Model Driven Solutions

Where Business Meets Technology

A Division of Data Access Technologies, Inc.

OBJECT MANAGEMENT GROUP

OMG SOA SIG Activity

Ed Harrington
ed-h “at” modeldriven.com

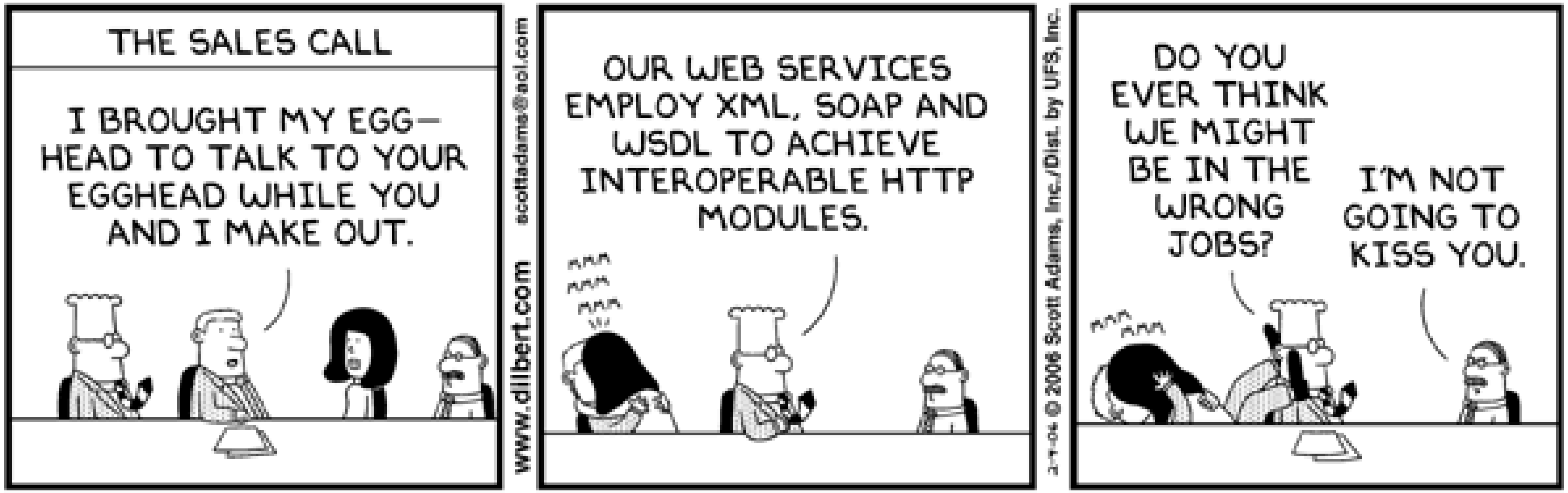
San Diego, California, USA
4 February 2009

Agenda

- OMG SOA SIG: Mission & Participants
- OMG's approach to SOA & Value Proposition
- Summary of gap analysis effort
- Summary of Standards in development
- SoaML
- SIG Next Steps

SOA – the fun part! *Dilbert's SOA Predicament!*

http://www.j2eegeek.com/blog/2006/02/04/dilbert-does-web-services/


THE SALES CALL
I BROUGHT MY EGG-HEAD TO TALK TO YOUR EGGHEAD WHILE YOU AND I MAKE OUT.
scottadams@aol.com
www.dilbert.com
OUR WEB SERVICES EMPLOY XML, SOAP AND WSDL TO ACHIEVE INTEROPERABLE HTTP MODULES.
MMM
MMM
MMM
2-7-06 © 2006 Scott Adams, Inc./Dist. by UFS, Inc.
DO YOU EVER THINK WE MIGHT BE IN THE WRONG JOBS?
I'M NOT GOING TO KISS YOU.
MMM
MMM

© Scott Adams, Inc./Dist. by UFS, Inc.

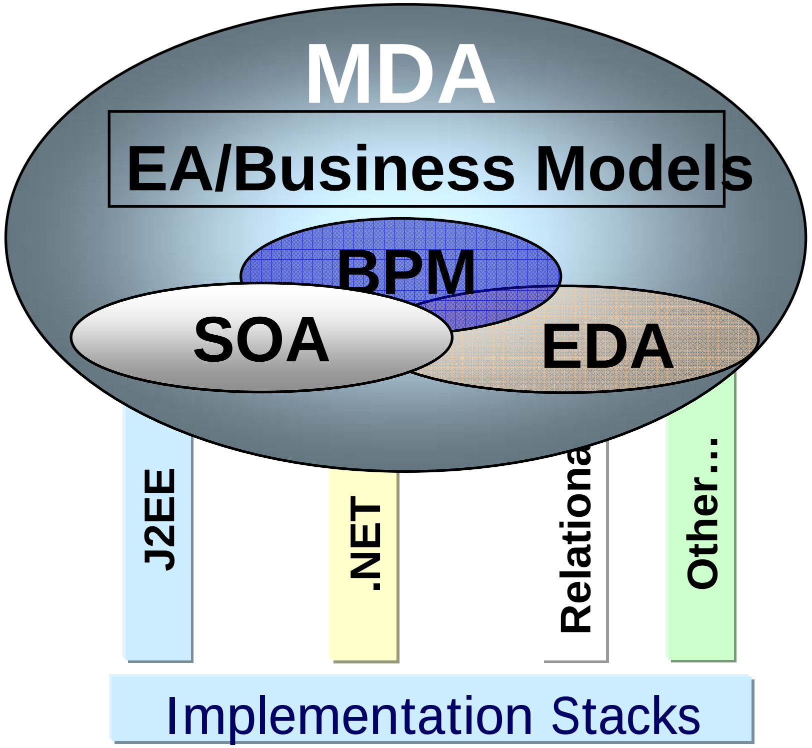
- **Problem is… the business folks have no idea what the Eggheads are talking about**
  - **Truth be told, one of biggest stumbling blocks to SOA Success is the difficulty of establishing a meaningful dialogue between business and IT – a.k.a.**

"The SOA Chasm"

SOA SIG Mission & Participants

Mission

- Support a Model Driven Architecture (MDA) approach to SOA that links architectural, business and technology views of Services, including Business Process Management (BPM) and Event-Driven Architecture (EDA)
- Identify and foster development of OMG modeling standards for SOA that integrate with and complement standards developed by other organizations such as W3C, Open Group and OASIS…
- Formally established in Q4, 2005



Participants

IBM, HP, SAP, BEA, Unisys, EDS, RTI, Model Driven Solutions, Adaptive, MEGA, MetLife, VISA, Bank of America, CitiGroup, Boeing, Raytheon, Rhysome, AptSoft, XAware, SOA COP, SOA Consortium, The Open Group, OASIS, US EPA, ebizQ, Georgetown University, NIST, Liberty-Mutual, Penn National, Prima-Solutions, OSOA, Nortel, Indiana University, Capgemini…

OMG’s Approach & Value Proposition

- While other organizations have focused on specific standards for integration or web services protocols (e.g., WS-* standards), OMG complements their efforts by taking a Platform Independent View and applying a Modeling Approach to SOA

Modeling is OMG’s Core Competence

- Modeling offers the capability to design a complete SOA solution
- Intellectual property documented through models can have a longer lifespan, allowing an organization a choice of the best technical platform for SOA implementation

Summary of Gap Analysis Effort

- Modeling approach to SOA*
- Service Traceability to Business Processes*
- Service Relationship to Events*
- Service Discovery & Assets
- SOA Life Cycle Metrics and Maturity
- SOA Governance & Compliance*

* Gaps OMG SOA SIG has started to work on

Summary of Standards

- UML Profile and Metamodel for Services (UPMS) (renamed to SoaML)
  - Service vocabulary, Specification, Contract, correlation to business process…http://www.omg.org/cgi-bin/doc?ad/08-11-01
- Event Modeling and Profile (EMP) – RFP (Issued Sept. 2008, http://www.omg.org/cgi-bin/doc?ad/08-09-15)
  - Event vocabulary, relationship between EDA, BPM and SOA
    - Event Metamodel and UML Profile
    - Event metadata exchange using standard formats like XMI
    - Event Traceability and causality in relation to Services
    - Processing of complex events
- Agent Metamodel and Profile (AMP) RFP (issued Sept. 2008, http://www.omg.org/cgi-bin/doc?ad/08-09-05)
  - Metamodel and Profile for extending UML with capabilities applicable to agents and agent-based software
  - Clarify semantics concerned with modeling agents
  - Enable Agent model interchange between tools via XMI

OMG SoaML

Service Oriented Architecture Modeling Language UML Profile and Metamodel for Services

Summary Overview of Capability-base Service Modeling

Dr. Arne J. Berre, SINTEF,
Jim Amsden, IBM
Cory Casanave, Model Driven Solutions

The Submission Team

- Submitters
  - 88Solutions
  - Adaptive
  - EDS
  - Model Driven Solutions
  - Capgemini
  - Fujitsu
  - Fundacion European Software Institute
  - Hewlett-Packard
  - International Business Machines
  - MEGA International
  - MID GmbH
  - Rhysome
  - Softeam
  - Telelogic AB

- Supporters
  - Everware-CBDI
  - General Services Administration
  - VisumPoint
  - Mega
  - BAE Systems
  - DERI – University of Innsbruck
  - DFKI
  - France Telecom R&D
  - NKUA – University of Athens
  - Oslo Software
  - SINTEF
  - THALES Group
  - University of Augsburg
  - Wilton Consulting Group

Goals

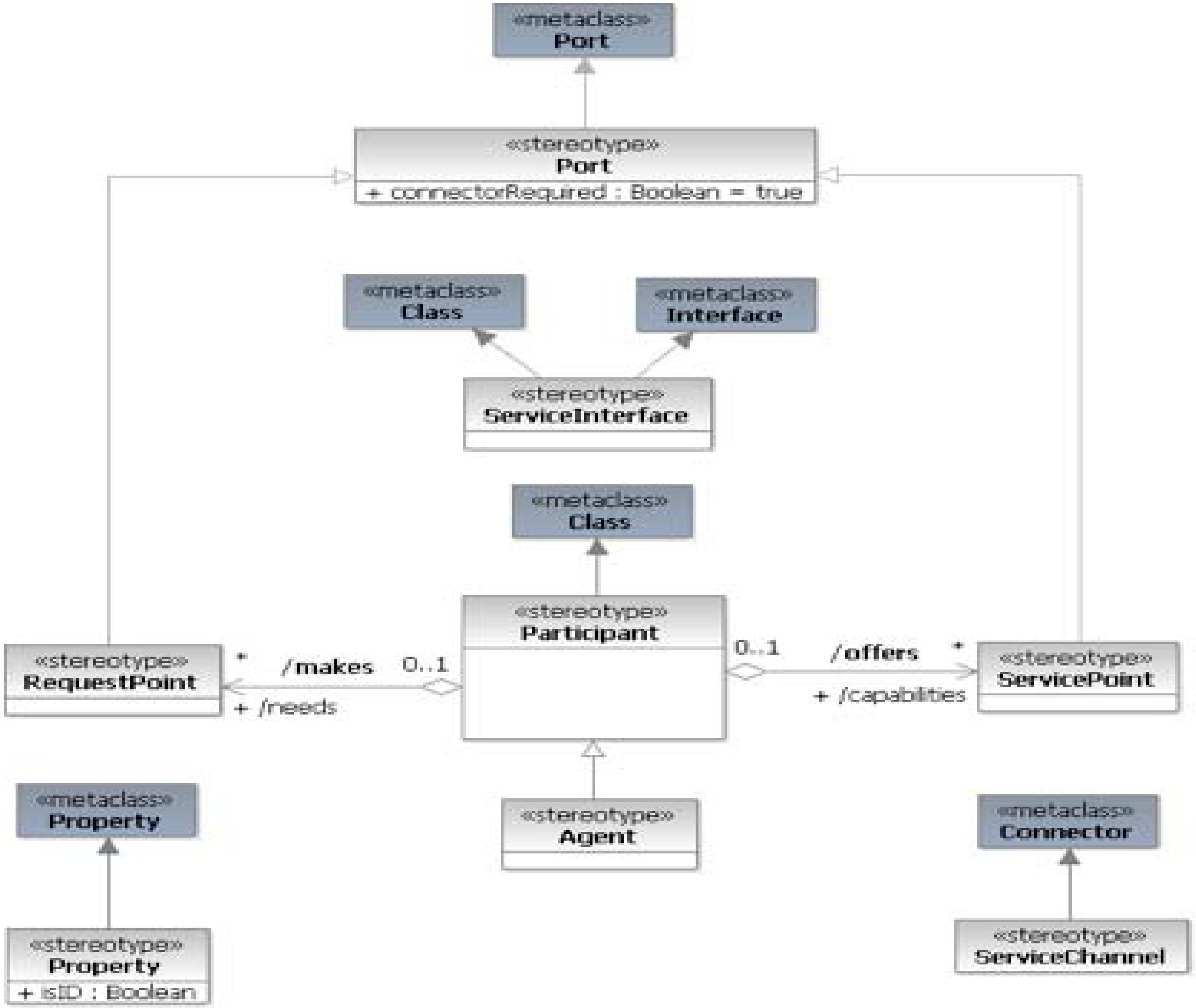
- **Intuitive and complete** support for modeling services in UML
- Support for **bi-directional asynchronous services** between multiple parties
- Support for **Services Architectures** where parties provide and use multiple services.
- Support for **services defined to contain other services**
- Easily 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 modeling**
- **Design by contract** or **dynamic adaptation** of services
- To specify and relate the **service capability and its contract**
- **No changes to UML**

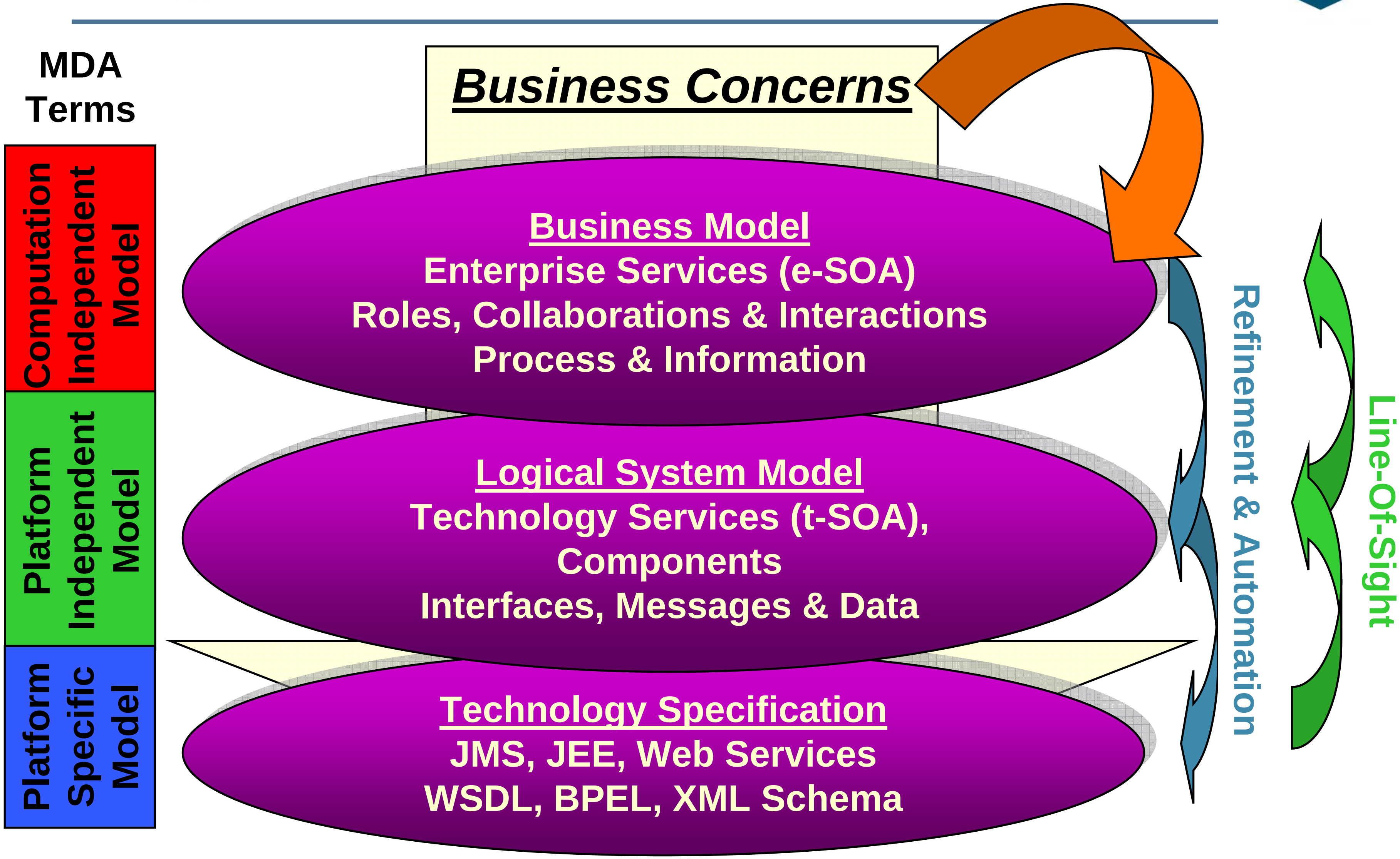
Service

- Service (noun) is the work or action performed by one for another, enabled by one or more capabilities.

- Here, the access to the service is provided using a prescribed interface and is exercised consistent with constraints and policies as specified by the service contract. A service is provided by a participant acting as the *provider of the service—for use by others. The eventual consumers of the service may not be known to the service provider and may demonstrate uses of the service beyond the scope originally conceived by the provider. [OASIS RM]*

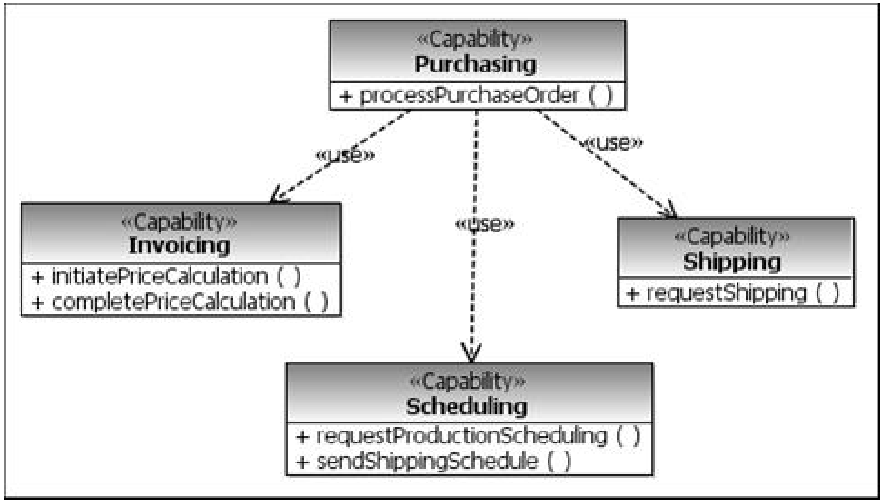
Metamodel


«metaclass»
Port
«stereotype»
Port
+ connectorRequired : Boolean = true
«metaclass»
Class
«metaclass»
Interface
«stereotype»
ServiceInterface
«metaclass»
Class
«stereotype»
Participant
«stereotype»
RequestPoint
*
/makes
0..1
+ /needs
0..1
/offers
*
+ /capabilities
«stereotype»
ServicePoint
«metaclass»
Property
«stereotype»
Agent
«metaclass»
Connector
«stereotype»
Property
+ isID : Boolean
«stereotype»
ServiceChannel

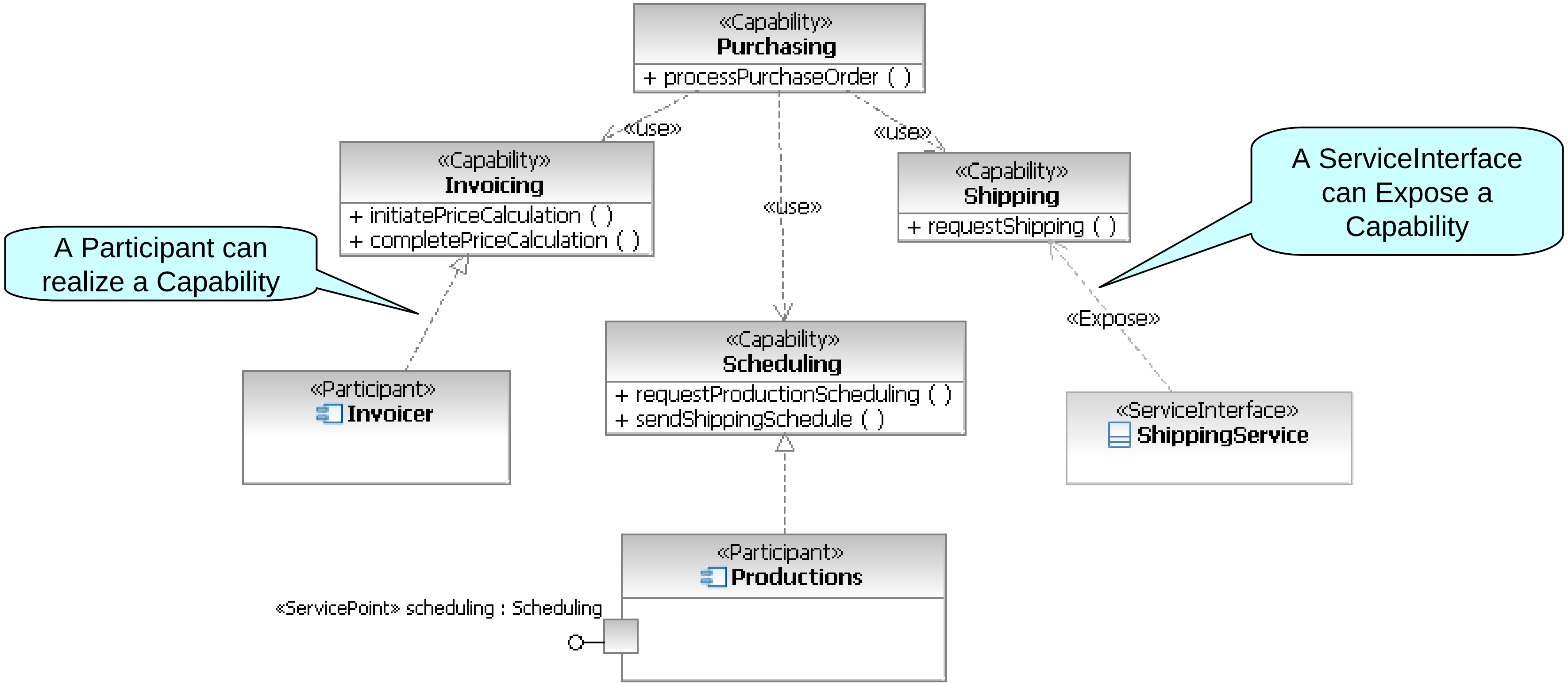

MDA
Terms
Computation Independent Model
Platform Independent Model
Platform Specific Model
Business Concerns
Business Model
Enterprise Services (e-SOA)
Roles, Collaborations & Interactions
Process & Information
Logical System Model
Technology Services (t-SOA),
Components
Interfaces, Messages & Data
Technology Specification
JMS, JEE, Web Services
WSDL, BPEL, XML Schema
Refinement & Automation
Line-Of-Sight

Modeling Capabilities

- Capabilities model the ability to accomplish some desired result or real-world effect
- A network of capabilities helps to identify and define services
- Capabilities can be identified using a variety of techniques:
  - Business goals and objectives
  - Business strategies and tactics
  - Business processes
  - Functional decomposition
  - Existing assets
- Capabilities can then be exposed by service interfaces and participants


«Capability»
Purchasing
+ processPurchaseOrder ( )
«use»
«use»
«use»
«Capability»
Invoicing
+ initiatePriceCalculation ( )
+ completePriceCalculation ( )
«Capability»
Shipping
+ requestShipping ( )
«Capability»
Scheduling
+ requestProductionScheduling ( )
+ sendShippingSchedule ( )

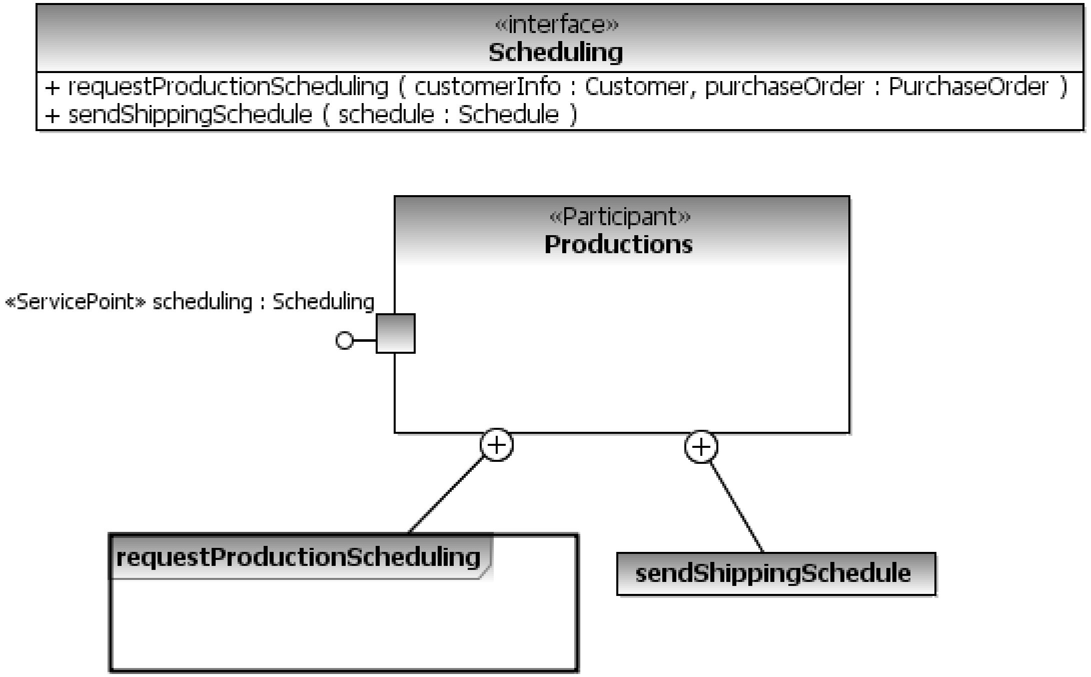
Capabilities Realized by Participants can be Exposed through Service Interfaces



A Participant can realize a Capability

A ServiceInterface can Expose a Capability

Participants Provide Services through ServicePoints

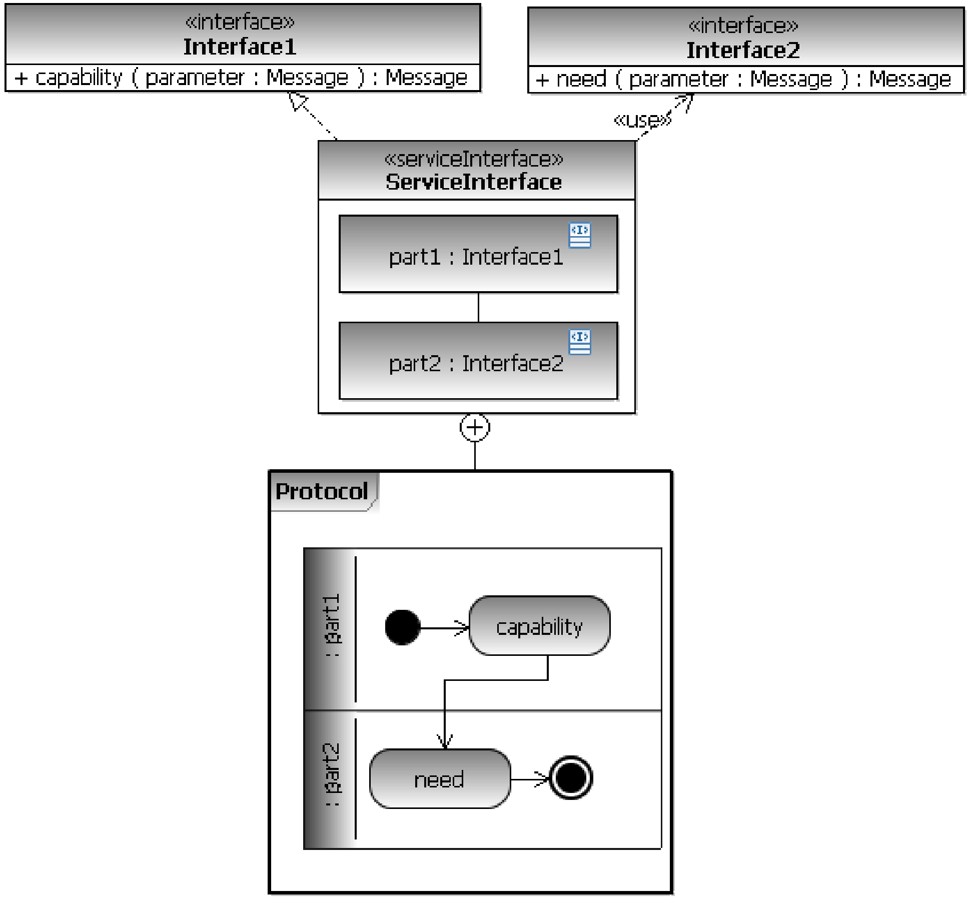

«interface»
Scheduling
+ requestProductionScheduling (customerInfo : Customer, purchaseOrder : PurchaseOrder)
+ sendShippingSchedule (schedule : Schedule)
«Participant»
Productions
«ServicePoint» scheduling : Scheduling
requestProductionScheduling
sendShippingSchedule

A **ServicePoint** is the offer of a service by one participant to others using well defined terms, conditions and interfaces. A ServicePoint defines the connection point through which a Participant offers its capabilities and provides a service to clients.

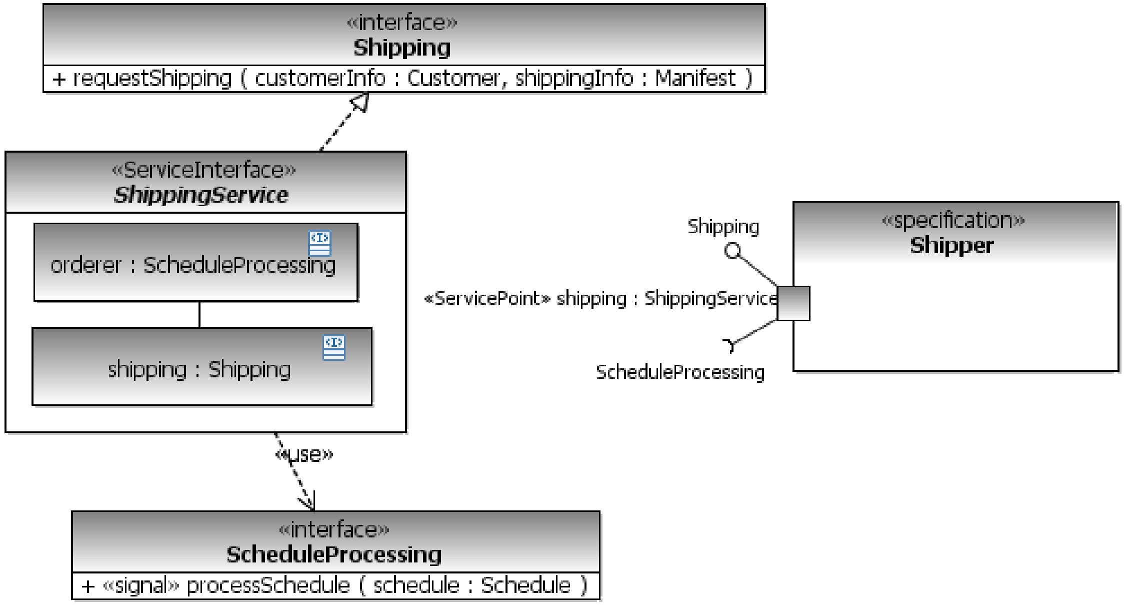
A ServicePoint is a mechanism by which a provider Participant makes available services that meet the needs of consumer requests as defined by ServiceInterfaces, Interfaces and ServiceContracts. A ServicePoint is represented by a UML Port on a Participant stereotyped as a «ServicePoint», .

A ServiceInterface is used to Define a Service

a **ServiceInterface** can be the type of a service or request point. The service interface has the additional feature that it can specify a bi-directional service – where both the provider and consumer have responsibilities to send and receive messages and events. The service interface can be defined from the perspective of either the service consumer or provider using three primary sections: the provided and required Interfaces, the ServiceInterface class and the protocol Behavior.

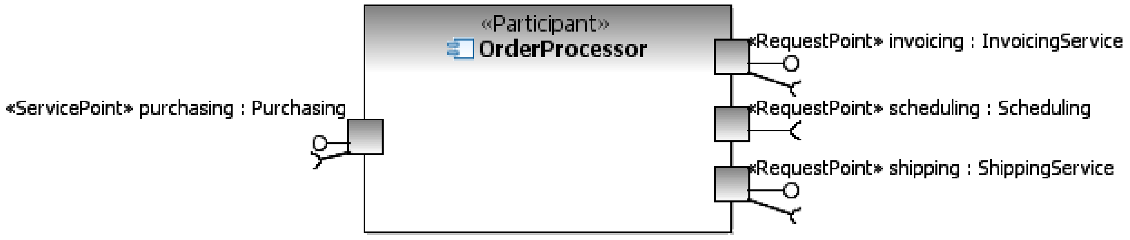


Participant with ServicePoint


«interface»
Shipping
+ requestShipping (customerInfo : Customer, shippingInfo : Manifest)
«ServiceInterface»
ShippingService
orderer : ScheduleProcessing
shipping : Shipping
«use»
«interface»
ScheduleProcessing
+ «signal» processSchedule (schedule : Schedule)
Shipping
«ServicePoint» shipping : ShippingService
ScheduleProcessing
«specification»
Shipper

A ServicePoint offered by a participant can be typed by a ServiceInterface which describes the interaction the participant expects with other participants through that interaction point.

Participants Consume Services through RequestPoints


«Participant»
OrderProcessor
«ServicePoint» purchasing : Purchasing
«RequestPoint» invoicing : InvoicingService
«RequestPoint» scheduling : Scheduling
«RequestPoint» shipping : ShippingService

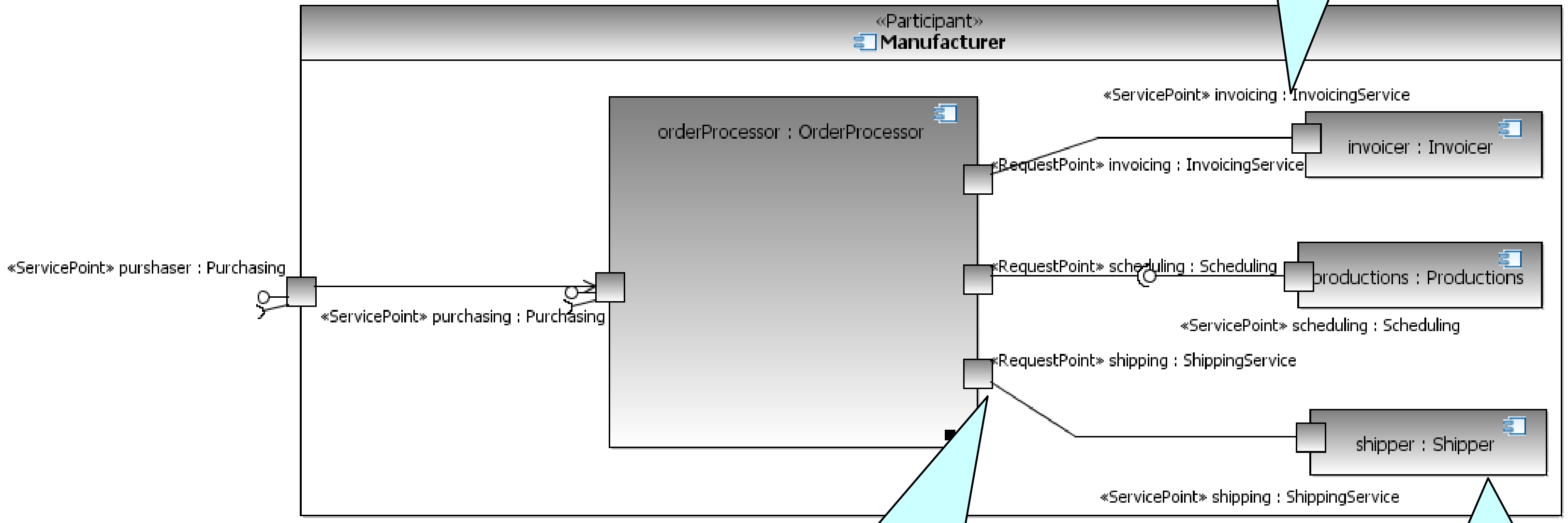
The type of a **RequestPoint** is also a ServiceInterface, or UML Interface, as it is with a Service point. The RequestPoint is the conjugate of a ServicePoint in that it defines the use of a service rather than its provision. This will allow us to connect service providers and consumers in a Participant.

Participants May be Assemblies of Other Participants

Participants can be assembled into a **service value chain** by connecting participants with **goals, needs and expectations** with other participants with compatible **value propositions, capabilities and commitments**

Participant

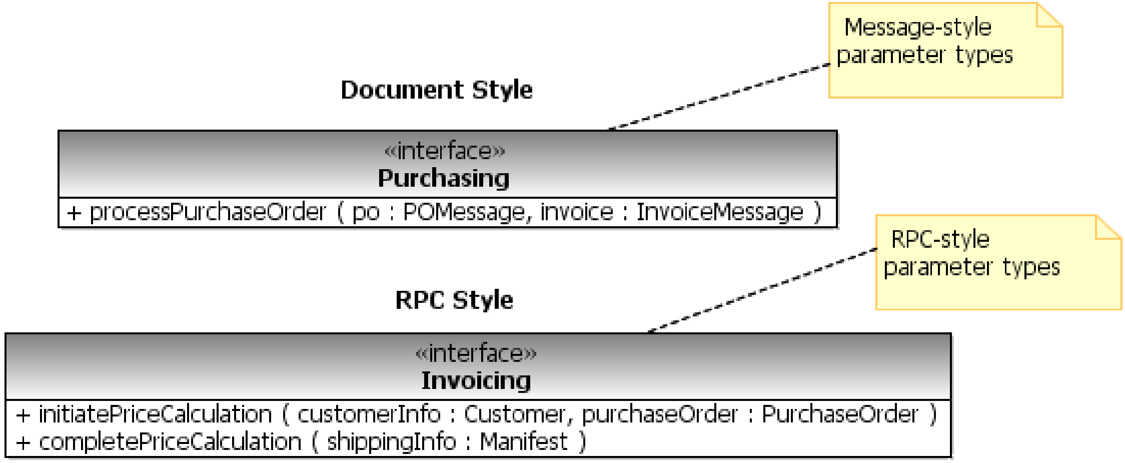
ServicePoint – capabilities typed by ServiceInterface


«Participant»
Manufacturer
orderProcessor : OrderProcessor
«ServicePoint» invoicing : InvoicingService
invoicer : Invoicer
«RequestPoint» invoicing : InvoicingService
«ServicePoint» purshaser : Purchasing
«ServicePoint» purchasing : Purchasing
«RequestPoint» scheduling : Scheduling
productions : Productions
«ServicePoint» scheduling : Scheduling
«RequestPoint» shipping : ShippingService
shipper : Shipper
«ServicePoint» shipping : ShippingService

RequestPoint – needs typed by ServiceInterface

Participant part

Document and RPC Style Service Operation Parameters



Document Style

«interface»
Purchasing

+ processPurchaseOrder (po : POMessage, invoice : InvoiceMessage)

Message-style parameter types

RPC Style

«interface»
Invoicing

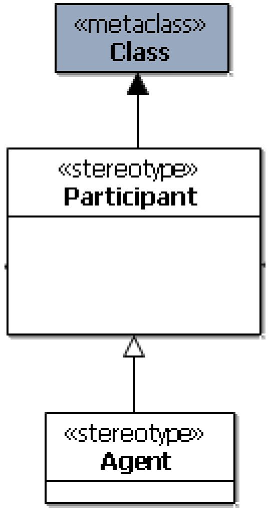
+ initiatePriceCalculation (customerInfo : Customer, purchaseOrder : PurchaseOrder)
+ completePriceCalculation (shippingInfo : Manifest)

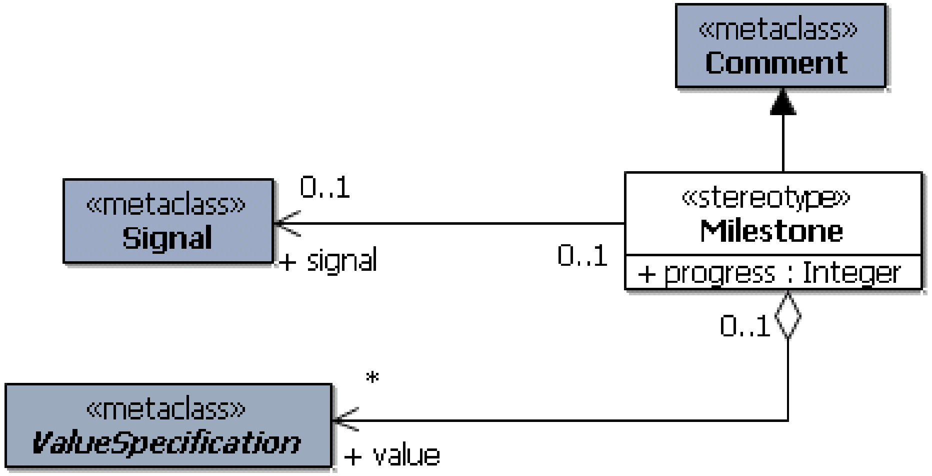
RPC-style parameter types

Agent and Milestones

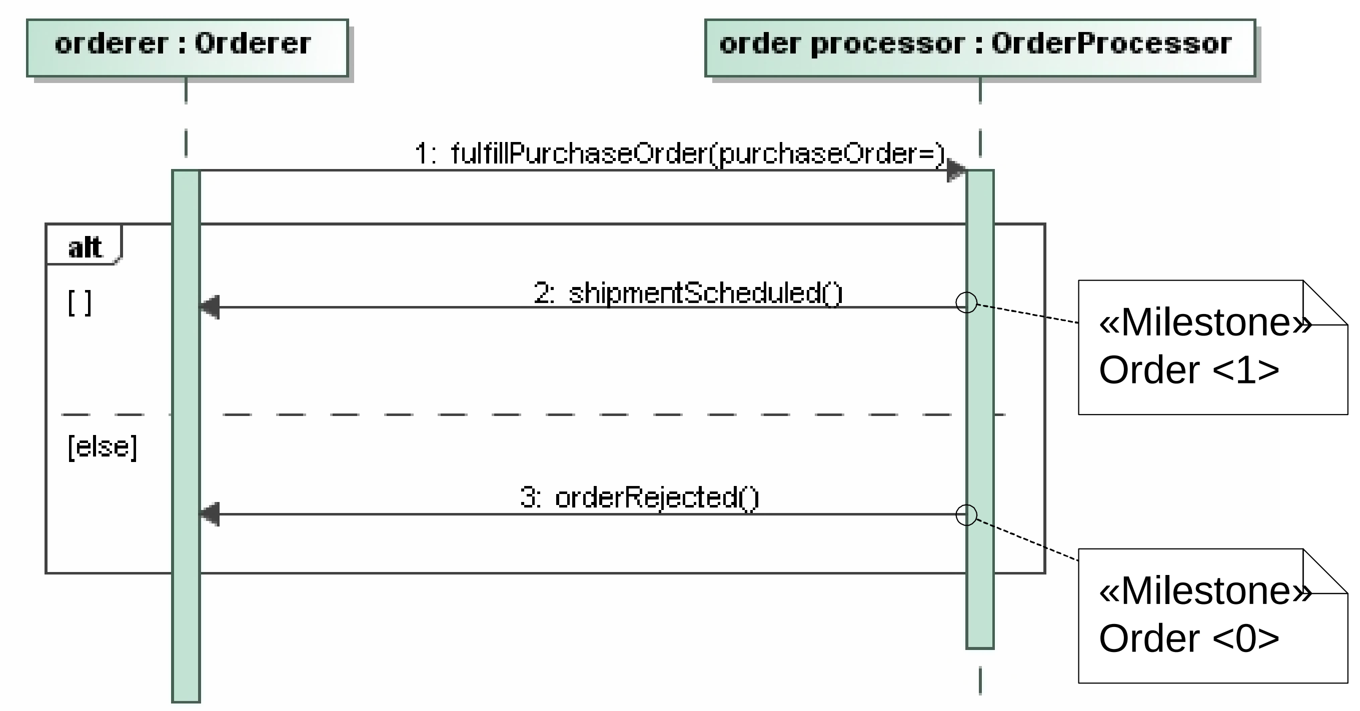
- ***Agent***
  - autonomous entity
  - has its own lifecycle behavior
  - can adapt to the environment

- ***Milestone***
  - defines a value of progress
  - attached to behavioral elements
  - is used especially for dynamic analysis of behavior that does not necessarily end





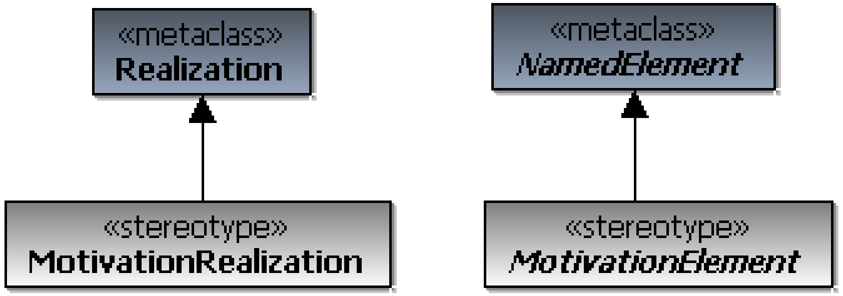
Milestone



A Milestone is a means for depicting progress in behaviors in order to analyze liveness. Milestones are particularly useful for behaviors that are long lasting or even infinite.

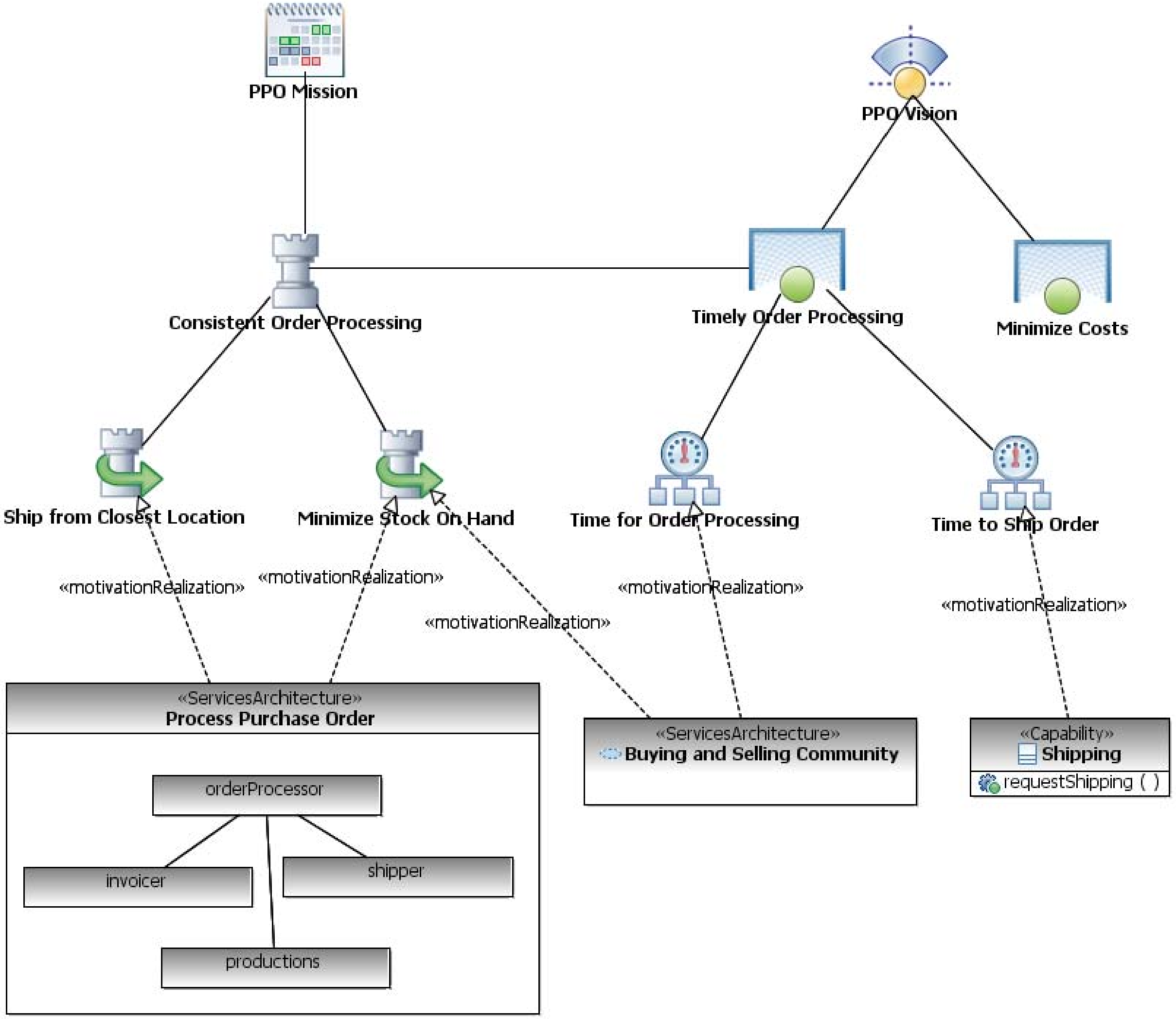
A Milestone can be understood as a “mythical” Signal. A mythical Signal is a conceptual signal that is sent from the behavior every time a point connected to the Milestone is passed during execution. The signal is sent to a conceptual observer outside the system that is able to record the origin of the signal, the signal itself and its progress value.

Integration with Business Motivation and Strategy



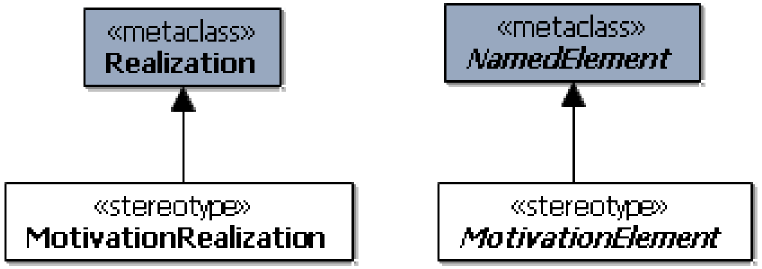
- Business requirements can be captured using the OMG Business Motivation Model (BMM).
- Any UML BehavioredClassifier including (for example a ServicesContract) may realize the BMM Motivation concept of *motivation realization*. This allows services models to be connected to the business motivation and strategy linking the services to the things that make them business relevant.

BMM with MeansRealizations



Relating to Other Standards

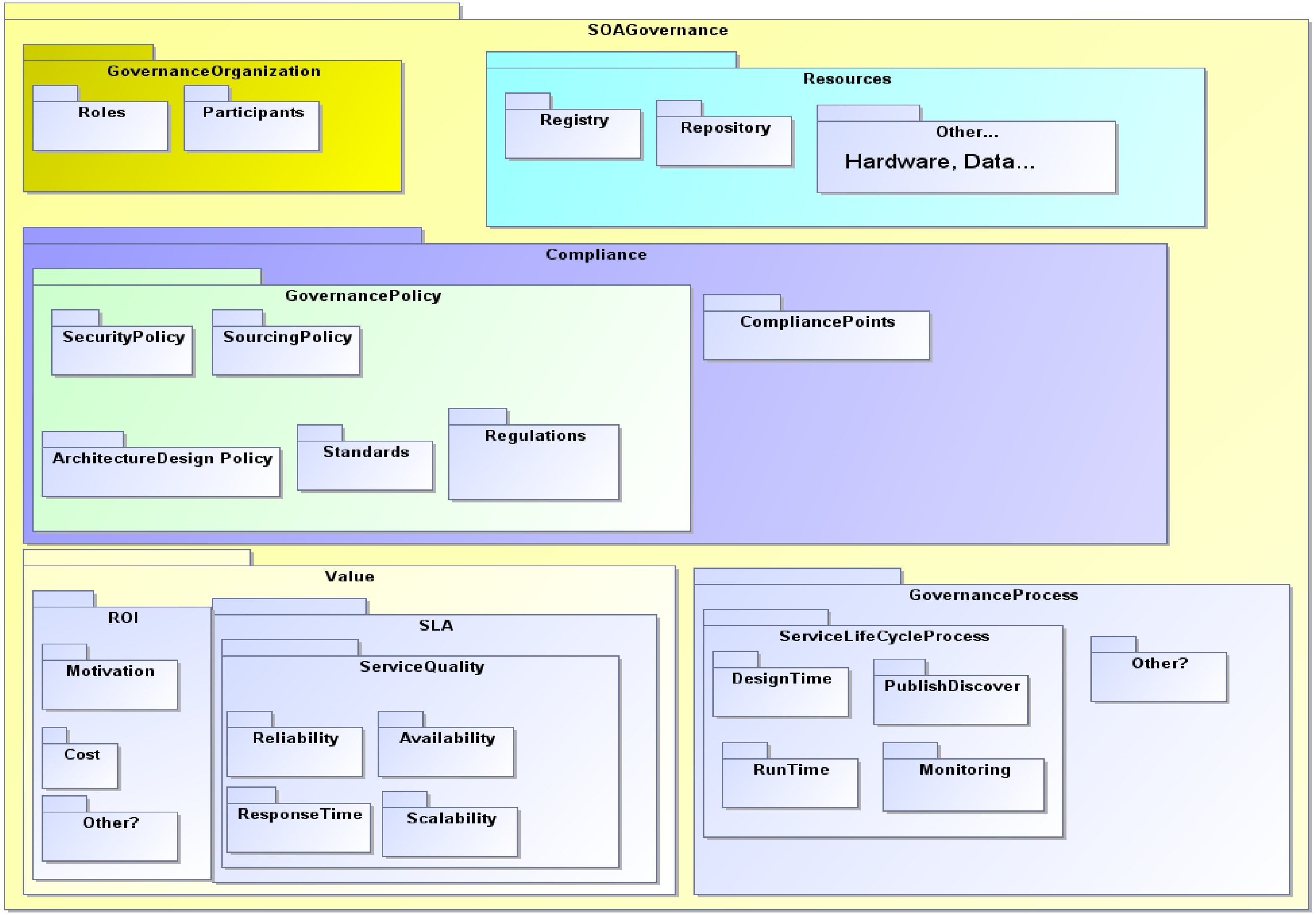
- SoaML integration with BPMN 2.0 and BPDM will be related to the ongoing BPMN 2.0 standardization
- Extensions for Agents and semantic services will also relate to semantics, ontologies and other OMG metamodels like ODM and SBVR
- Limited BMM integration is included to tie services to the business



SIG next steps/roadmap

- SoaML: voting and finalization in progress
- EMP RFP: Letter of Intent (LOI) due Feb. 09, initial submission due May 209
- AMP RFP: LOI Feb. 09, initial submission work started
- Service Categorization RFP in the works
  - Scope: Service categorization and description to facilitate multiple classifications, automated discovery etc.
- SOA Governance RFP: scope, requirements in progress, plan to issue in June 2009
- Collaborative projects:
  - HSSP (Healthcare Service Specification Project): defining Services such as: Retrieve, Locate and Update Service (RLUS), Entity Identification Service http://hssp.wikispaces.com/

SOA Governance RFP: Potential scope



FYI

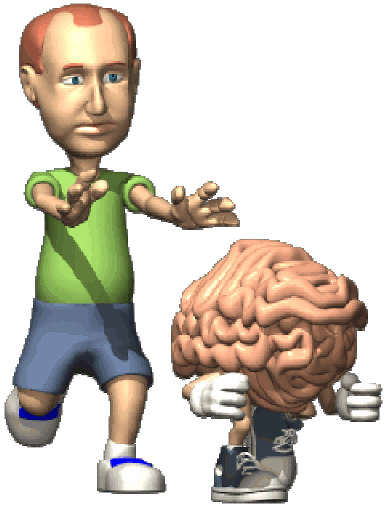
- OMG SOA SIG (http://soa.omg.org) meets every Tuesday at the quarterly OMG technical meetings (for 2009 calendar: http://www.omg.org/news/schedule/upcoming.htm)
- Mailing lists:
  - SOA SIG: soa@omg.org (to join, please send email to: request@omg.org)
  - Agent modeling work: amp-team@omg.org, http://www.omgwiki.org/AMP-team
  - SOA Governance RFP: soagov@omg.org http://www.omgwiki.org/soagov/doku.php

Contact Info:

Harsh Sharma, Co-Chair, OMG SOA SIG, hsharma@meta-guru.com
James Odell, Co-Char, OMG SOA SIG, CSC, email@jamesodell.com

Ed Harrington, Model Driven Solutions: ed-h@modeldriven.com

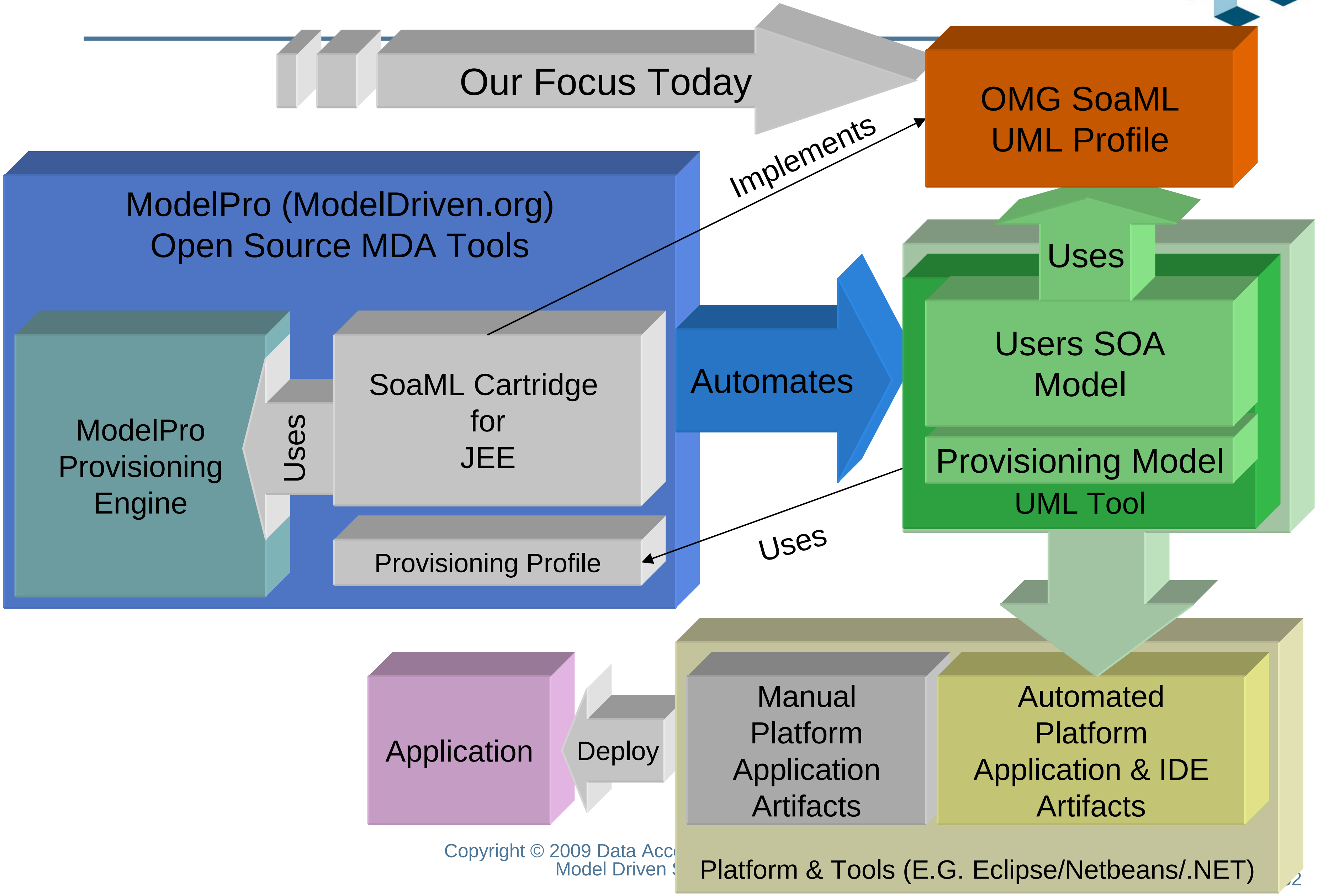
Thank You



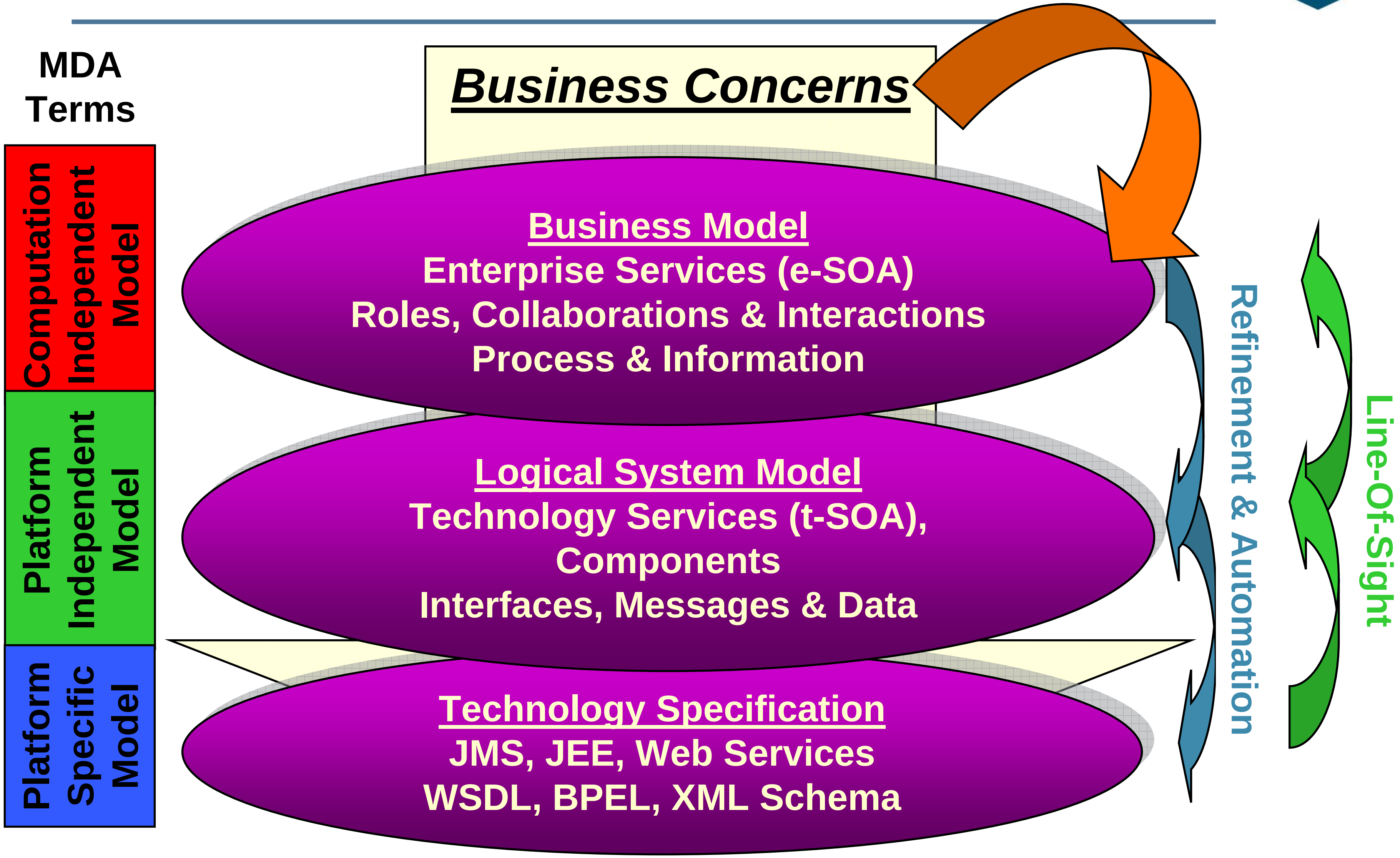
Supplemental Materials

- Model Driven SOA
- Social Security Administration / ORSIS SOA Modeling Example

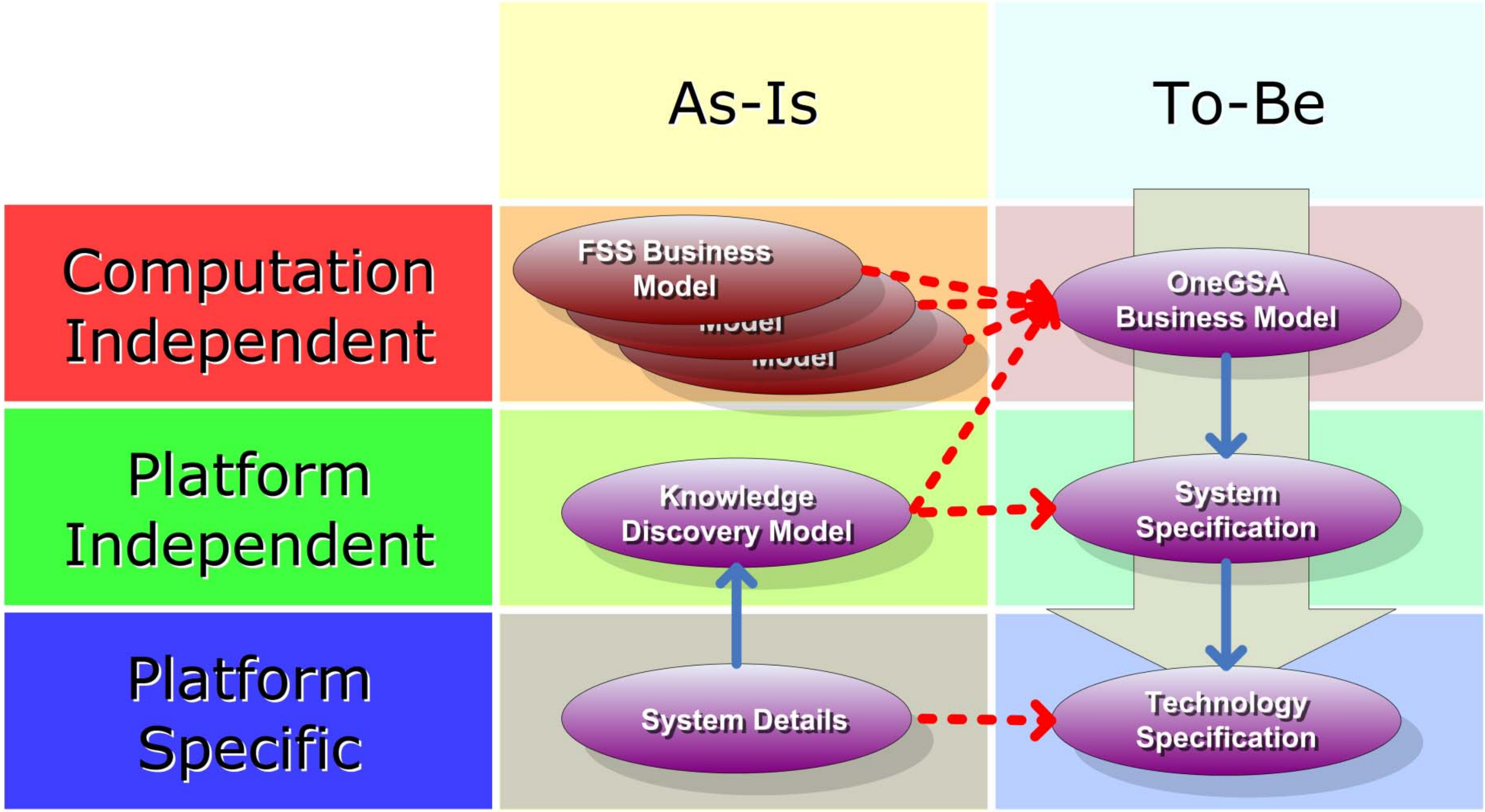
Relating the Parts for Model Driven SOA


Our Focus Today
OMG SoaML UML Profile
Implements
ModelPro (ModelDriven.org) Open Source MDA Tools
ModelPro Provisioning Engine
Uses
SoaML Cartridge for JEE
Provisioning Profile
Automates
Uses
Users SOA Model
Provisioning Model
UML Tool
Uses
Application
Deploy
Manual Platform Application Artifacts
Automated Platform Application & IDE Artifacts
Platform & Tools (E.G. Eclipse/Netbeans/.NET)

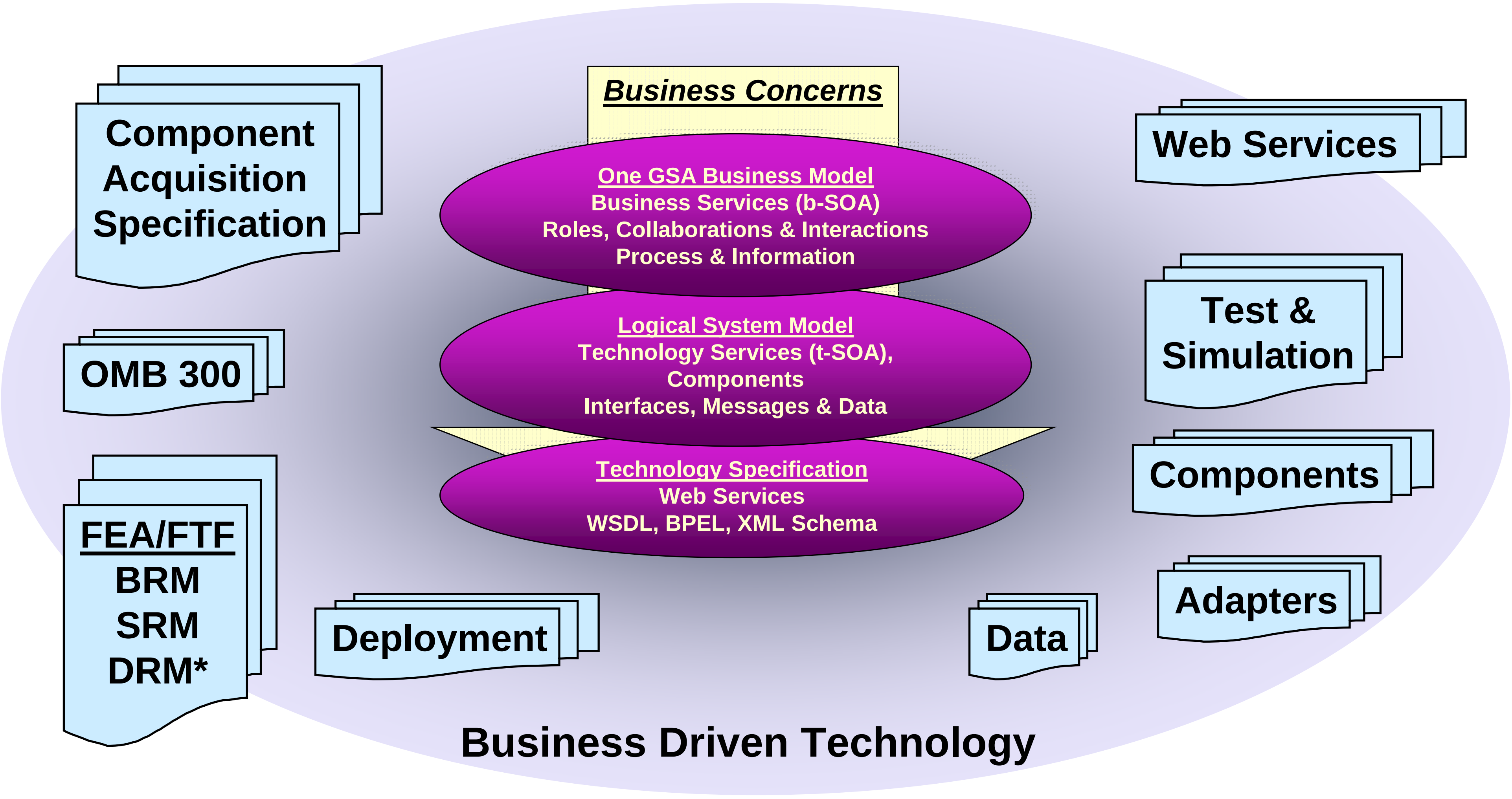
Business Focused SOA Using Model Driven Architecture



Incorporating Legacy Analysis



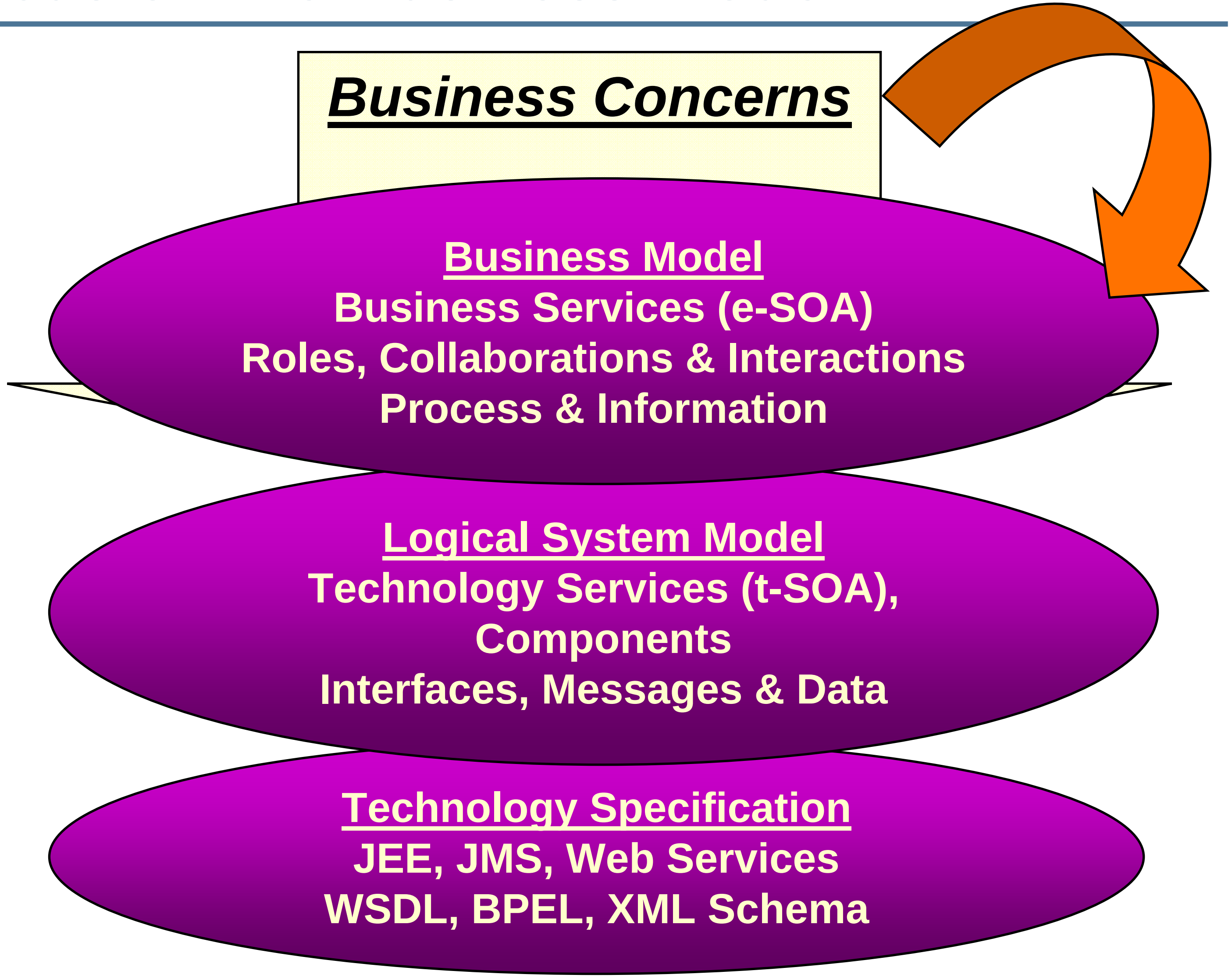
Value derived from the architecture



Business Driven Technology

Facilitating Business Processes

Focus on the Business Model



Business Concerns

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

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

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

Model Driven Solutions
Where Business Meets Technology
A Division of Data Access Technologies, Inc.

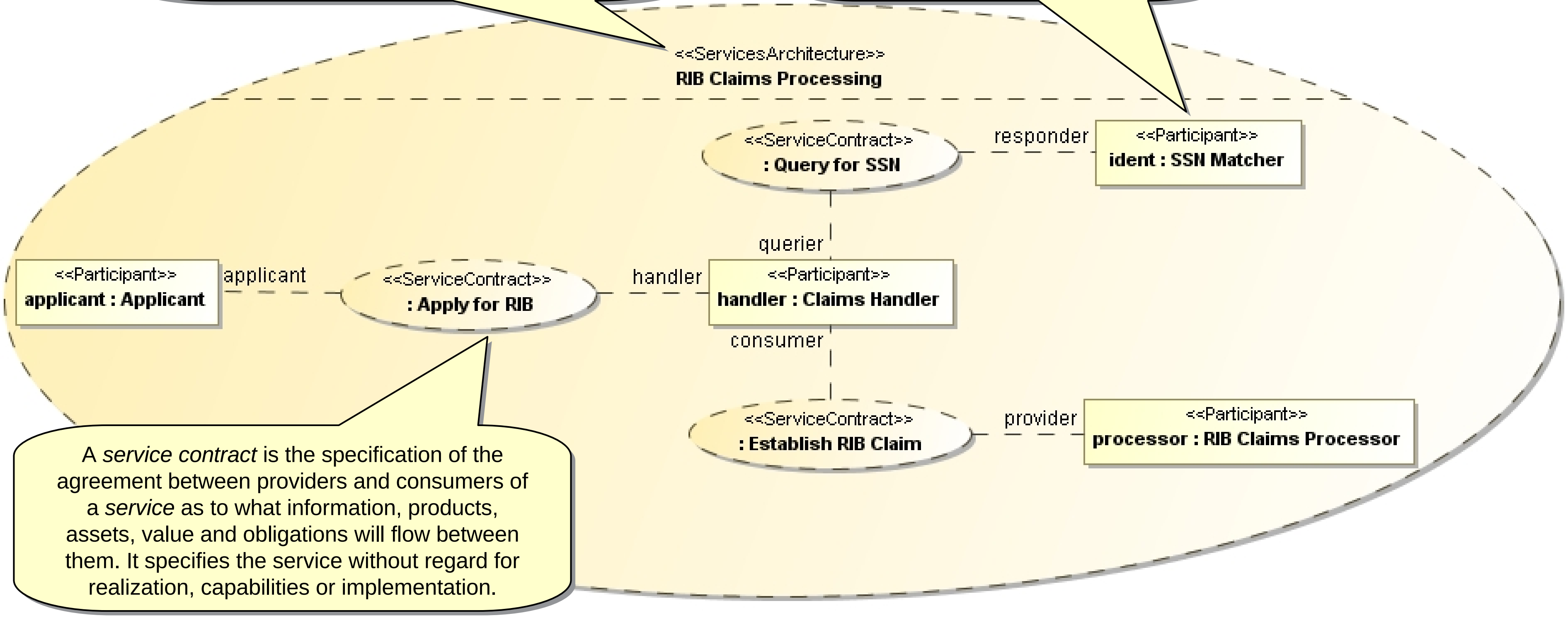
Social Security Administration / ORSIS Service Oriented Architecture (SOA) Modeling Example

Ed Seidewitz

Computation Independent Model (CIM)

- RIB Claims Processing Services Architecture
  - RIB Claims Processing Business Process
- Apply for RIB Service Contract
  - RIB Application Service Interface
- Query for SSN Service Contract
  - SSN Query Service Interface
- Establish RIB Claim Service Contract
  - RIB Establishment Service Interface
- RIB Claims Processing Participants

RIB Claims Processing Services Architecture

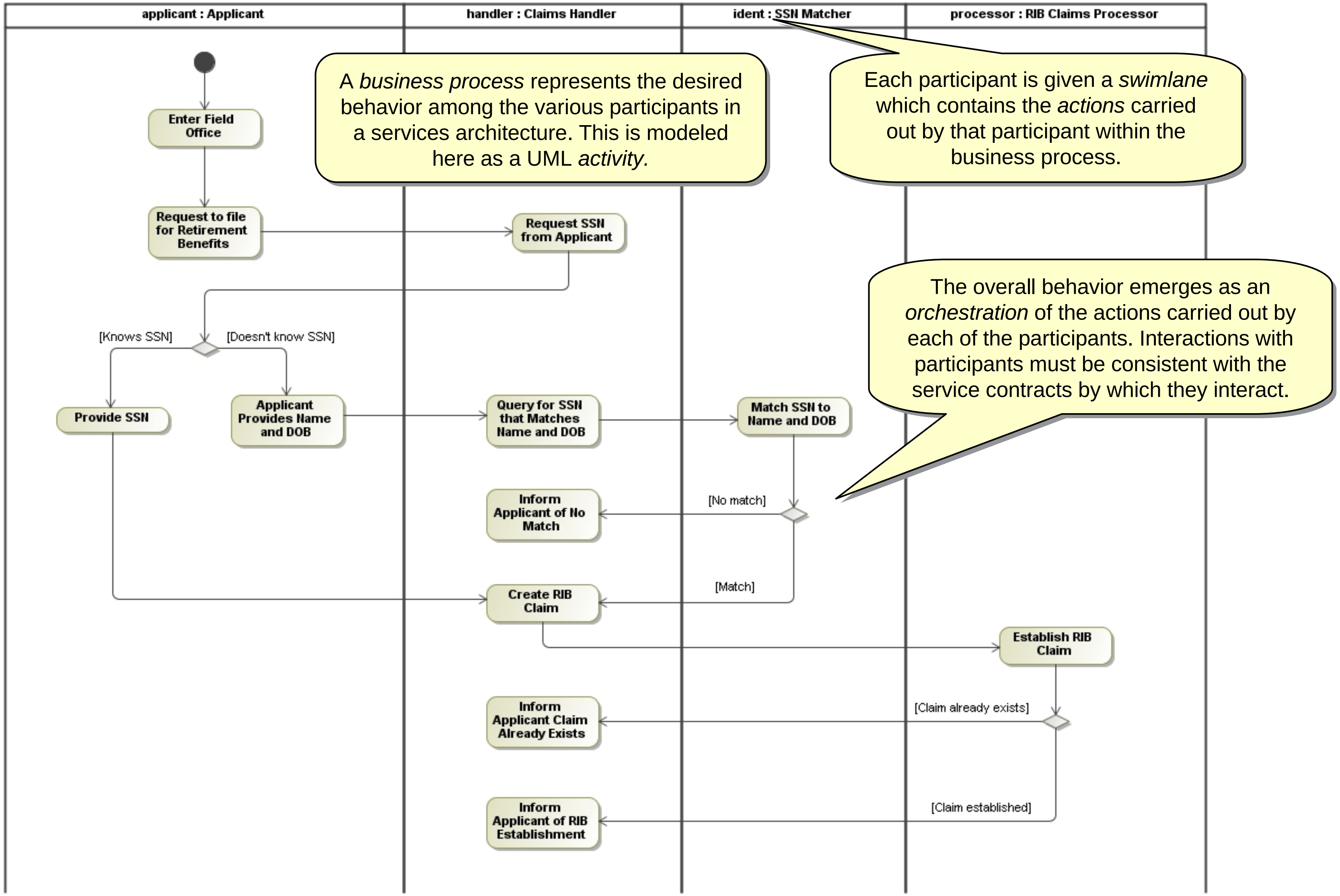


A *services architecture* describes how *participants* work together for a purpose by providing and using services expressed as *service contracts*. It is modeled as a UML *collaboration.*

A *participant* represents some party that provides and/or consumes services. Participants may represent people, organizations or systems.

A *service contract* is the specification of the agreement between providers and consumers of a *service* as to what information, products, assets, value and obligations will flow between them. It specifies the service without regard for realization, capabilities or implementation.

RIB Claims Processing Business Process


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

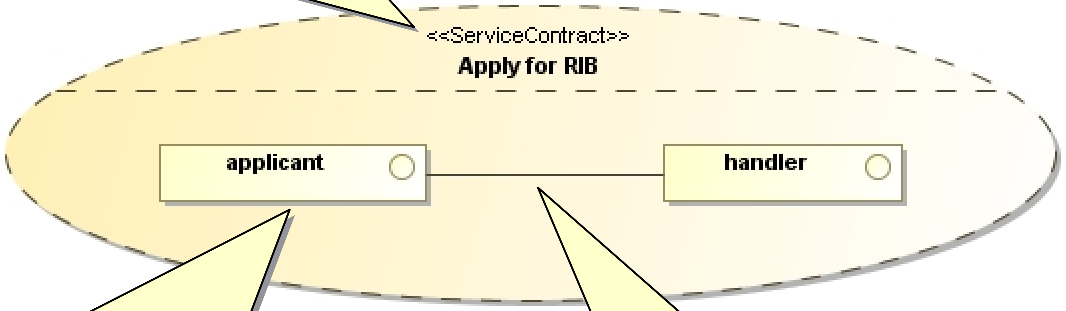
A *business process* represents the desired behavior among the various participants in a services architecture. This is modeled here as a UML *activity.*

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

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

Apply for RIB Service Contract

A *service contract* is the specification of the agreement between providers and consumers of a *service* as to what information, products, assets, value and obligations will flow between them. It specifies the service without regard for realization, capabilities or implementation. It is modeled as a UML *collaboration.*

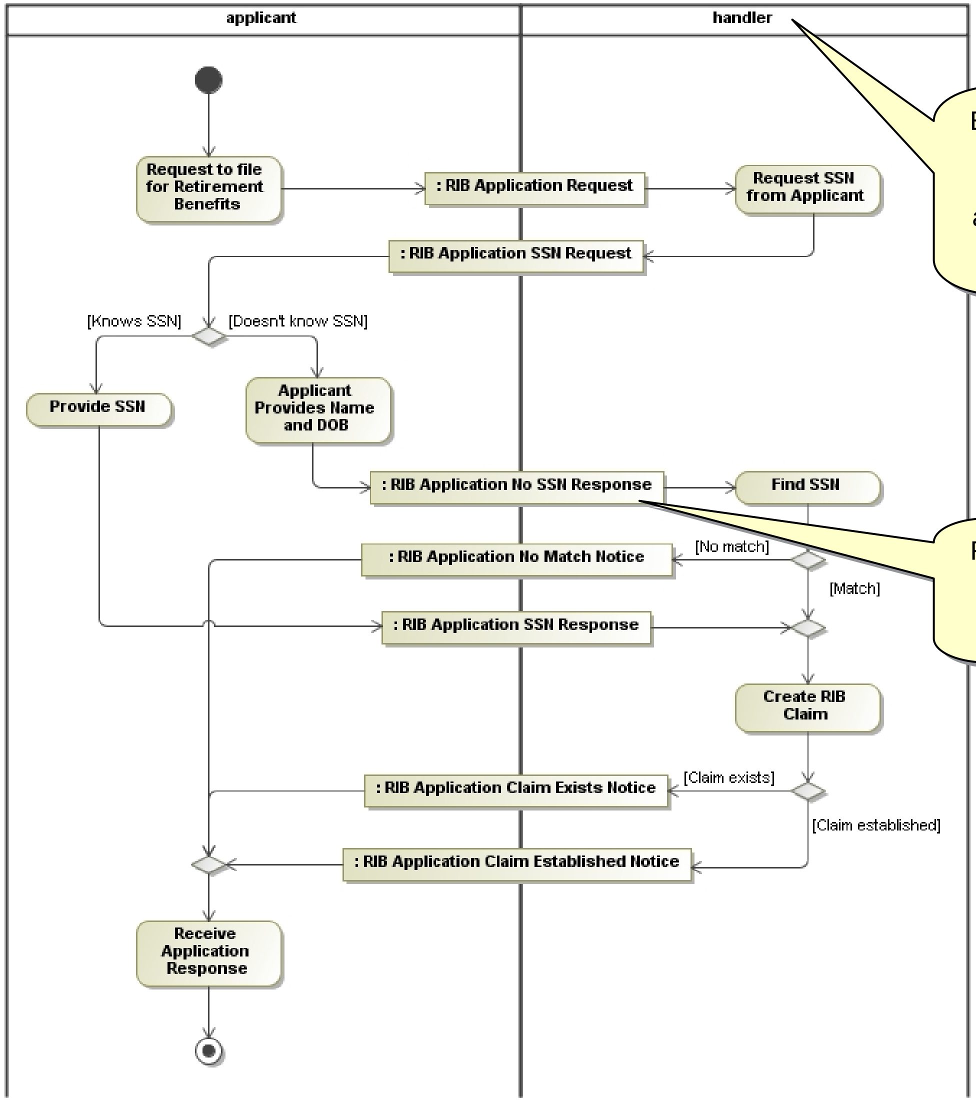


The service contract defines the *roles* to be played by consumers and providers of the service. Many service contracts have only two roles, one a consumer and one a provider. But any number are allowed.

The service contract also defines the *connections* across which roles may interact.

Apply for RIB Behavior

A service contract may have a ***behavior*** **that *choreographs* the allowed interactions between parties in the contract. This is modeled here as a UML *activity.***

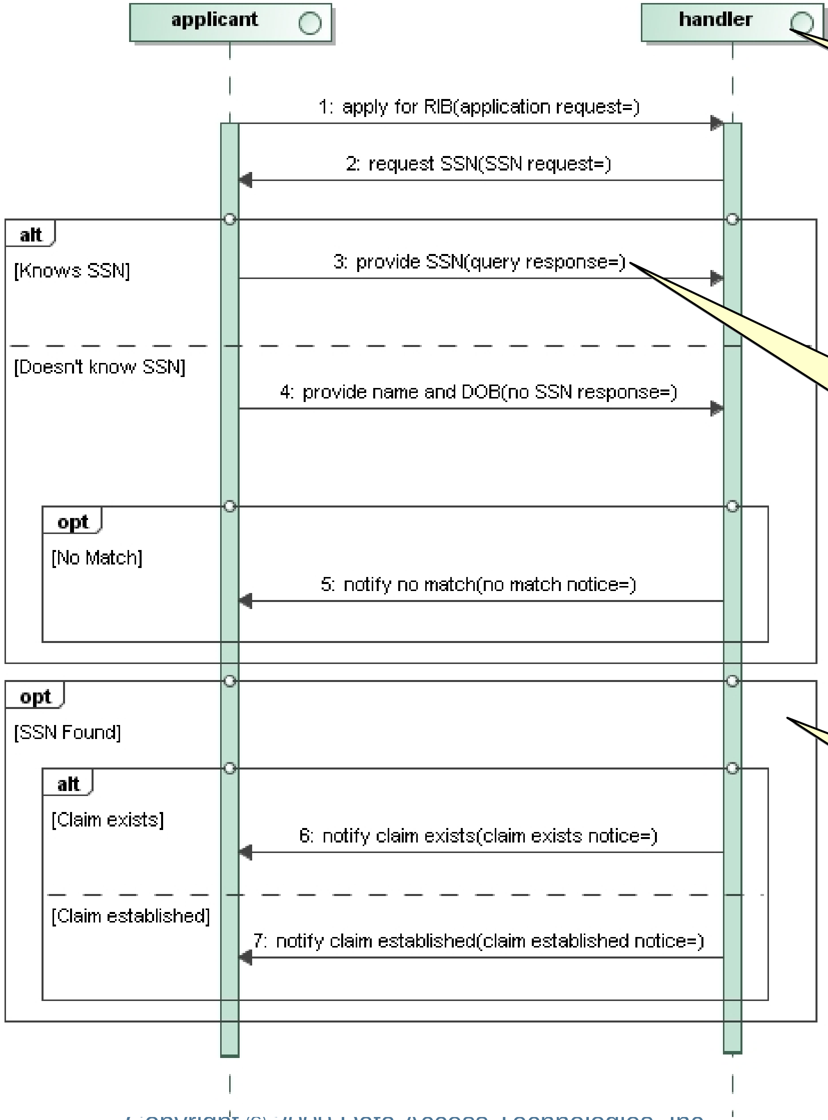

applicant
handler
Request to file for Retirement Benefits
: RIB Application Request
Request SSN from Applicant
: RIB Application SSN Request
[Knows SSN]
[Doesn't know SSN]
Provide SSN
Applicant Provides Name and DOB
: RIB Application No SSN Response
Find SSN
: RIB Application No Match Notice
[No match]
[Match]
: RIB Application SSN Response
Create RIB Claim
: RIB Application Claim Exists Notice
[Claim exists]
[Claim established]
: RIB Application Claim Established Notice
Receive Application Response

Each role in the contract is given a *swimlane* which contains the *actions* that are expected to be carried out by that role.

Roles may exchange information via *message* objects.

Apply for RIB Interaction

The behavior of a service contract may also be modeled using other kinds of UML interaction models. It is modeled here as an *interaction* using a *sequence diagram.*



applicant
handler
1: apply for RIB(application request=)
2: request SSN(SSN request=)
alt
[Knows SSN]
3: provide SSN(query response=)
[Doesn't know SSN]
4: provide name and DOB(no SSN response=)
opt
[No Match]
5: notify no match(no match notice=)
opt
[SSN Found]
alt
[Claim exists]
6: notify claim exists(claim exists notice=)
[Claim established]
7: notify claim established(claim established notice=)

Each role in the contract is given a *lifeline* which acts as the source and target for the sending of *messages*.

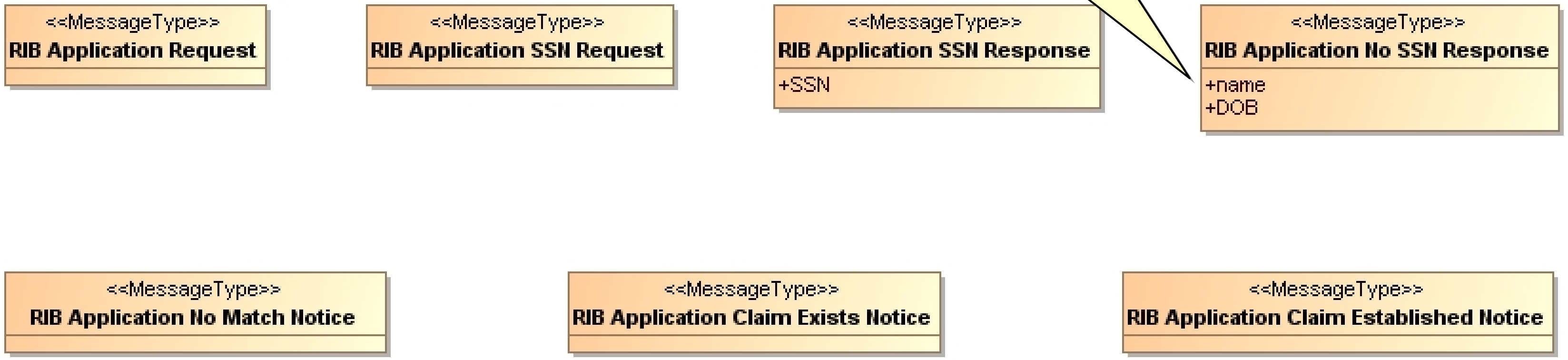
Messages are modeled as being passed via calls to *operations* on the *interfaces* to the roles.

Condition flows can be modeled using *interaction fragment* constructs within the sequence diagram.

RIB Application Messages

The messages passed between roles in a service contract are specified using *message types.* Message types are modeled as UML *classes*.

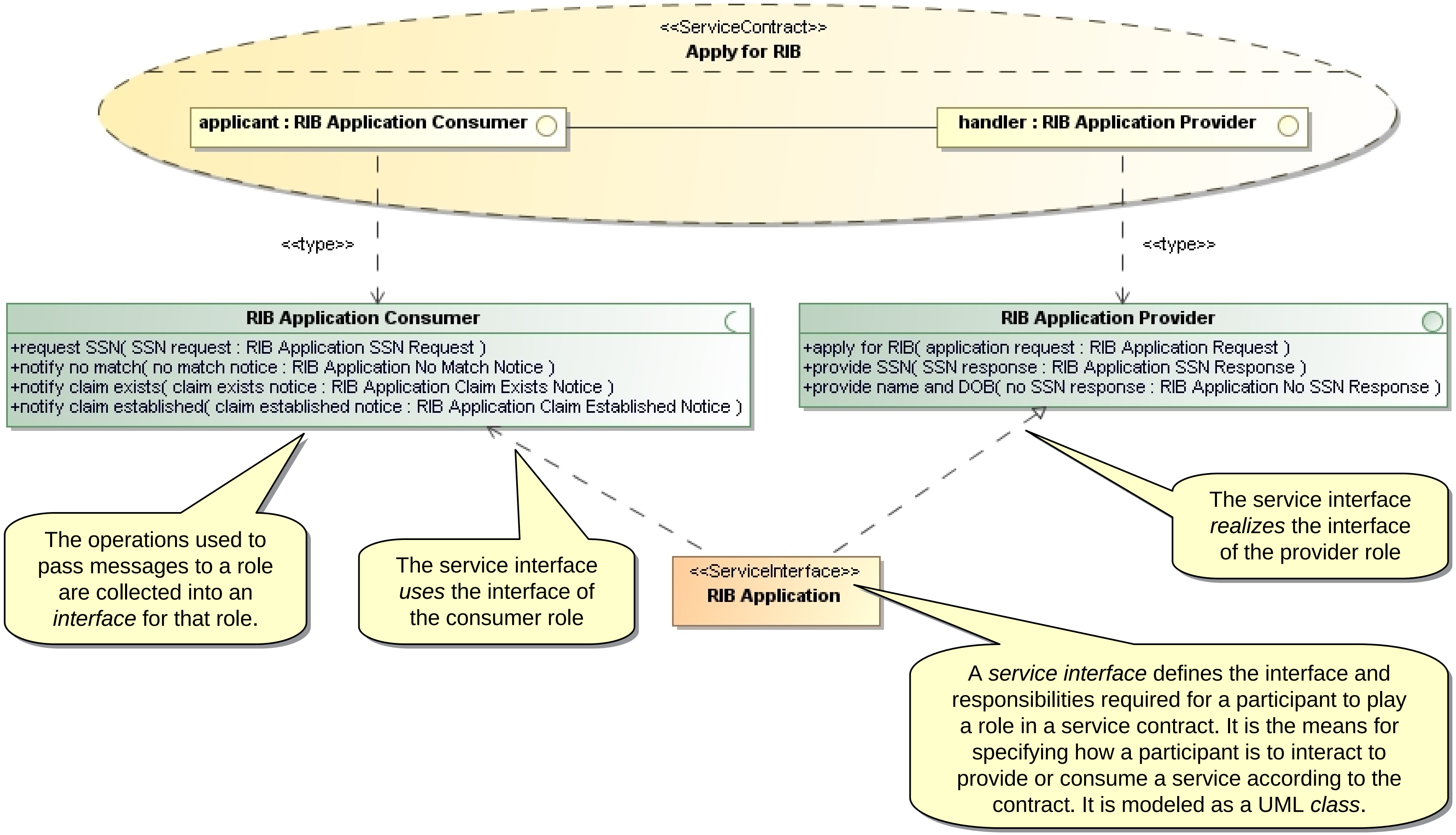
A message type may have data *attributes* but no operations or other behavior.



<<MessageType>>
RIB Application Request
<<MessageType>>
RIB Application SSN Request
<<MessageType>>
RIB Application SSN Response
+SSN
<<MessageType>>
RIB Application No SSN Response
+name
+DOB
<<MessageType>>
RIB Application No Match Notice
<<MessageType>>
RIB Application Claim Exists Notice
<<MessageType>>
RIB Application Claim Established Notice

Note: Message information model has not been fully elaborated yet

RIB Application Service Interface



<<ServiceContract>>
Apply for RIB

applicant : RIB Application Consumer

handler : RIB Application Provider

<<type>>

<<type>>

RIB Application Consumer

+request SSN(SSN request : RIB Application SSN Request)
+notify no match(no match notice : RIB Application No Match Notice)
+notify claim exists(claim exists notice : RIB Application Claim Exists Notice)
+notify claim established(claim established notice : RIB Application Claim Established Notice)

RIB Application Provider

+apply for RIB(application request : RIB Application Request)
+provide SSN(SSN response : RIB Application SSN Response)
+provide name and DOB(no SSN response : RIB Application No SSN Response)

<<ServiceInterface>>
RIB Application

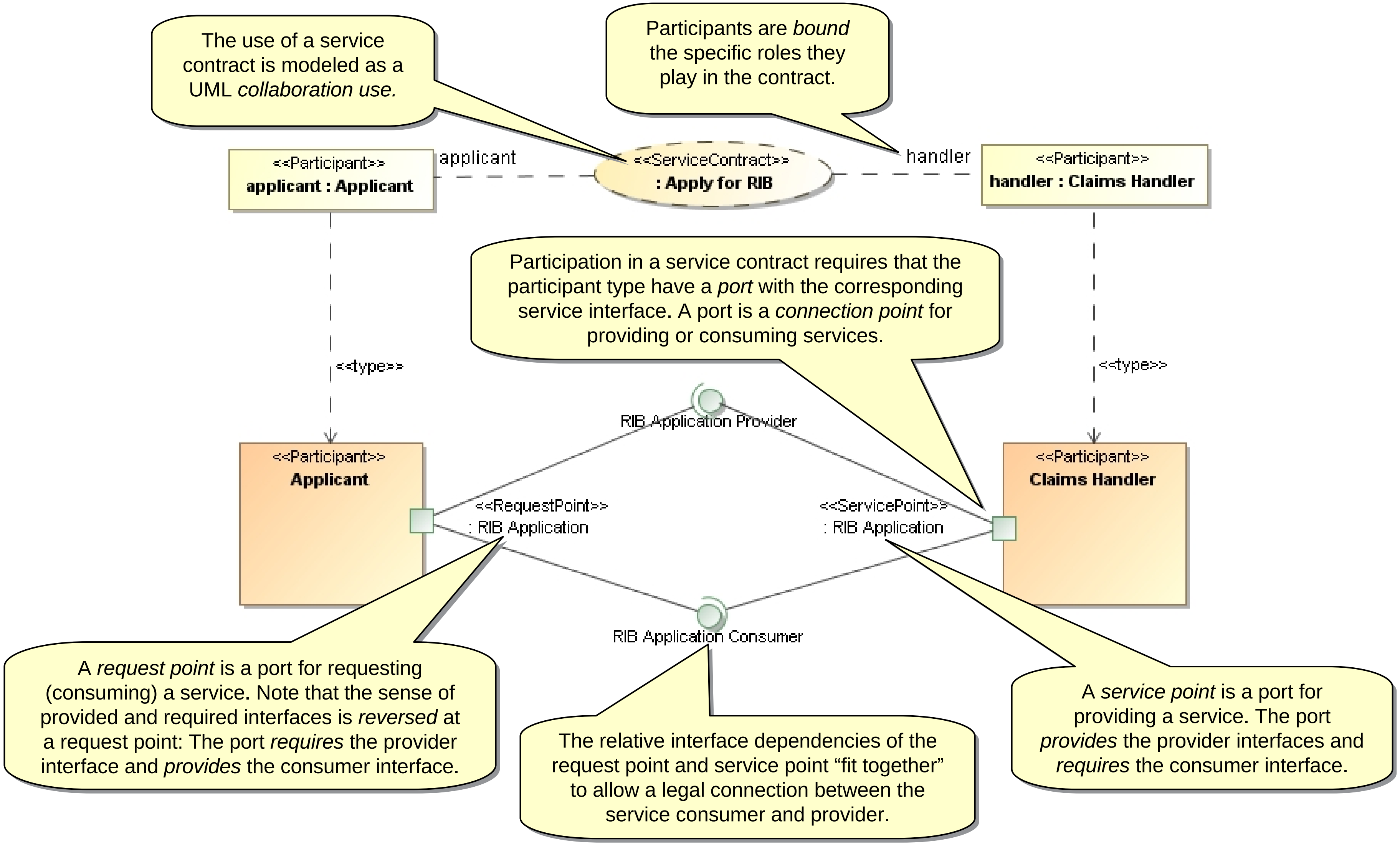
The operations used to pass messages to a role are collected into an *interface* for that role.

The service interface *uses* the interface of the consumer role

The service interface *realizes* the interface of the provider role

A *service interface* defines the interface and responsibilities required for a participant to play a role in a service contract. It is the means for specifying how a participant is to interact to provide or consume a service according to the contract. It is modeled as a UML *class*.

RIB Application Service Usage



The use of a service contract is modeled as a UML *collaboration use.*

Participants are *bound* the specific roles they play in the contract.

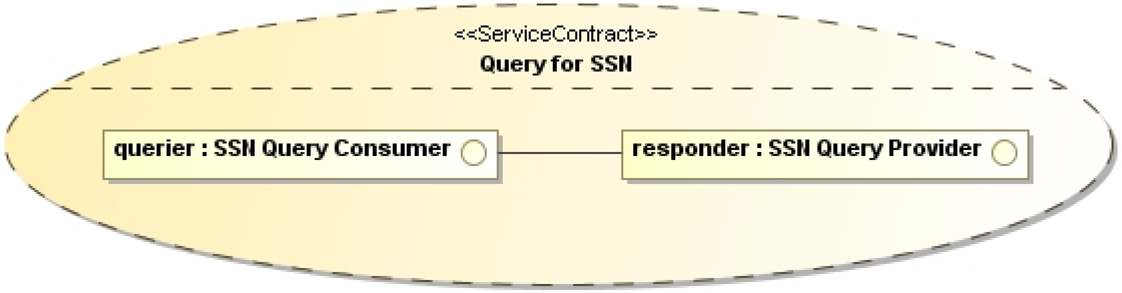
Participation in a service contract requires that the participant type have a *port* with the corresponding service interface. A port is a *connection point* for providing or consuming services.

A *request point* is a port for requesting (consuming) a service. Note that the sense of provided and required interfaces is *reversed* at a request point: The port *requires* the provider interface and *provides* the consumer interface.

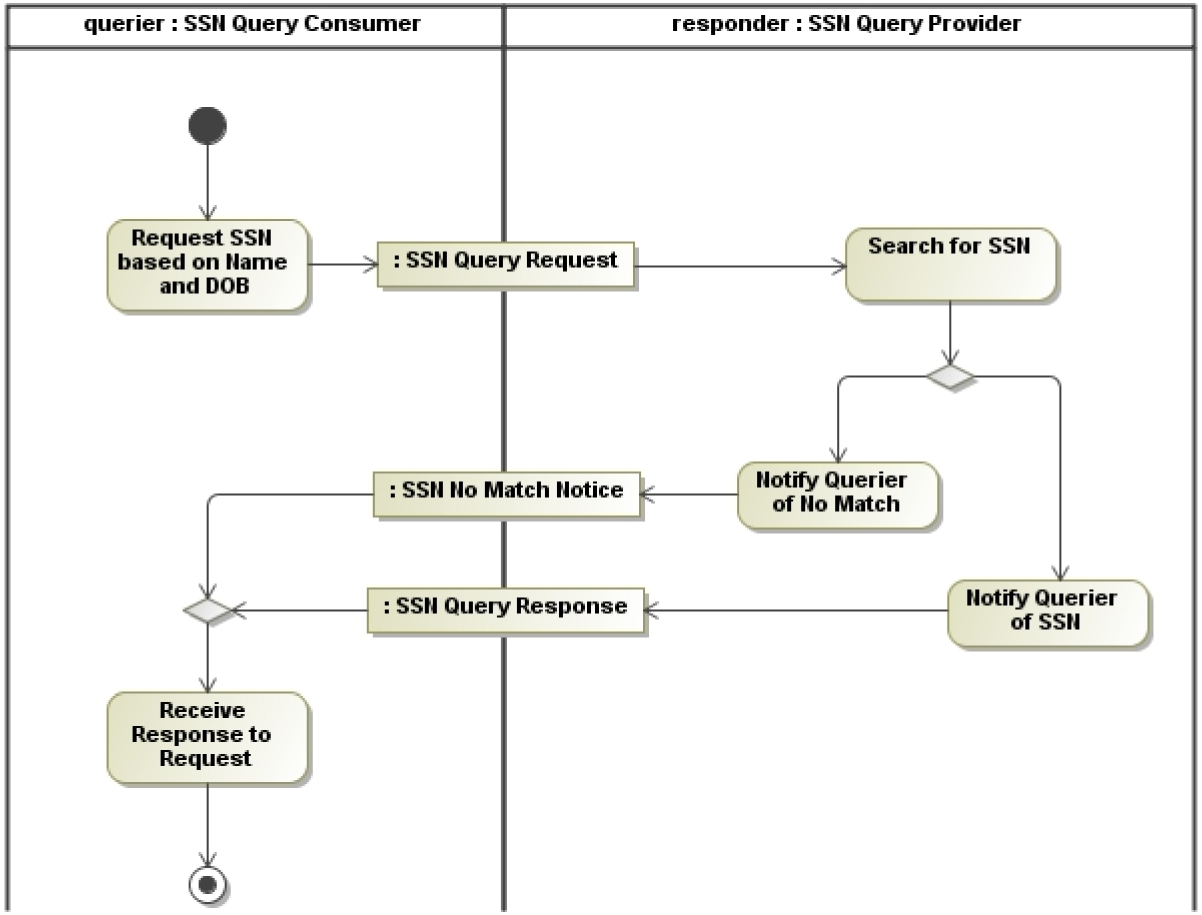
The relative interface dependencies of the request point and service point “fit together” to allow a legal connection between the service consumer and provider.

A *service point* is a port for providing a service. The port *provides* the provider interfaces and *requires* the consumer interface.

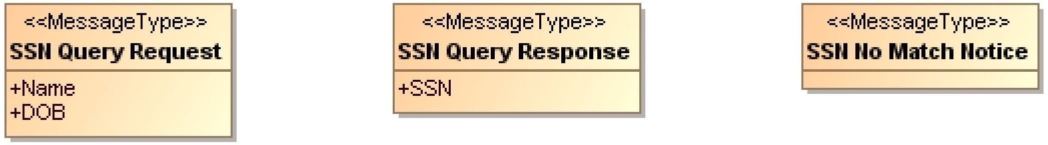
Query for SSN Service Contract



Query for SSN Behavior


querier : SSN Query Consumer
responder : SSN Query Provider
Request SSN based on Name and DOB
: SSN Query Request
Search for SSN
Notify Querier of No Match
: SSN No Match Notice
Notify Querier of SSN
: SSN Query Response
Receive Response to Request

SSN *Query* Messages

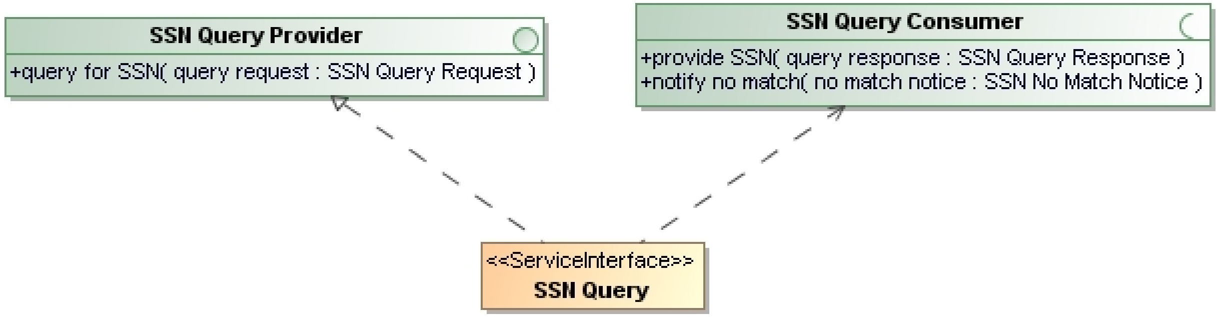


<<MessageType>>
SSN Query Request
+Name
+DOB

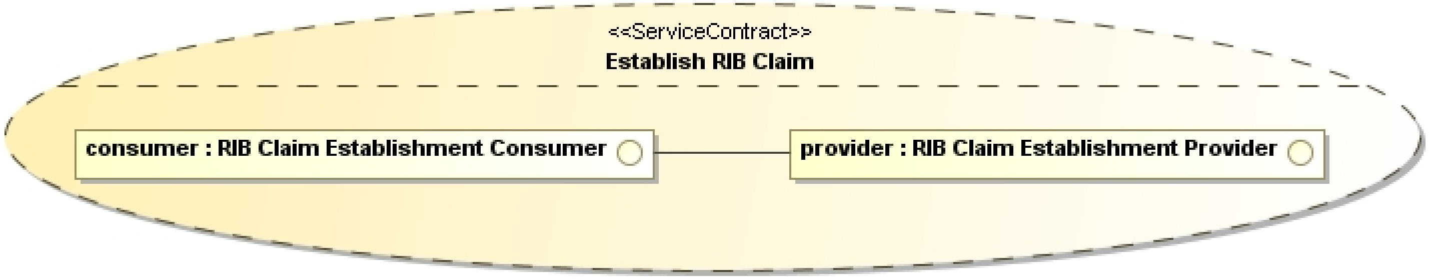
<<MessageType>>
SSN Query Response
+SSN

<<MessageType>>
SSN No Match Notice

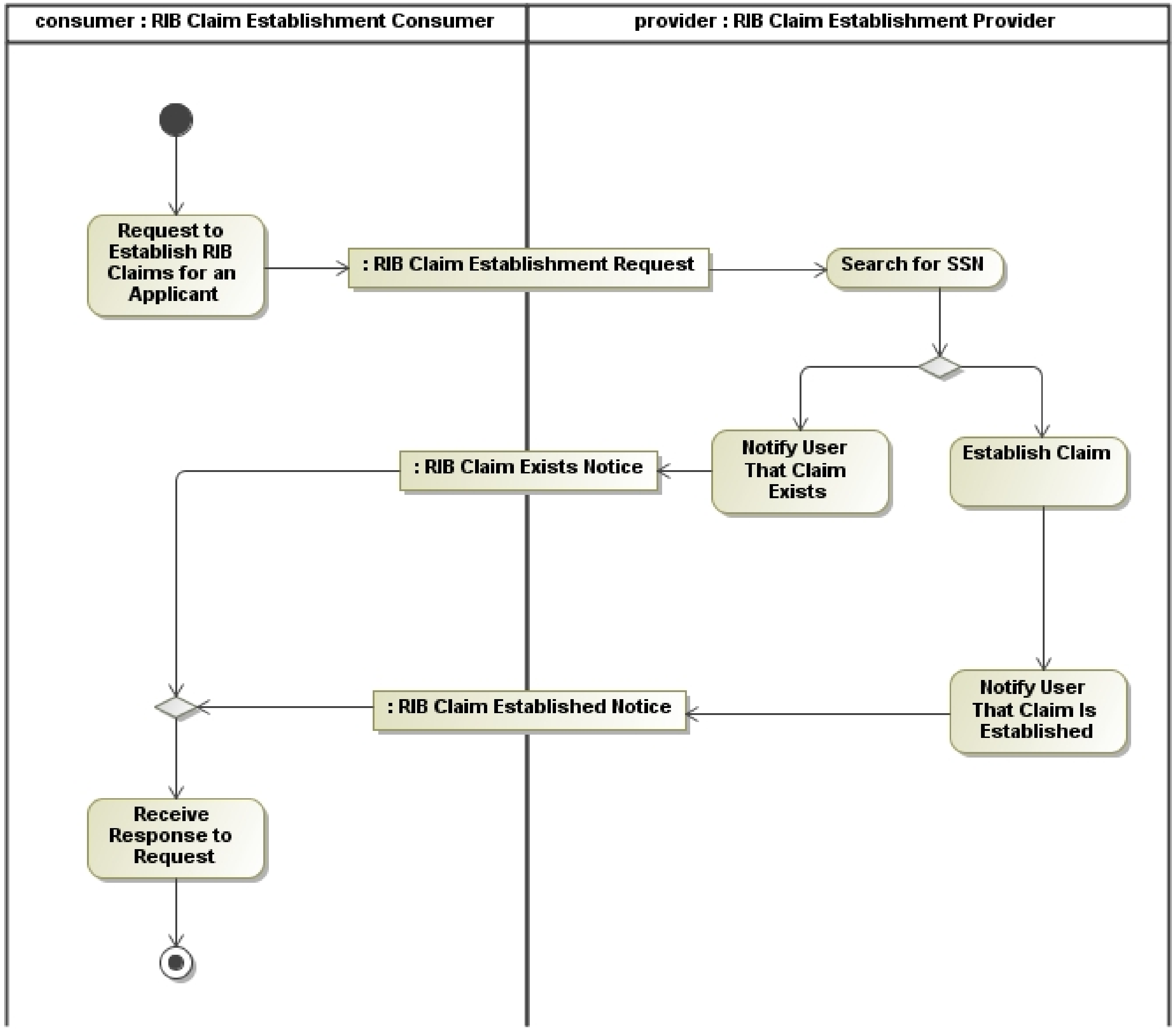
SSN Query Service Interface


SSN Query Provider
+query for SSN(query request : SSN Query Request)
SSN Query Consumer
+provide SSN(query response : SSN Query Response)
+notify no match(no match notice : SSN No Match Notice)
<<ServiceInterface>>
SSN Query

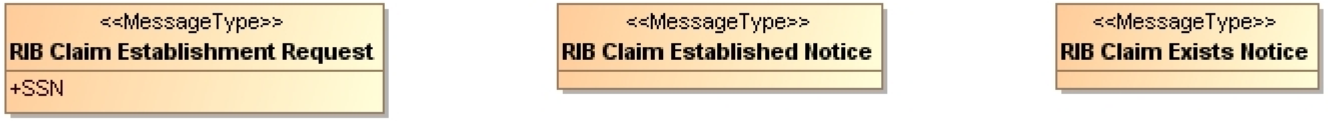
Establish RIB Claim Service Contract


<<ServiceContract>>
Establish RIB Claim
consumer : RIB Claim Establishment Consumer
provider : RIB Claim Establishment Provider

Establish RIB Claim Behavior


consumer : RIB Claim Establishment Consumer
provider : RIB Claim Establishment Provider
Request to Establish RIB Claims for an Applicant
: RIB Claim Establishment Request
Search for SSN
Notify User That Claim Exists
Establish Claim
: RIB Claim Exists Notice
Notify User That Claim Is Established
: RIB Claim Established Notice
Receive Response to Request

RIB Claim Establishment Messages

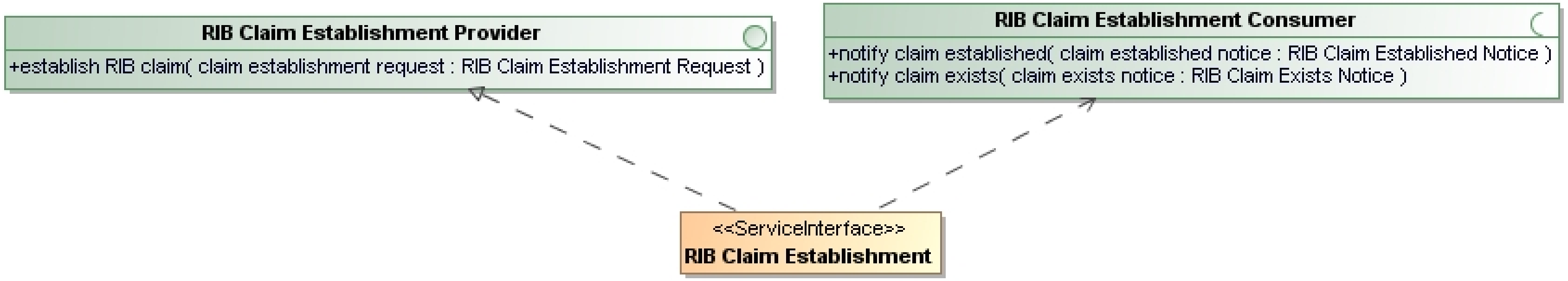


<<MessageType>>
RIB Claim Establishment Request
+SSN

<<MessageType>>
RIB Claim Established Notice

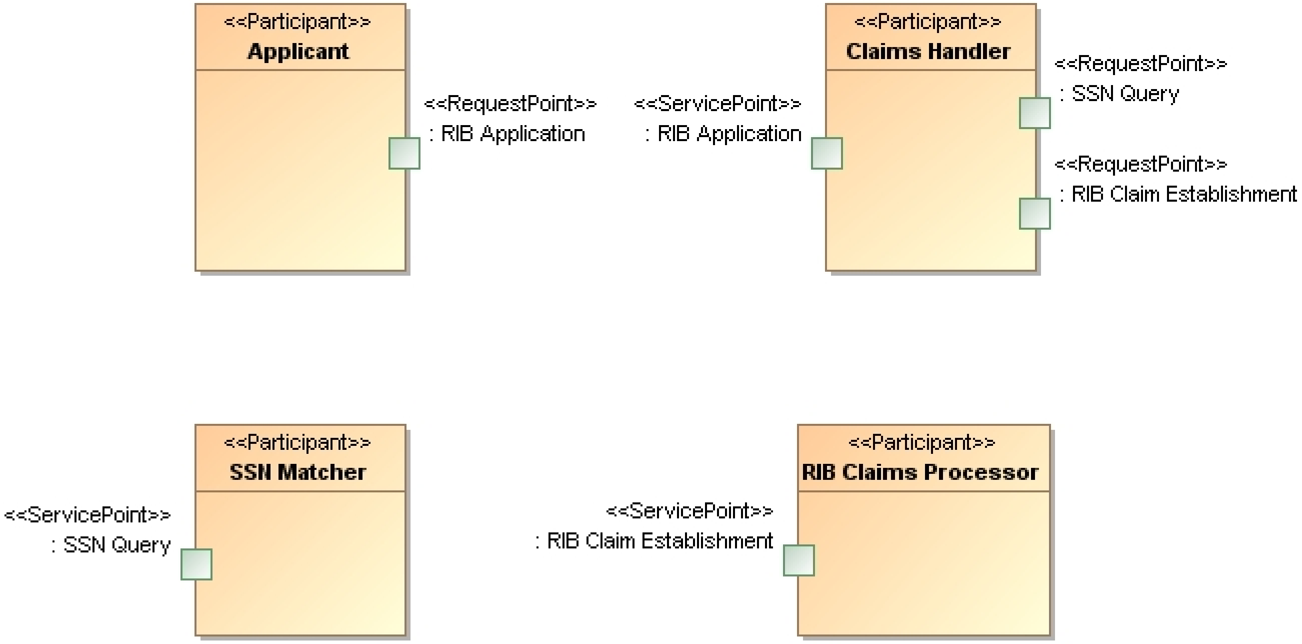
<<MessageType>>
RIB Claim Exists Notice

RIB Claim Establishment Service Interface

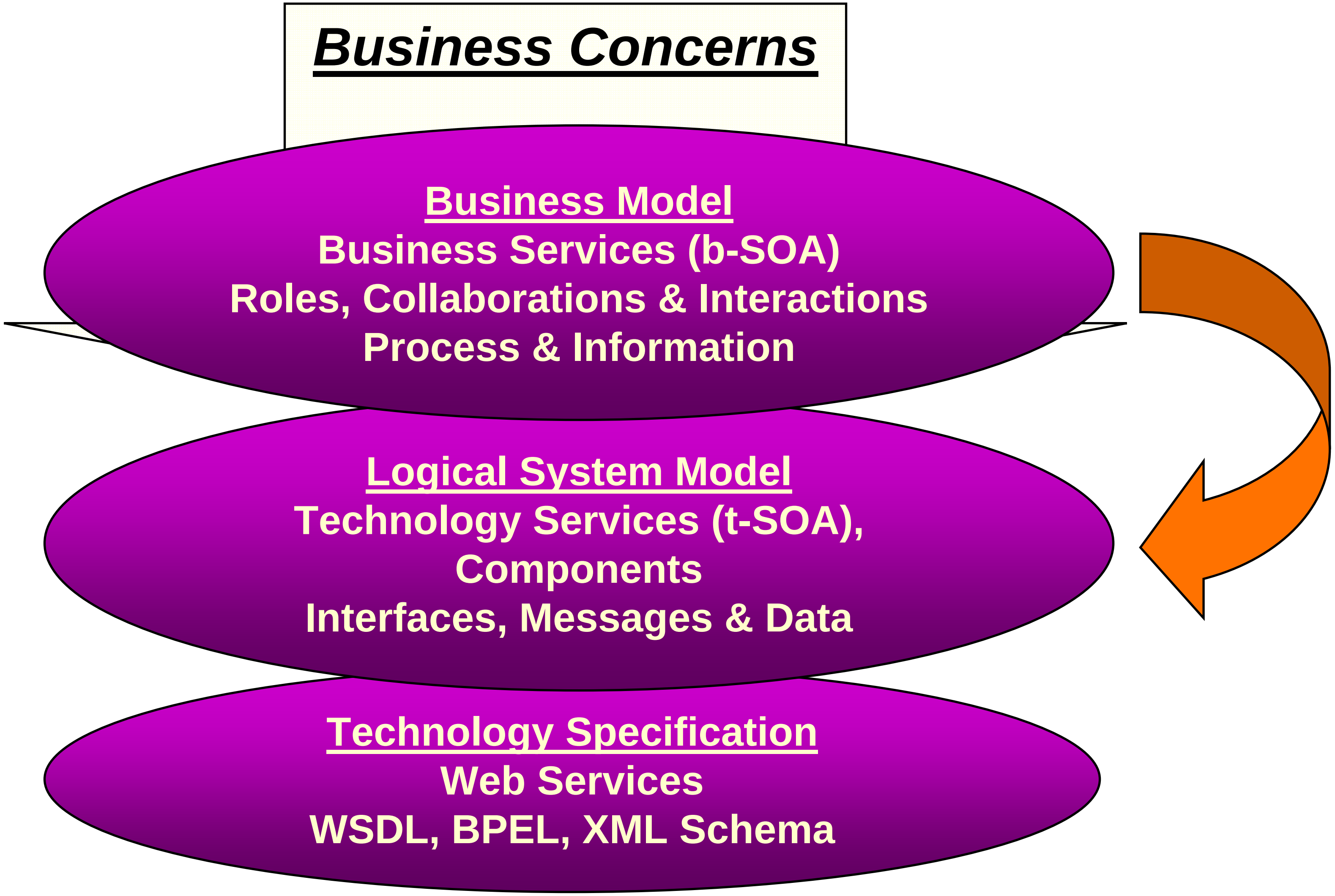

RIB Claim Establishment Provider
+establish RIB claim(claim establishment request : RIB Claim Establishment Request)
RIB Claim Establishment Consumer
+notify claim established(claim established notice : RIB Claim Established Notice)
+notify claim exists(claim exists notice : RIB Claim Exists Notice)
<<ServiceInterface>>
RIB Claim Establishment

RIB Claims Processing Participants

The full specification of a participant includes ports for every service contract in which the participant participates within the services architecture.



Producing the logical systems model



Business Concerns

Business Model

Business Services (b-SOA)

Roles, Collaborations & Interactions

Process & Information

Logical System Model

Technology Services (t-SOA),

Components

Interfaces, Messages & Data

Technology Specification

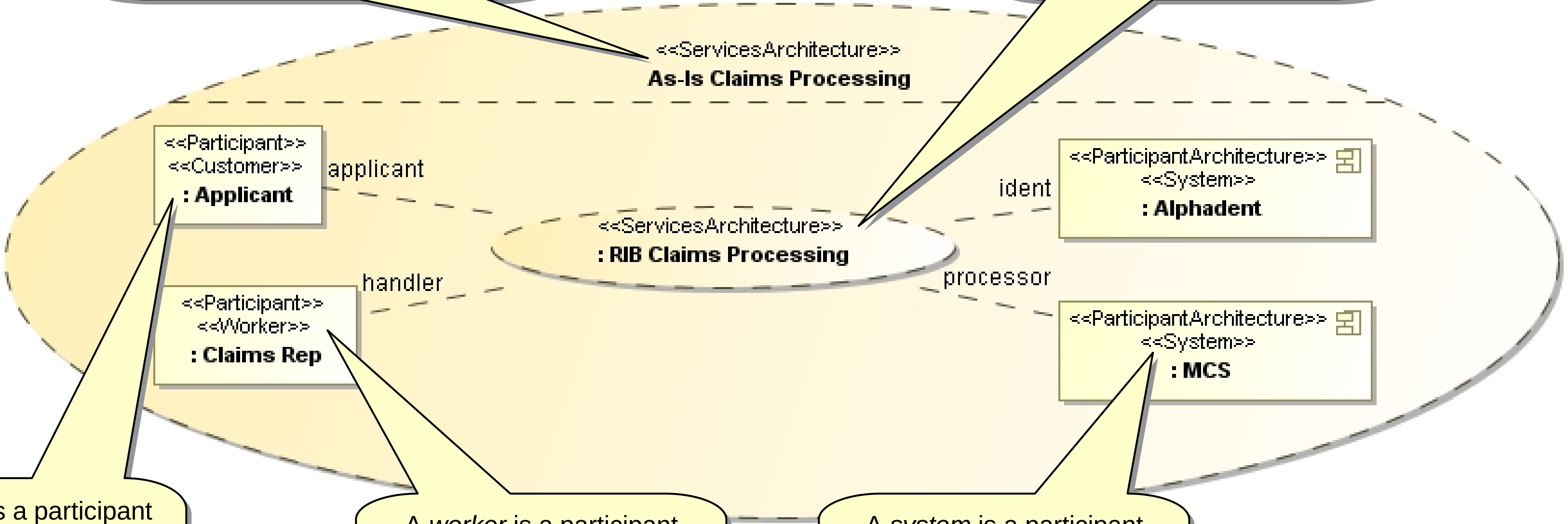
Web Services

WSDL, BPEL, XML Schema

Platform Independent Model (PIM)

- As-Is Claims Processing Services Architecture
  - Human Participants
  - System Participant Architectures
- MCS: Potential Tiered Replacement Architecture
- Claims Processing System: Potential Replacement Architecture
  - Citizen Self Service
  - Claims Rep Assisted Service

As-Is Claims Processing Services Architecture



<<ServicesArchitecture>>
As-Is Claims Processing
<<Participant>> <<Customer>> : Applicant
applicant
<<Participant>> <<Worker>> : Claims Rep
handler
<<ServicesArchitecture>> : RIB Claims Processing
ident
processor
<<ParticipantArchitecture>> <<System>> : Alphadent
<<ParticipantArchitecture>> <<System>> : MCS

The as-is claims processing architecture is modeled here as a services architecture showing how the roles CIM-level business architecture are currently being played.

The business process being carried out is defined by the CIM-level services architecture, which defines the process roles and desired behavior.

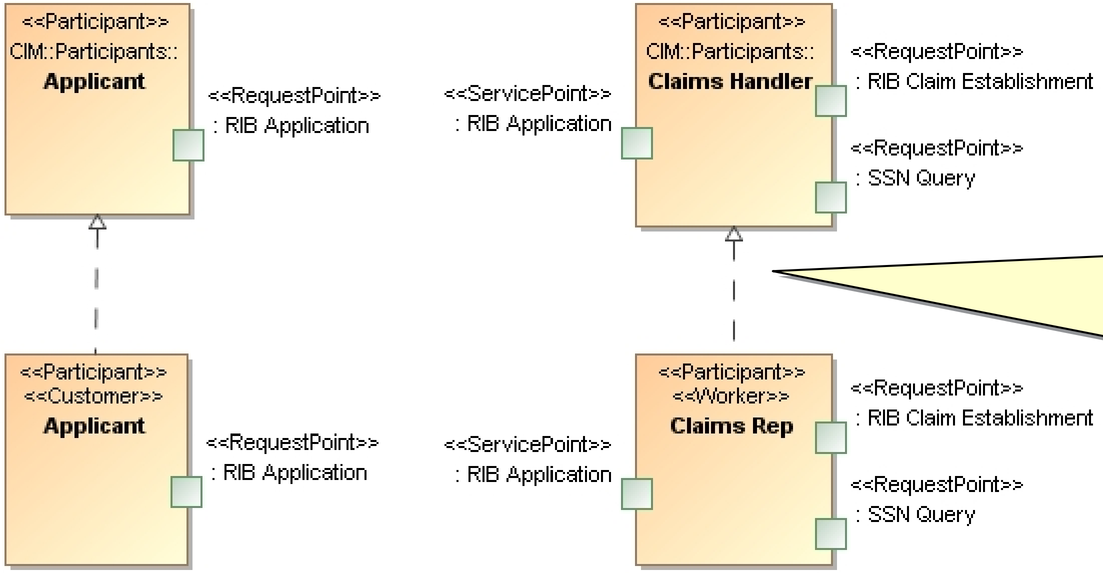
A *customer* is a participant who is external to, and being served by, the enterprise carrying out the business process.

A *worker* is a participant who is internal to the enterprise carrying out the business process.

A *system* is a participant that whose responsibilities are being automated using an IT system.

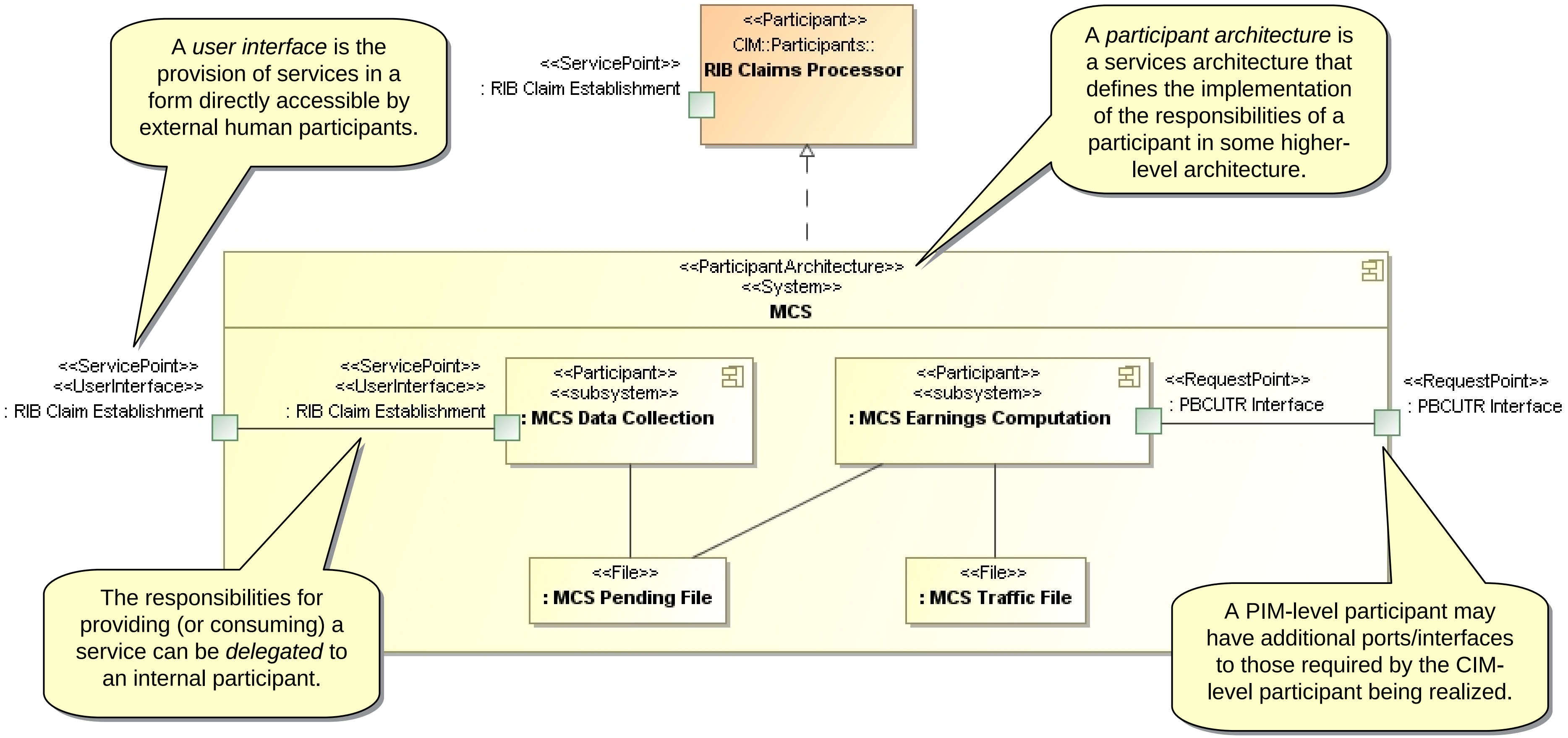
As-Is Claims Processing Human Participants

At the PIM-level, some participants may be known not to be automated. Such participant types generally represent *positions* filled by people in the enterprise.



Participants at the PIM level can *realize* (one or more) participants at the CIM level. This indicates the intended way the PIM-level participants are to participate in various business processes. The PIM-level participant model must have ports that conform to all the ports of the CIM-level participant.

MCS System Architecture



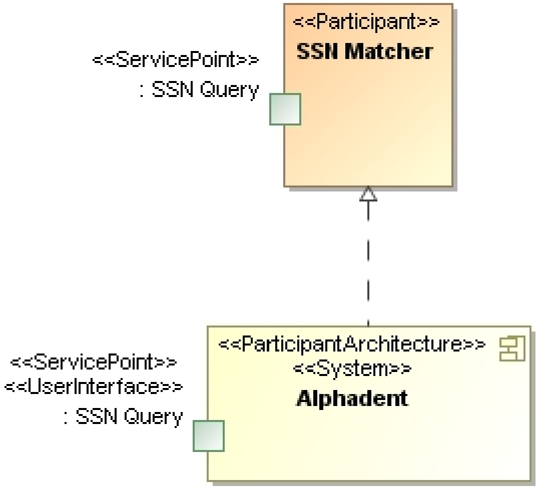
A *user interface* is the provision of services in a form directly accessible by external human participants.

A *participant architecture* is a services architecture that defines the implementation of the responsibilities of a participant in some higher-level architecture.

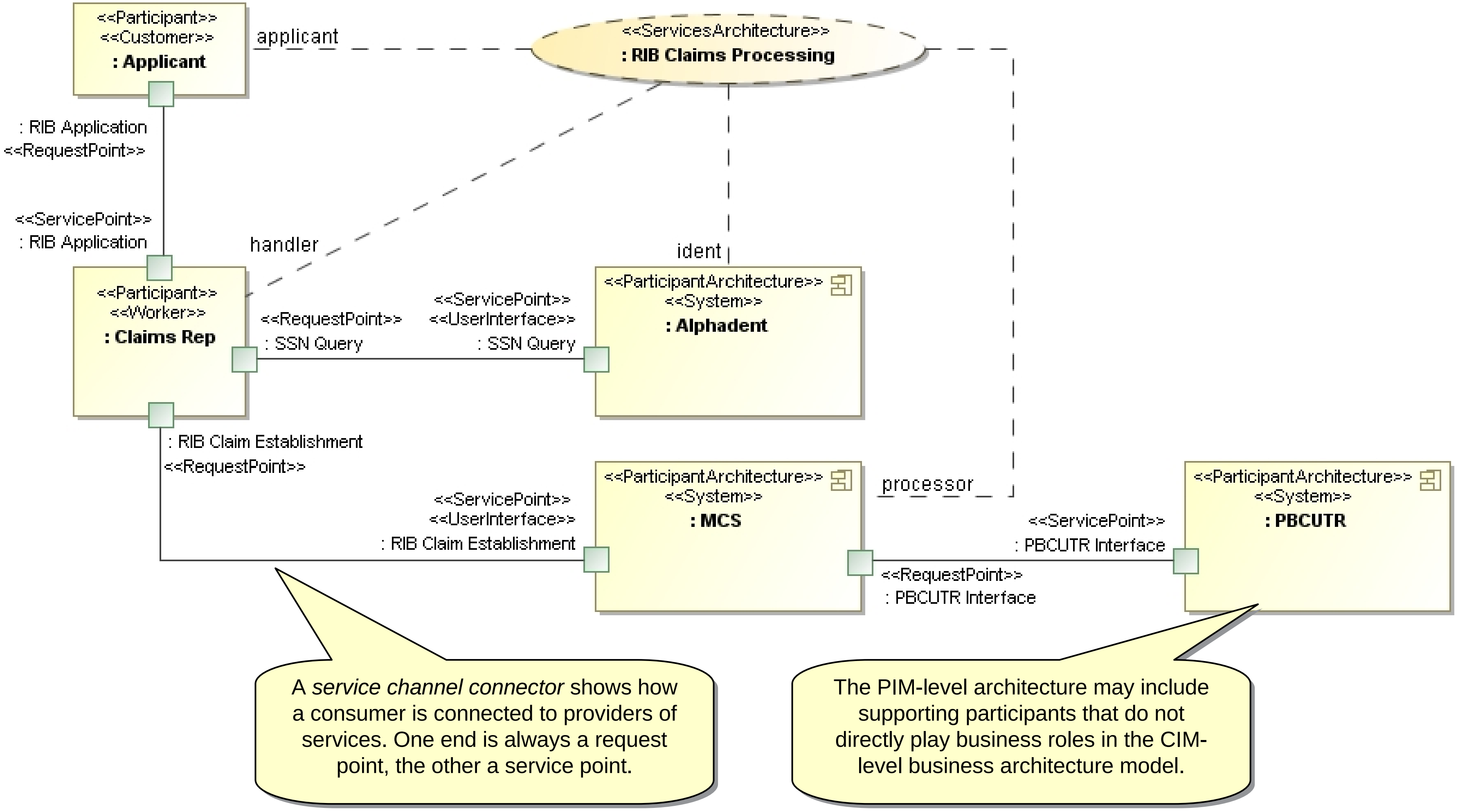
The responsibilities for providing (or consuming) a service can be *delegated* to an internal participant.

A PIM-level participant may have additional ports/interfaces to those required by the CIM-level participant being realized.

Alphadent System Architecture



As-Is Claim Processing Composite Structure

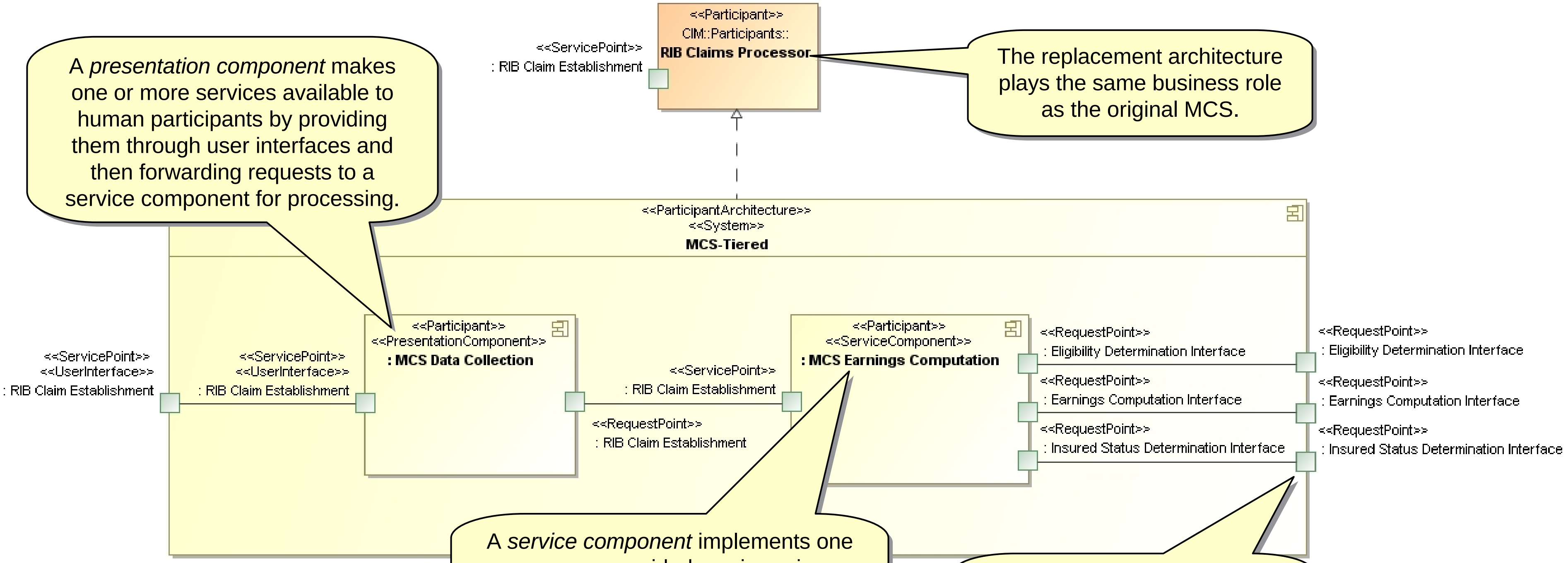


A *service channel connector* shows how a consumer is connected to providers of services. One end is always a request point, the other a service point.

The PIM-level architecture may include supporting participants that do not directly play business roles in the CIM-level business architecture model.

MCS Potential Tiered Replacement Architecture

This is a simplified example of a logically tiered replacement architecture for MCS, in which internal file interfaces have been replaced by a service interface.



<<Participant>>
CIM::Participants::
RIB Claims Processor
<<ServicePoint>>
: RIB Claim Establishment
<<ParticipantArchitecture>>
<<System>>
MCS-Tiered
<<ServicePoint>>
<<UserInterface>>
: RIB Claim Establishment
<<ServicePoint>>
<<UserInterface>>
: RIB Claim Establishment
<<Participant>>
<<PresentationComponent>>
: MCS Data Collection
<<ServicePoint>>
: RIB Claim Establishment
<<RequestPoint>>
: RIB Claim Establishment
<<Participant>>
<<ServiceComponent>>
: MCS Earnings Computation
<<RequestPoint>>
: Eligibility Determination Interface
<<RequestPoint>>
: Earnings Computation Interface
<<RequestPoint>>
: Insured Status Determination Interface
<<RequestPoint>>
: Eligibility Determination Interface
<<RequestPoint>>
: Earnings Computation Interface
<<RequestPoint>>
: Insured Status Determination Interface

The replacement architecture plays the same business role as the original MCS.

A *presentation component* makes one or more services available to human participants by providing them through user interfaces and then forwarding requests to a service component for processing.

A *service component* implements one or more provided services via interfaces available to other system components (as opposed to humans).

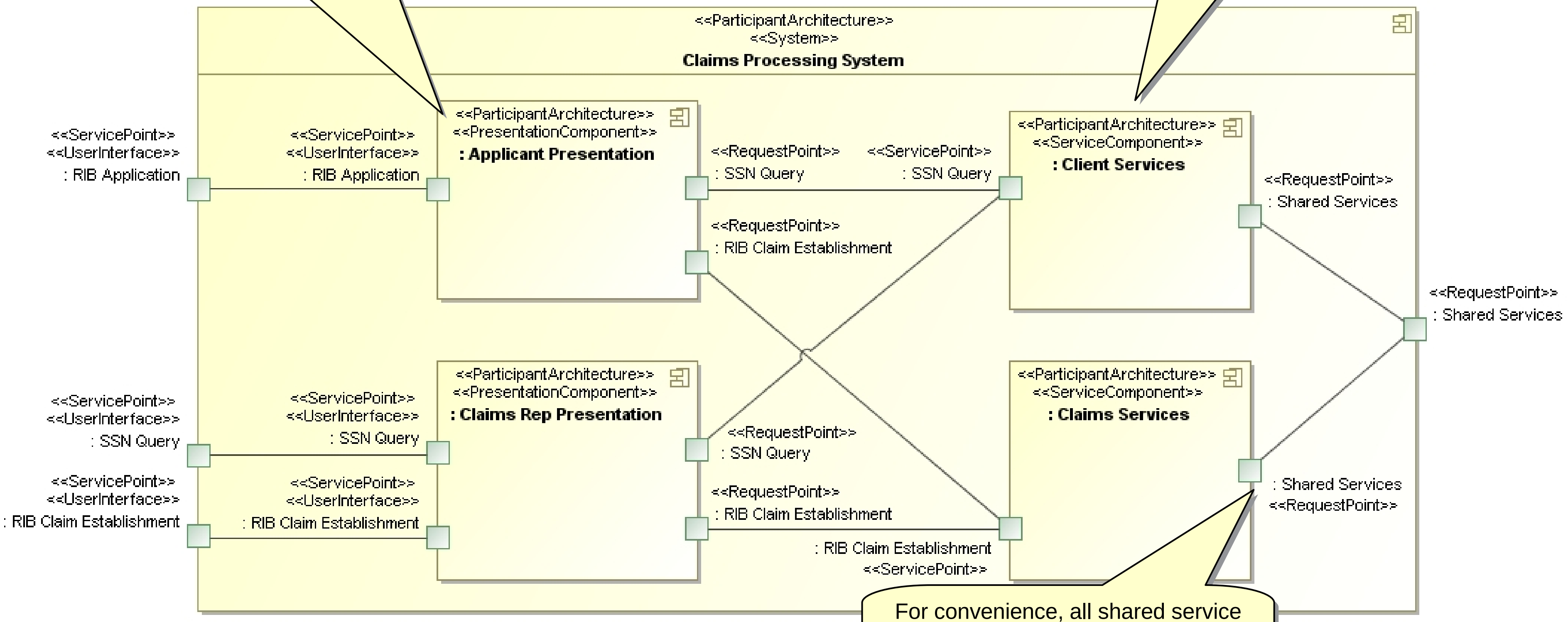
These services are presumed to be available from some shared service provider.

Claims Processing System Potential Replacement Architecture

An applicant presentation component allows for customer self-service, mediating the use of the same services internally that are directly available to a claims rep.

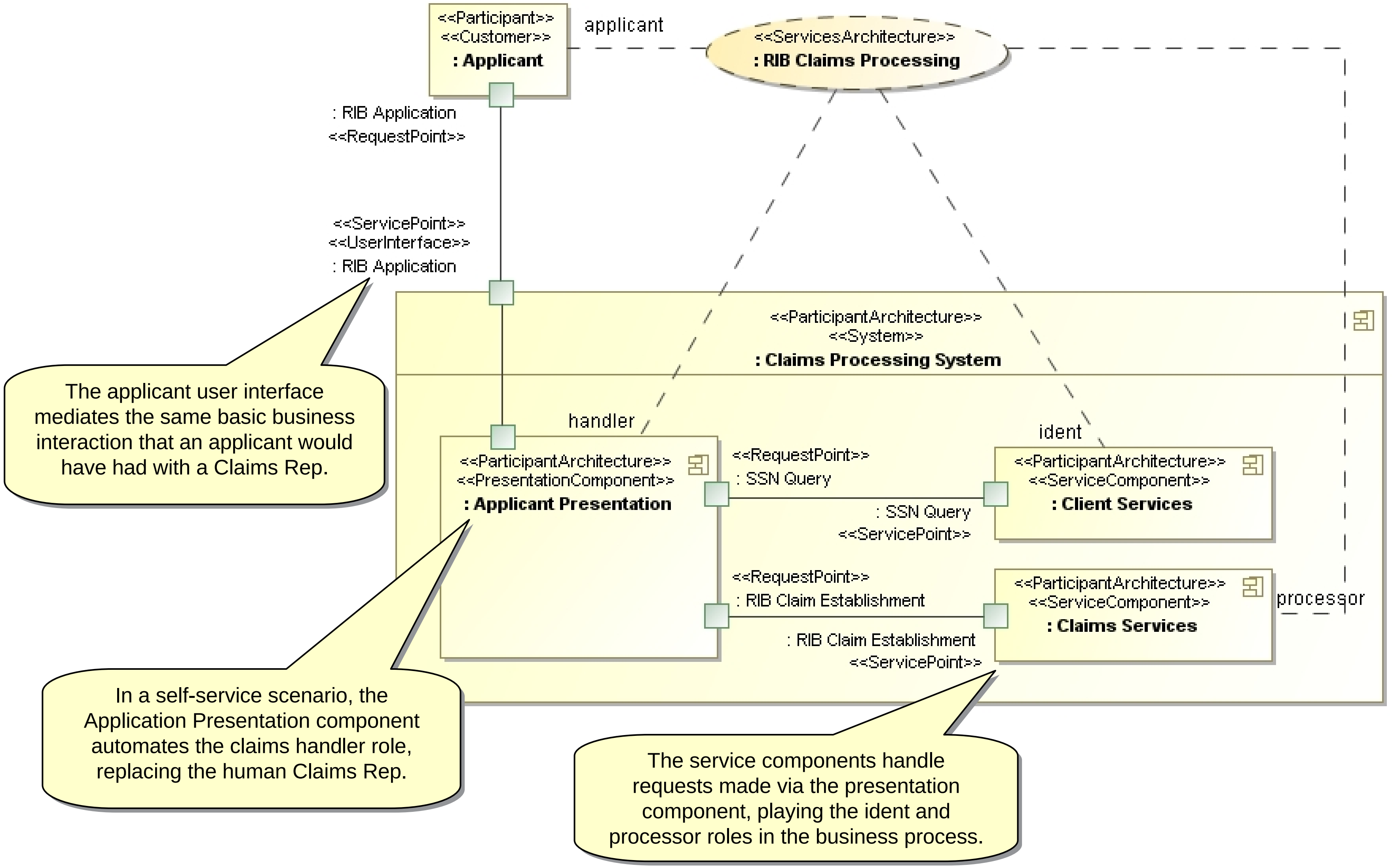
This is a simplified example of a more sweeping replacement for the entire claims processing architecture.

Service components are designed to provide services specifically related to different business entities.



For convenience, all shared service interfaces have been grouped into a single *composite* service interface.

To-Be Claims Processing Architecture: Citizen Self Service

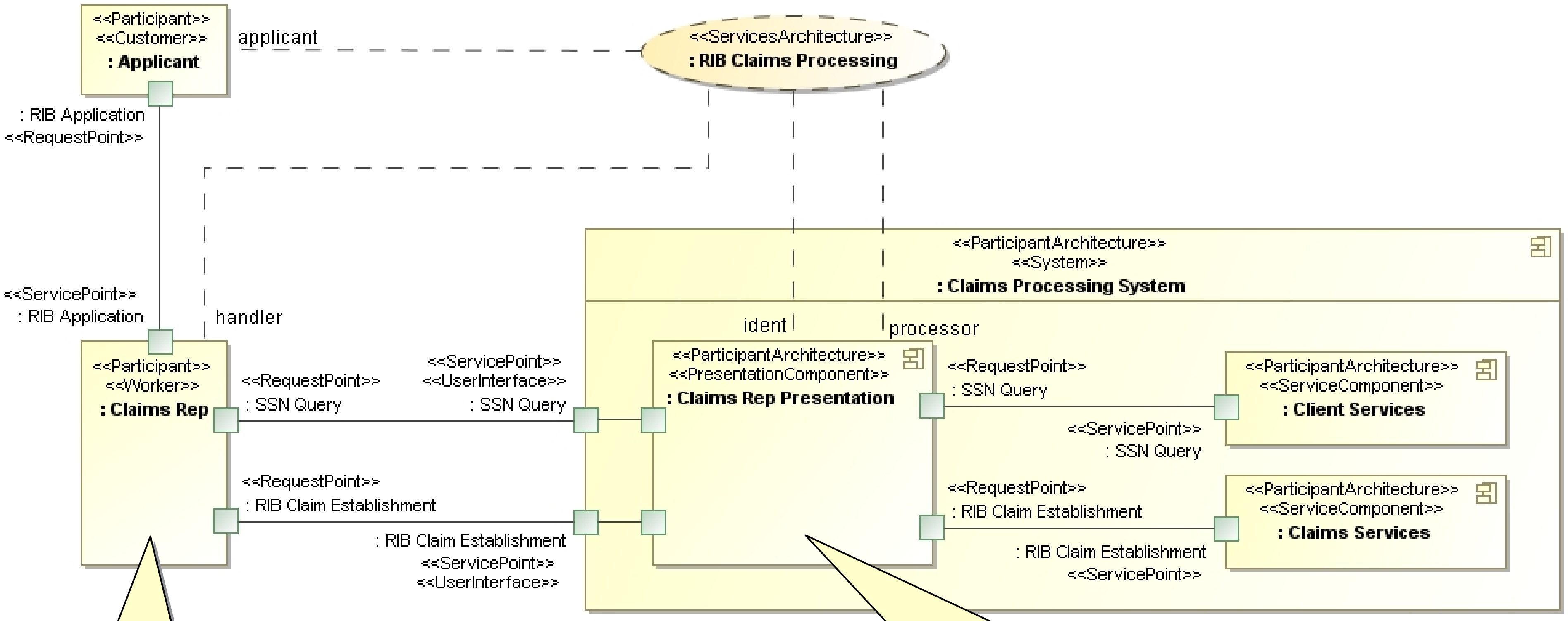


The applicant user interface mediates the same basic business interaction that an applicant would have had with a Claims Rep.

In a self-service scenario, the Application Presentation component automates the claims handler role, replacing the human Claims Rep.

The service components handle requests made via the presentation component, playing the ident and processor roles in the business process.

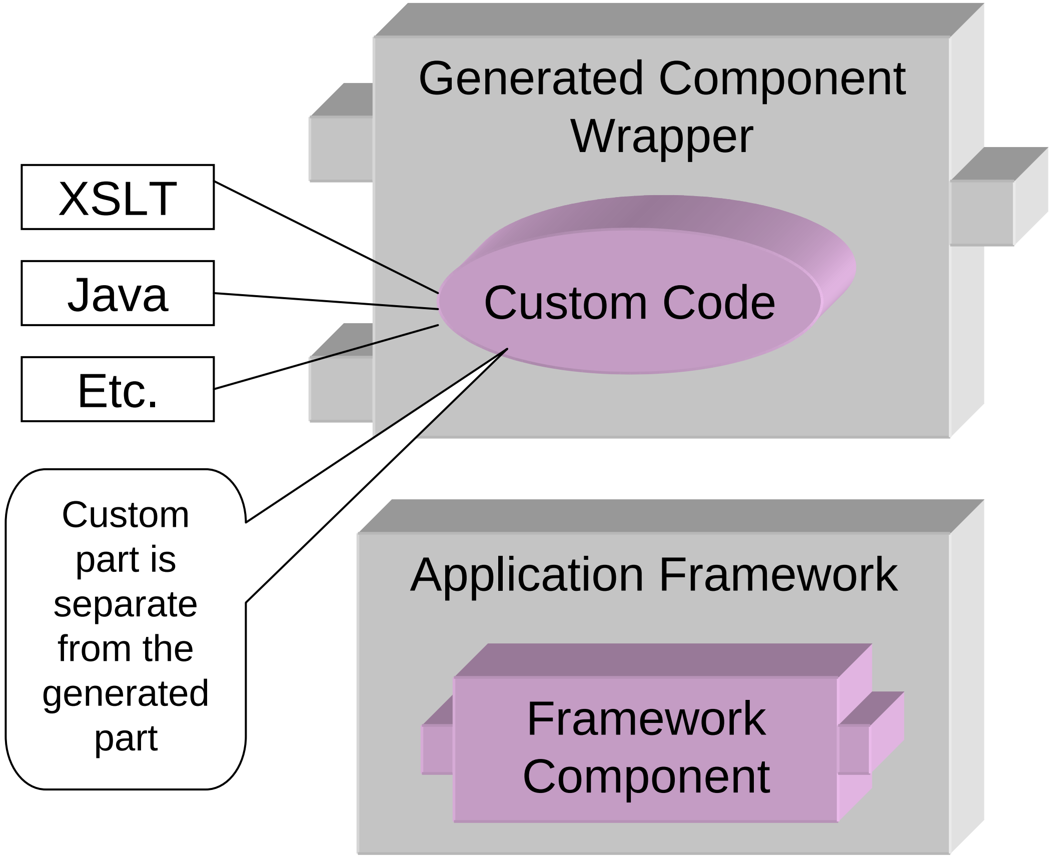
To-Be Claims Processing Architecture: Claims Rep Assisted Service



In an assisted-service scenario, the Claims Rep still plays the handler role for the Applicant.

The Claims Rep Presentation component acts as a *façade* allowing the Claims Rep access to the underlying services provided by the service components. Relative to the Claims Rep it effectively plays *both* the ident and processor roles.

Custom Business Logic Components


Generated Component Wrapper
Custom Code
XSLT
Java
Etc.
Custom part is separate from the generated part
Application Framework
Framework Component

Application components provide service implementations with user supplied logic. These “plug into” the users architecture as composite application components

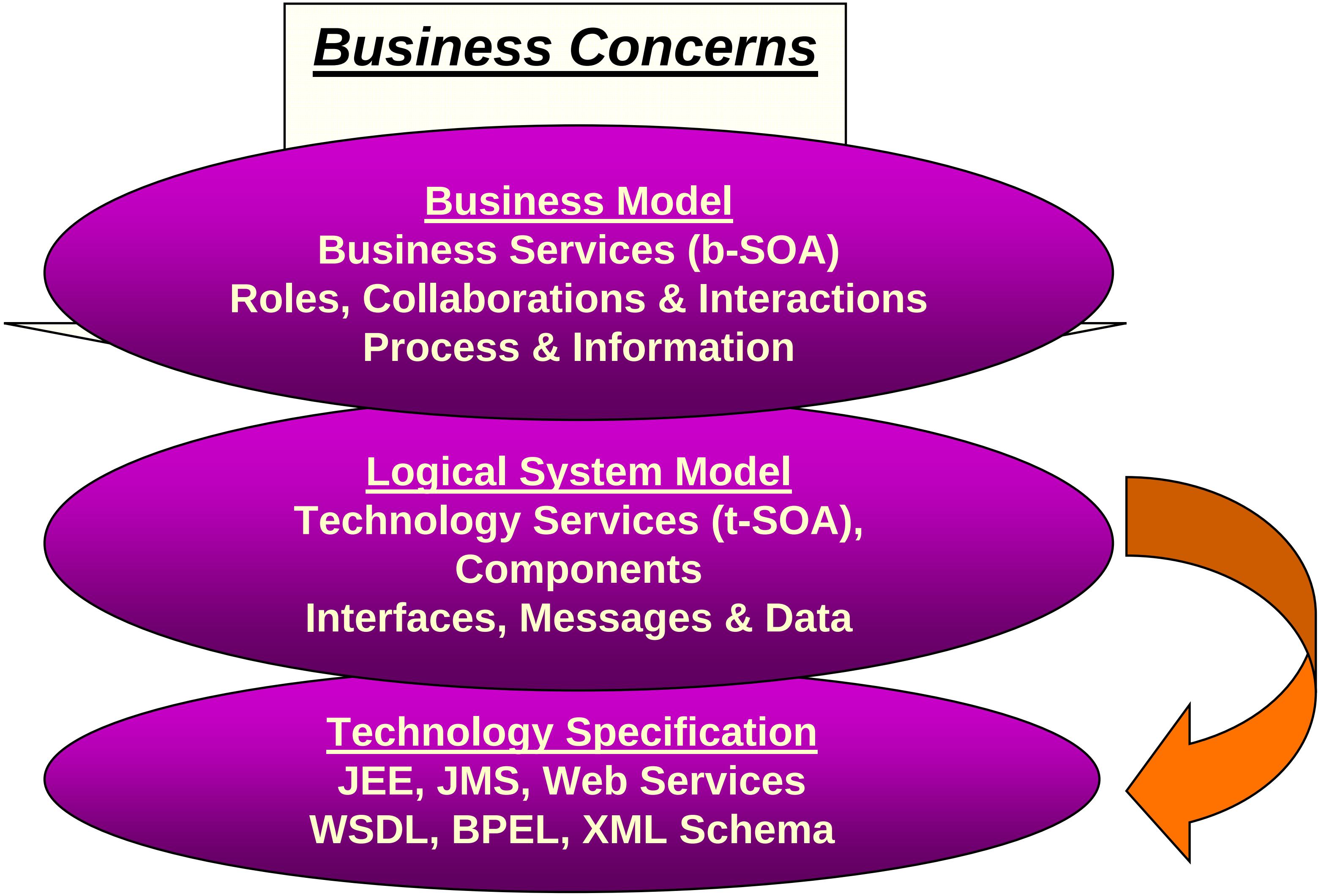
Framework components add infrastructural capabilities by extending the platform (E.G. JBI) and are called by the provisioned code or platform configuration

As MDA progresses, there will be less and less need for custom components, but the capability will remain.

Platform Specific Model (PSM)

- MCS Tiered Deployment
- Claims Processing System Tiered Deployment

Technology Architecture



Business Concerns

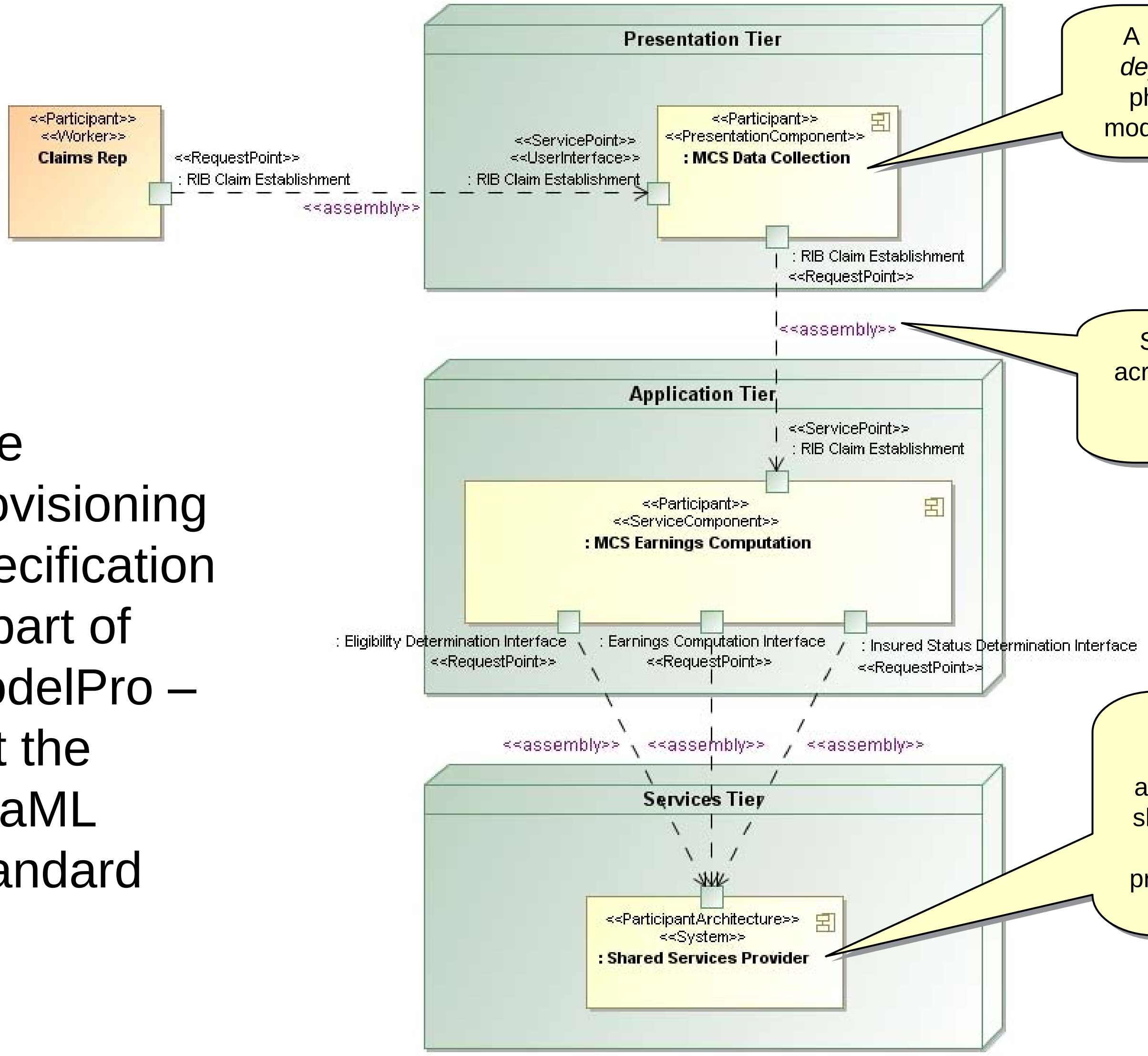
Business Model
Business Services (b-SOA)
Roles, Collaborations & Interactions
Process & Information

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

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

MCS Tiered Deployment

The provisioning specification is part of ModelPro – not the SoaML Standard



Presentation Tier

<<Participant>>
<<Worker>>
Claims Rep

<<RequestPoint>>
: RIB Claim Establishment

<<assembly>>

<<ServicePoint>>
<<UserInterface>>
: RIB Claim Establishment

<<Participant>>
<<PresentationComponent>>
: MCS Data Collection

: RIB Claim Establishment
<<RequestPoint>>

<<assembly>>

Application Tier

<<ServicePoint>>
: RIB Claim Establishment

<<Participant>>
<<ServiceComponent>>
: MCS Earnings Computation

: Eligibility Determination Interface
<<RequestPoint>>

: Earnings Computation Interface
<<RequestPoint>>

: Insured Status Determination Interface
<<RequestPoint>>

<<assembly>>

<<assembly>>

<<assembly>>

Services Tier

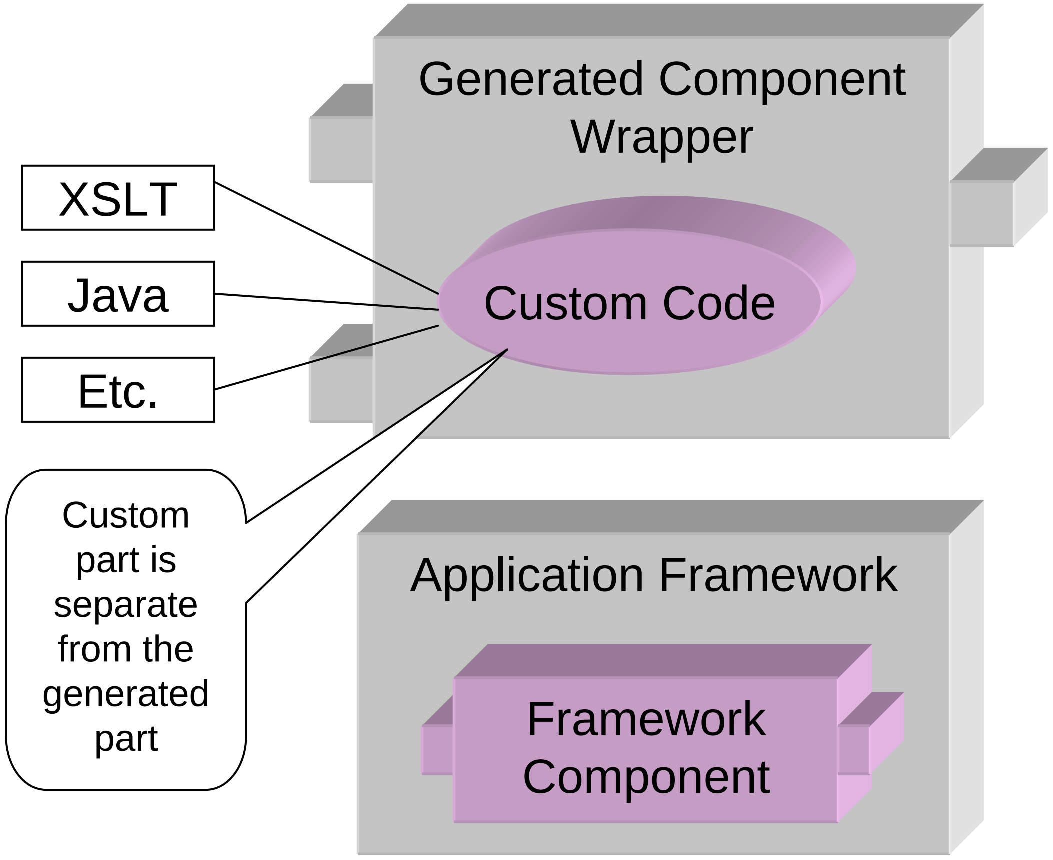
<<ParticipantArchitecture>>
<<System>>
: Shared Services Provider

A components can be *deployed* on a specific physical *tier*. A tier is modeled as a UML *node.*

Service connections across tiers are modeled as UML *assembly dependencies.*

Shared services may be provided to multiple applications. (The provider shown is illustrative—there may be more than one provider for different sets of shared services.)

Custom Business Logic Components

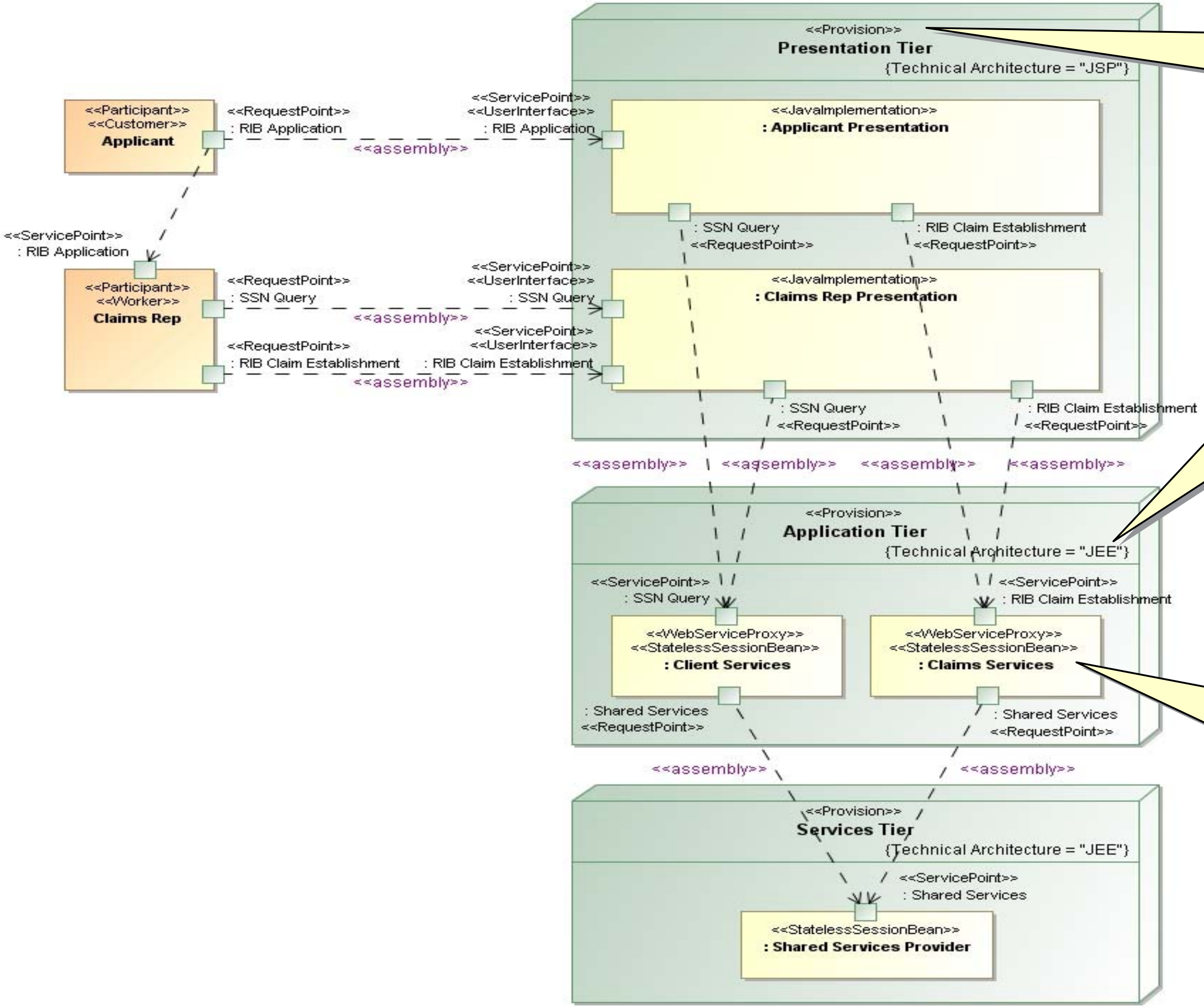

Generated Component Wrapper
XSLT
Java
Etc.
Custom Code
Custom part is separate from the generated part
Application Framework
Framework Component

Application components provide service implementations with user supplied logic. These “plug into” the users architecture as composite application components

Framework components add infrastructural capabilities by extending the platform (E.G. JBI) and are called by the provisioned code or platform configuration

As MDA progresses, there will be less and less need for custom components, but the capability will remain.

To-Be Claims Processing Tiered Deployment



Provisioning is the generation of some or all system implementation artifacts directly from the models.

The *technical architecture* for a tier defines the implementation platform to use when provisioning components on that tier.

Within the context of the technical architecture, component deployments can be stereotyped to affect how they are provisioned.

Application Provisioning

- Platform technologies are provisioned from the model based on the technology specified
  - XSD
  - WSDL
  - Application Server Configuration
  - Java Interfaces & Implementation
  - XSLT
  - IDE Project
  - SQL
  - Documentation
  - Tests
  - …

Details of what is provisioned for a particular technology are beyond the scope of this presentation