / Communication Patterns

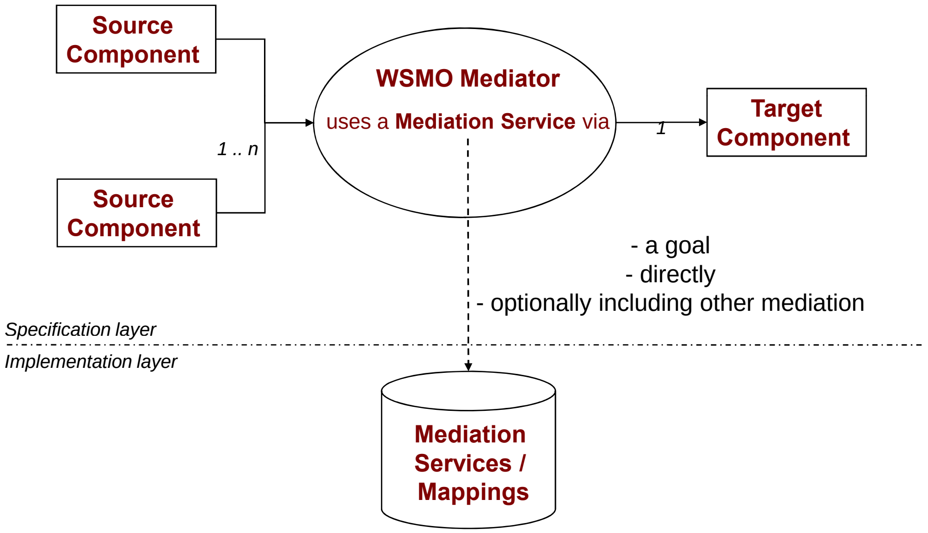
Mediator Specification (1)

- Imported Ontologies: importing existing ontologies where no heterogeneities arise
- Used Mediators:
  - OO Mediator: importing ontologies with mismatch resolution
  - GG Mediator:
    - allows goal definition by reusing an already existing goal
    - allows definition of **Goal Taxonomies**
  - WW Mediator: resolving data, process and protocol heterogeneity between Web services
  - WG Mediator: resolving data, process and protocol heterogeneity between a Goal and Web services
- Non-functional Properties

Mediator Specification (2)

- Source(s)
  - defining entities that are the sources of the mediator
- Target
  - defining the entity that is the target of the mediator
- Mediation Service
  - the mediation service points to:
    - a goal that declaratively describes the mapping, or to
    - a Web service that actually implements the mapping, or to a
    - wwMediator that links to a Web service that actually implements the mapping

Mediator Structure



Source Component

Source Component

1 .. n

WSMO Mediator

uses a **Mediation Service** via

1

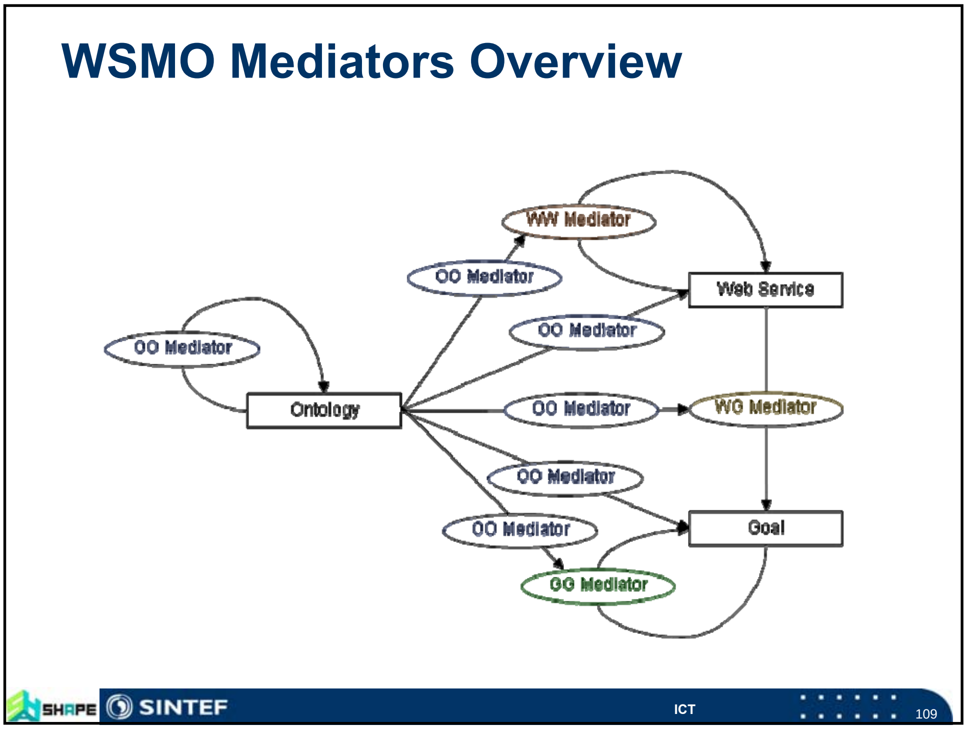
Target Component

- a goal
- directly
- optionally including other mediation

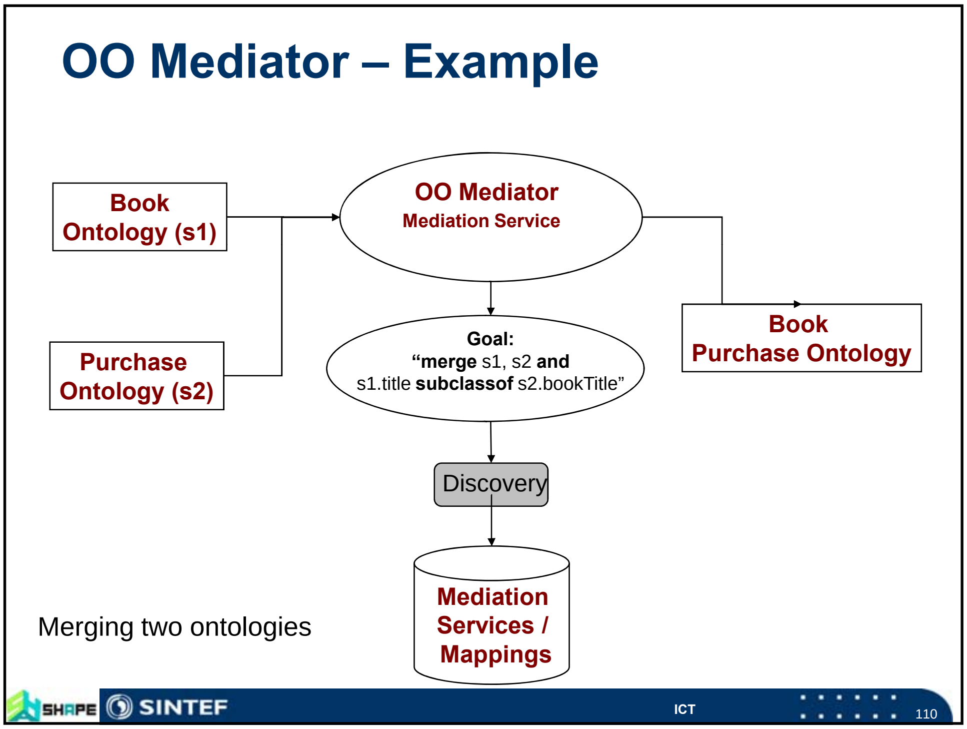
Specification layer

Implementation layer

Mediation Services / Mappings



WSMO Mediators Overview



OO Mediator – Example

Book Ontology (s1)

Purchase Ontology (s2)

OO Mediator
Mediation Service

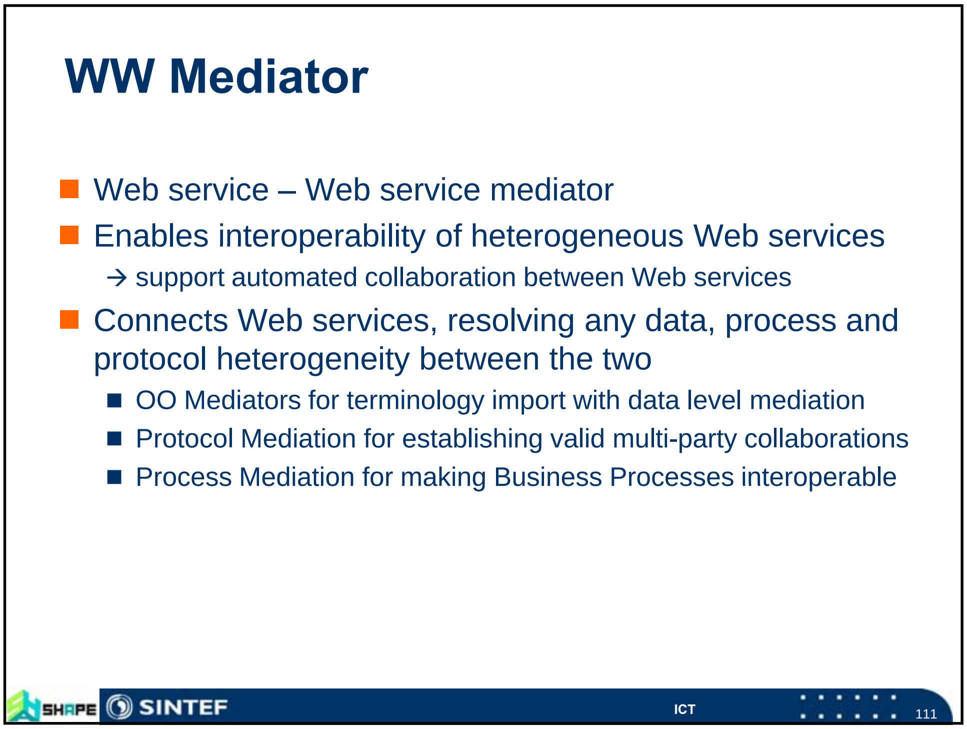
Goal:
"**merge** s1, s2 **and** s1.title **subclassof** s2.bookTitle"

Discovery

Mediation Services / Mappings

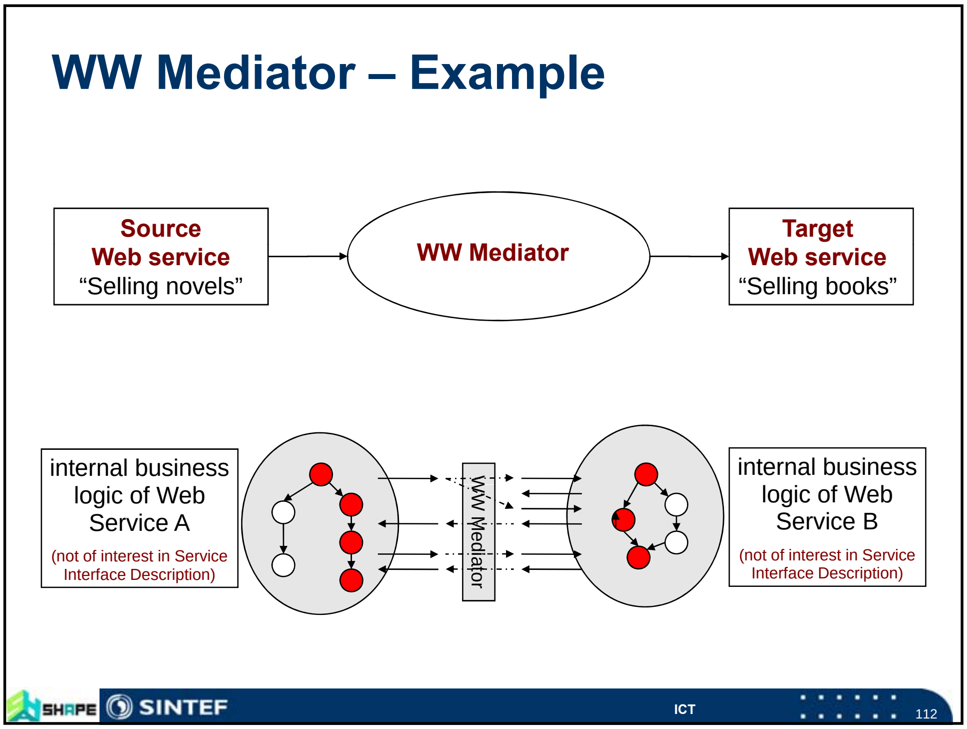
Book Purchase Ontology

Merging two ontologies



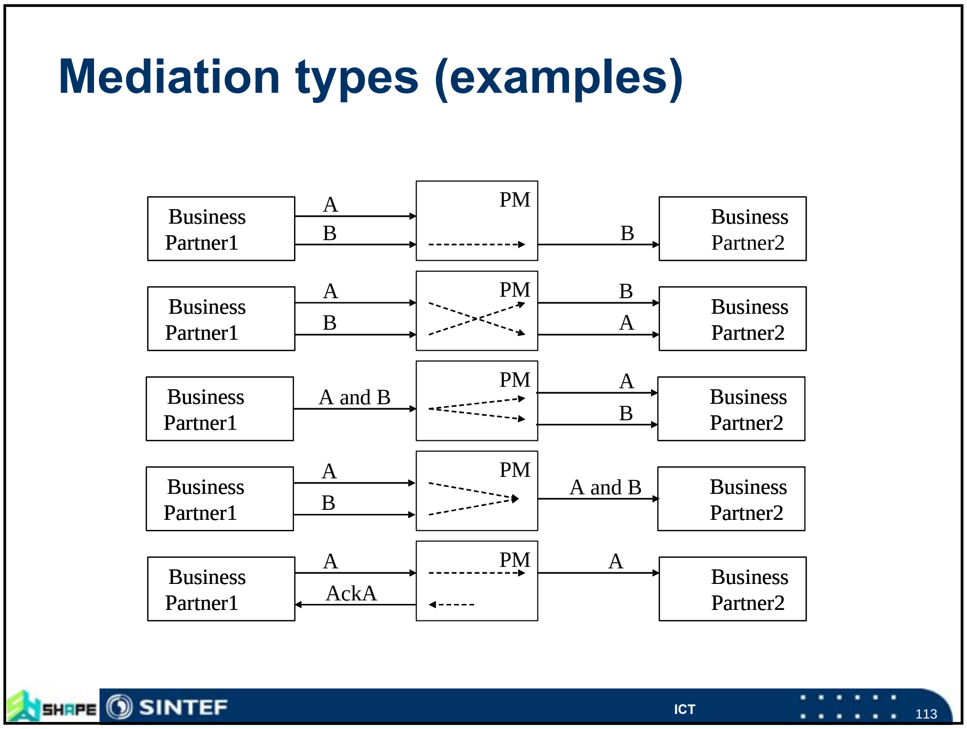
WW Mediator

- Web service – Web service mediator
- Enables interoperability of heterogeneous Web services
  → support automated collaboration between Web services
- Connects Web services, resolving any data, process and protocol heterogeneity between the two
  - OO Mediators for terminology import with data level mediation
  - Protocol Mediation for establishing valid multi-party collaborations
  - Process Mediation for making Business Processes interoperable



WW Mediator – Example

Source
Web service
“Selling novels”
WW Mediator
Target
Web service
“Selling books”
internal business logic of Web Service A
(not of interest in Service Interface Description)
WW Mediator
internal business logic of Web Service B
(not of interest in Service Interface Description)



Mediation types (examples)

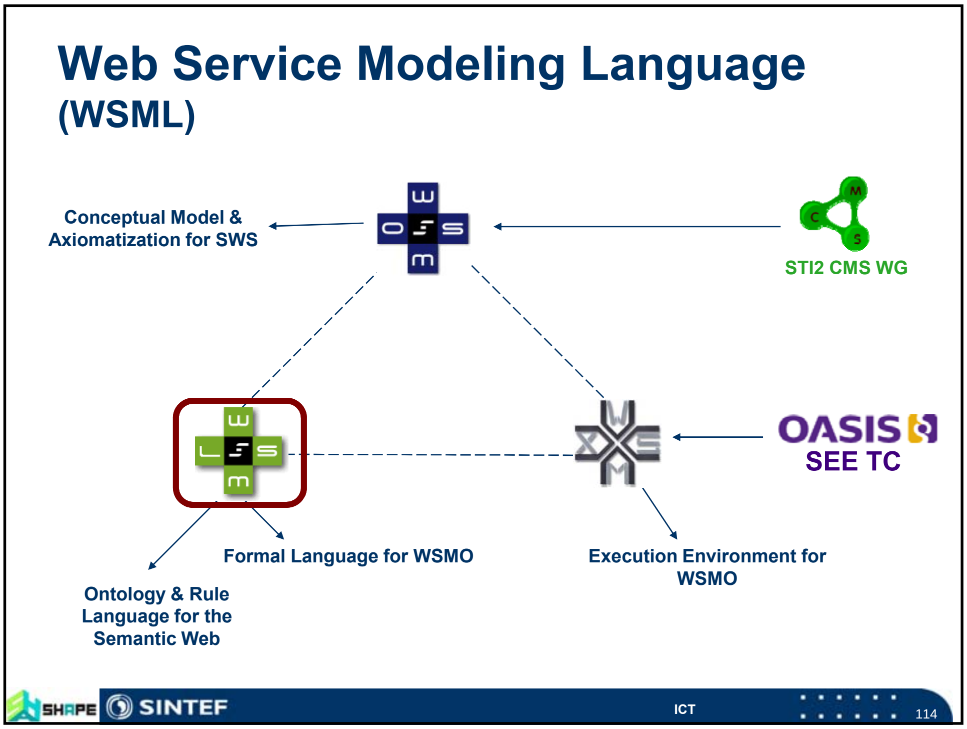
Business Partner1 — A, B → PM → B → Business Partner2

Business Partner1 — A, B → PM → B, A → Business Partner2

Business Partner1 — A and B → PM → A, B → Business Partner2

Business Partner1 — A, B → PM → A and B → Business Partner2

Business Partner1 — A → PM → A → Business Partner2; AckA ← PM



Web Service Modeling Language (WSML)

Conceptual Model & Axiomatization for SWS

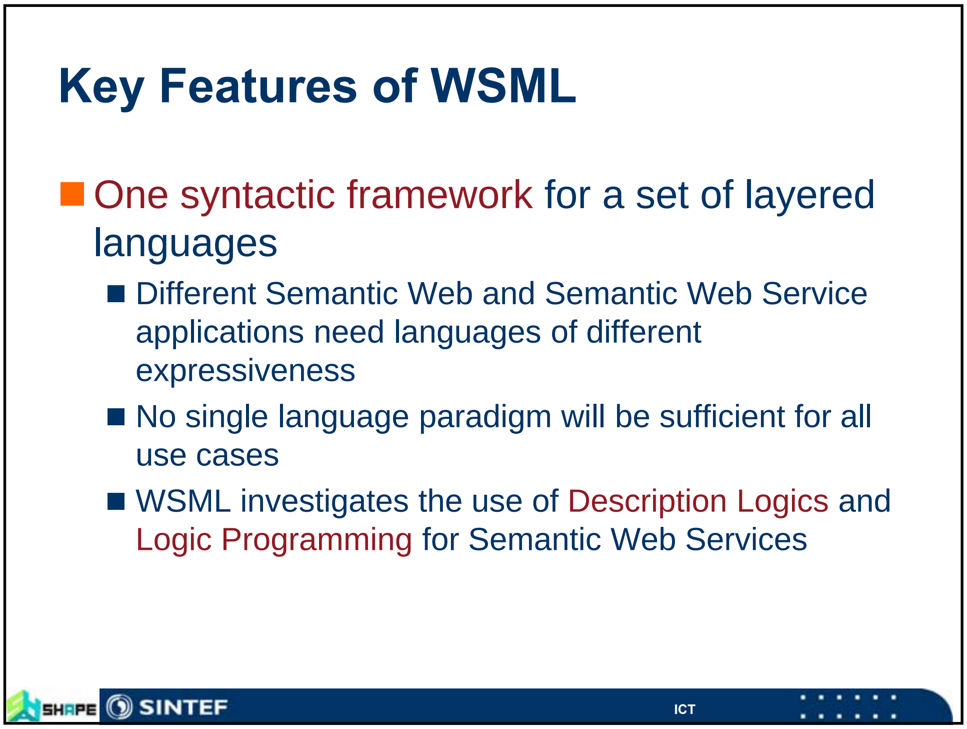
STI2 CMS WG

OASIS SEE TC

Formal Language for WSMO

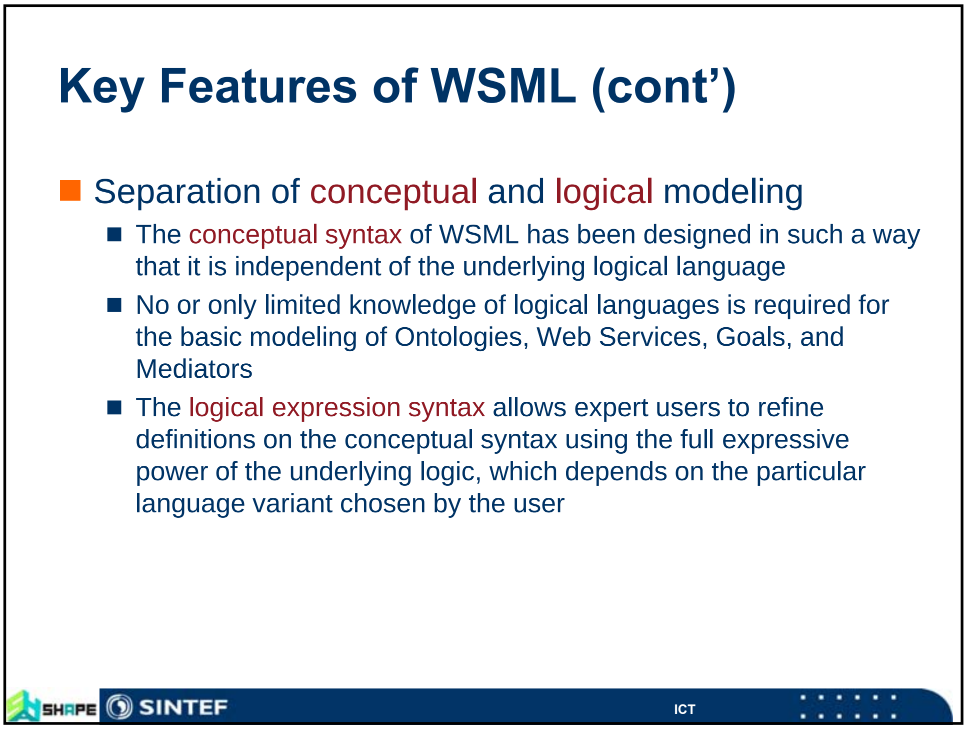
Execution Environment for WSMO

Ontology & Rule Language for the Semantic Web



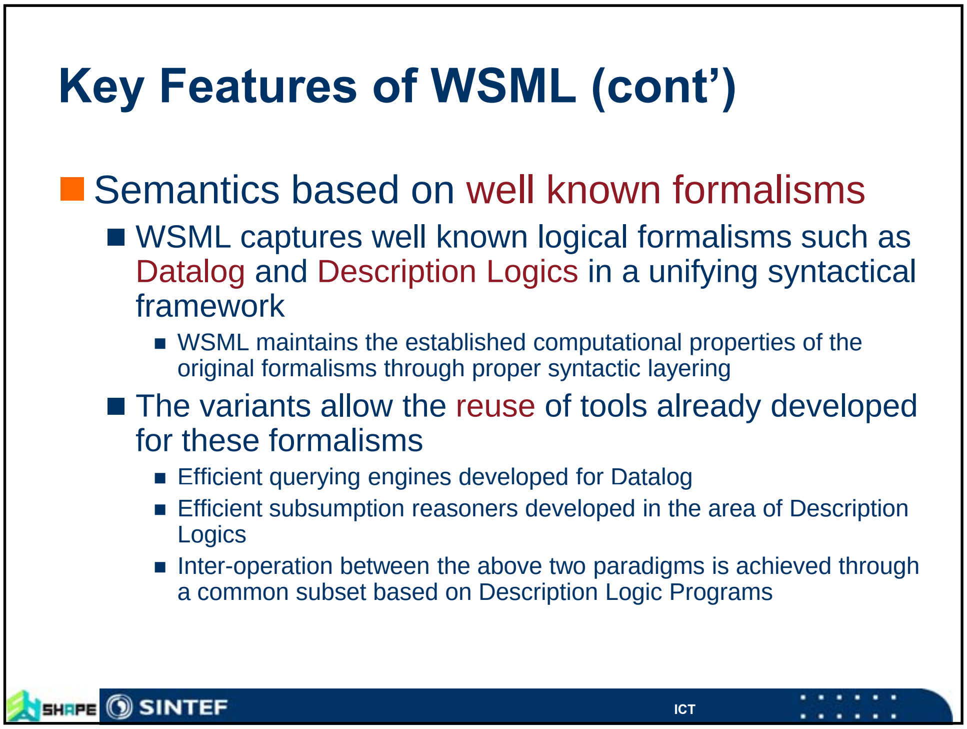
Key Features of WSML

- One syntactic framework for a set of layered languages
  - Different Semantic Web and Semantic Web Service applications need languages of different expressiveness
  - No single language paradigm will be sufficient for all use cases
  - WSML investigates the use of Description Logics and Logic Programming for Semantic Web Services



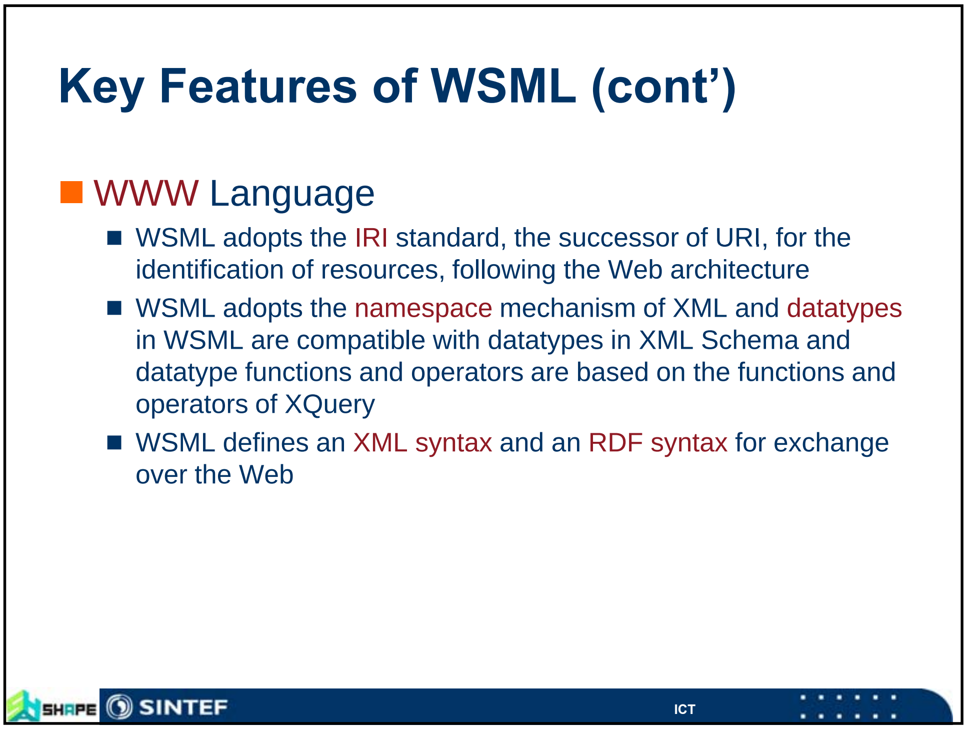
Key Features of WSML (cont')

- Separation of conceptual and logical modeling
  - The conceptual syntax of WSML has been designed in such a way that it is independent of the underlying logical language
  - No or only limited knowledge of logical languages is required for the basic modeling of Ontologies, Web Services, Goals, and Mediators
  - The logical expression syntax allows expert users to refine definitions on the conceptual syntax using the full expressive power of the underlying logic, which depends on the particular language variant chosen by the user



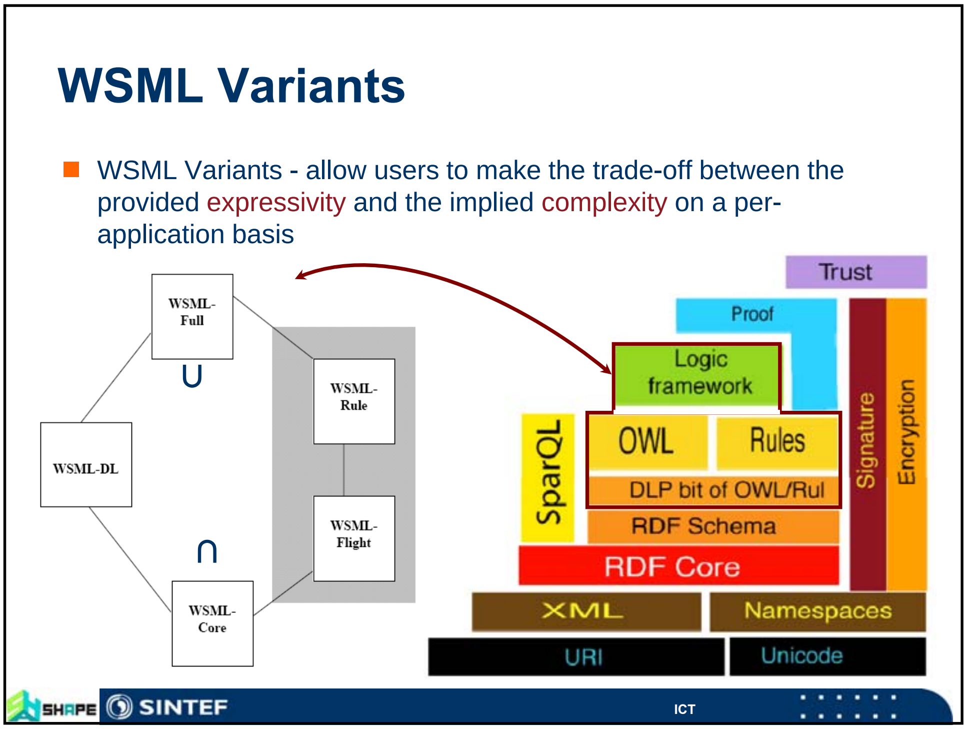
Key Features of WSML (cont')

- Semantics based on well known formalisms
  - WSML captures well known logical formalisms such as Datalog and Description Logics in a unifying syntactical framework
    - WSML maintains the established computational properties of the original formalisms through proper syntactic layering
  - The variants allow the reuse of tools already developed for these formalisms
    - Efficient querying engines developed for Datalog
    - Efficient subsumption reasoners developed in the area of Description Logics
    - Inter-operation between the above two paradigms is achieved through a common subset based on Description Logic Programs



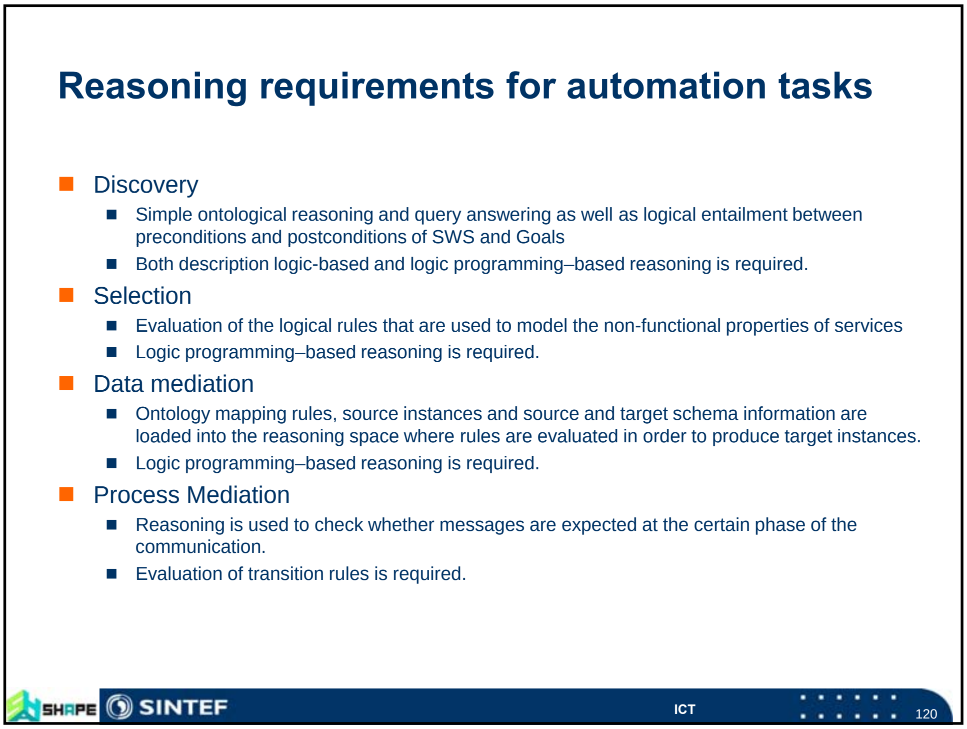
Key Features of WSML (cont')

- WWW Language
  - WSML adopts the IRI standard, the successor of URI, for the identification of resources, following the Web architecture
  - WSML adopts the namespace mechanism of XML and datatypes in WSML are compatible with datatypes in XML Schema and datatype functions and operators are based on the functions and operators of XQuery
  - WSML defines an XML syntax and an RDF syntax for exchange over the Web



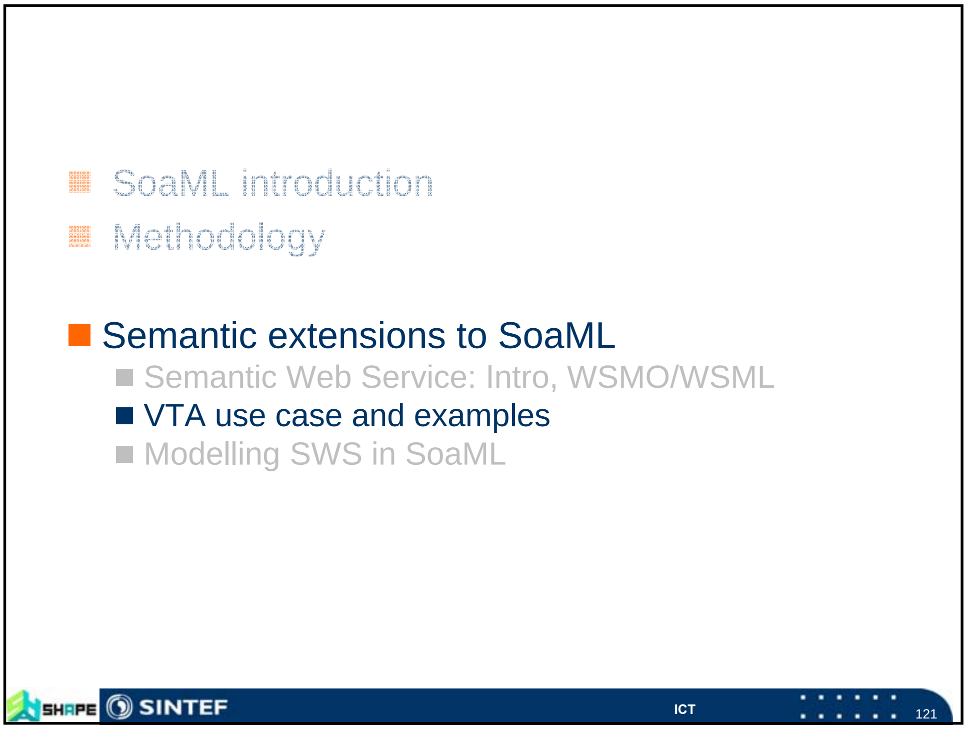
WSML Variants

- WSML Variants - allow users to make the trade-off between the provided expressivity and the implied complexity on a per-application basis



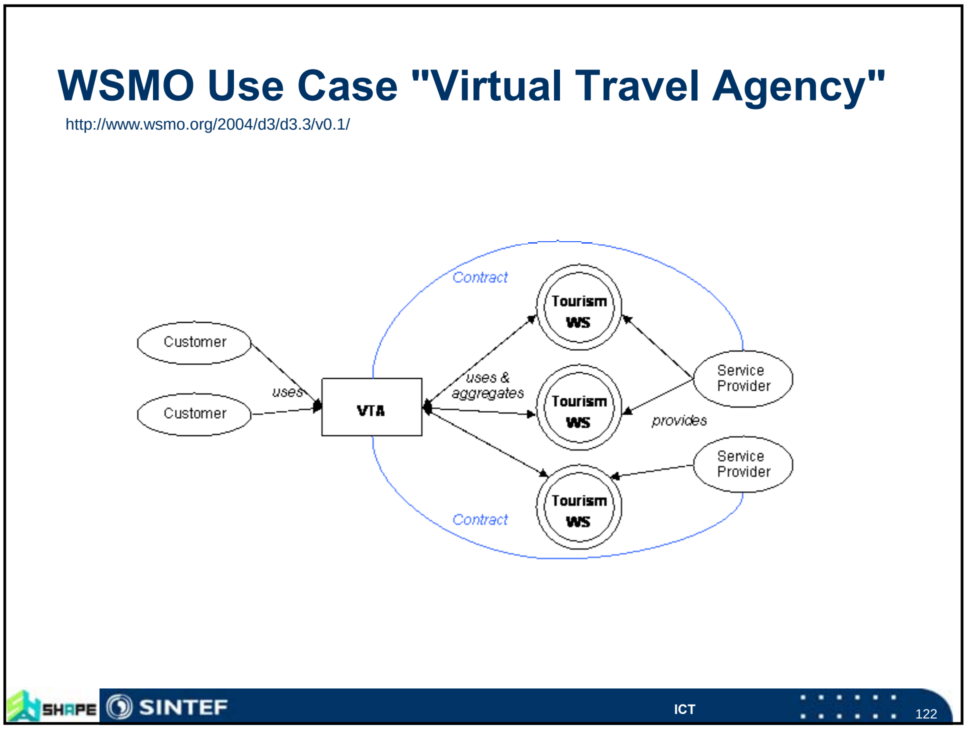
Reasoning requirements for automation tasks

- Discovery
  - Simple ontological reasoning and query answering as well as logical entailment between preconditions and postconditions of SWS and Goals
  - Both description logic-based and logic programming–based reasoning is required.
- Selection
  - Evaluation of the logical rules that are used to model the non-functional properties of services
  - Logic programming–based reasoning is required.
- Data mediation
  - Ontology mapping rules, source instances and source and target schema information are loaded into the reasoning space where rules are evaluated in order to produce target instances.
  - Logic programming–based reasoning is required.
- Process Mediation
  - Reasoning is used to check whether messages are expected at the certain phase of the communication.
  - Evaluation of transition rules is required.



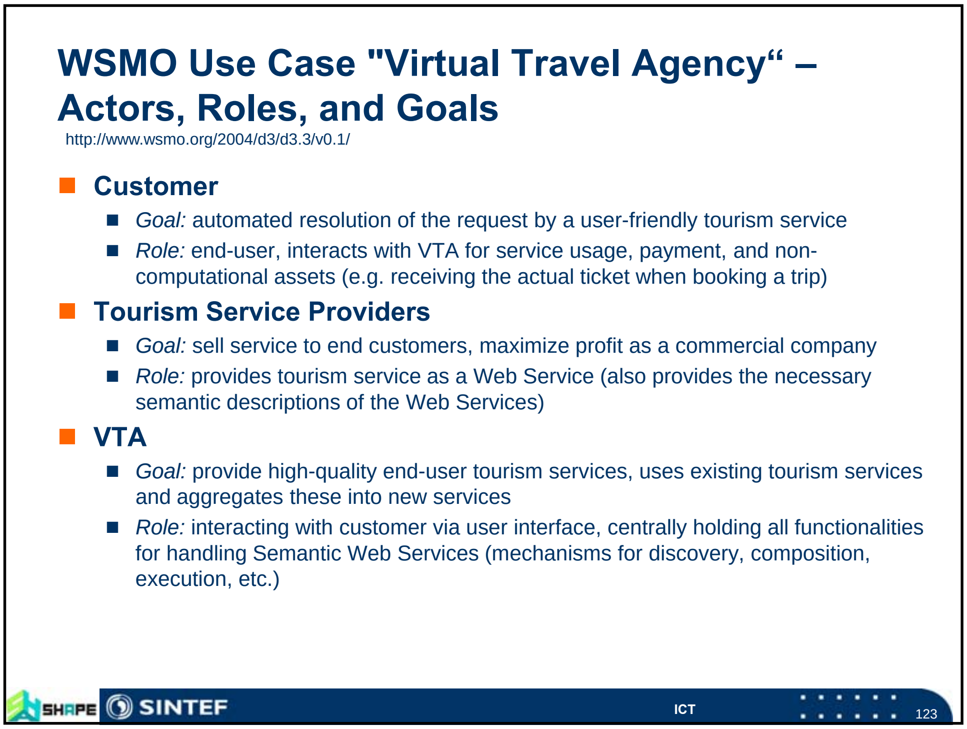
- SoaML introduction
- Methodology

- Semantic extensions to SoaML
  - Semantic Web Service: Intro, WSMO/WSML
  - VTA use case and examples
  - Modelling SWS in SoaML



WSMO Use Case "Virtual Travel Agency"

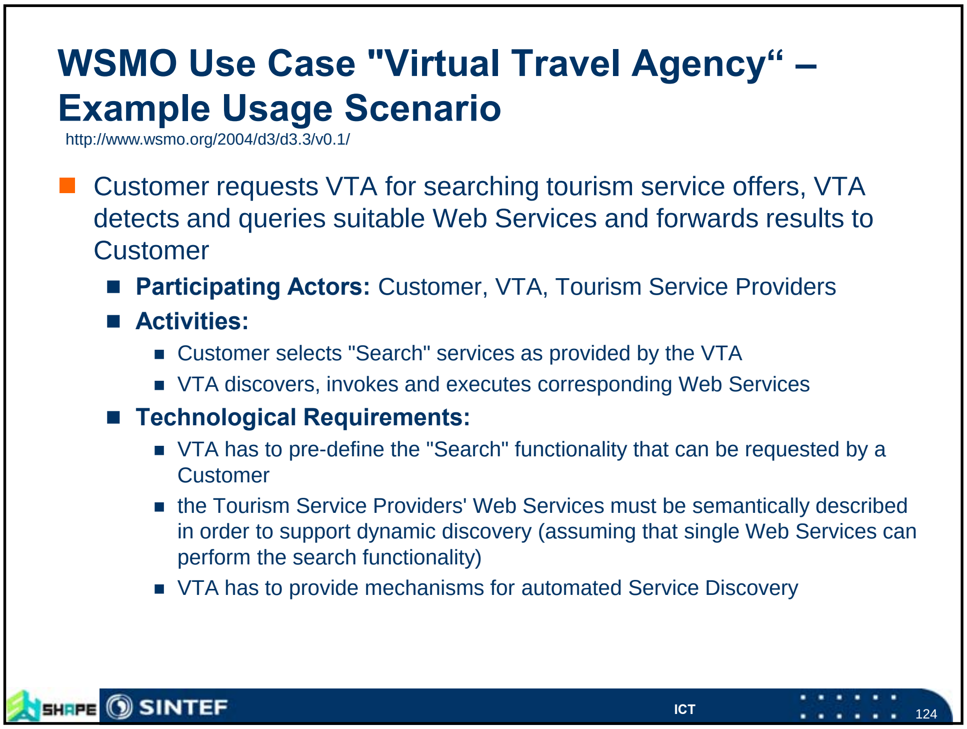
http://www.wsmo.org/2004/d3/d3.3/v0.1/



WSMO Use Case "Virtual Travel Agency“ – Actors, Roles, and Goals

http://www.wsmo.org/2004/d3/d3.3/v0.1/

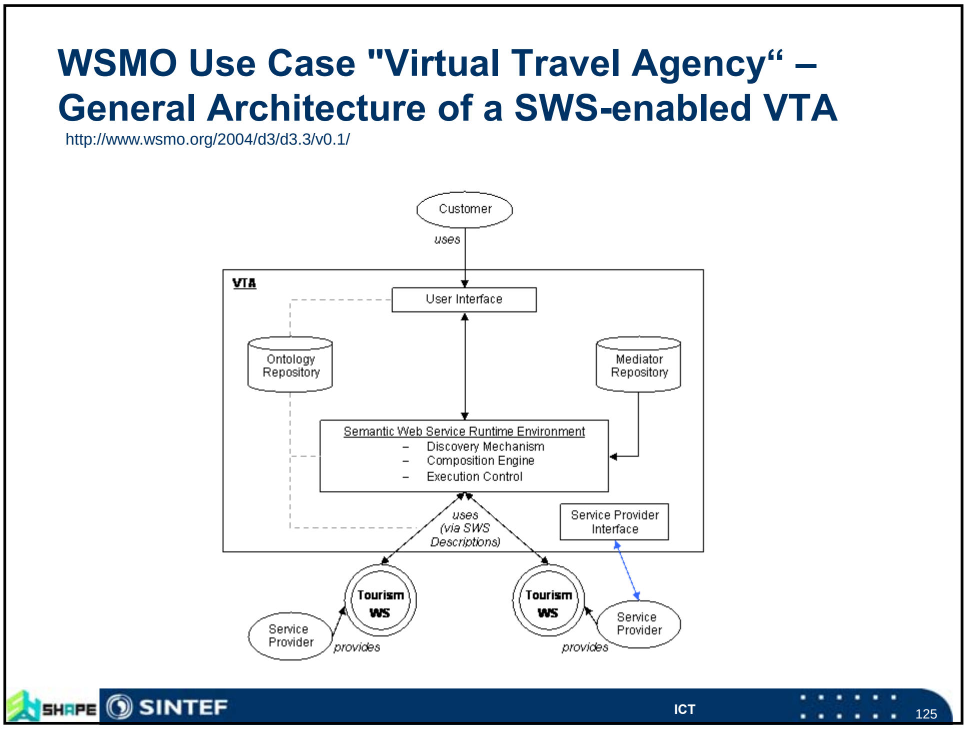
- **Customer**
  - *Goal:* automated resolution of the request by a user-friendly tourism service
  - *Role:* end-user, interacts with VTA for service usage, payment, and non-computational assets (e.g. receiving the actual ticket when booking a trip)
- **Tourism Service Providers**
  - *Goal:* sell service to end customers, maximize profit as a commercial company
  - *Role:* provides tourism service as a Web Service (also provides the necessary semantic descriptions of the Web Services)
- **VTA**
  - *Goal:* provide high-quality end-user tourism services, uses existing tourism services and aggregates these into new services
  - *Role:* interacting with customer via user interface, centrally holding all functionalities for handling Semantic Web Services (mechanisms for discovery, composition, execution, etc.)



WSMO Use Case "Virtual Travel Agency“ – Example Usage Scenario

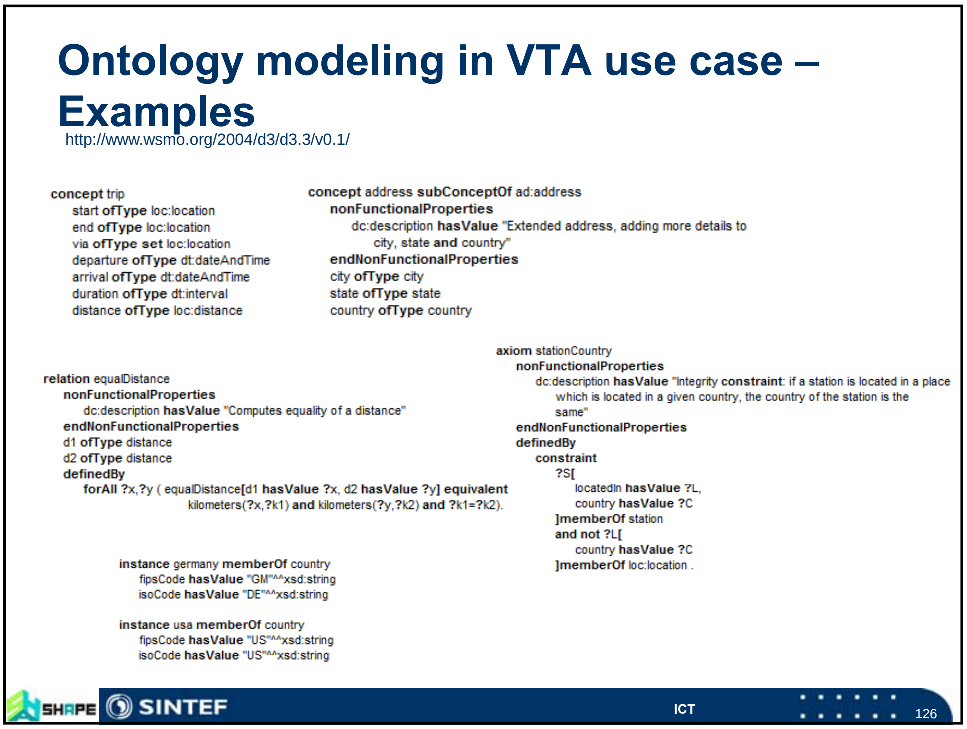
http://www.wsmo.org/2004/d3/d3.3/v0.1/

- Customer requests VTA for searching tourism service offers, VTA detects and queries suitable Web Services and forwards results to Customer
  - **Participating Actors:** Customer, VTA, Tourism Service Providers
  - **Activities:**
    - Customer selects "Search" services as provided by the VTA
    - VTA discovers, invokes and executes corresponding Web Services
  - **Technological Requirements:**
    - VTA has to pre-define the "Search" functionality that can be requested by a Customer
    - the Tourism Service Providers' Web Services must be semantically described in order to support dynamic discovery (assuming that single Web Services can perform the search functionality)
    - VTA has to provide mechanisms for automated Service Discovery



WSMO Use Case "Virtual Travel Agency“ – General Architecture of a SWS-enabled VTA

http://www.wsmo.org/2004/d3/d3.3/v0.1/



Ontology modeling in VTA use case – Examples

http://www.wsmo.org/2004/d3/d3.3/v0.1/

```
concept trip
    start ofType loc:location
    end ofType loc:location
    via ofType set loc:location
    departure ofType dt:dateAndTime
    arrival ofType dt:dateAndTime
    duration ofType dt:interval
    distance ofType loc:distance
```

```
concept address subConceptOf ad:address
    nonFunctionalProperties
        dc:description hasValue "Extended address, adding more details to
            city, state and country"
    endNonFunctionalProperties
    city ofType city
    state ofType state
    country ofType country
```

```
relation equalDistance
    nonFunctionalProperties
        dc:description hasValue "Computes equality of a distance"
    endNonFunctionalProperties
    d1 ofType distance
    d2 ofType distance
    definedBy
        forAll ?x,?y ( equalDistance[d1 hasValue ?x, d2 hasValue ?y] equivalent
                            kilometers(?x,?k1) and kilometers(?y,?k2) and ?k1=?k2).
```

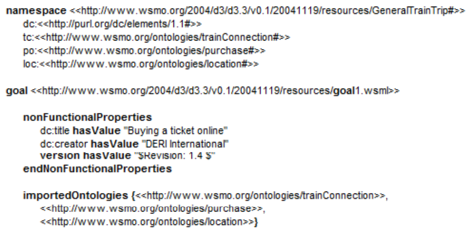
```
axiom stationCountry
    nonFunctionalProperties
        dc:description hasValue "Integrity constraint: if a station is located in a place
            which is located in a given country, the country of the station is the
            same"
    endNonFunctionalProperties
    definedBy
        constraint
            ?S[
                locatedIn hasValue ?L,
                country hasValue ?C
            ]memberOf station
            and not ?L[
                country hasValue ?C
            ]memberOf loc:location .
```

```
instance germany memberOf country
    fipsCode hasValue "GM"^^xsd:string
    isoCode hasValue "DE"^^xsd:string

instance usa memberOf country
    fipsCode hasValue "US"^^xsd:string
    isoCode hasValue "US"^^xsd:string
```

Goal modeling in VTA use case – Example

http://www.wsmo.org/2004/d3/d3.3/v0.1/

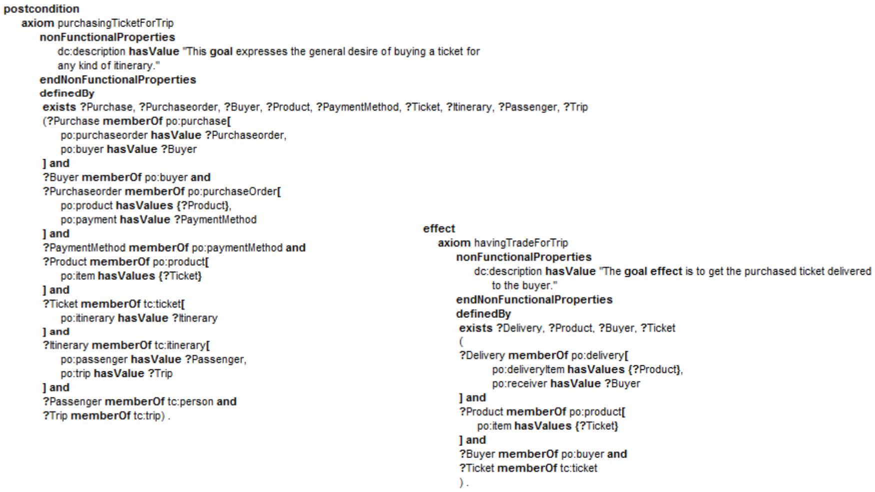
- Goal: buy a ticket online
  - **postcondition:** a purchase for a buyer with a ticket as the product
  - **effect:** the purchased ticket is delivered to the buyer



```
namespace <<http://www.wsmo.org/2004/d3/d3.3/v0.1/20041119/resources/GeneralTrainTrip#>>
    dc:<<http://purl.org/dc/elements/1.1#>>
    tc:<<http://www.wsmo.org/ontologies/trainConnection#>>
    po:<<http://www.wsmo.org/ontologies/purchase#>>
    loc:<<http://www.wsmo.org/ontologies/location#>>

goal <<http://www.wsmo.org/2004/d3/d3.3/v0.1/20041119/resources/goal1.wsml>>

    nonFunctionalProperties
        dc:title hasValue "Buying a ticket online"
        dc:creator hasValue "DERI International"
        version hasValue "$Revision: 1.4 $"
    endNonFunctionalProperties

    importedOntologies {<<http://www.wsmo.org/ontologies/trainConnection>>,
        <<http://www.wsmo.org/ontologies/purchase>>,
        <<http://www.wsmo.org/ontologies/location>>}
```

Goal modeling in VTA use case – Example (cont')

http://www.wsmo.org/2004/d3/d3.3/v0.1/



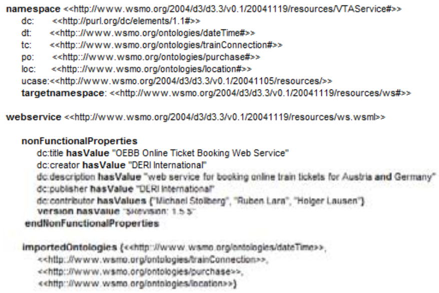
```
postcondition
    axiom purchasingTicketForTrip
        nonFunctionalProperties
            dc:description hasValue "This goal expresses the general desire of buying a ticket for
            any kind of itinerary."
        endNonFunctionalProperties
        definedBy
         exists ?Purchase, ?Purchaseorder, ?Buyer, ?Product, ?PaymentMethod, ?Ticket, ?Itinerary, ?Passenger, ?Trip
         (?Purchase memberOf po:purchase[
             po:purchaseorder hasValue ?Purchaseorder,
             po:buyer hasValue ?Buyer
         ] and
         ?Buyer memberOf po:buyer and
         ?Purchaseorder memberOf po:purchaseOrder[
             po:product hasValues {?Product},
             po:payment hasValue ?PaymentMethod
         ] and
         ?PaymentMethod memberOf po:paymentMethod and
         ?Product memberOf po:product[
             po:item hasValues {?Ticket}
         ] and
         ?Ticket memberOf tc:ticket[
             po:itinerary hasValue ?Itinerary
         ] and
         ?Itinerary memberOf tc:itinerary[
             po:passenger hasValue ?Passenger,
             po:trip hasValue ?Trip
         ] and
         ?Passenger memberOf tc:person and
         ?Trip memberOf tc:trip) .
```

```
effect
    axiom havingTradeForTrip
        nonFunctionalProperties
            dc:description hasValue "The goal effect is to get the purchased ticket delivered
                to the buyer."
        endNonFunctionalProperties
        definedBy
         exists ?Delivery, ?Product, ?Buyer, ?Ticket
         (
         ?Delivery memberOf po:delivery[
                po:deliveryItem hasValues {?Product},
                po:receiver hasValue ?Buyer
         ] and
         ?Product memberOf po:product[
             po:item hasValues {?Ticket}
         ] and
         ?Buyer memberOf po:buyer and
         ?Ticket memberOf tc:ticket
         ) .
```

Web Service modeling in VTA use case – Example

http://www.wsmo.org/2004/d3/d3.3/v0.1/

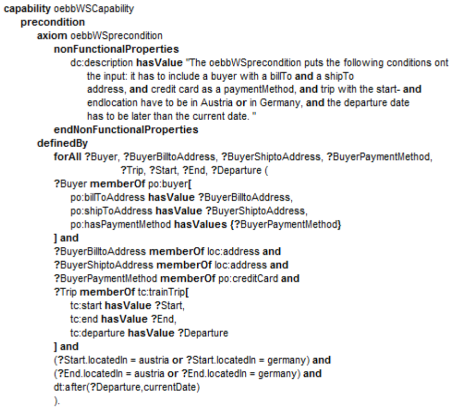
- Web service: booking online train tickets
  - **precondition:** the input has to be the information about the buyer, a train trip that a ticket is to be purchased for, and information on the passenger for whom the ticket shall be valid. To be valid input, the following restrictions are defined for the trip: the start and end locations are restricted to stations in Austria or Germany; the departure date for the trip has to be after the current date; and the payment method of the buyer has to be a credit card.
  - **effect:** the sold ticket is delivered to the buyer shipping address, either by a drop ship carrieror via online delivery.



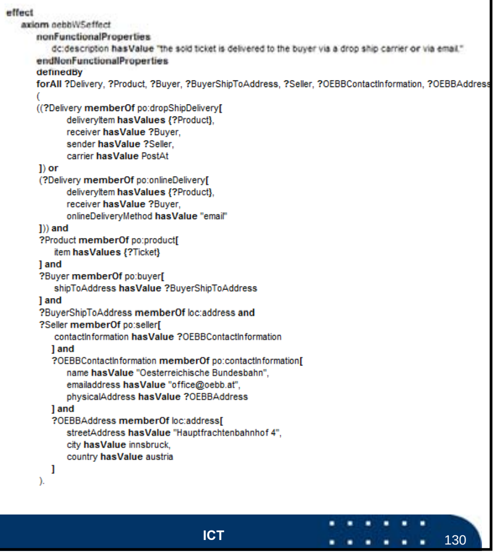
```
namespace <<http://www.wsmo.org/2004/d3/d3.3/v0.1/20041119/resources/VTAService#>>
    dc:      <<http://purl.org/dc/elements/1.1#>>
    dt:      <<http:://www.wsmo.org/ontologies/dateTime#>>
    tc:      <<http:://www.wsmo.org/ontologies/trainConnection#>>
    po:      <<http:://www.wsmo.org/ontologies/purchase#>>
    loc:     <<http:://www.wsmo.org/ontologies/location#>>
    ucase:<<http://www.wsmo.org/2004/d3/d3.3/v0.1/20041105/resources/>>
    targetnamespace: <<http://www.wsmo.org/2004/d3/d3.3/v0.1/20041119/resources/ws#>>

webservice <<http://www.wsmo.org/2004/d3/d3.3/v0.1/20041119/resources/ws.wsml>>

    nonFunctionalProperties
        dc:title hasValue "OEBB Online Ticket Booking Web Service"
        dc:creator hasValue "DERI International"
        dc:description hasValue "web service for booking online train tickets for Austria and Germany"
        dc:publisher hasValue "DERI International"
        dc:contributor hasValues {"Michael Stolberg", "Ruben Lara", "Holger Lausen"}
        version hasValue "$Revision: 1.5 $"
     endNonFunctionalProperties

    importedOntologies {<<http:://www.wsmo.org/ontologies/dateTime>>,
        <<http:://www.wsmo.org/ontologies/trainConnection>>,
        <<http:://www.wsmo.org/ontologies/purchase>>,
        <<http:://www.wsmo.org/ontologies/location>>}
```

Web Service modeling in VTA use case – Example (cont')

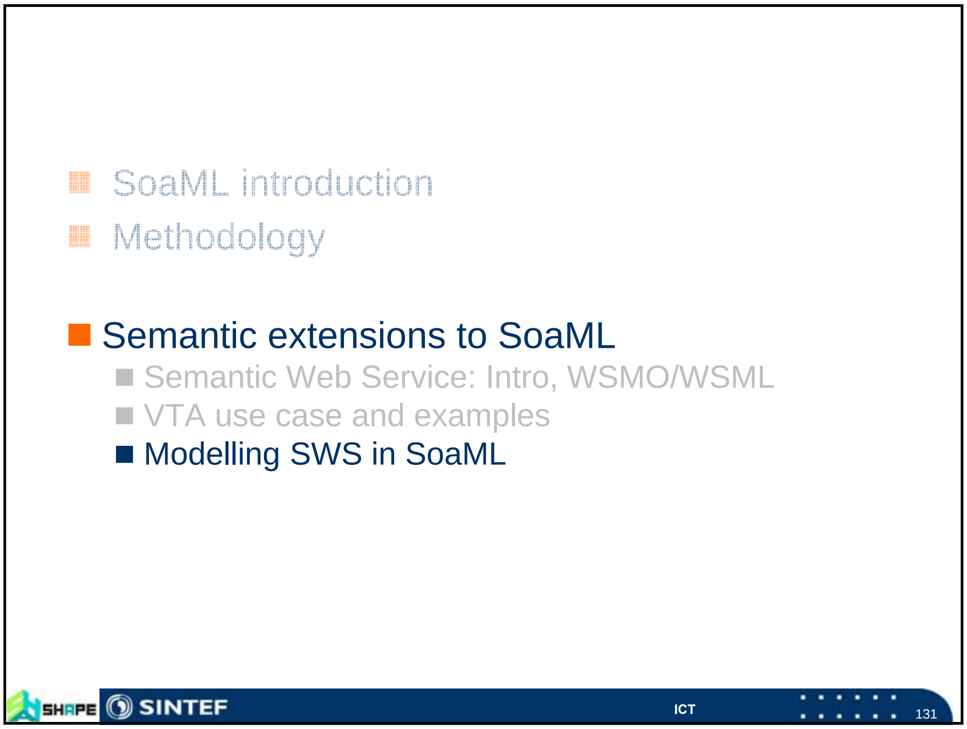
http://www.wsmo.org/2004/d3/d3.3/v0.1/



```
capability oebbWSCapability
    precondition
        axiom oebbWSprecondition
            nonFunctionalProperties
                dc:description hasValue "The oebbWSprecondition puts the following conditions ont
                    the input: it has to include a buyer with a billTo and a shipTo
                    address, and credit card as a paymentMethod, and trip with the start- and
                    endlocation have to be in Austria or in Germany, and the departure date
                    has to be later than the current date. "
            endNonFunctionalProperties
        definedBy
            forAll ?Buyer, ?BuyerBilltoAddress, ?BuyerShiptoAddress, ?BuyerPaymentMethod,
                        ?Trip, ?Start, ?End, ?Departure (
            ?Buyer memberOf po:buyer[
                po:billToAddress hasValue ?BuyerBilltoAddress,
                po:shipToAddress hasValue ?BuyerShiptoAddress,
                po:hasPaymentMethod hasValues {?BuyerPaymentMethod}
            ] and
            ?BuyerBilltoAddress memberOf loc:address and
            ?BuyerShiptoAddress memberOf loc:address and
            ?BuyerPaymentMethod memberOf po:creditCard and
            ?Trip memberOf tc:trainTrip[
                tc:start hasValue ?Start,
                tc:end hasValue ?End,
                tc:departure hasValue ?Departure
            ] and
            (?Start.locatedIn = austria or ?Start.locatedIn = germany) and
            (?End.locatedIn = austria or ?End.locatedIn = germany) and
            dt:after(?Departure,currentDate)
            ).
```

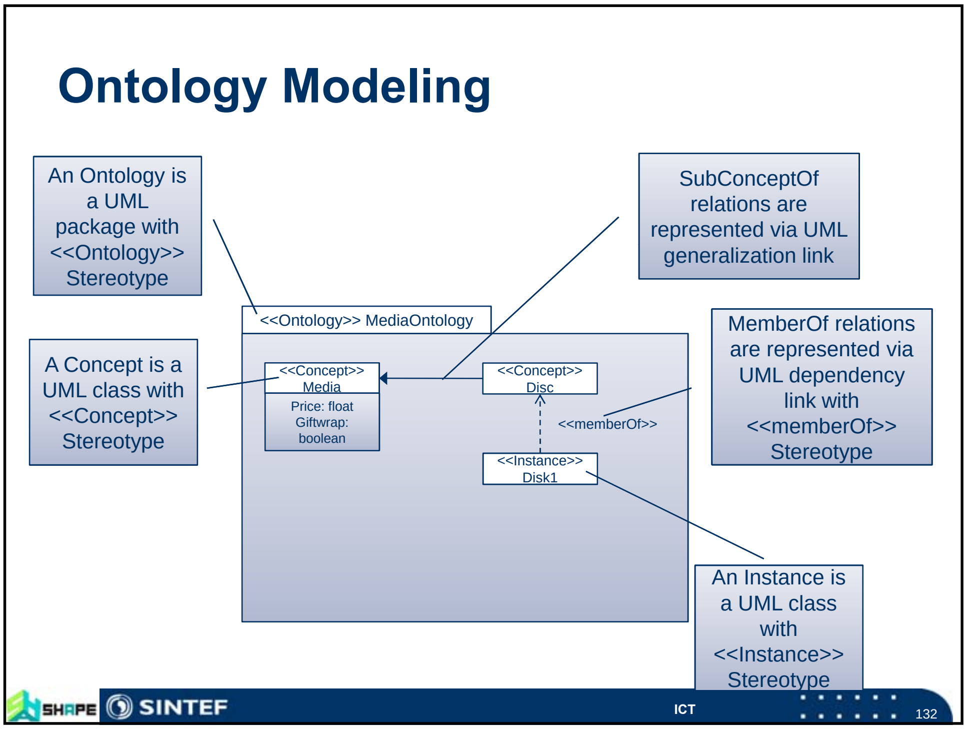


```
effect
    axiom oebbWSeffect
        nonFunctionalProperties
            dc:description hasValue "the sold ticket is delivered to the buyer via a drop ship carrier or via email."
        endNonFunctionalProperties
        definedBy
        forAll ?Delivery, ?Product, ?Buyer, ?BuyerShipToAddress, ?Seller, ?OEBBContactInformation, ?OEBBAddress
        (
        ((?Delivery memberOf po:dropShipDelivery[
                deliveryItem hasValues {?Product},
                receiver hasValue ?Buyer,
                sender hasValue ?Seller,
                carrier hasValue PostAt
         ]) or
         (?Delivery memberOf po:onlineDelivery[
                deliveryItem hasValues {?Product},
                receiver hasValue ?Buyer,
                onlineDeliveryMethod hasValue "email"
         ])) and
         ?Product memberOf po:product[
             item hasValues {?Ticket}
         ] and
         ?Buyer memberOf po:buyer[
             shipToAddress hasValue ?BuyerShipToAddress
         ] and
         ?BuyerShipToAddress memberOf loc:address and
         ?Seller memberOf po:seller[
             contactInformation hasValue ?OEBBContactInformation
            ] and
            ?OEBBContactInformation memberOf po:contactInformation[
                name hasValue "Oesterreichische Bundesbahn",
                emailaddress hasValue "office@oebb.at",
                physicalAddress hasValue ?OEBBAddress
            ] and
            ?OEBBAddress memberOf loc:address[
                streetAddress hasValue "Hauptfrachtenbahnhof 4",
                city hasValue innsbruck,
                country hasValue austria
            ]
         ).
```



- SoaML introduction
- Methodology

- Semantic extensions to SoaML
  - Semantic Web Service: Intro, WSMO/WSML
  - VTA use case and examples
  - Modelling SWS in SoaML



Ontology Modeling

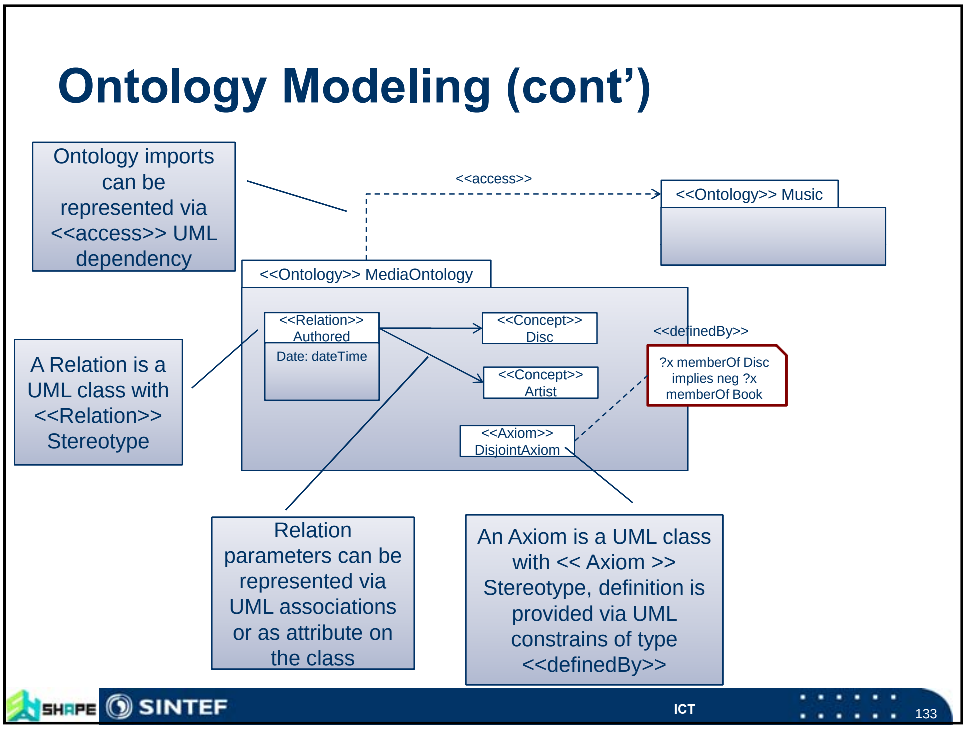
An Ontology is a UML package with <<Ontology>> Stereotype

A Concept is a UML class with <<Concept>> Stereotype

SubConceptOf relations are represented via UML generalization link

MemberOf relations are represented via UML dependency link with <<memberOf>> Stereotype

An Instance is a UML class with <<Instance>> Stereotype



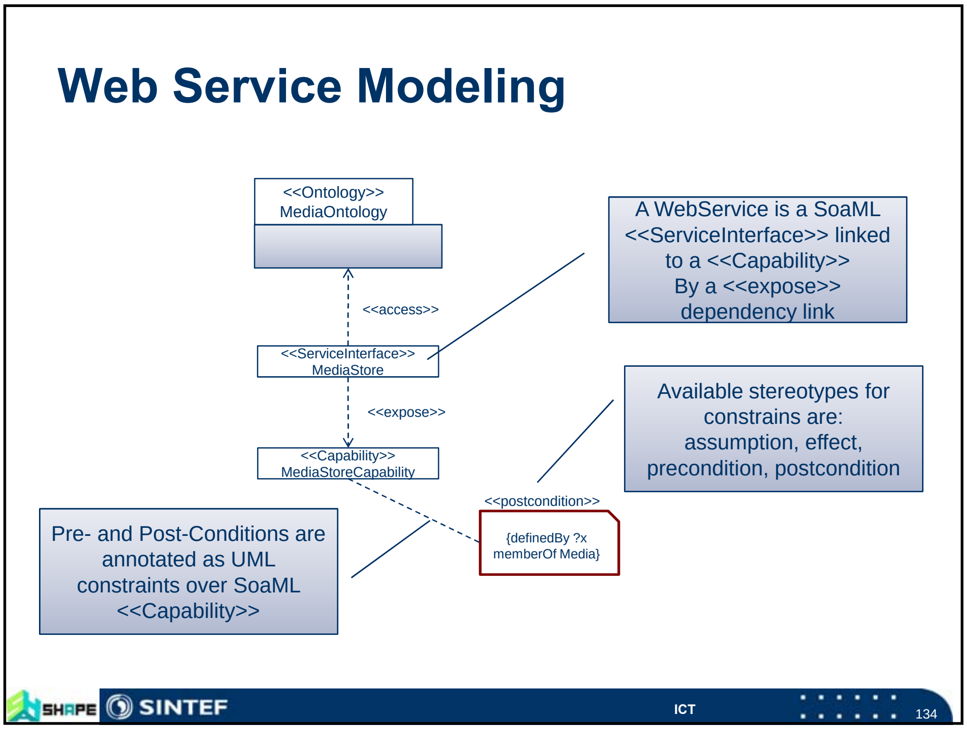
Ontology Modeling (cont')

Ontology imports can be represented via <<access>> UML dependency

A Relation is a UML class with <<Relation>> Stereotype

Relation parameters can be represented via UML associations or as attribute on the class

An Axiom is a UML class with << Axiom >> Stereotype, definition is provided via UML constrains of type <<definedBy>>



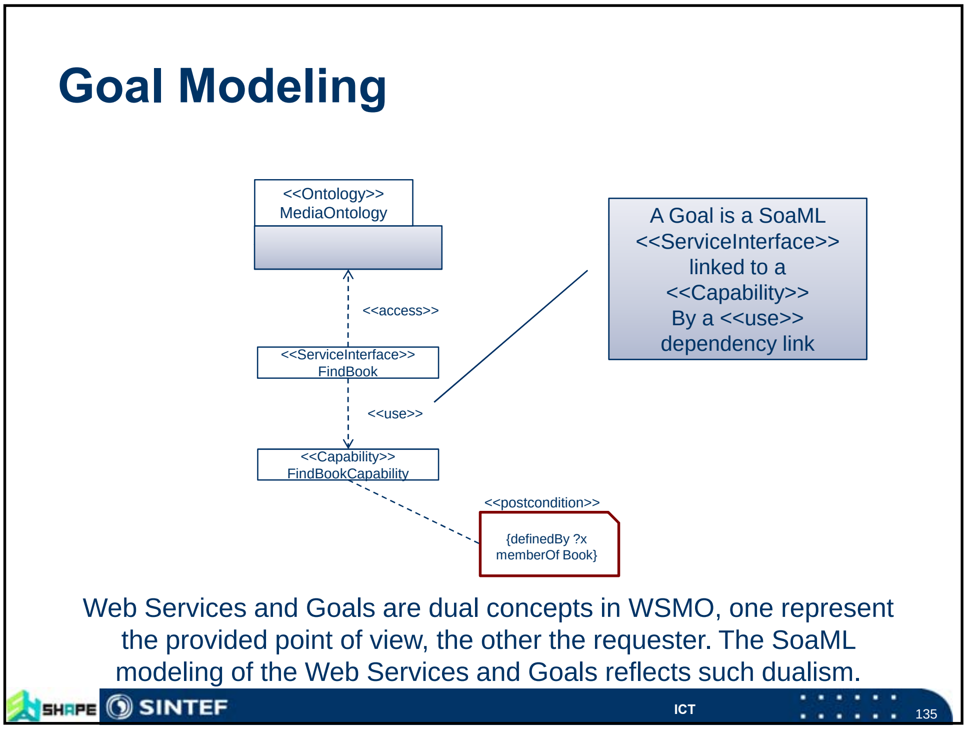
Web Service Modeling

A WebService is a SoaML <<ServiceInterface>> linked to a <<Capability>> By a <<expose>> dependency link

Available stereotypes for constrains are: assumption, effect, precondition, postcondition

Pre- and Post-Conditions are annotated as UML constraints over SoaML <<Capability>>

<<Ontology>> MediaOntology
<<access>>
<<ServiceInterface>> MediaStore
<<expose>>
<<Capability>> MediaStoreCapability
<<postcondition>>
{definedBy ?x memberOf Media}



Goal Modeling

<<Ontology>>
MediaOntology

<<access>>

<<ServiceInterface>>
FindBook

<<use>>

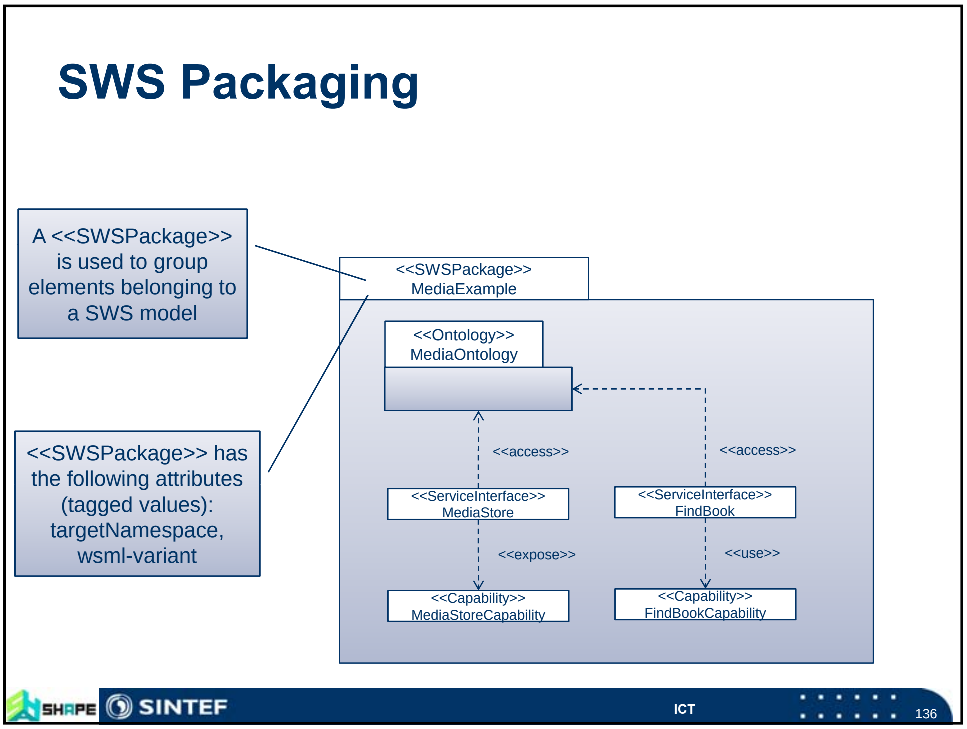
<<Capability>>
FindBookCapability

<<postcondition>>

{definedBy ?x memberOf Book}

A Goal is a SoaML <<ServiceInterface>> linked to a <<Capability>> By a <<use>> dependency link

Web Services and Goals are dual concepts in WSMO, one represent the provided point of view, the other the requester. The SoaML modeling of the Web Services and Goals reflects such dualism.



SWS Packaging

A <<SWSPackage>> is used to group elements belonging to a SWS model

<<SWSPackage>> has the following attributes (tagged values): targetNamespace, wsml-variant

<<SWSPackage>>
MediaExample

<<Ontology>>
MediaOntology

<<access>>

<<access>>

<<ServiceInterface>>
MediaStore

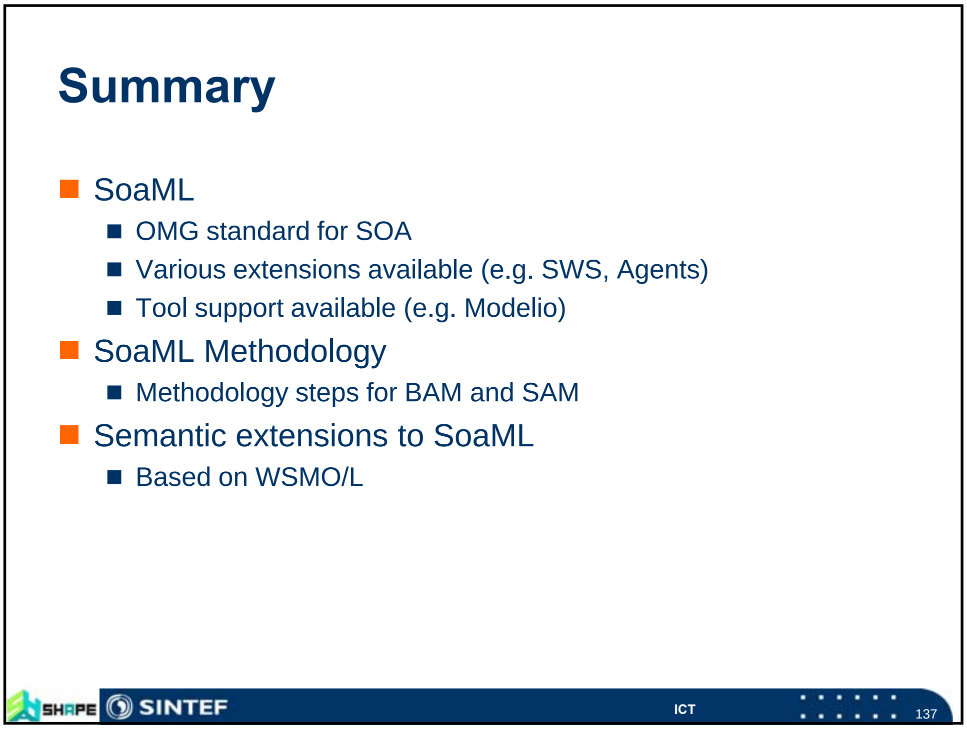
<<ServiceInterface>>
FindBook

<<expose>>

<<use>>

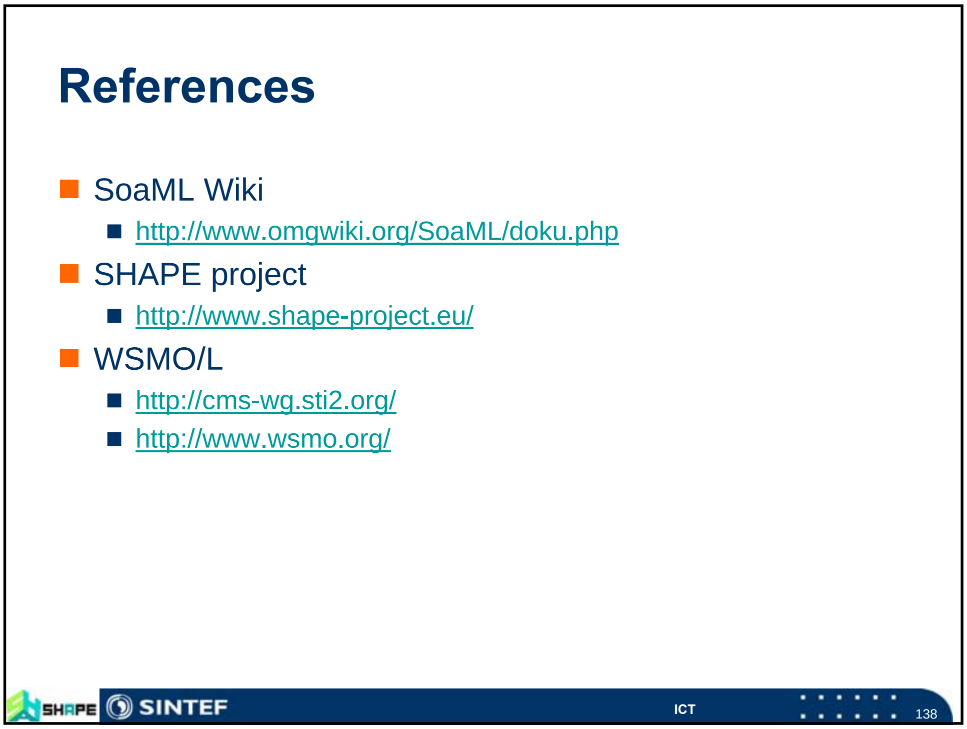
<<Capability>>
MediaStoreCapability

<<Capability>>
FindBookCapability



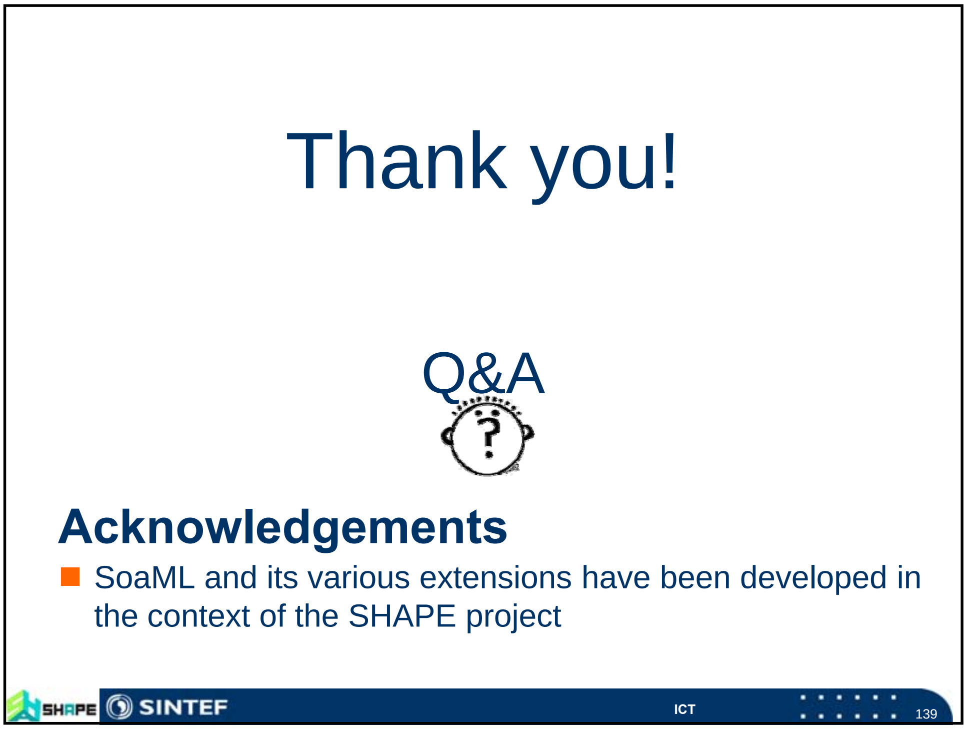
Summary

- SoaML
  - OMG standard for SOA
  - Various extensions available (e.g. SWS, Agents)
  - Tool support available (e.g. Modelio)
- SoaML Methodology
  - Methodology steps for BAM and SAM
- Semantic extensions to SoaML
  - Based on WSMO/L



References

- SoaML Wiki
  - http://www.omgwiki.org/SoaML/doku.php
- SHAPE project
  - http://www.shape-project.eu/
- WSMO/L
  - http://cms-wg.sti2.org/
  - http://www.wsmo.org/



Thank you!

Q&A

Acknowledgements

- SoaML and its various extensions have been developed in the context of the SHAPE project

SHAPE SINTEF ICT