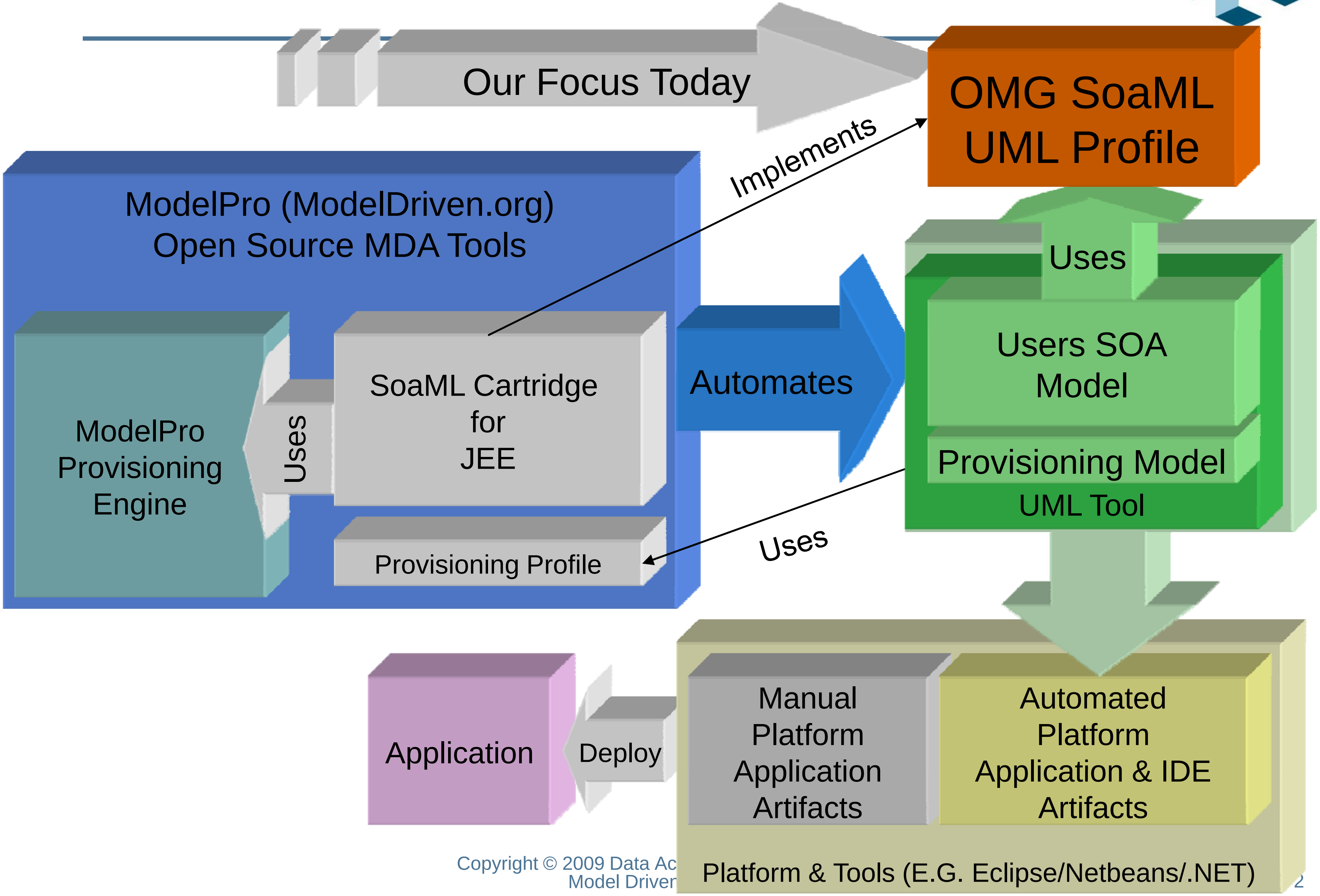
Enterprise-SOA with SoaML by Example
SOA Consortium

Cory Casanave, CEO

Cory-c (at) modeldriven.com

09 January 2009

Relating the Parts for Model Driven SOA



SoaML 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**

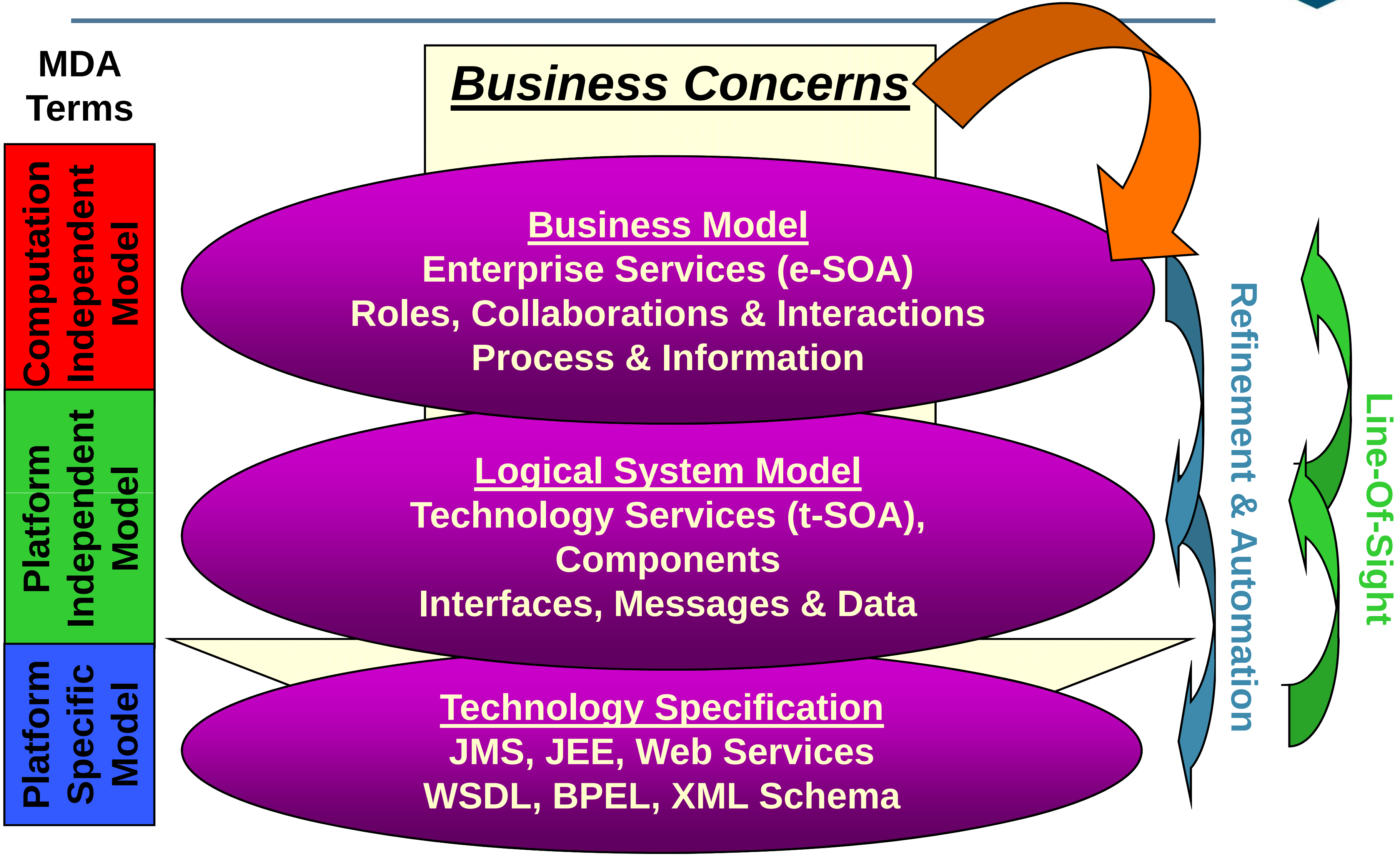
The SoaML submission team

- 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

Business Focused SOA Using Model Driven Architecture



MDA Terms

Computation Independent Model

Platform Independent Model

Platform Specific Model

Business Concerns

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

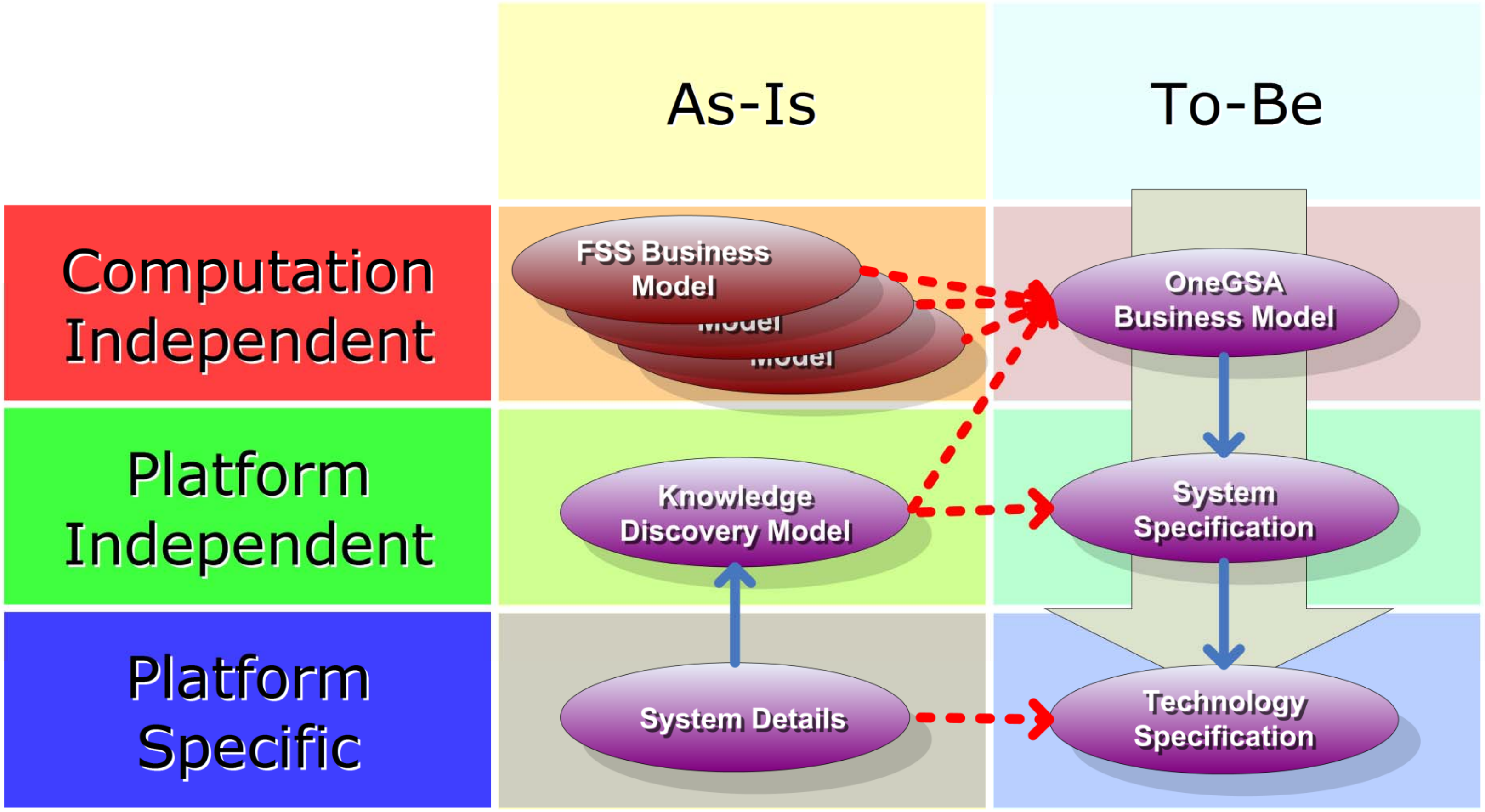
Logical System Model
Technology Services (t-SOA),
Components
Interfaces, Messages & Data

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

Refinement & Automation

Line-Of-Sight

Incorporating Legacy Analysis



As-Is

To-Be

Computation Independent

Platform Independent

Platform Specific

FSS Business Model

Model

Model

OneGSA Business Model

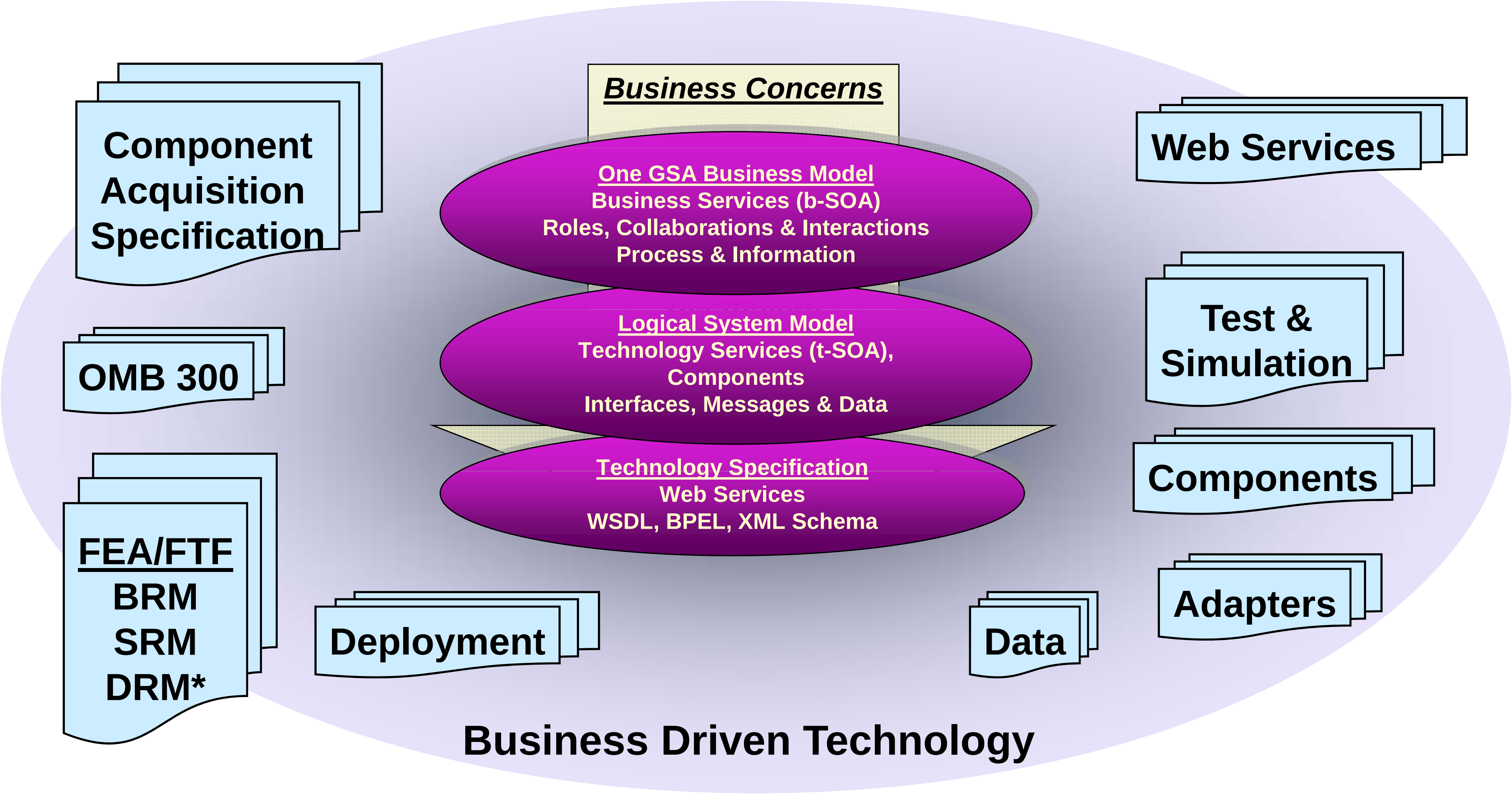
Knowledge Discovery Model

System Specification

System Details

Technology Specification

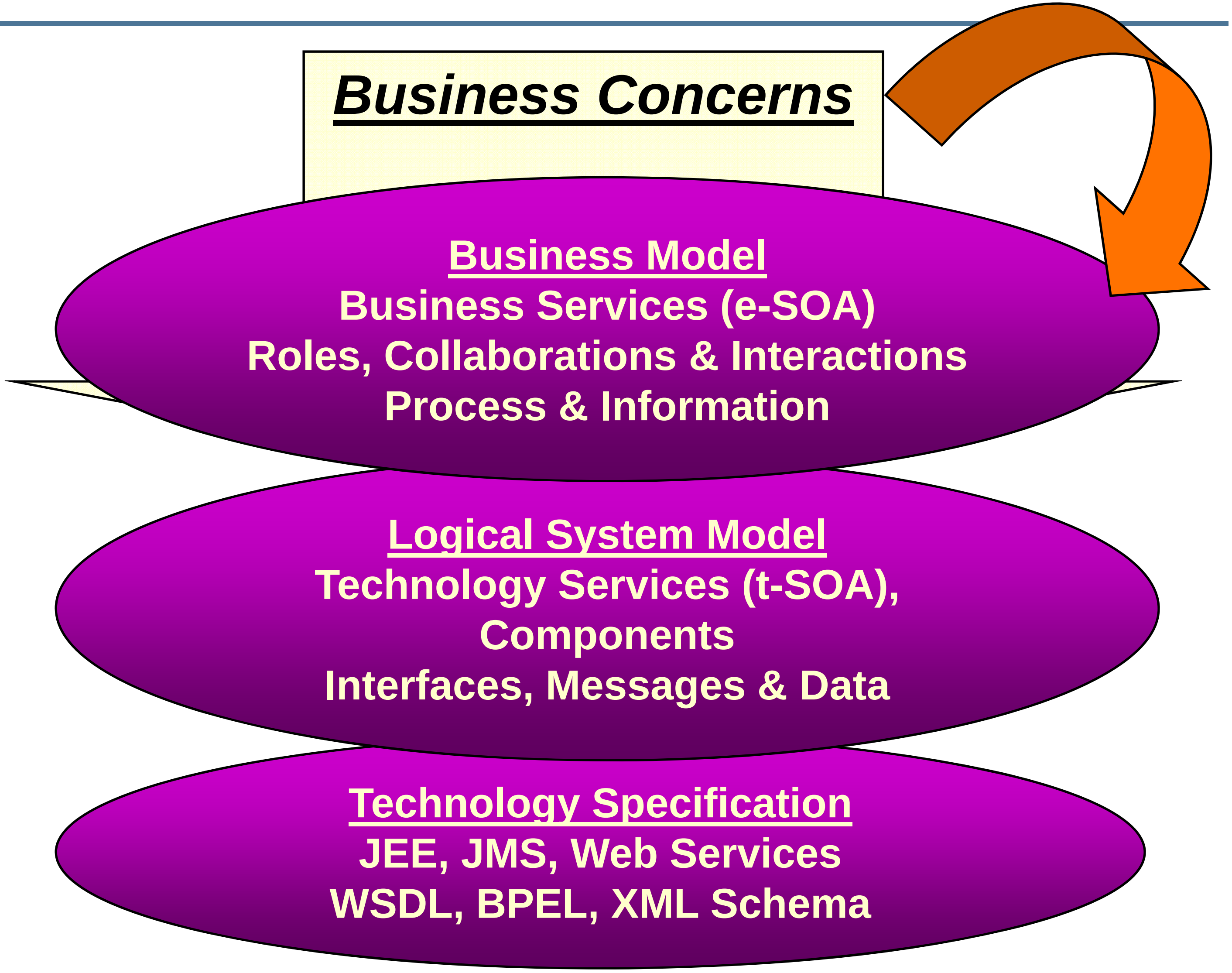
Value derived from the architecture



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)

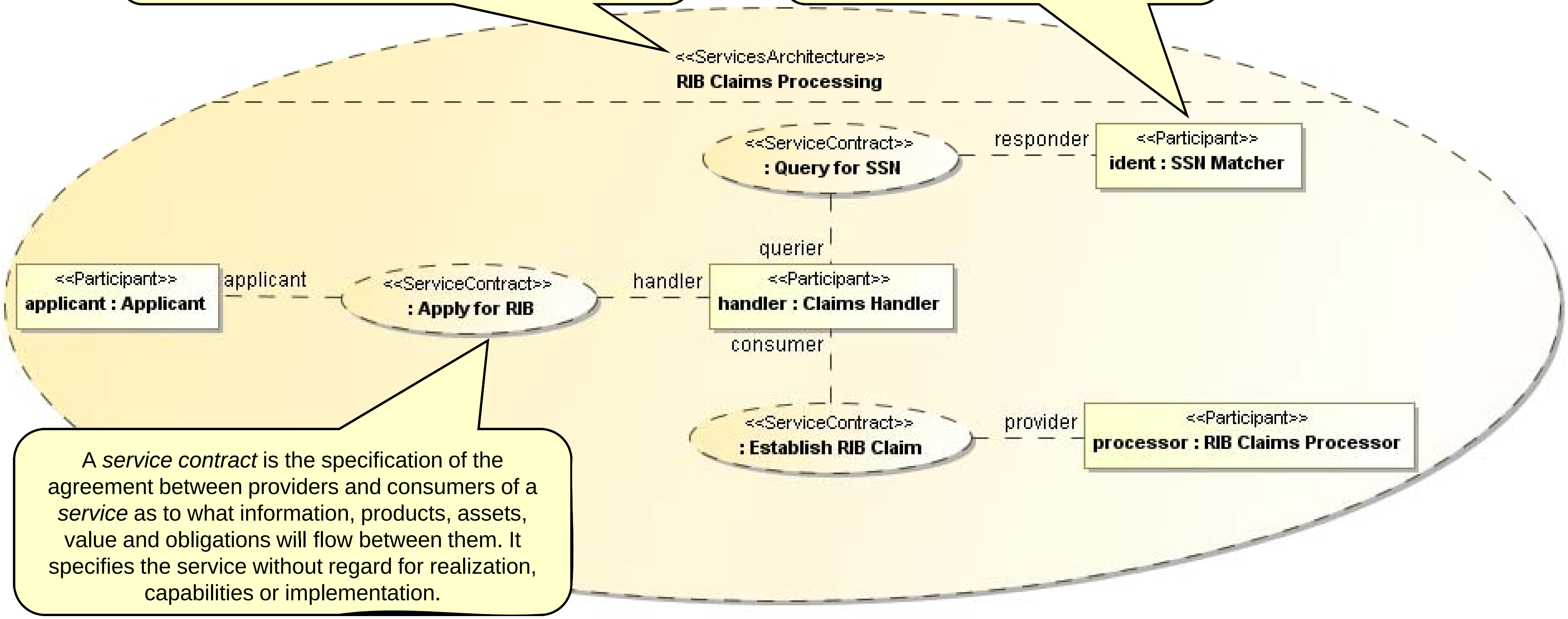
- RIB* Claims Processing Services Architecture
  - RIB Claims Processing Business Process
- Apply for RIB Service Contract
  - RIB Application Service Interface
- Query for SSN Service Contract
  - SSN Query Service Interface
- Establish RIB Claim Service Contract
  - RIB Establishment Service Interface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

“RIB” Is a domain term meaningful to the user meaning “Retirement Insurance Benefit”

RIB Claims Processing Services Architecture

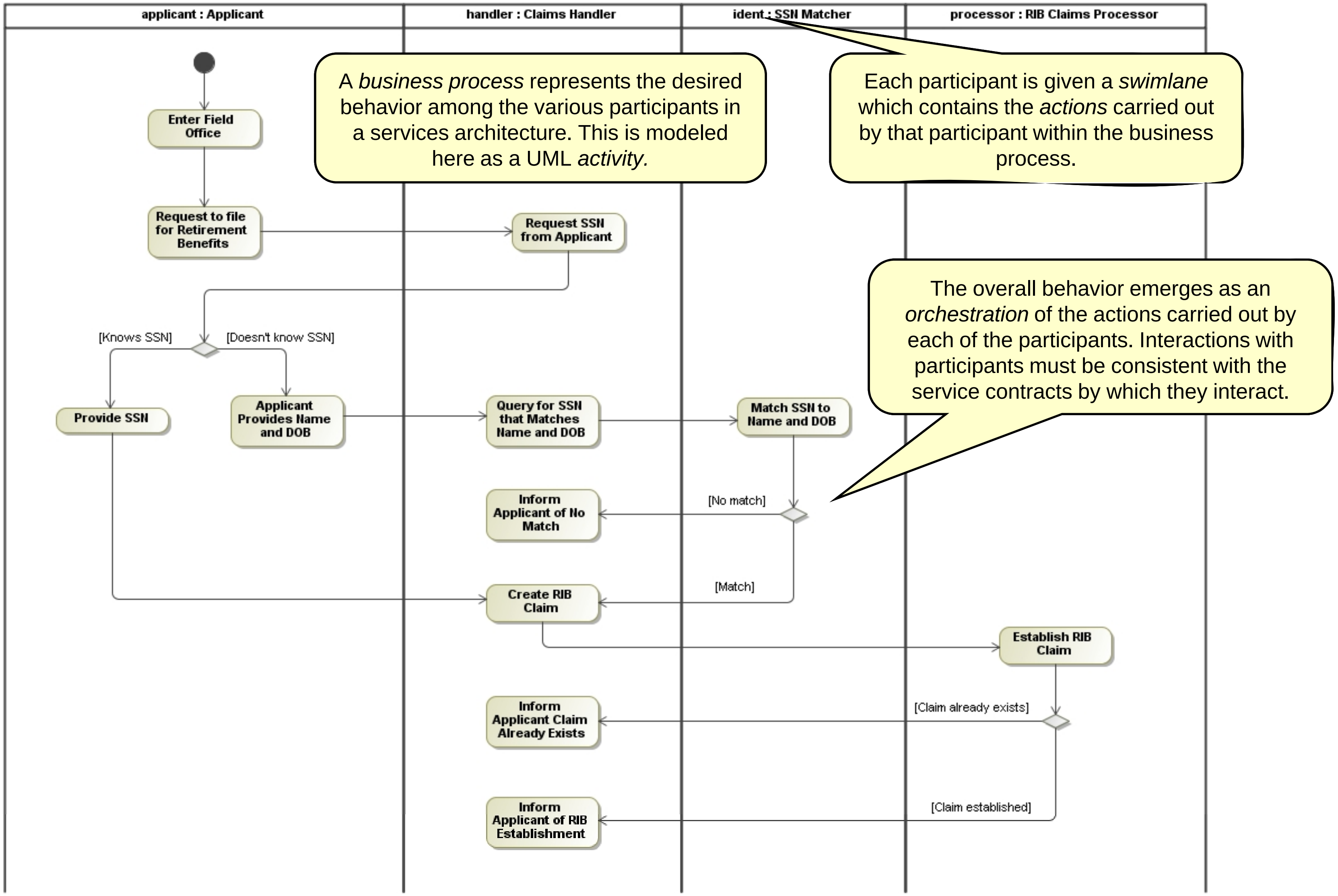
A *services architecture* describes how *participants* work together for a purpose by providing and using services expressed as *service contracts*. It is modeled as a UML *collaboration.*

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

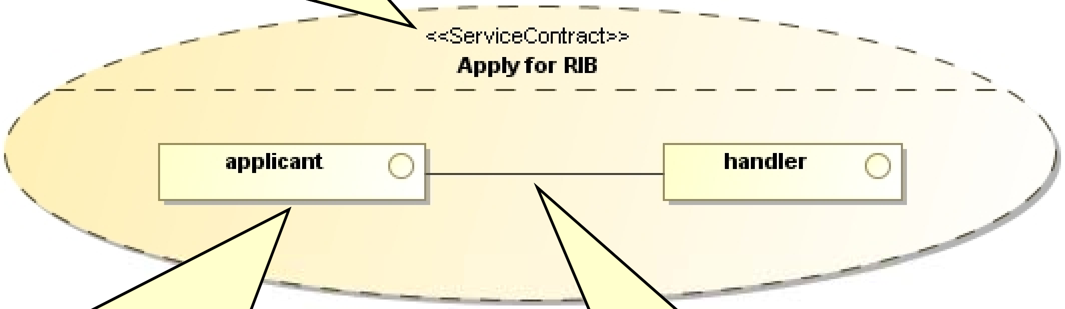
A *business process* represents the desired behavior among the various participants in a services architecture. This is modeled here as a UML *activity.*

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.*

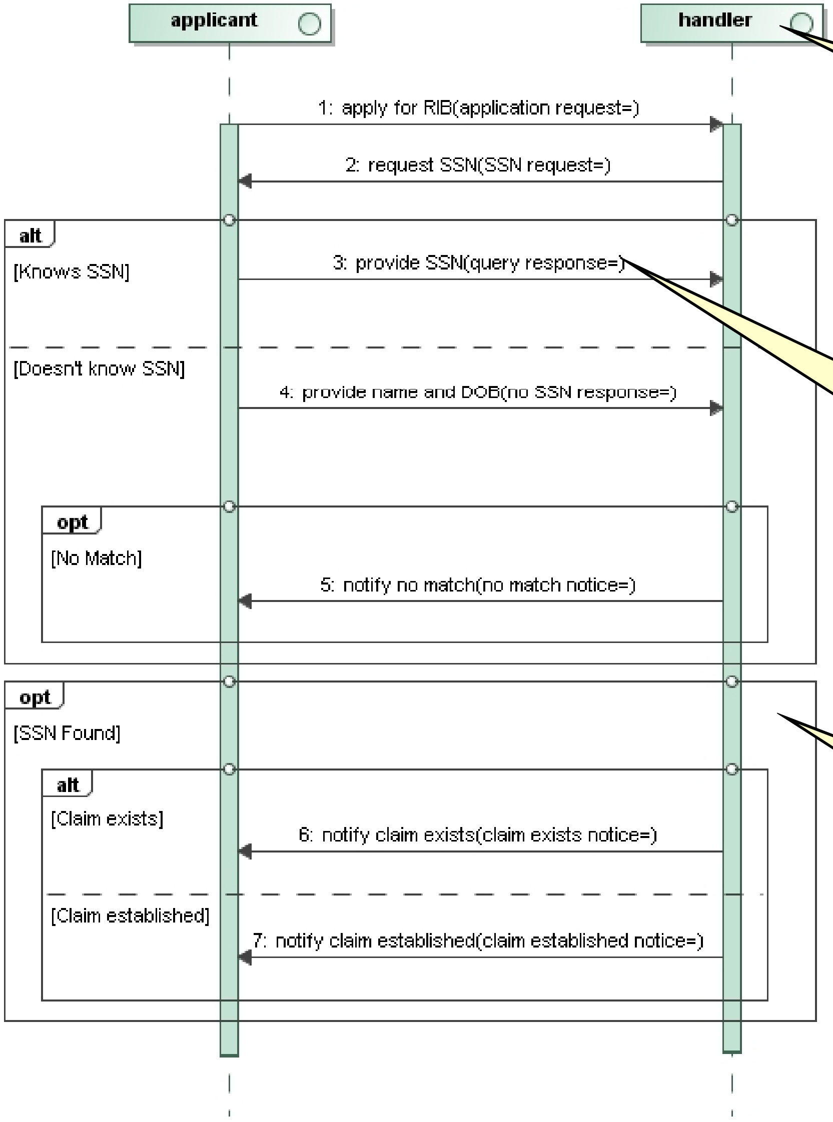


The service contract defines the *roles* to be played by consumers and providers of the service. Many service contracts have only two roles, one a consumer and one a provider. But any number are allowed.

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

Apply for RIB 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*.

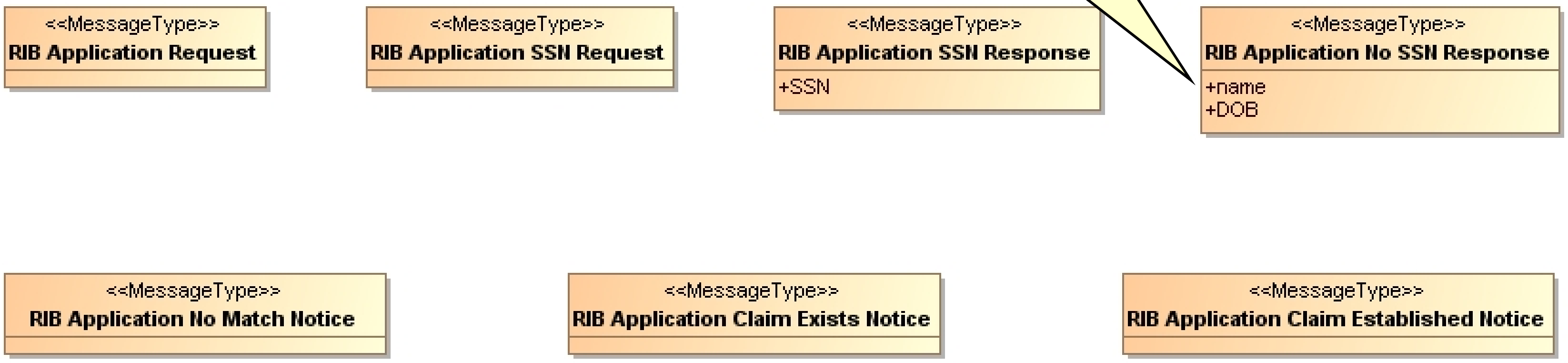
Messages are modeled as being passed via calls to *operations* on the *interfaces* to the roles.

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

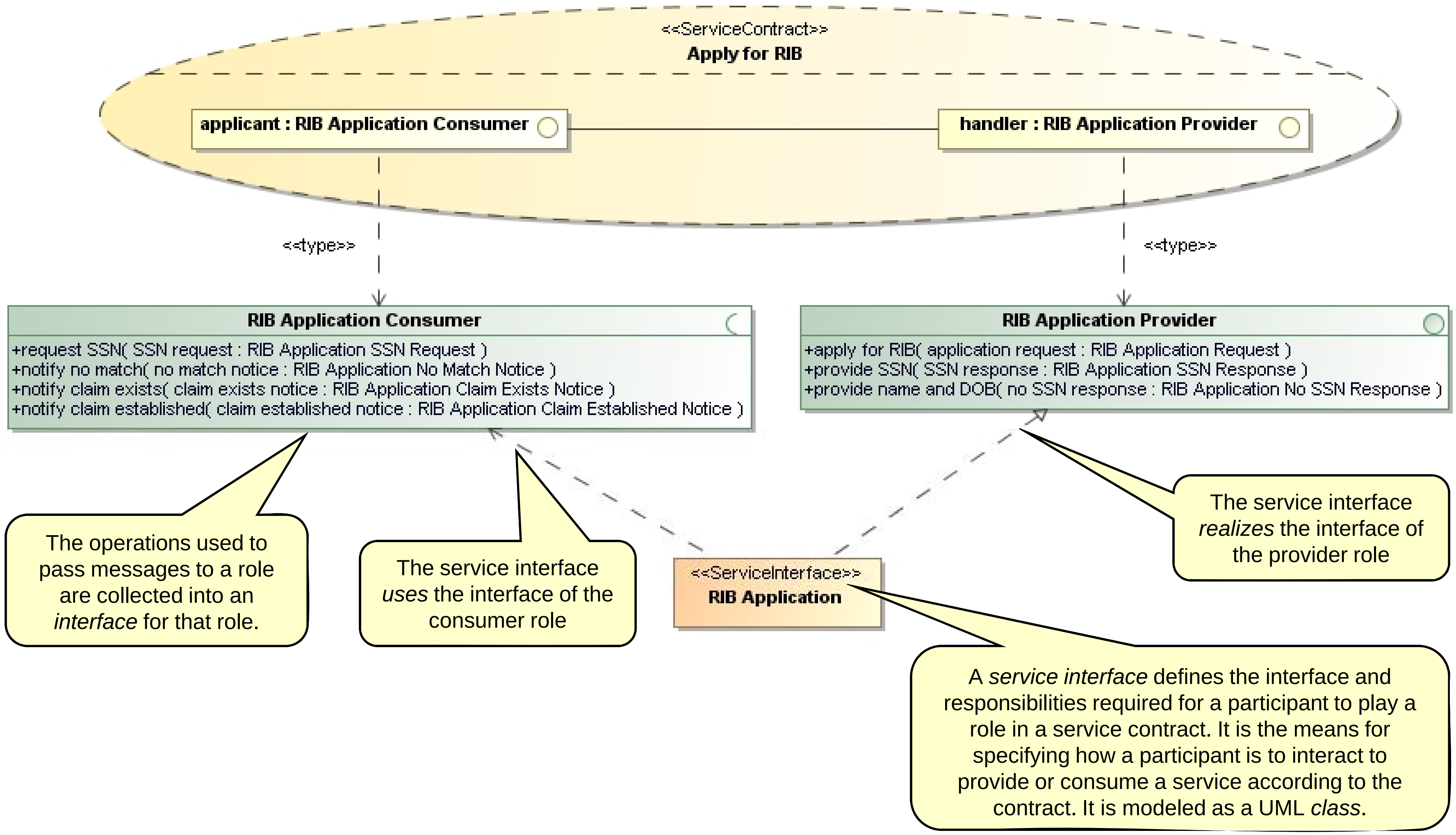
<<MessageType>>
RIB Application No Match Notice

<<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

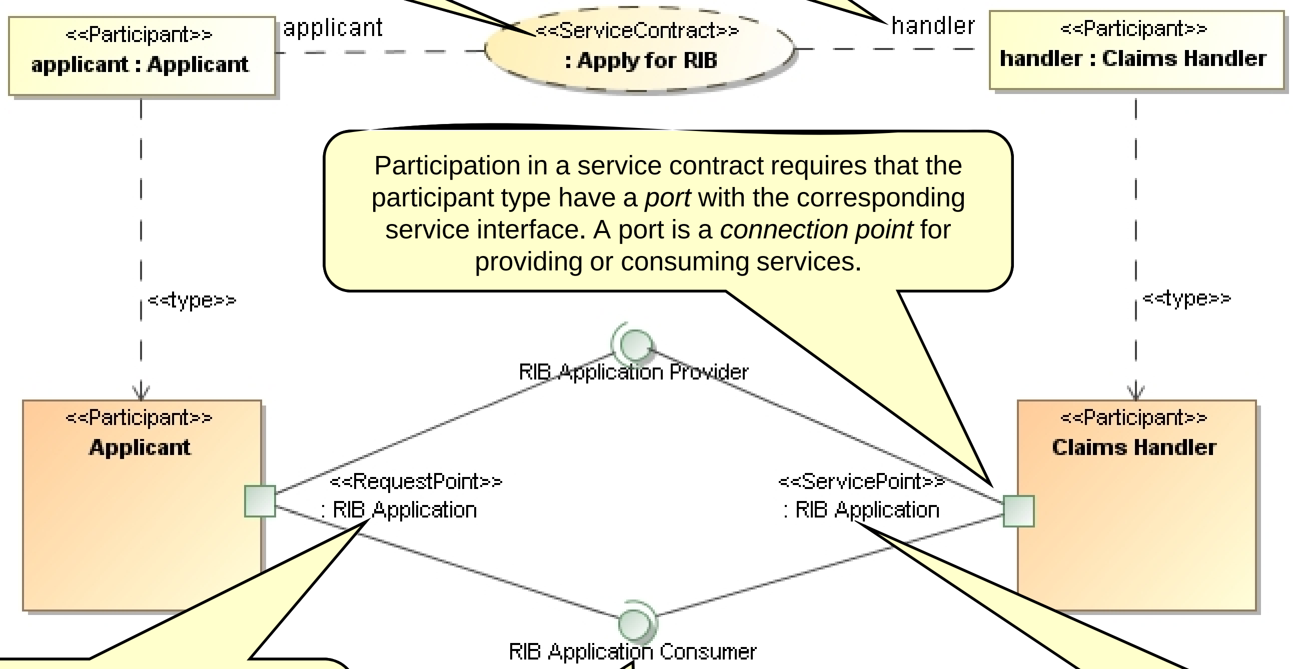
The operations used to pass messages to a role are collected into an *interface* for that role.

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.

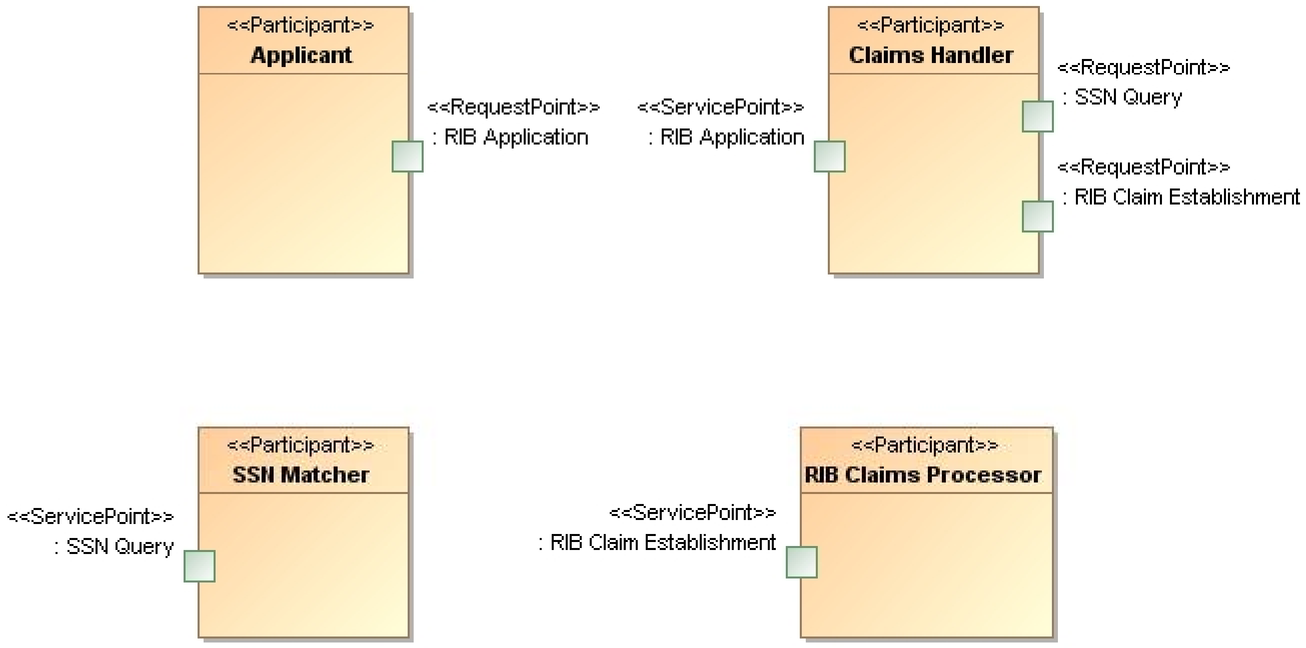
A *request point* is a port for requesting (consuming) a service. Note that the sense of provided and required interfaces is *reversed* at a request point: The port *requires* the provider interface and *provides* the consumer interface.

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

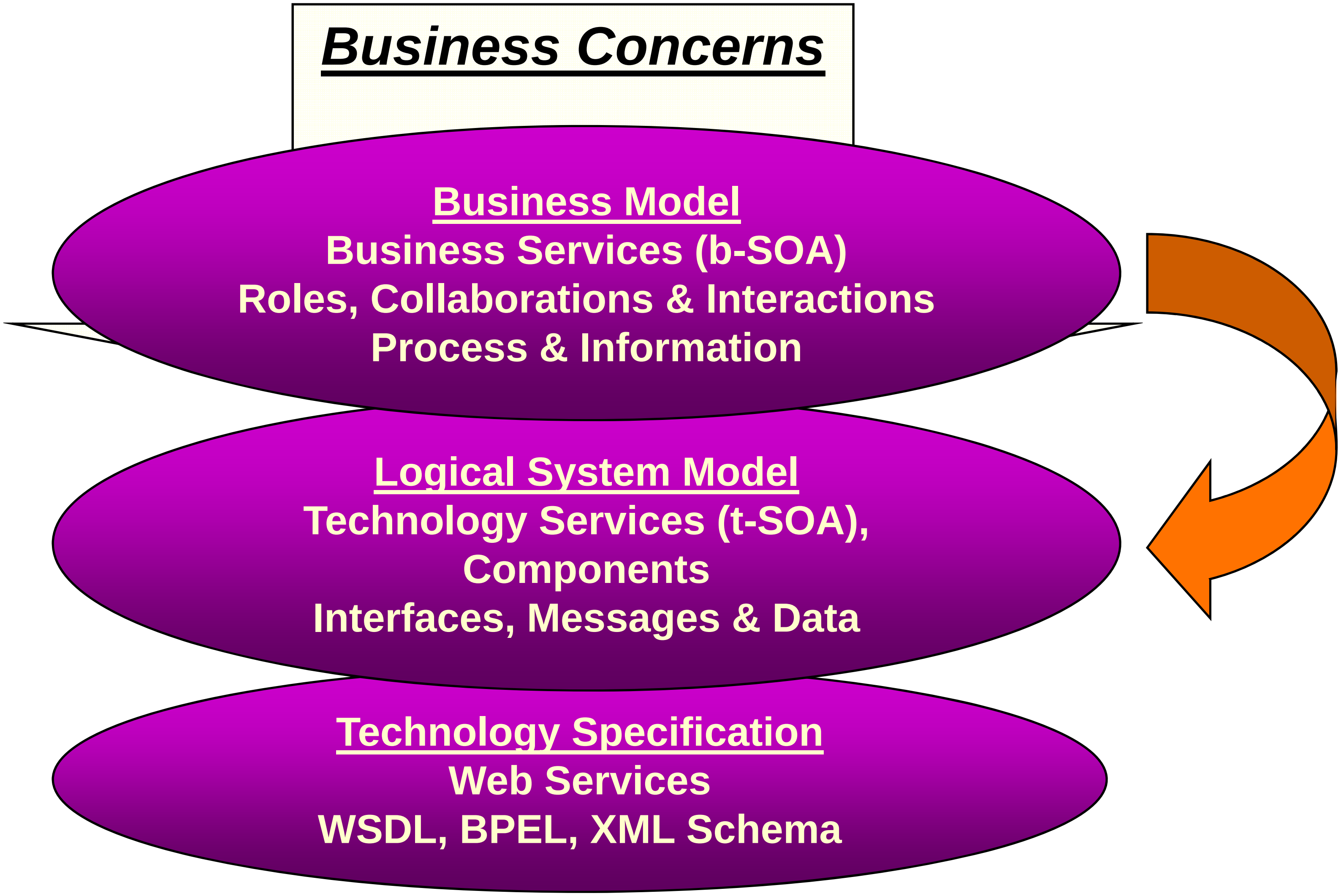
A *service point* is a port for providing a service. The port *provides* the provider interfaces and *requires* the consumer interface.

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

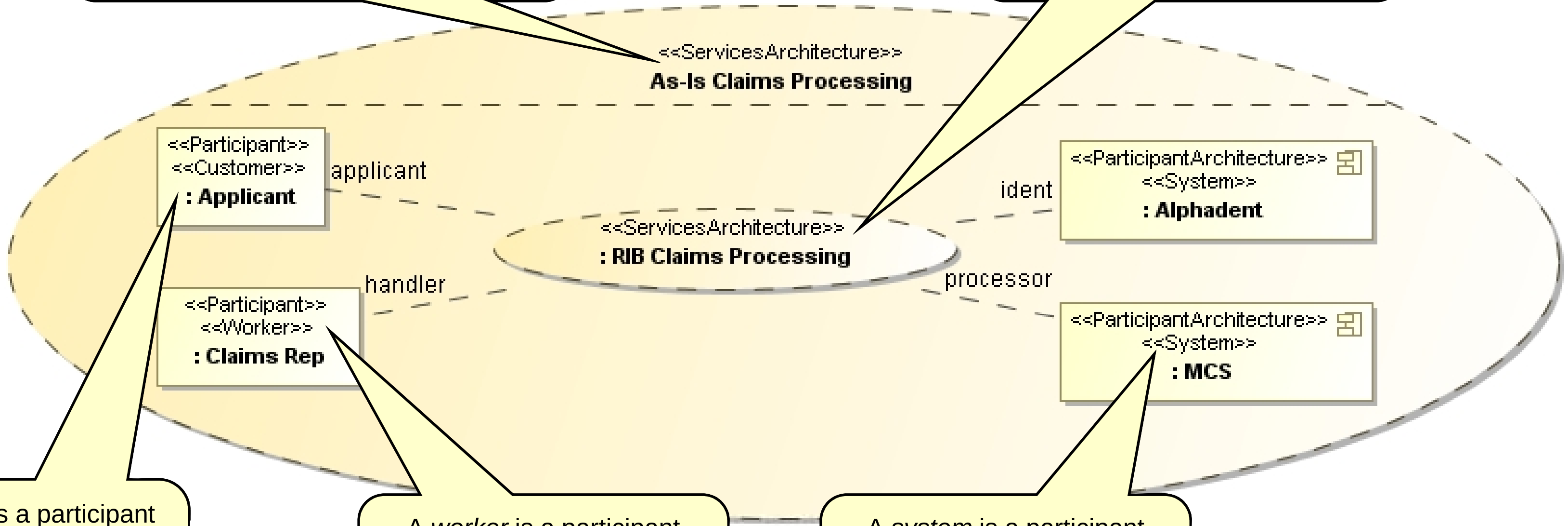
Platform Independent Model (PIM)

- 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

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.



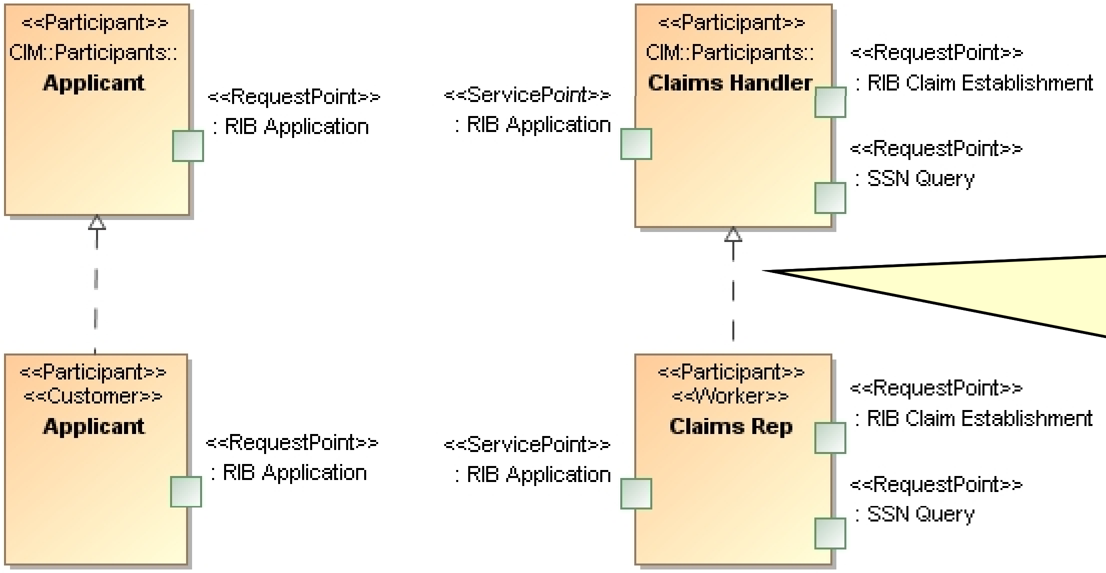
A *customer* is a participant who is external to, and being served by, the enterprise carrying out the business process.

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.

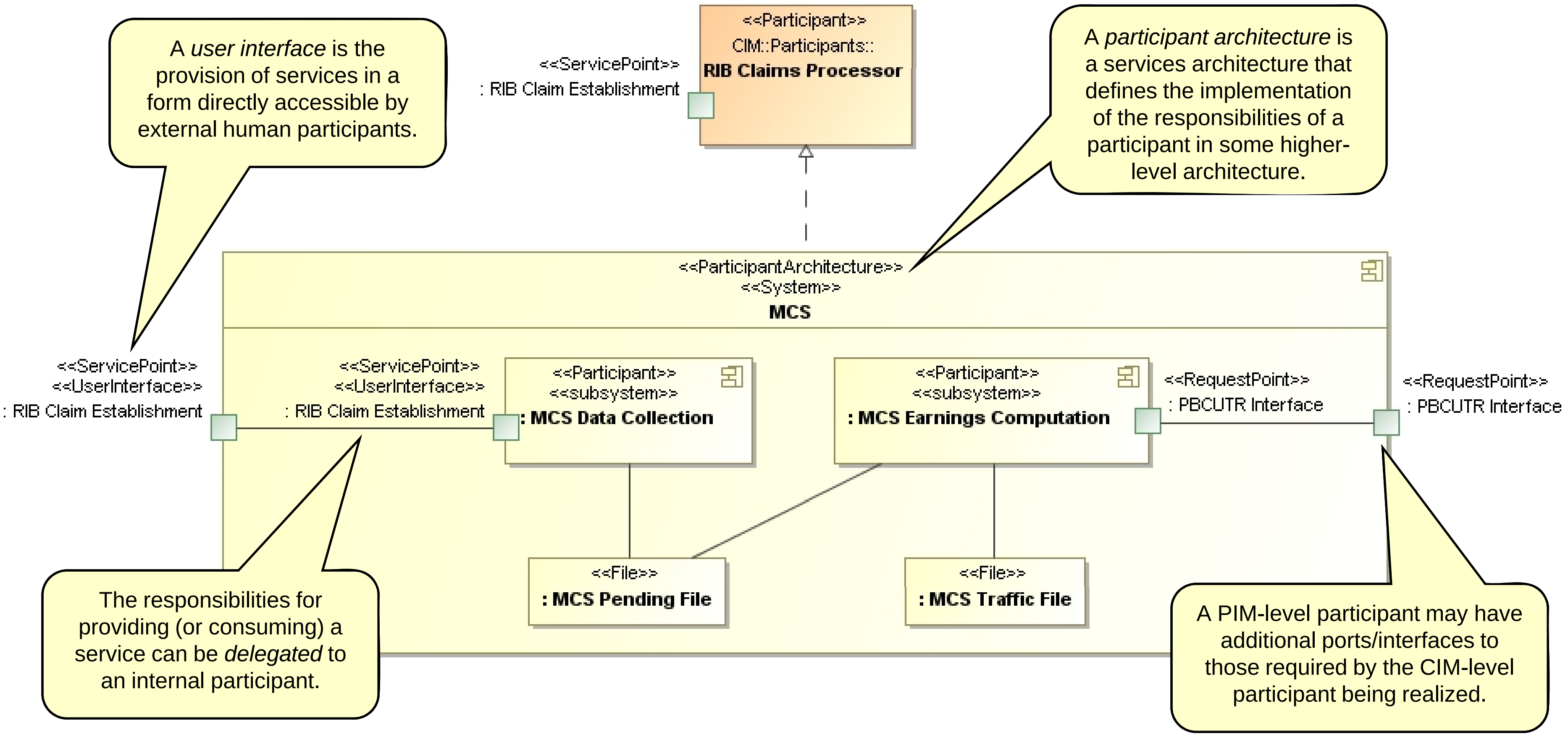
As-Is Claims Processing Human Participants

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



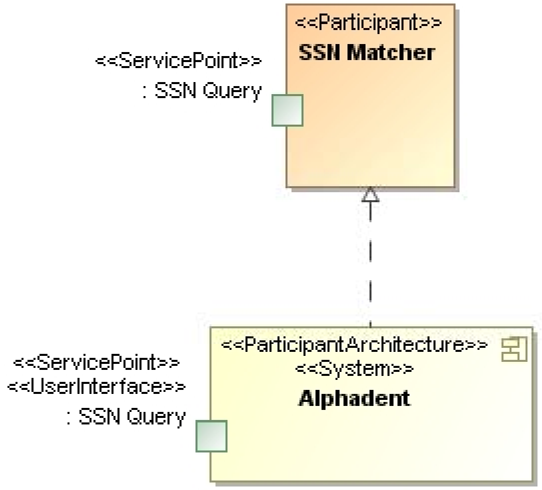
A *user interface* is the provision of services in a form directly accessible by external human participants.

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

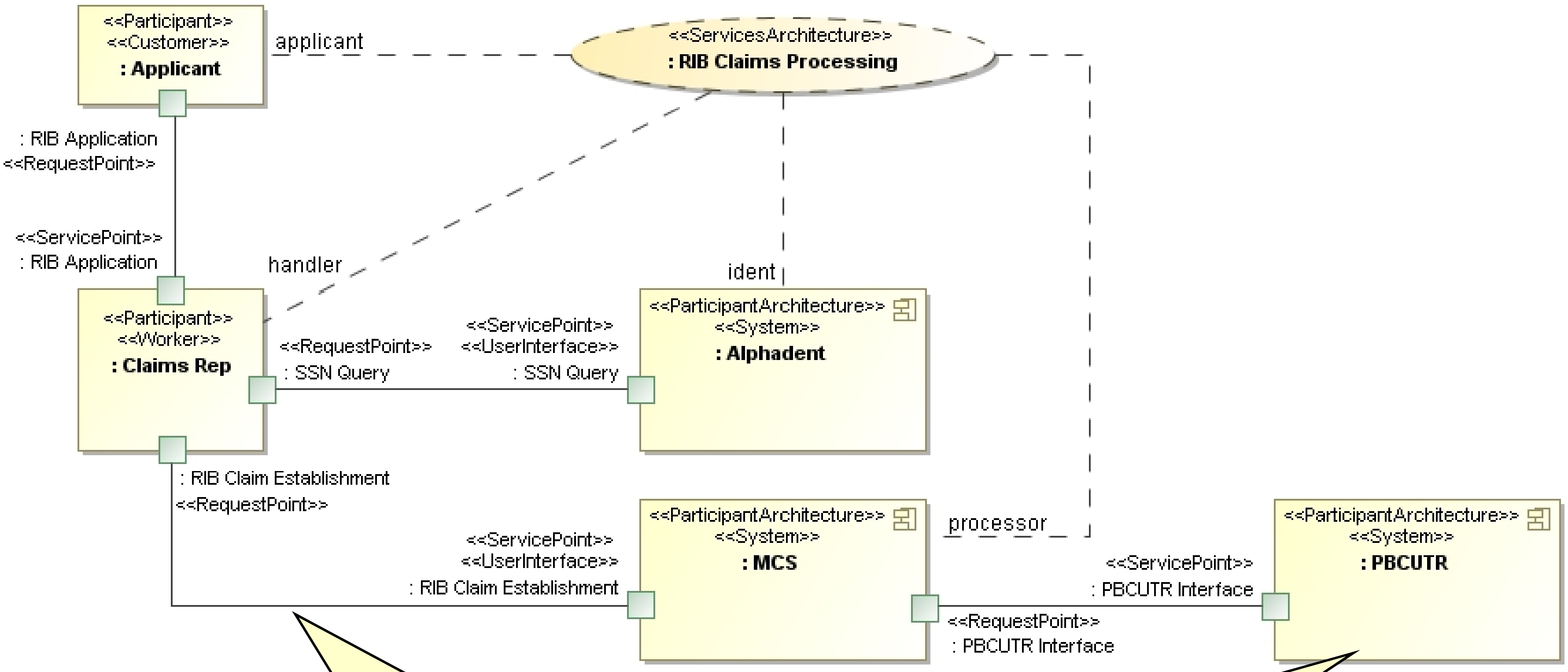
The responsibilities for providing (or consuming) a service can be *delegated* to an internal participant.

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

Alphadent System Architecture



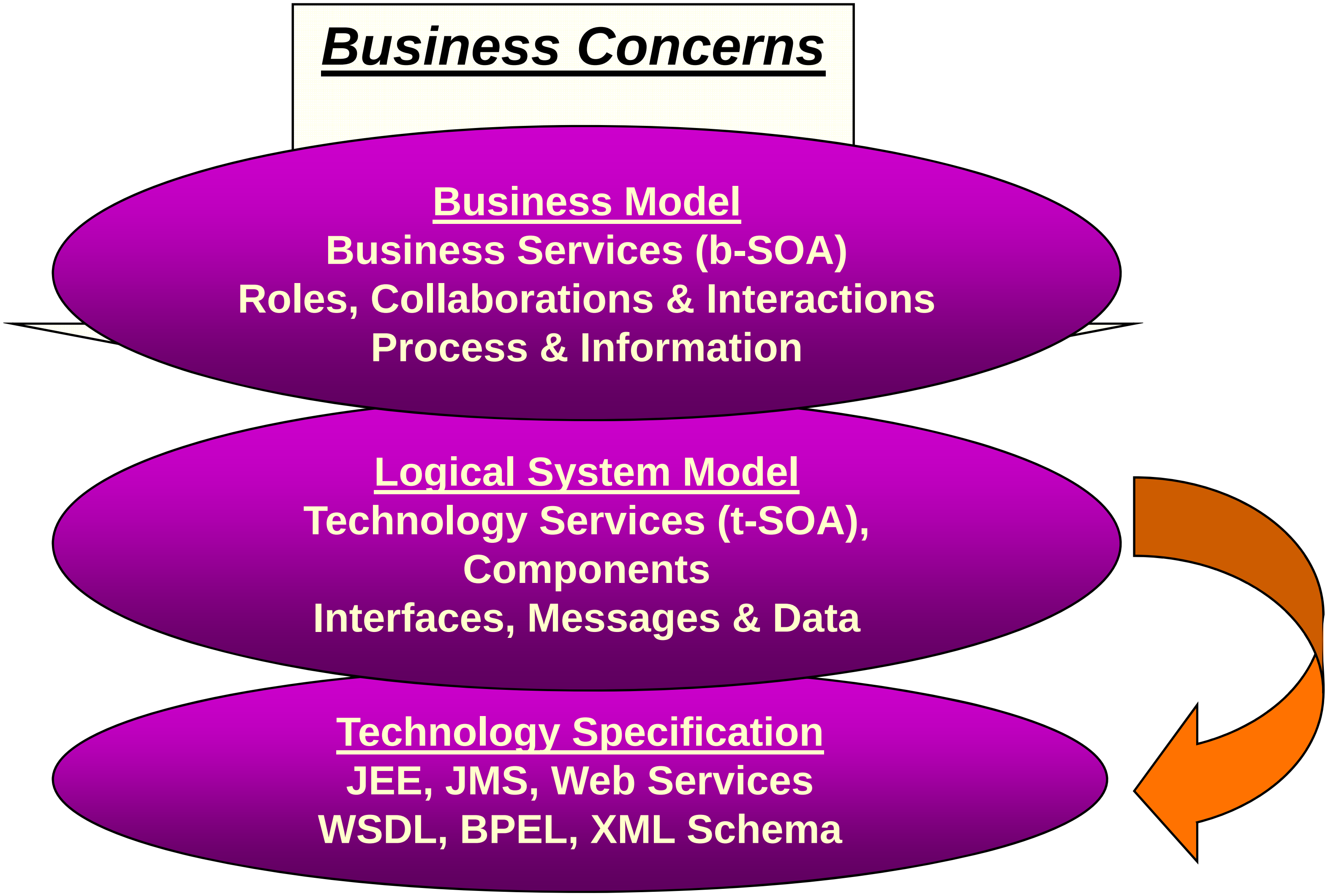
As-Is Claim Processing Composite Structure



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.

Technology Architecture



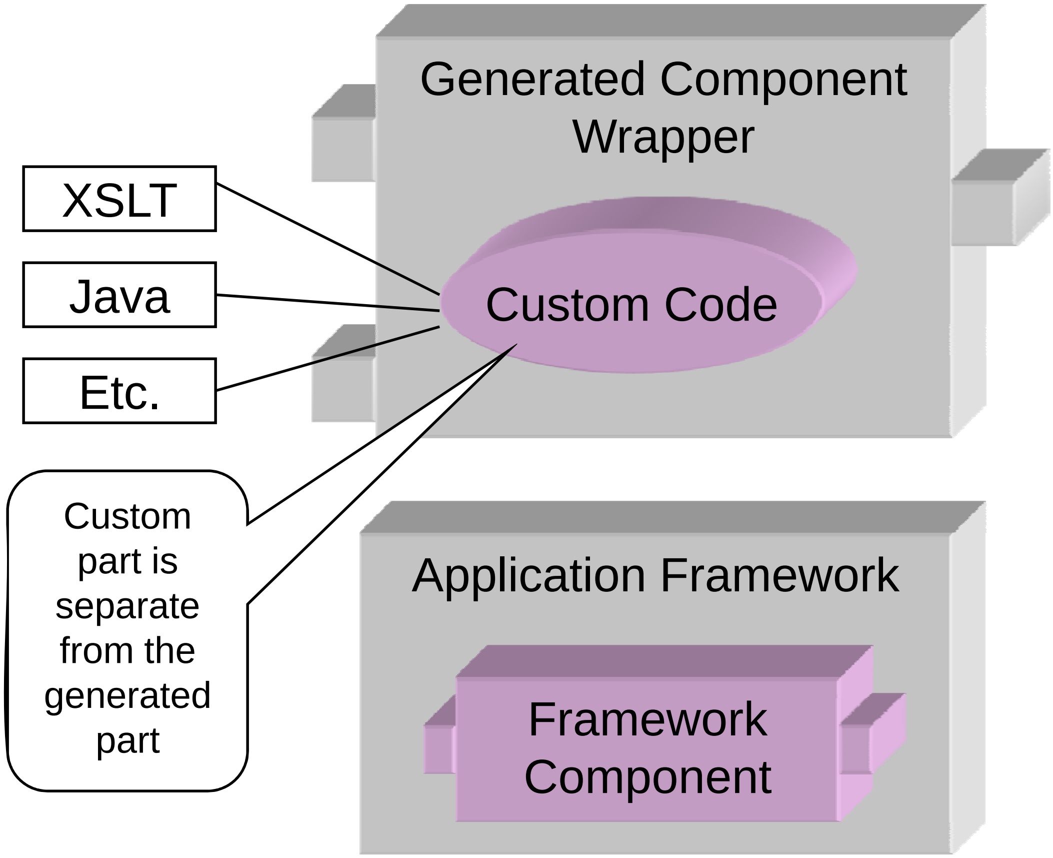
Business Concerns

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

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

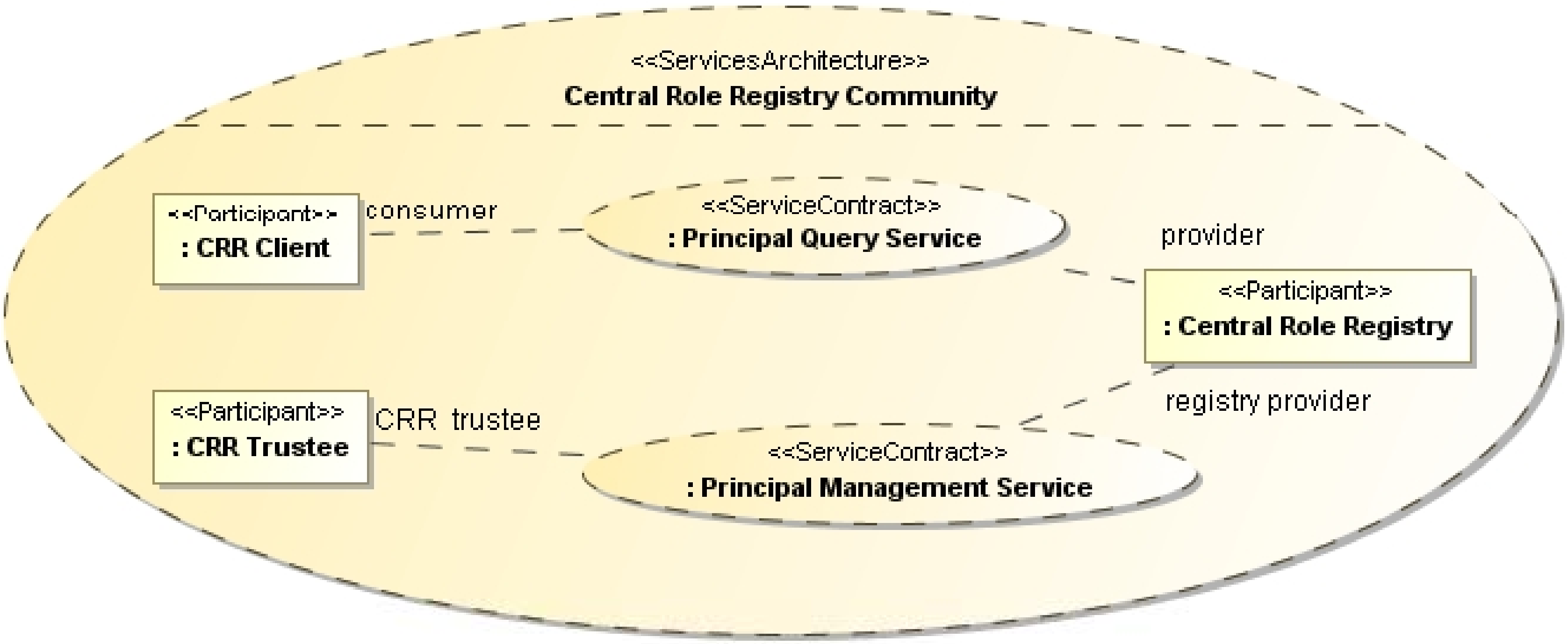
As MDA progresses, there will be less and less need for custom components, but the capability will remain.

Application Provisioning

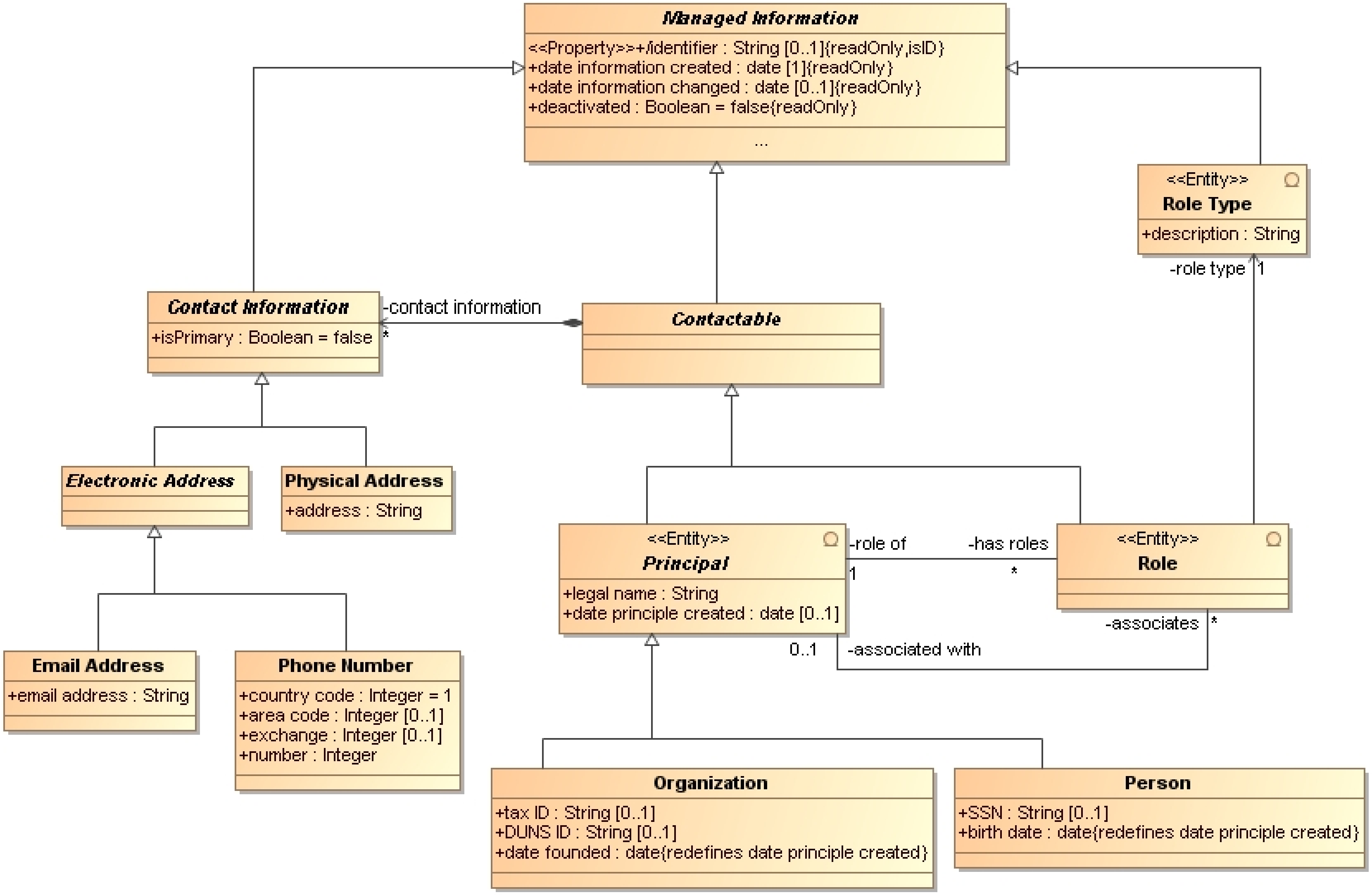
- Platform technologies are provisioned from the model based on the technology specified
  - XSD
  - WSDL
  - Application Server Configuration
  - Java Interfaces & Implementation
  - XSLT
  - IDE Project
  - SQL
  - Documentation
  - Tests
  - …

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

Executable Example Services Architecture



Example 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

Email Address
+email address : String

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

<<Entity>>
Principal
+legal name : String
+date principle created : date [0..1]

-role of 1

-has roles *

<<Entity>>
Role

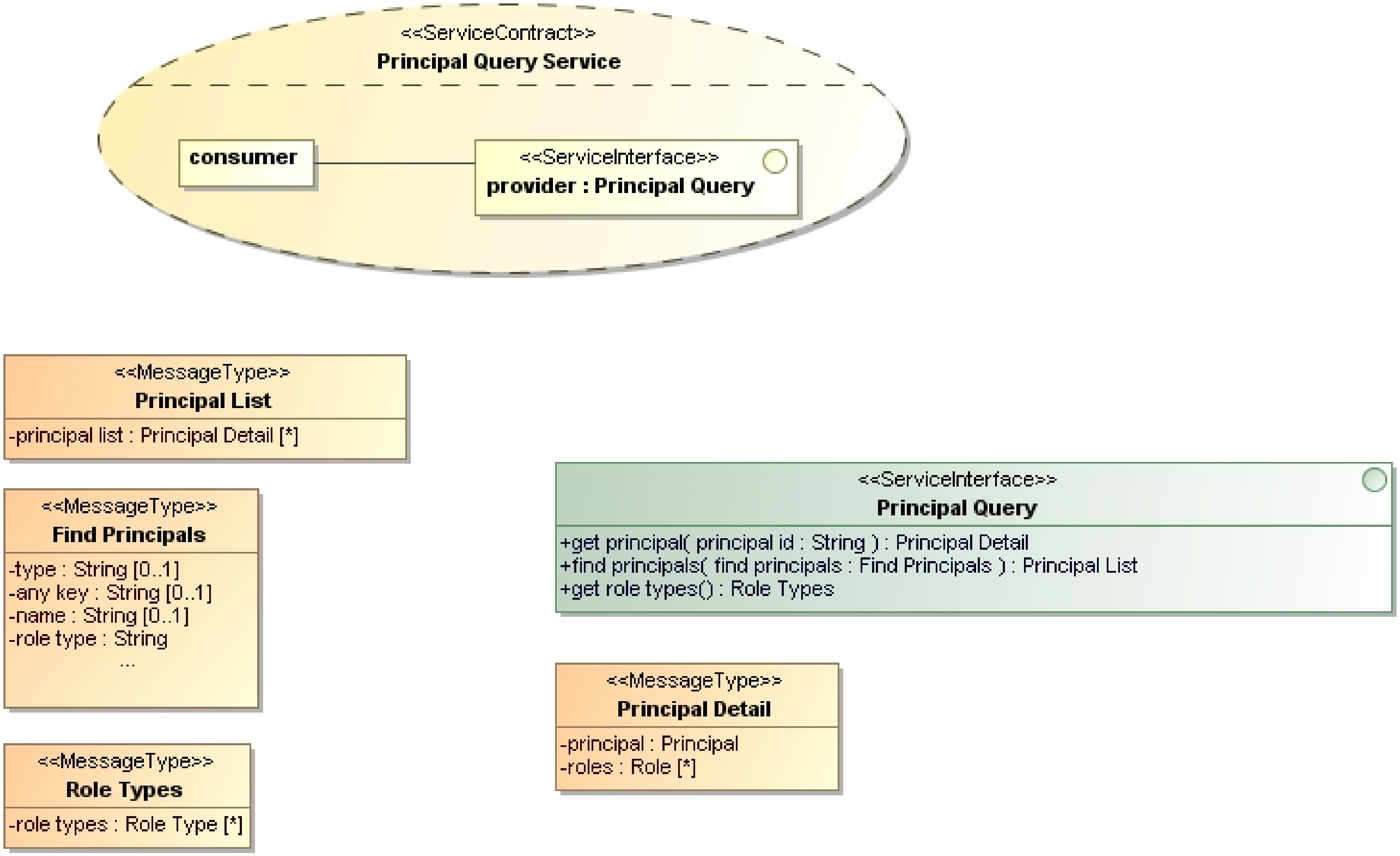
-associates *

0..1 -associated with

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

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

Example Service Contract & Messages



<<ServiceContract>>
Principal Query Service

consumer

<<ServiceInterface>>
provider : Principal Query

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

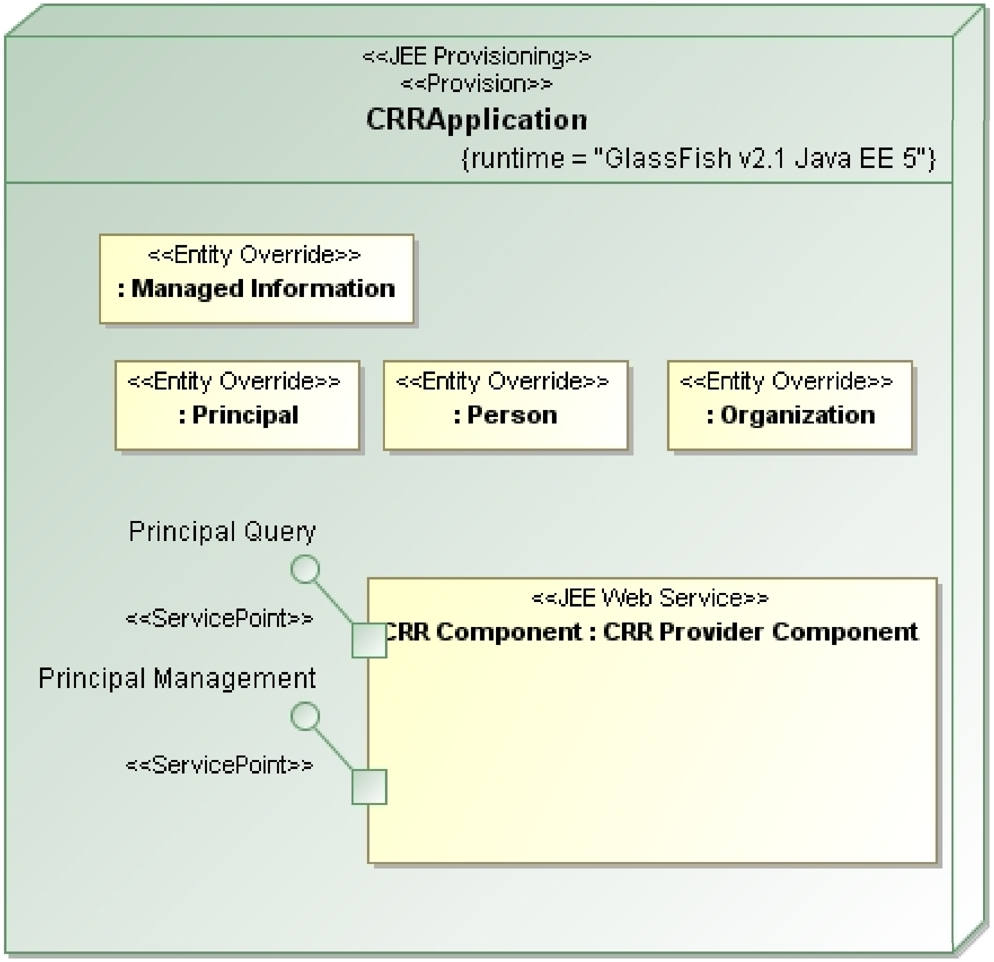
<<MessageType>>
Find Principals
-type : String [0..1]
-any key : String [0..1]
-name : String [0..1]
-role type : String
...

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

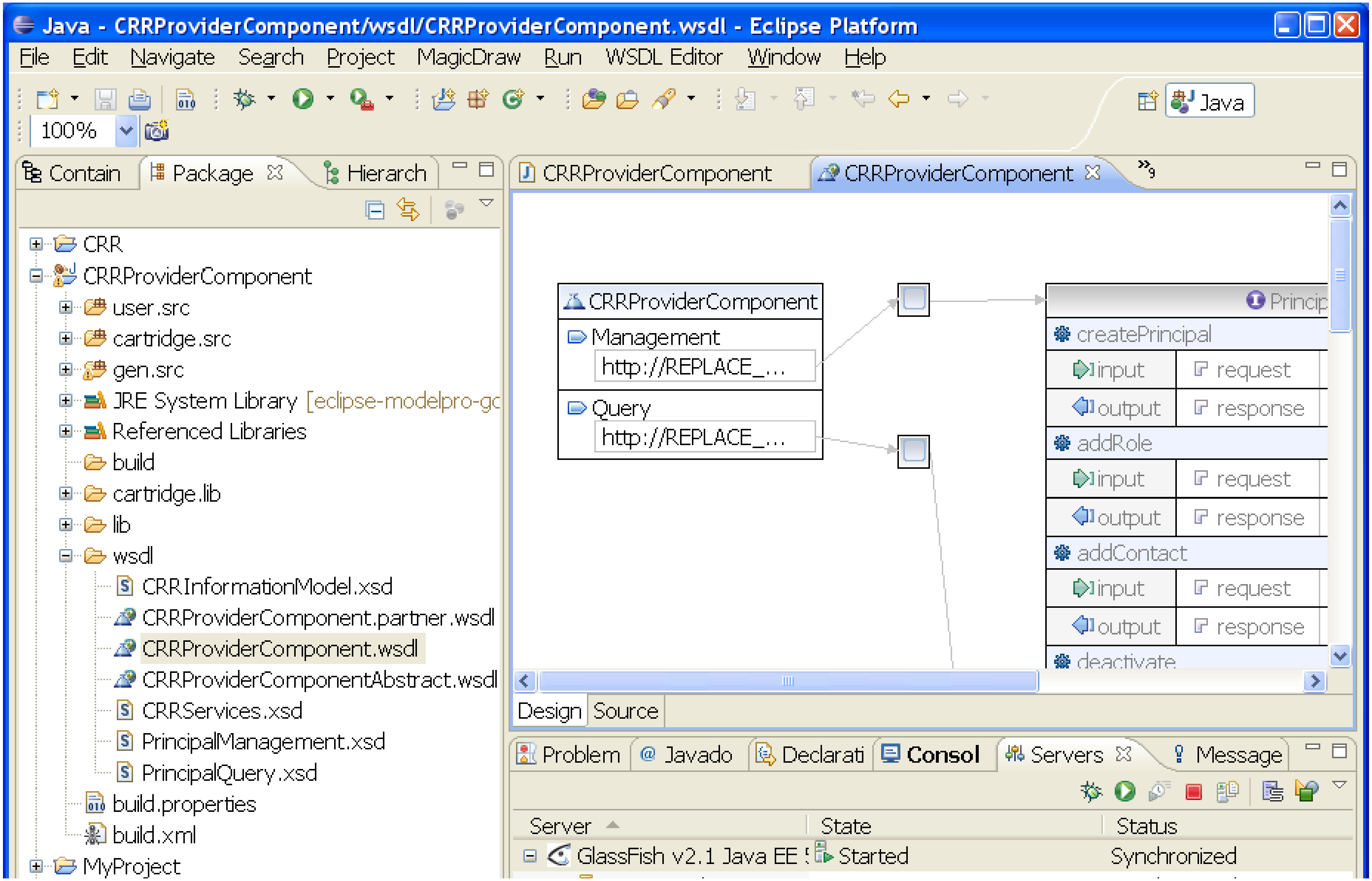
<<ServiceInterface>>
Principal Query
+get principal(principal id : String) : Principal Detail
+find principals(find principals : Find Principals) : Principal List
+get role types() : Role Types

<<MessageType>>
Principal Detail
-principal : Principal
-roles : Role [*]

Example Provisioning to JEE Web Services

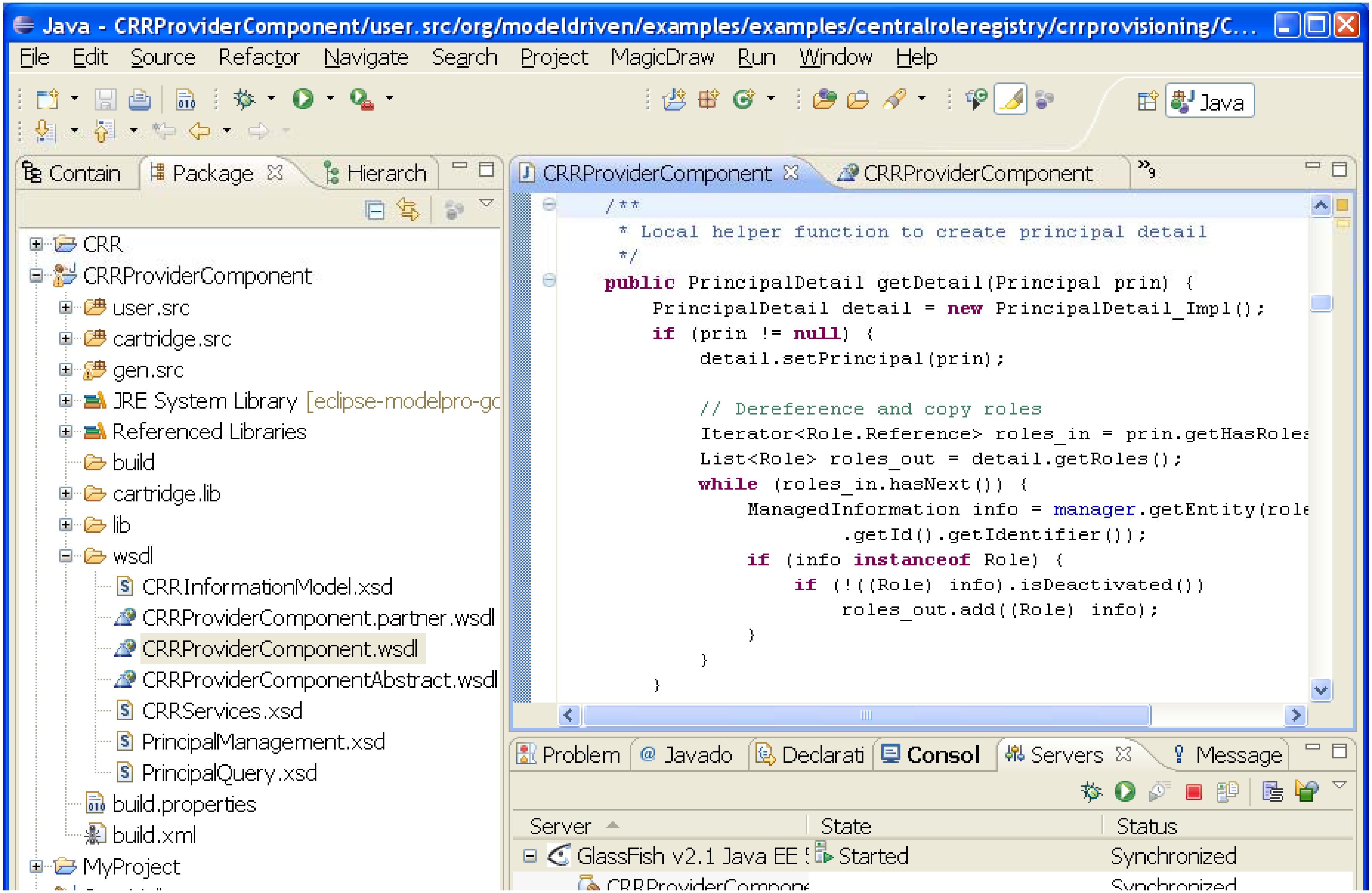


Generated Artifacts in Java IDE



Server	State	Status
GlassFish v2.1 Java EE 5	Started	Synchronized

Java Override Code

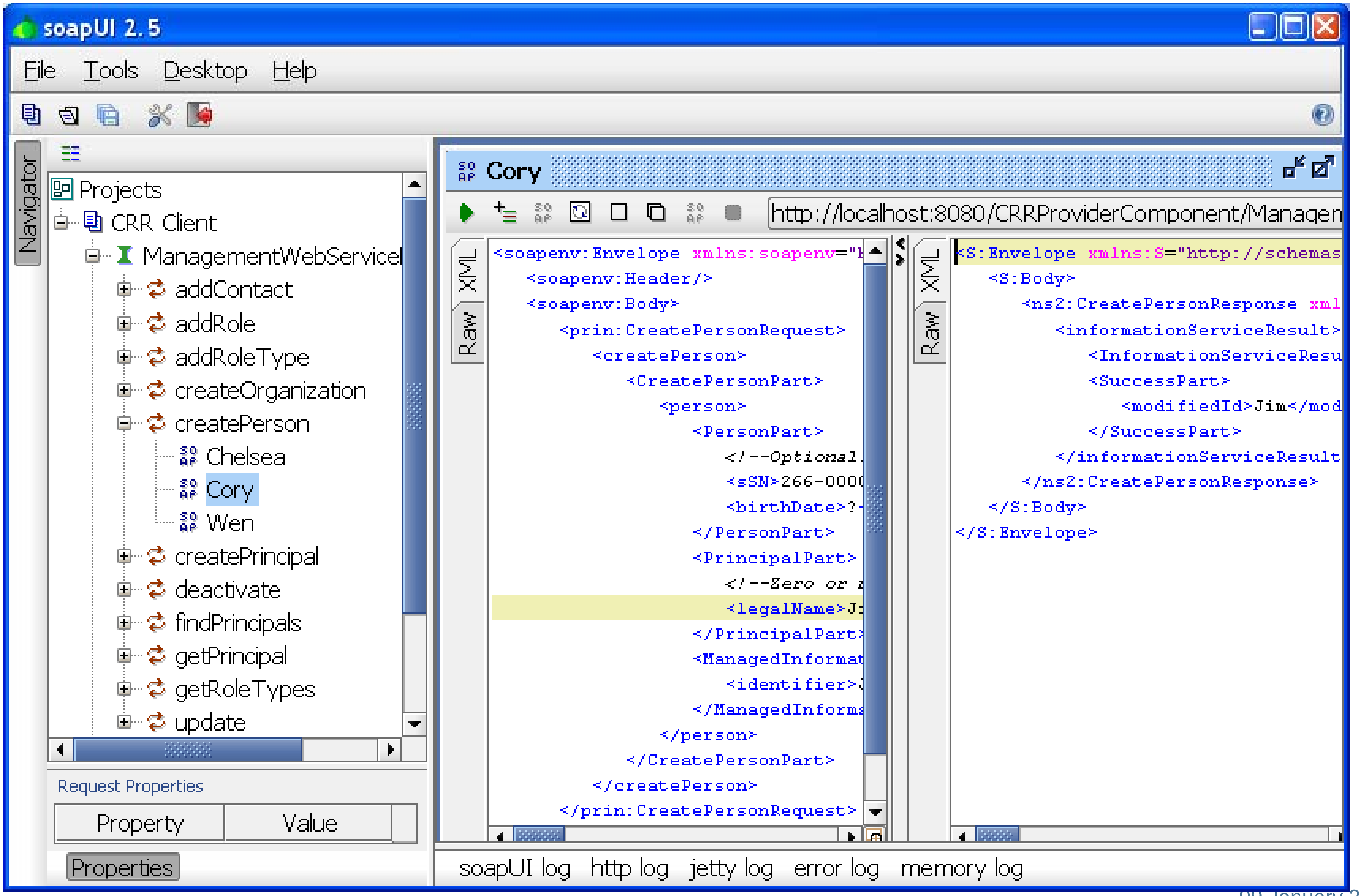


```
    /**
     * Local helper function to create principal detail
     */
    public PrincipalDetail getDetail(Principal prin) {
        PrincipalDetail detail = new PrincipalDetail_Impl();
        if (prin != null) {
            detail.setPrincipal(prin);

            // Dereference and copy roles
            Iterator<Role.Reference> roles_in = prin.getHasRoles
            List<Role> roles_out = detail.getRoles();
            while (roles_in.hasNext()) {
                ManagedInformation info = manager.getEntity(role
                        .getId().getIdentifier());
                if (info instanceof Role) {
                    if (!((Role) info).isDeactivated())
                        roles_out.add((Role) info);
                }
            }
        }
```

Server	State	Status
GlassFish v2.1 Java EE 5	Started	Synchronized
CRRProviderCompone		Synchronized

Using the deployed service from an ugly client



```
<soapenv:Envelope xmlns:soapenv="h
   <soapenv:Header/>
   <soapenv:Body>
      <prin:CreatePersonRequest>
         <createPerson>
            <CreatePersonPart>
               <person>
                  <PersonPart>
                     <!--Optional
                     <sSN>266-000
                     <birthDate>?
                  </PersonPart>
                  <PrincipalPart>
                     <!--Zero or
                     <legalName>J
                  </PrincipalPart>
                  <ManagedInformat
                     <identifier>
                  </ManagedInform
               </person>
            </CreatePersonPart>
         </createPerson>
      </prin:CreatePersonRequest>
```

```
<S:Envelope xmlns:S="http://schemas
   <S:Body>
      <ns2:CreatePersonResponse xml
         <informationServiceResult>
            <InformationServiceResu
            <SuccessPart>
               <modifiedId>Jim</mod
            </SuccessPart>
         </informationServiceResult
      </ns2:CreatePersonResponse>
   </S:Body>
</S:Envelope>
```

More Information

- Cory Casanave – cory-c (at) modeldriven.com
- SoaML Web Page – www.soaml.org
- ModelPro (Open Source) – www.ModelDriven.org
- Model Driven Solutions – www.ModelDriven.com
- Cameo SOA+ from NoMagic – soaplus.cameosuite.com