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

FMEA: An executable services oriented enterprise architecture for financial management

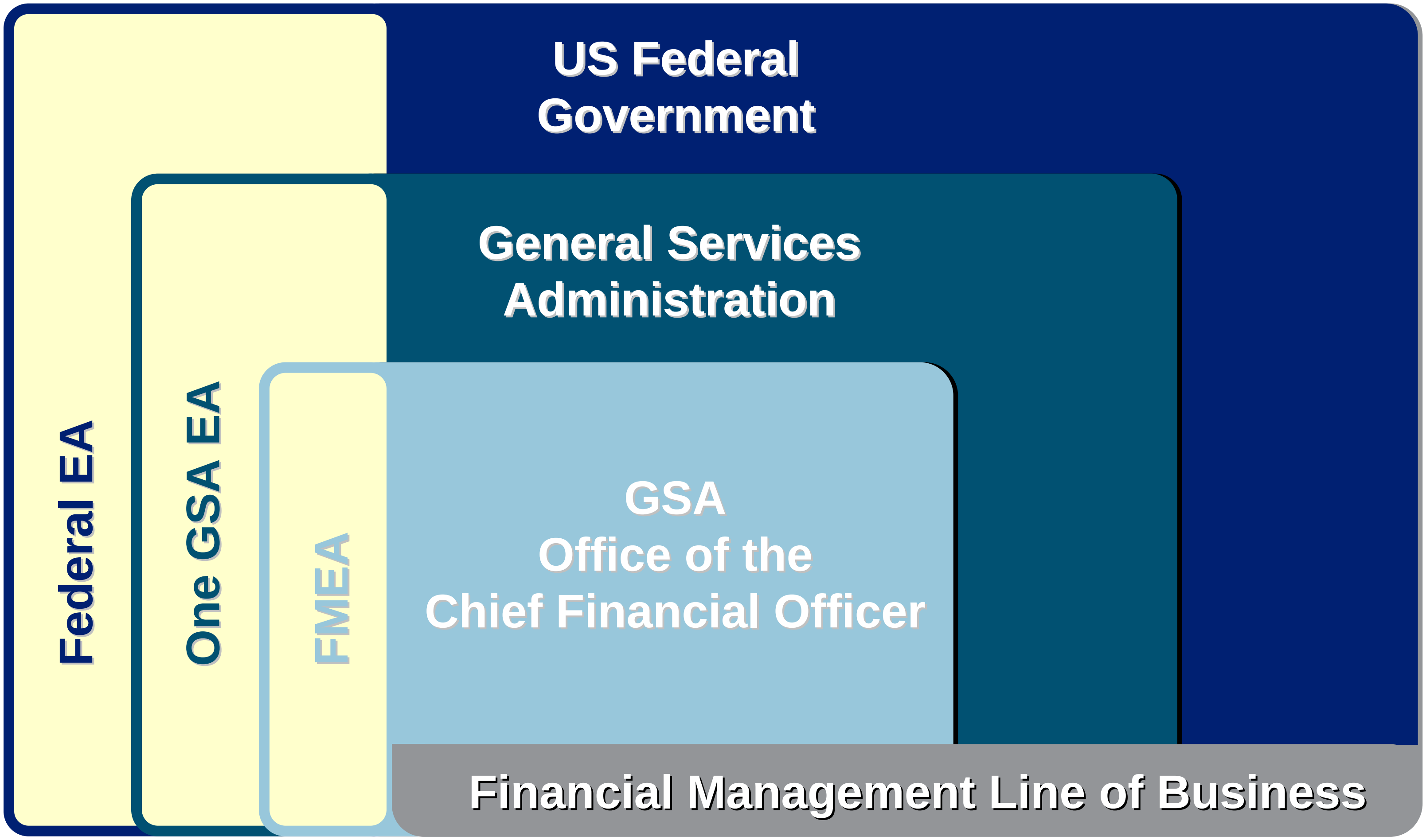
OMG's Maximizing BPM Investments with SOA Workshop

Cory Casanave

cory-c (at) modeldriven.com

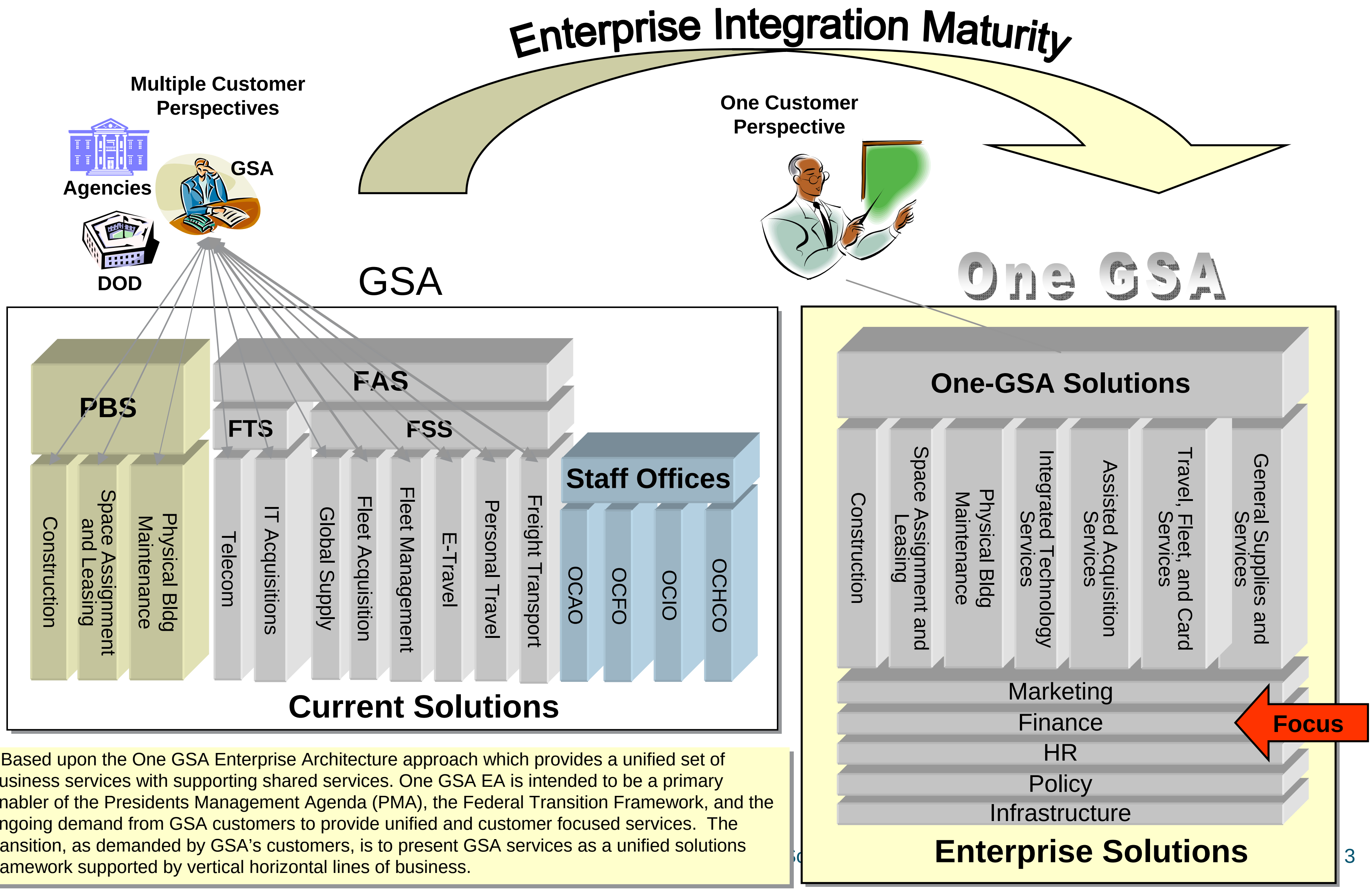
January 2008

FMEA in Context


US Federal Government
General Services Administration
GSA Office of the Chief Financial Officer
Federal EA
One GSA EA
FMEA
Financial Management Line of Business

Copyright © 2008 Model Driven Solutions

U.S. GSA Transition To A Federated Environment*



Enterprise Integration Maturity

Multiple Customer Perspectives

Agencies

GSA

DOD

One Customer Perspective

GSA

PBS

Construction

Space Assignment and Leasing

Physical Bldg Maintenance

FAS

FTS

Telecom

IT Acquisitions

FSS

Global Supply

Fleet Acquisition

Fleet Management

E-Travel

Personal Travel

Freight Transport

Staff Offices

OCAO

OCFO

OCIO

OCHCO

Current Solutions

One GSA

One-GSA Solutions

Construction

Space Assignment and Leasing

Physical Bldg Maintenance

Integrated Technology Services

Assisted Acquisition Services

Travel, Fleet, and Card Services

General Supplies and Services

Marketing

Finance

HR

Policy

Infrastructure

Focus

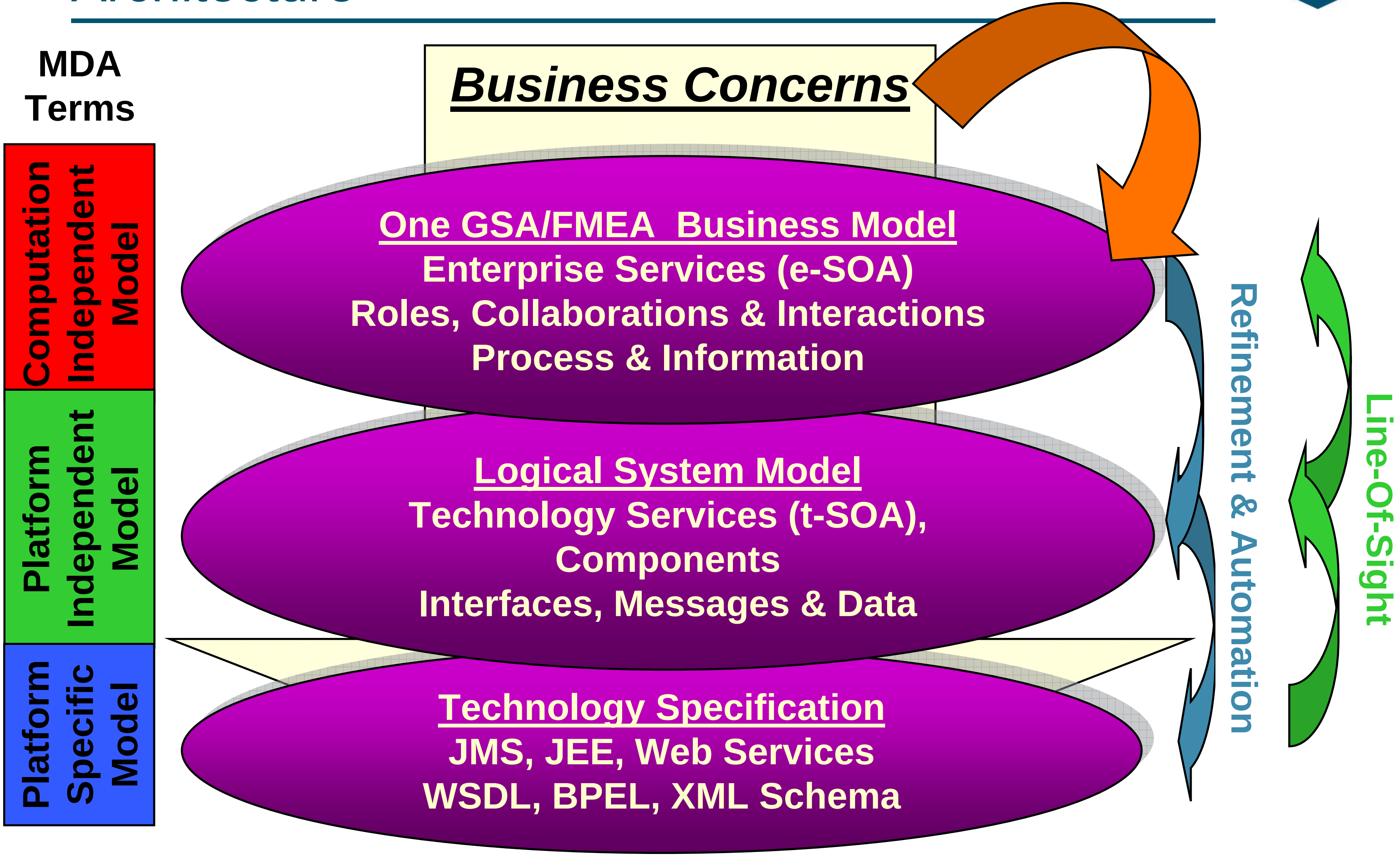
Enterprise Solutions

* Based upon the One GSA Enterprise Architecture approach which provides a unified set of business services with supporting shared services. One GSA EA is intended to be a primary enabler of the Presidents Management Agenda (PMA), the Federal Transition Framework, and the ongoing demand from GSA customers to provide unified and customer focused services. The transition, as demanded by GSA's customers, is to present GSA services as a unified solutions framework supported by vertical horizontal lines of business.

Approach

- Business focus, facilitated with technology
- Services Oriented Architecture (SOA) at both the business and technical level
- Described with Collaborative Role Interactions, Processes and Information models based on OMG standards
- Model Driven Architecture (MDA) to connect the business and technical architectures
- JEE, JMS & Web services as the technical interface to the line of business
- Tools Used
  - Magic draw UML
  - ModelDriven.org (open source project) for MDA provisioning
  - Eclipse
  - jBoss suite

Business Focus Using Model Driven Architecture



MDA Terms

Computation Independent Model

Platform Independent Model

Platform Specific Model

Business Concerns

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

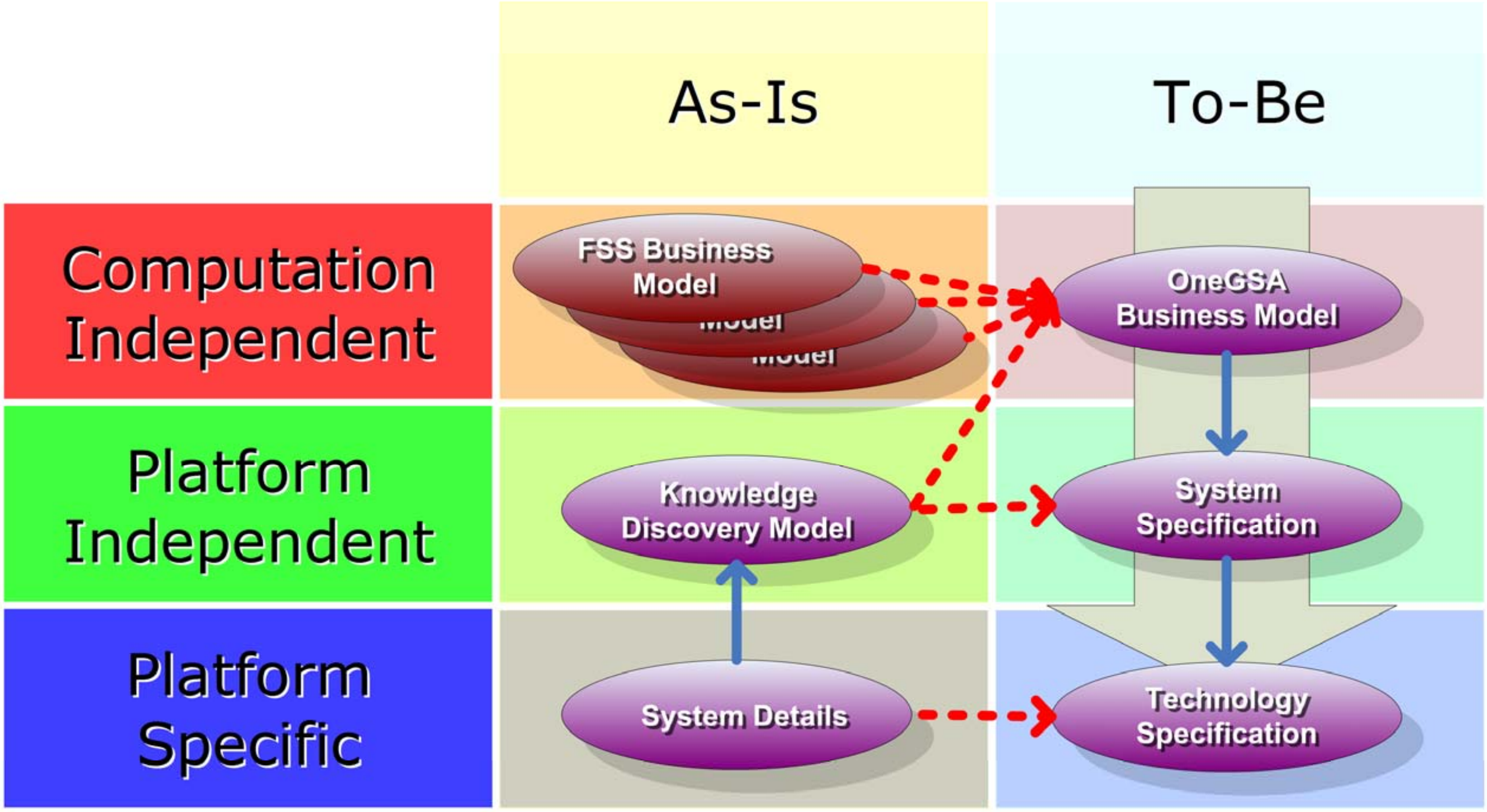
Logical System Model
Technology Services (t-SOA), Components
Interfaces, Messages & Data

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

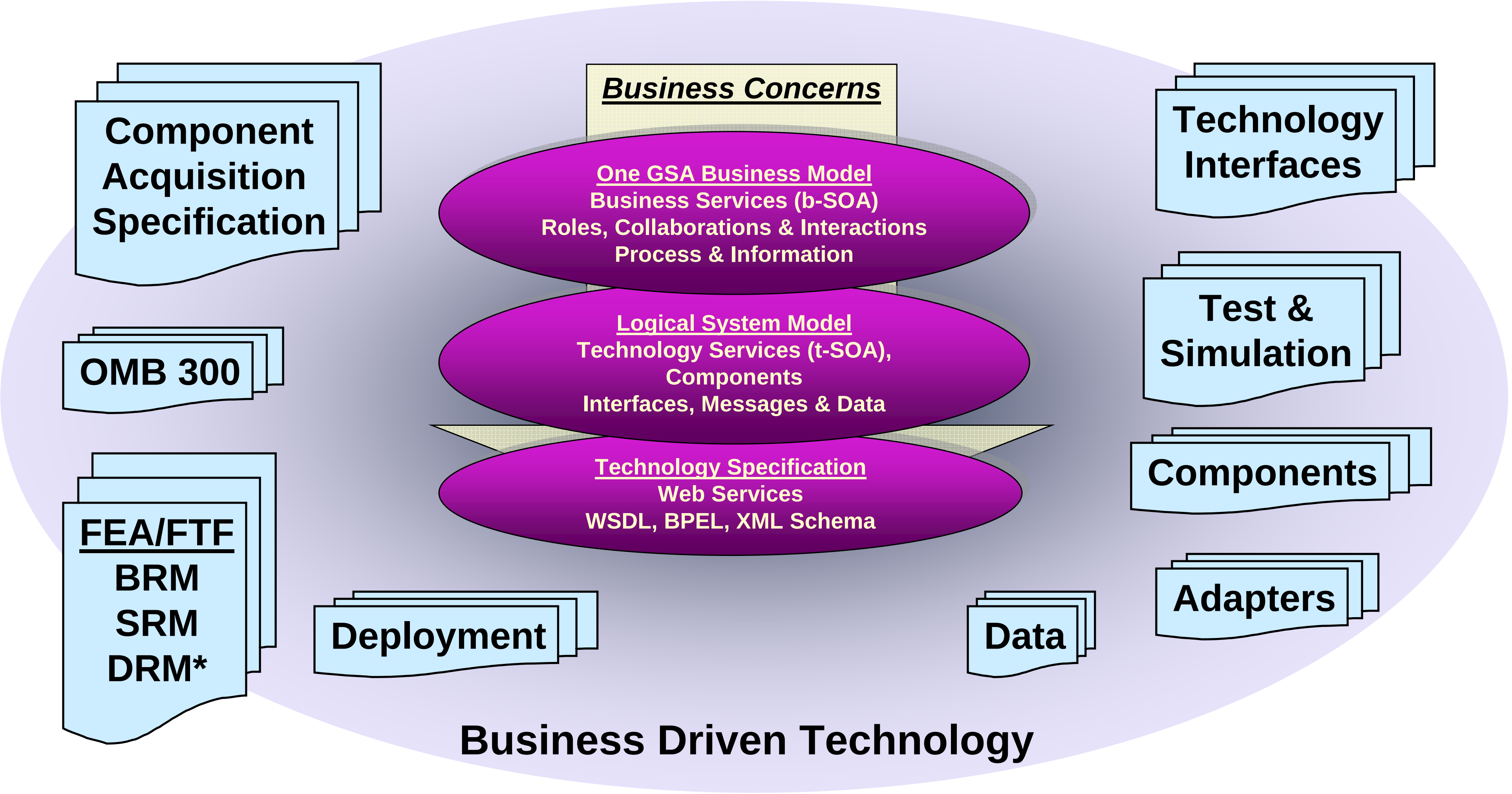
Refinement & Automation

Line-Of-Sight

Incorporating Legacy Analysis



Value derived from the architecture



Component Acquisition Specification

OMB 300

FEA/FTF
BRM
SRM
DRM*

Deployment

Business Concerns

One GSA 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

Data

Technology Interfaces

Test & Simulation

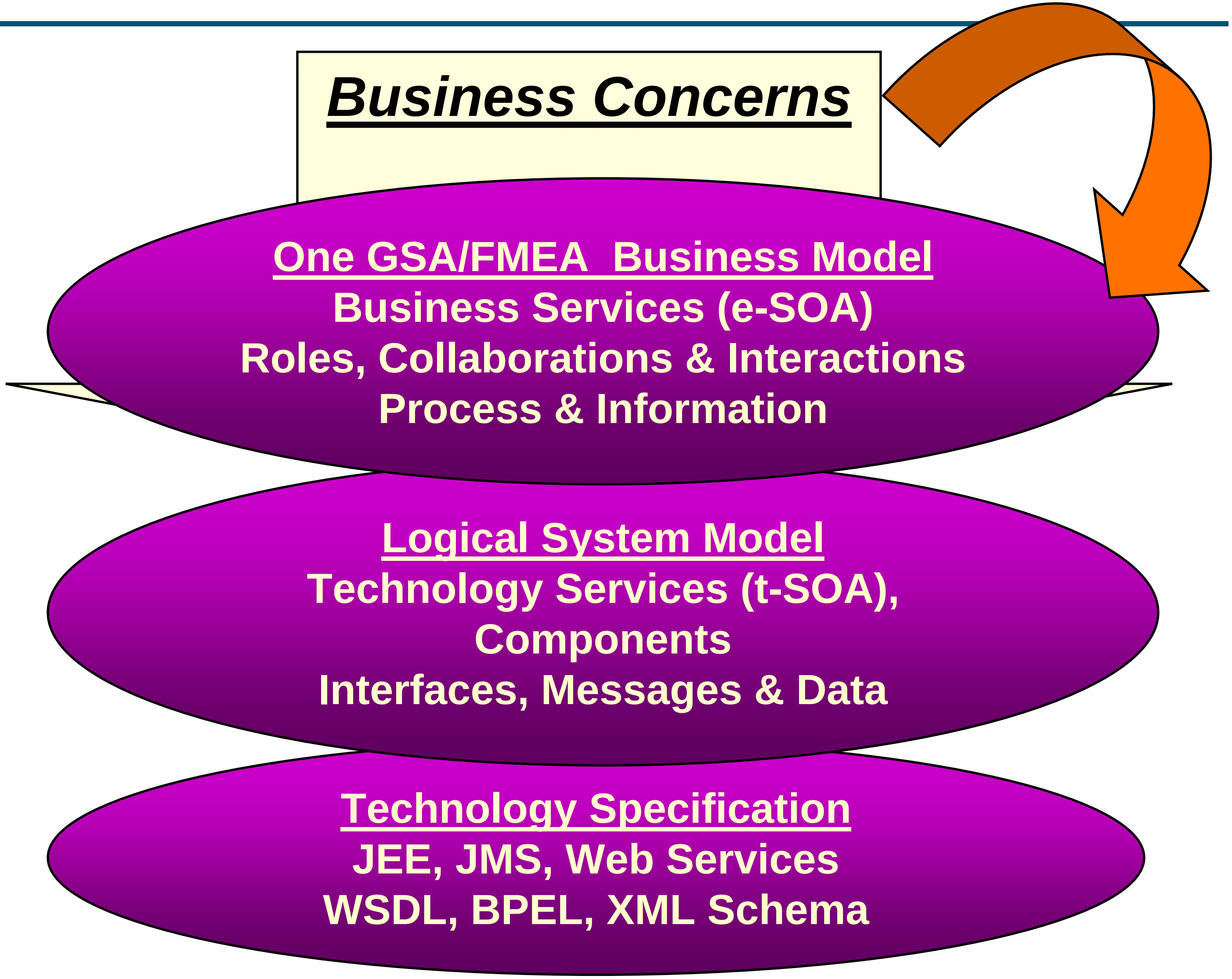
Components

Adapters

Business Driven Technology

Facilitating Business Processes

Focus on the Business Model



Business Concerns

One GSA/FMEA 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

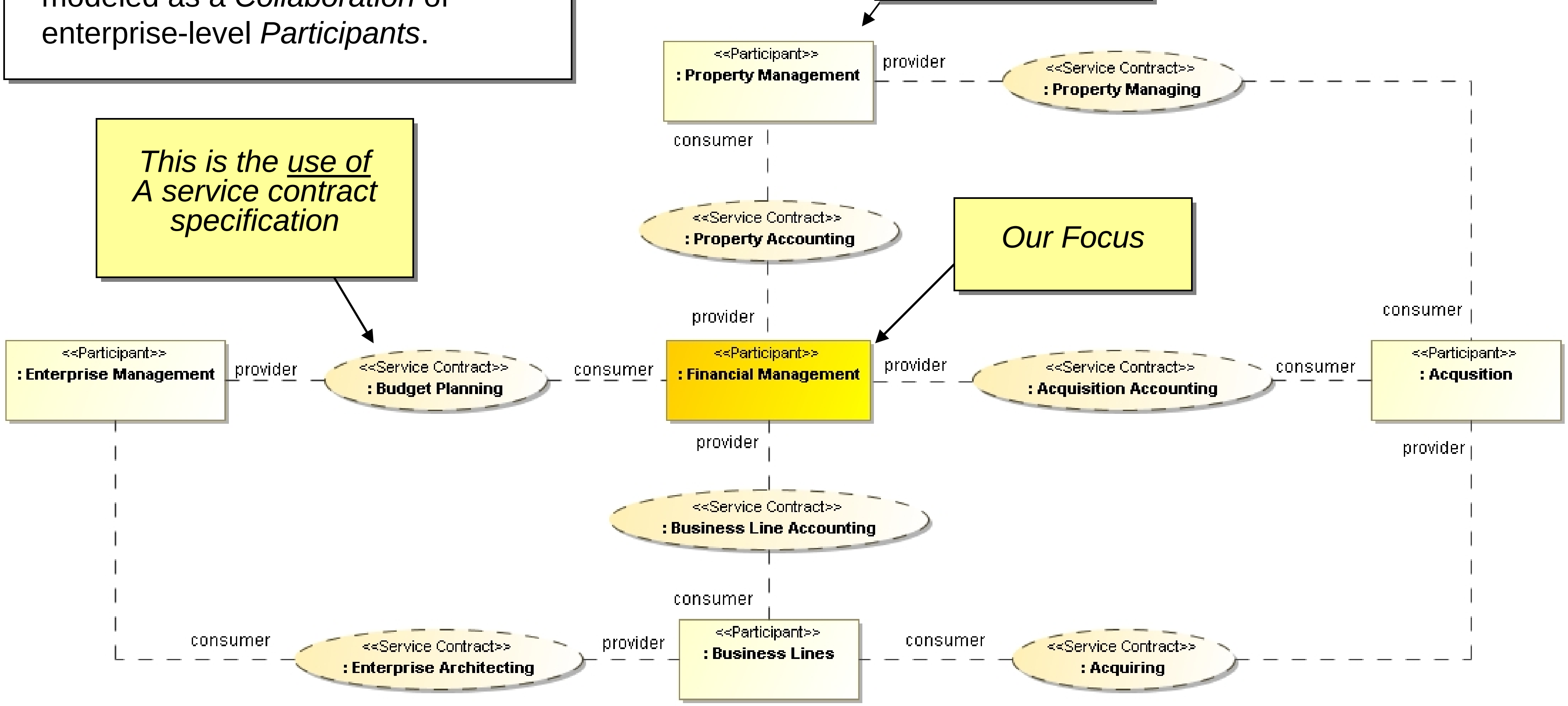
WSDL, BPEL, XML Schema

The enterprise as services

- Think about the enterprise as a set of interacting roles providing and using services to enable agility, cost savings and an effective transition framework
- **Externally**
  - The enterprise is part of the global supply chain, providing services to customers and using the services of suppliers
- **Internally**
  - Consider parts of the enterprise as providing services to other parts of the enterprise, and in term using the service of others
  - Like everything was outsourced as a service, it just happens to be done inside the organization.
- Business is modeled in terms of interacting roles – providing and using services – the essential concepts of enterprise SOA

Financial Management Enterprise Context

- The service-oriented business architecture of an enterprise is modeled as a *Collaboration* of enterprise-level *Participants*.



External enterprise level participants

<<Participant>>
: Property Management

provider

<<Service Contract>>
: Property Managing

consumer

<<Service Contract>>
: Property Accounting

provider

This is the <u>use of</u> A service contract specification

Our Focus

consumer

<<Participant>>
: Enterprise Management

provider

<<Service Contract>>
: Budget Planning

consumer

<<Participant>>
: Financial Management

provider

<<Service Contract>>
: Acquisition Accounting

consumer

<<Participant>>
: Acqusition

provider

provider

<<Service Contract>>
: Business Line Accounting

consumer

consumer

<<Service Contract>>
: Enterprise Architecting

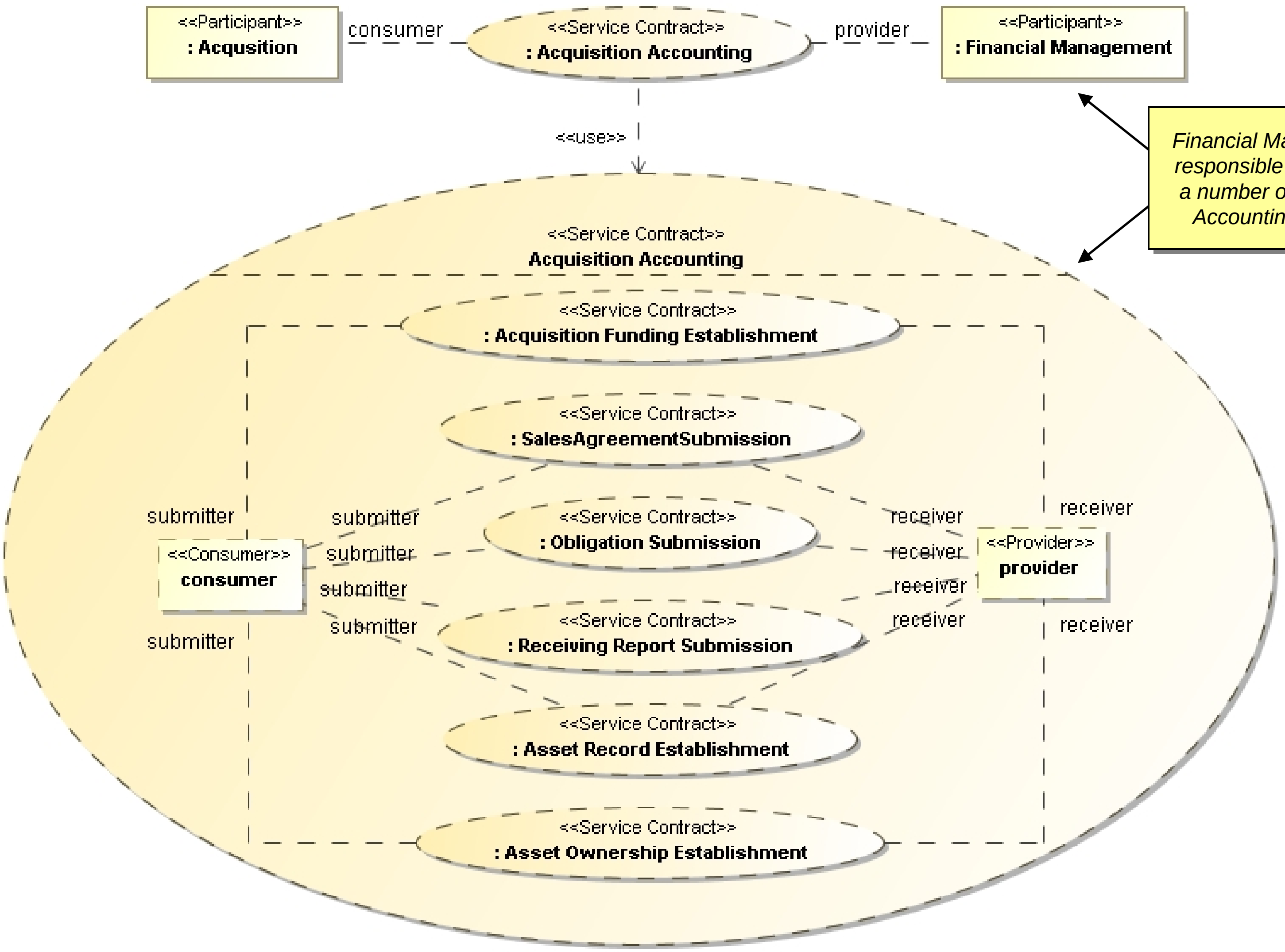
provider

<<Participant>>
: Business Lines

consumer

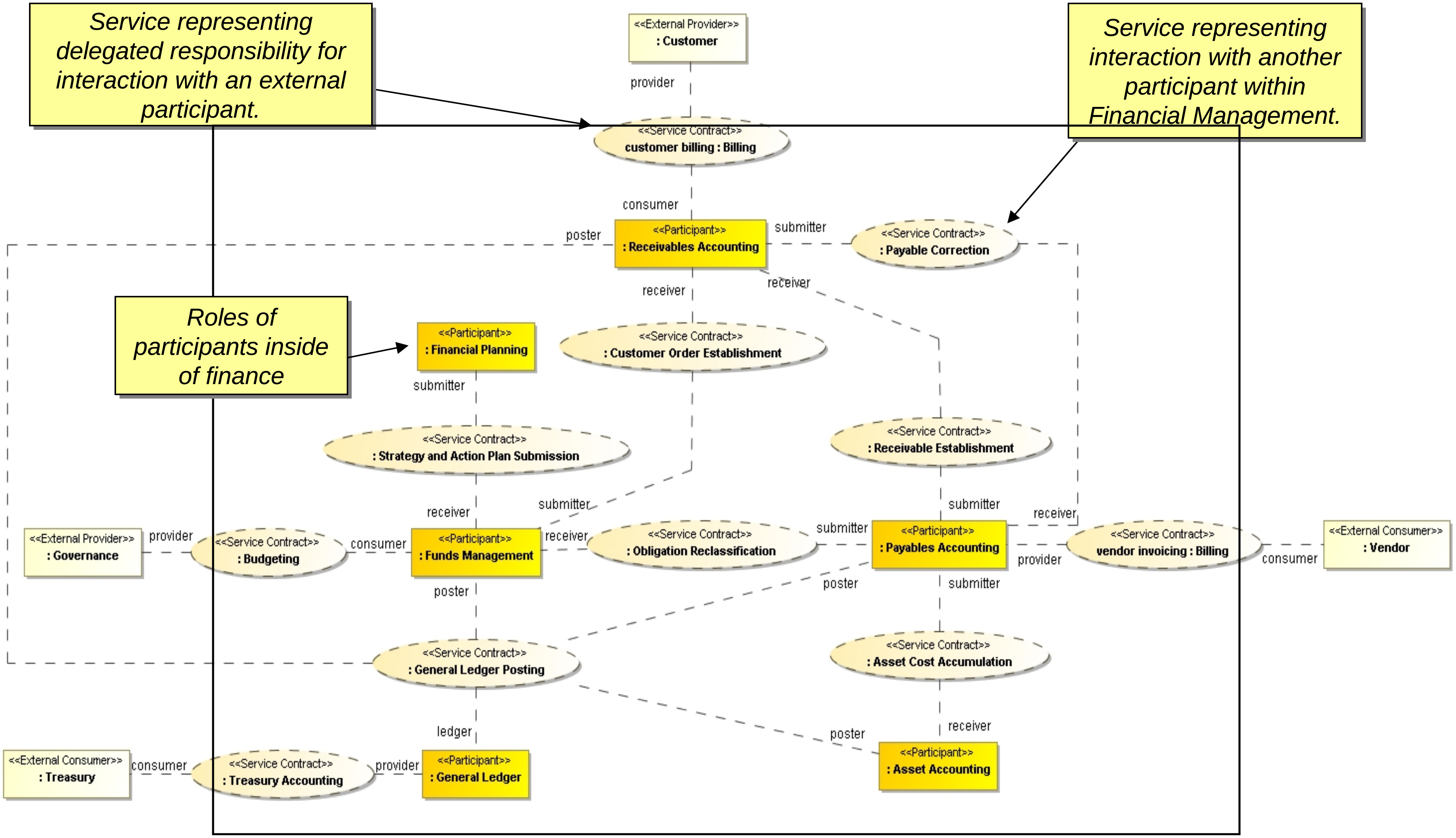
<<Service Contract>>
: Acquiring

A Composite Service Contract



Financial Management is responsible for providing a number of Acquisition Accounting services.

Inside Financial Management



Service representing delegated responsibility for interaction with an external participant.

Service representing interaction with another participant within Financial Management.

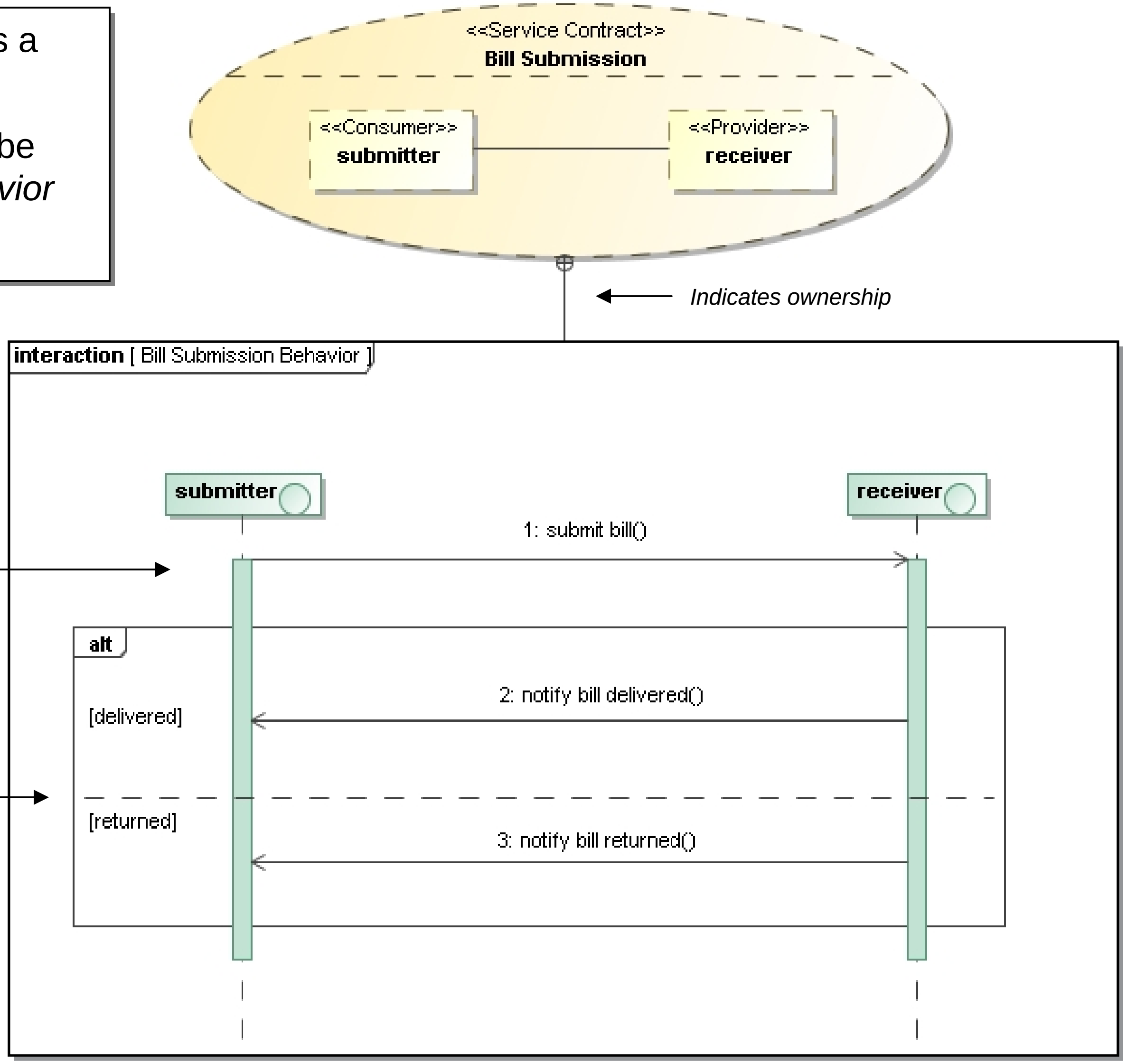
Roles of participants inside of finance

<<External Provider>> : Customer
provider
<<Service Contract>> customer billing : Billing
consumer
poster
<<Participant>> : Receivables Accounting
submitter
<<Service Contract>> : Payable Correction
receiver
receiver
<<Participant>> : Financial Planning
<<Service Contract>> : Customer Order Establishment
submitter
<<Service Contract>> : Strategy and Action Plan Submission
<<Service Contract>> : Receivable Establishment
receiver
submitter
submitter
receiver
<<External Provider>> : Governance
provider
<<Service Contract>> : Budgeting
consumer
<<Participant>> : Funds Management
receiver
<<Service Contract>> : Obligation Reclassification
submitter
<<Participant>> : Payables Accounting
<<Service Contract>> vendor invoicing : Billing
provider
consumer
<<External Consumer>> : Vendor
poster
poster
submitter
<<Service Contract>> : General Ledger Posting
<<Service Contract>> : Asset Cost Accumulation
ledger
poster
receiver
<<External Consumer>> : Treasury
consumer
<<Service Contract>> : Treasury Accounting
provider
<<Participant>> : General Ledger
<<Participant>> : Asset Accounting

Simple Bill Submission Service Contract

- A service contract is modeled as a *UML Collaboration*.
- The required conversation may be specified using an *Owned Behavior* (e.g., Interaction or Activity)

Note that, while one Participant requests the service and the other responds, information may flow both ways during the interaction.



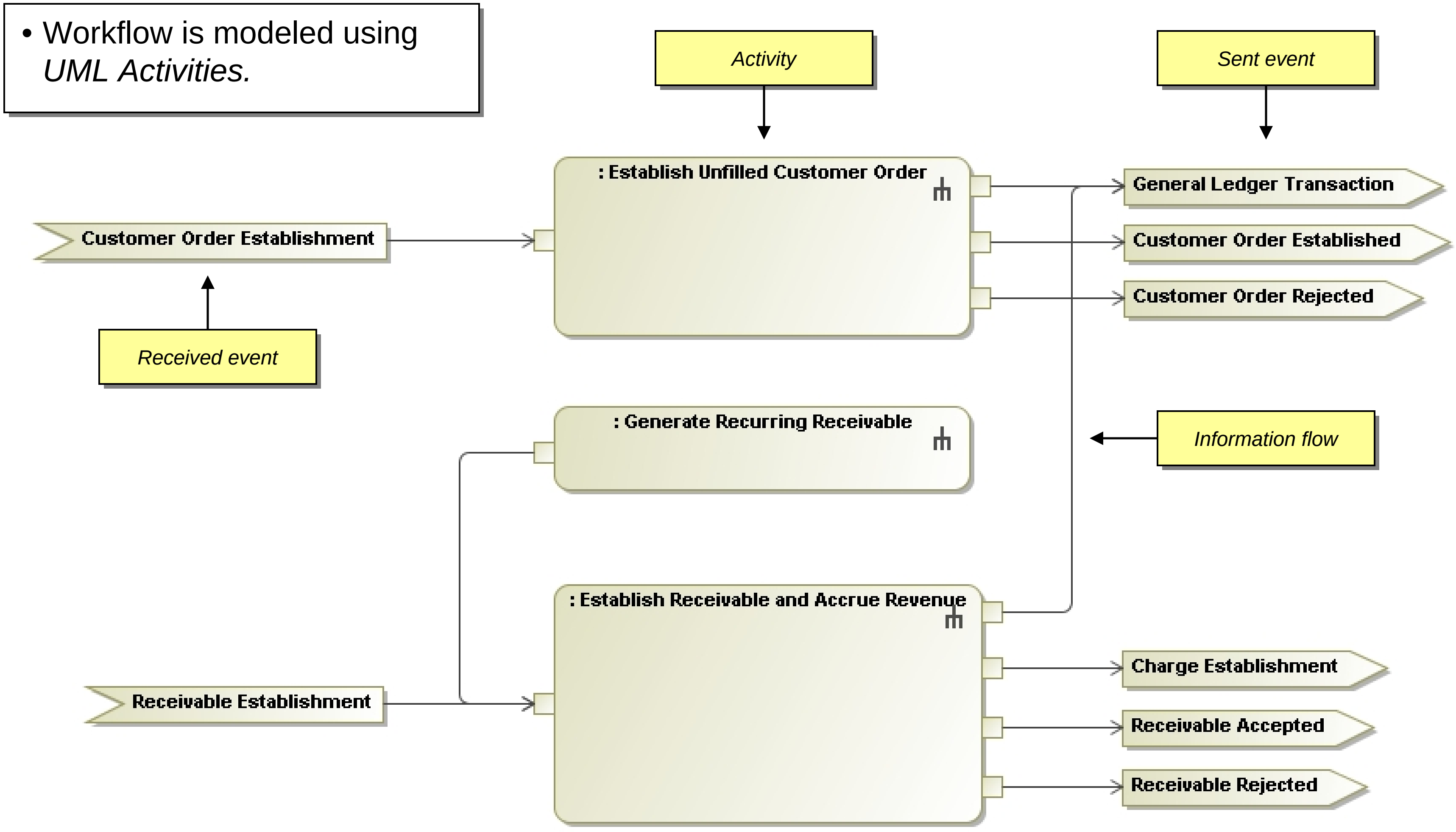
Indicates ownership

First the submitter submits a bill to the receiver…

…then either the bill is *successfully delivered or it is returned.*

Receivables Management Activities

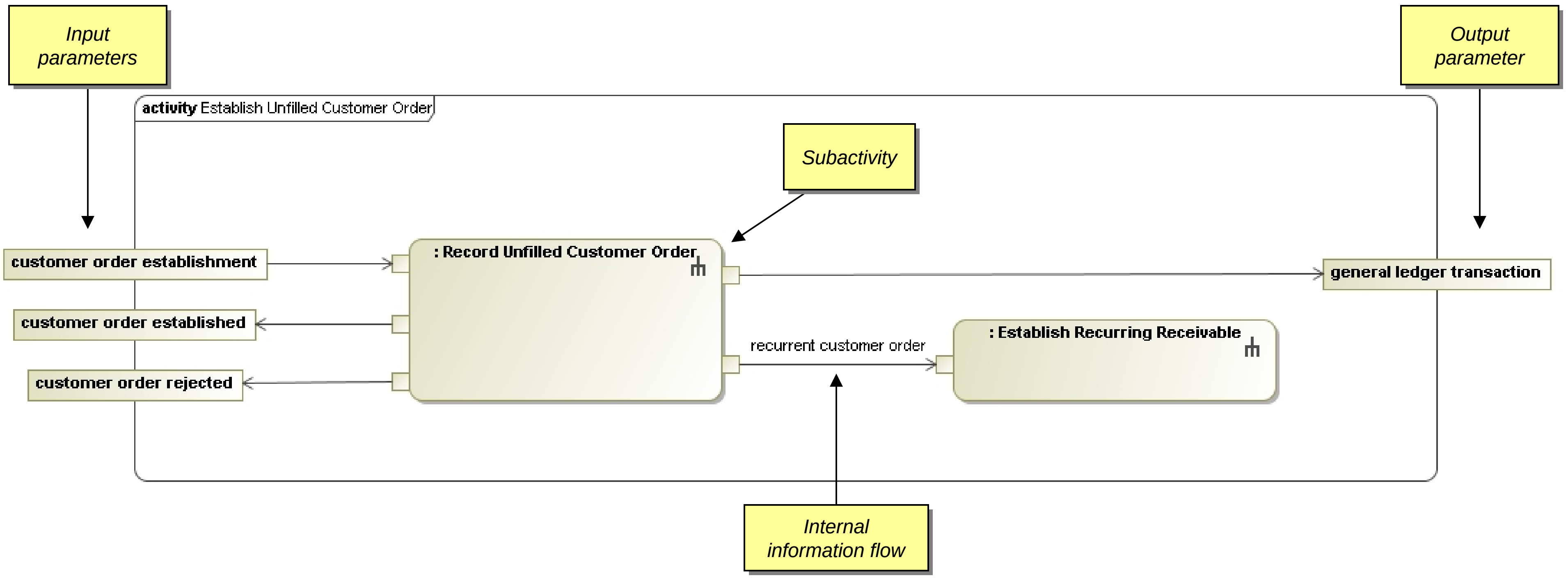
- Workflow is modeled using *UML Activities.*



Activity
Sent event
: Establish Unfilled Customer Order
General Ledger Transaction
Customer Order Establishment
Customer Order Established
Customer Order Rejected
Received event
: Generate Recurring Receivable
Information flow
: Establish Receivable and Accrue Revenue
Charge Establishment
Receivable Establishment
Receivable Accepted
Receivable Rejected

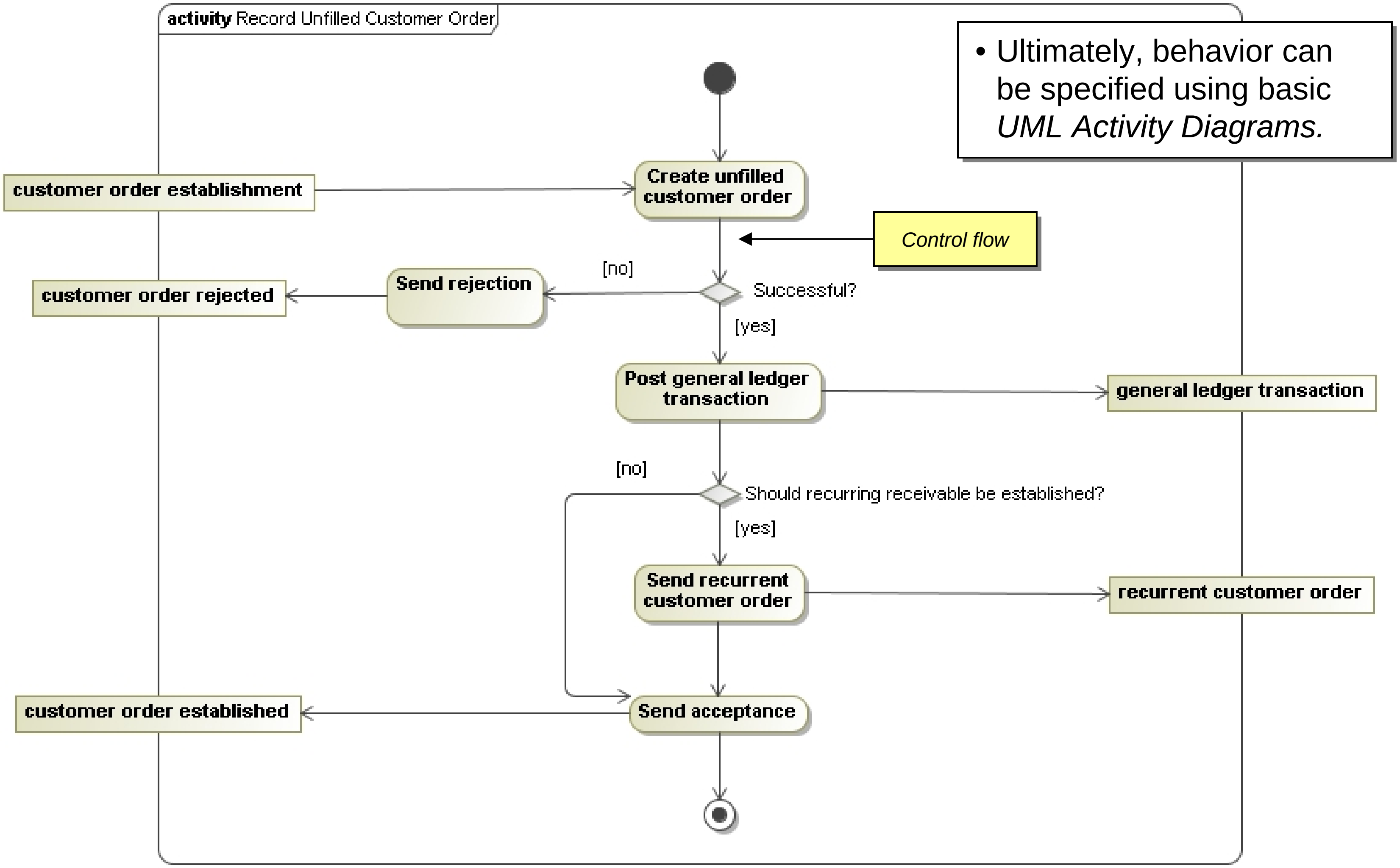
Establish Unfilled Customer Order Subactivities

- Complicated activities may be decomposed into subactivities.



Input parameters
Output parameter
activity Establish Unfilled Customer Order
Subactivity
customer order establishment
customer order established
customer order rejected
: Record Unfilled Customer Order
general ledger transaction
recurrent customer order
: Establish Recurring Receivable
Internal information flow

Record Unfilled Customer Order Behavior


activity Record Unfilled Customer Order
customer order establishment
Create unfilled customer order
Control flow
[no]
Successful?
Send rejection
customer order rejected
[yes]
Post general ledger transaction
general ledger transaction
[no]
Should recurring receivable be established?
[yes]
Send recurrent customer order
recurrent customer order
customer order established
Send acceptance

- Ultimately, behavior can be specified using basic *UML Activity Diagrams.*

Record Unfilled Customer Order Requirements

- Detailed requirements and business rules can be documented for activities separately from the process flow.

<u>Record Unfilled Customer Order</u>

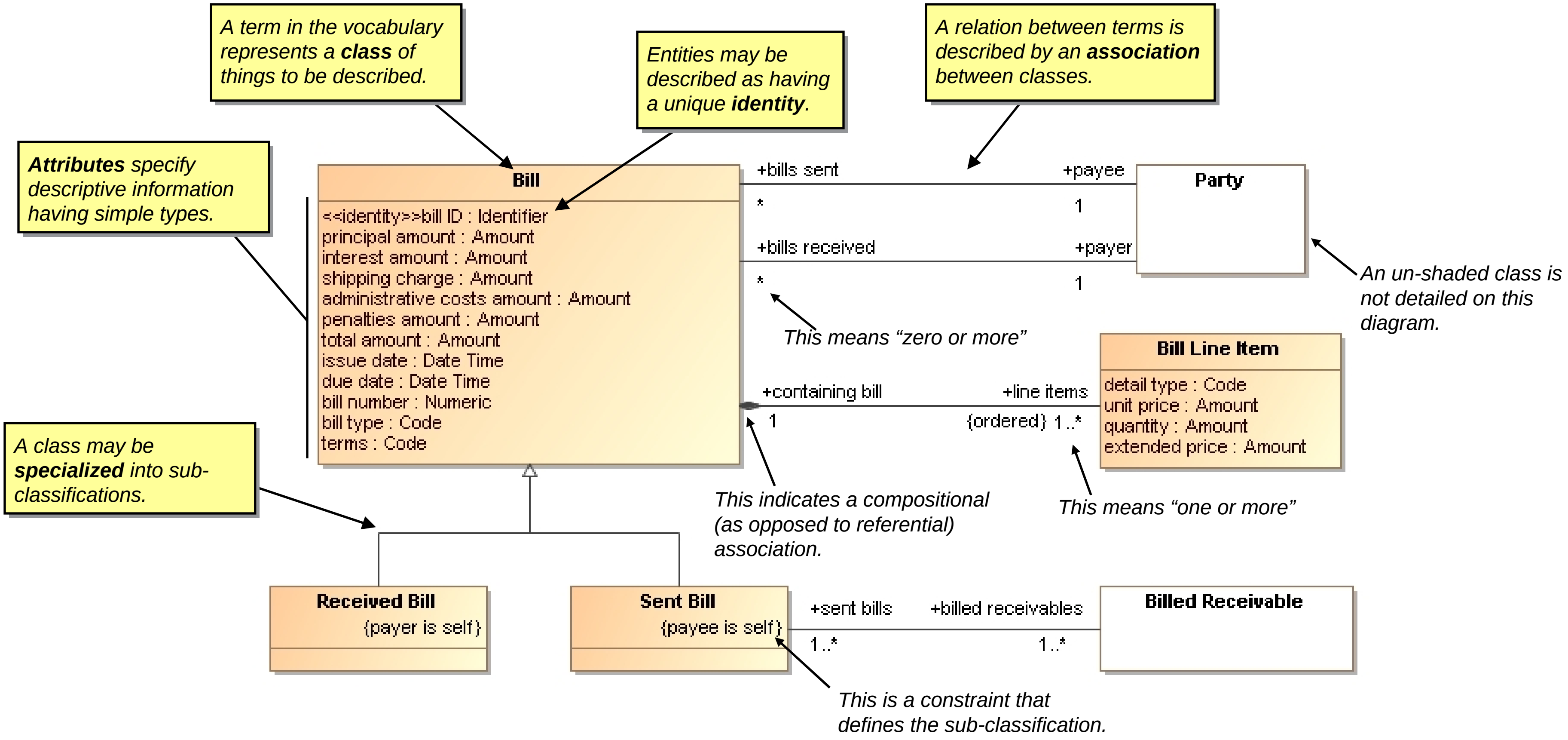
Description

- Record a new unfilled customer order, as established via a specific sales agreement.

Requirements

1. Generate general ledger transactions to increase Unfilled Customer Orders and decrease Anticipated Reimbursements.
2. If the Customer Order is against a Sales Agreement that requires recurring payments, establish a recurring receivable.
3. …

Information Model



A term in the vocabulary represents a ***class*** *of things to be described.*

Entities may be described as having a unique ***identity****.*

A relation between terms is described by an ***association*** *between classes.*

Attributes *specify descriptive information having simple types.*

An un-shaded class is not detailed on this diagram.

This means "zero or more"

A class may be ***specialized*** *into sub-classifications.*

This indicates a compositional (as opposed to referential) association.

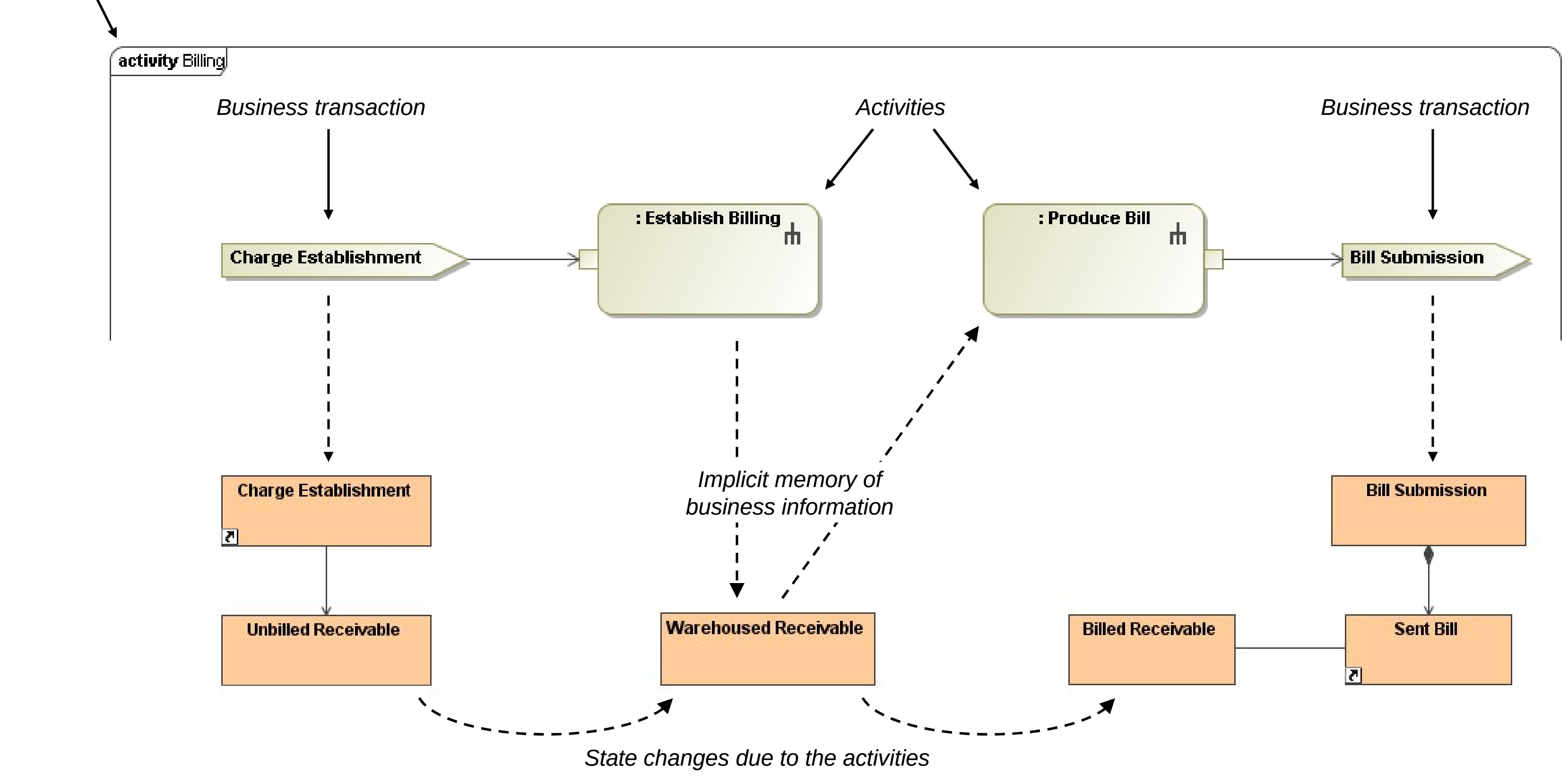
This means "one or more"

This is a constraint that defines the sub-classification.

Information Model: What Is It For?

*The **process model** describes how business activities are (or are to be) carried out.*

Workflow

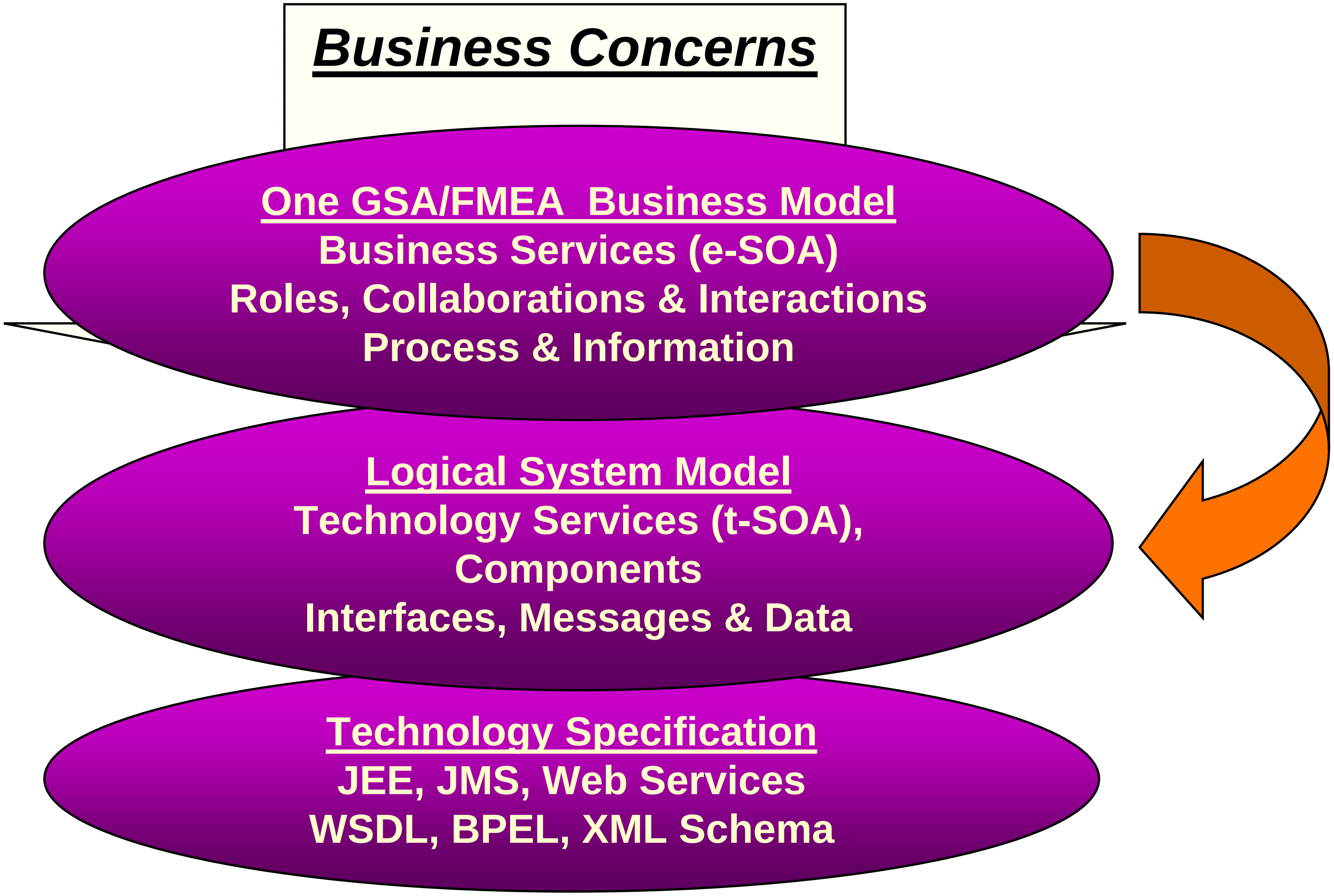


*The **information model** details the vocabulary of the business entities and transactions used in the process model.*

Summary

The CIM is a model of the business, not the information system

Producing the logical model



Business Concerns

One GSA/FMEA Business Model

Business Services (e-SOA)

Roles, Collaborations & Interactions

Process & Information

Logical System Model

Technology Services (t-SOA), Components

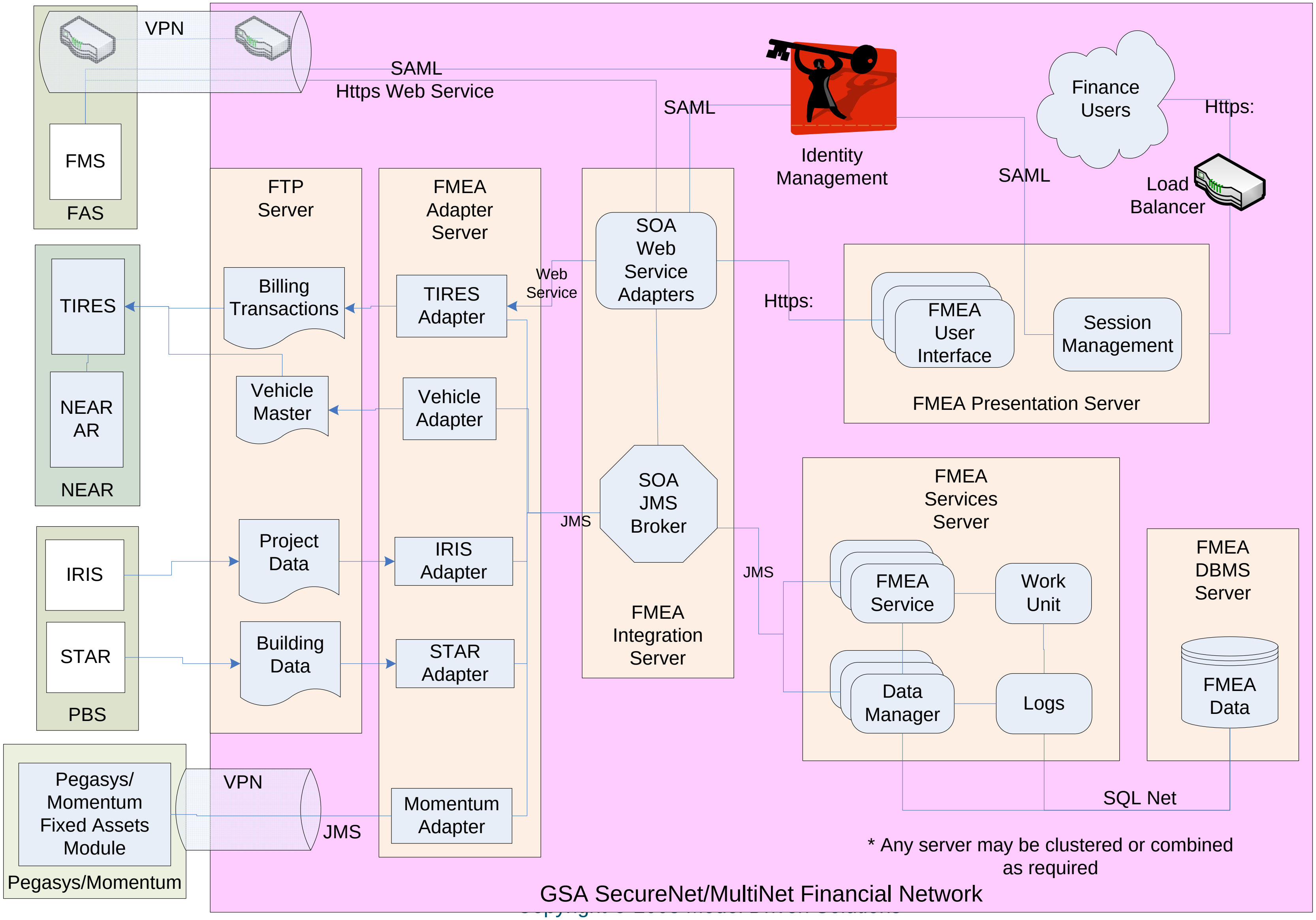
Interfaces, Messages & Data

Technology Specification

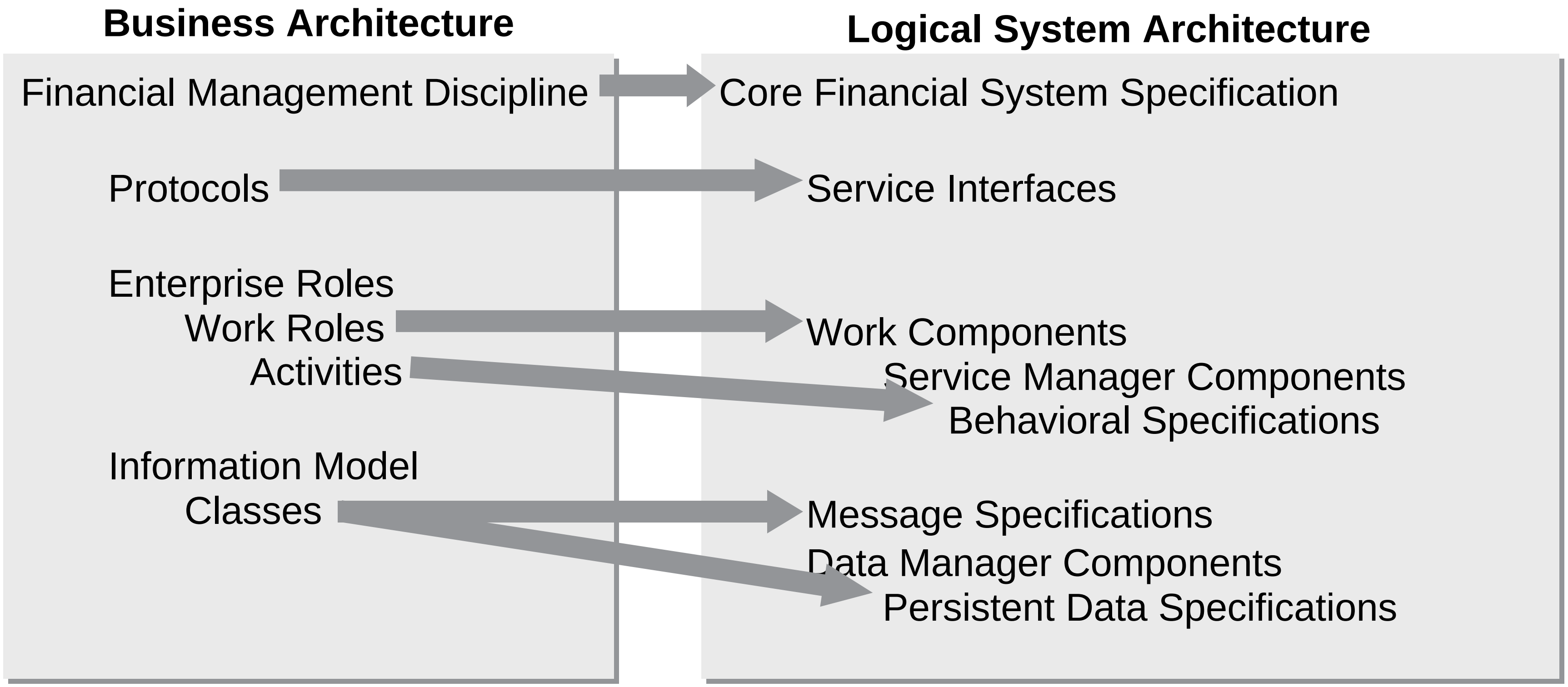
JEE, JMS, Web Services

WSDL, BPEL, XML Schema

FMEA Systems Architecture (High Level)



From Business Architecture to System Architecture



Business Architecture

Financial Management Discipline

Protocols

Enterprise Roles
Work Roles
Activities

Information Model
Classes

Logical System Architecture

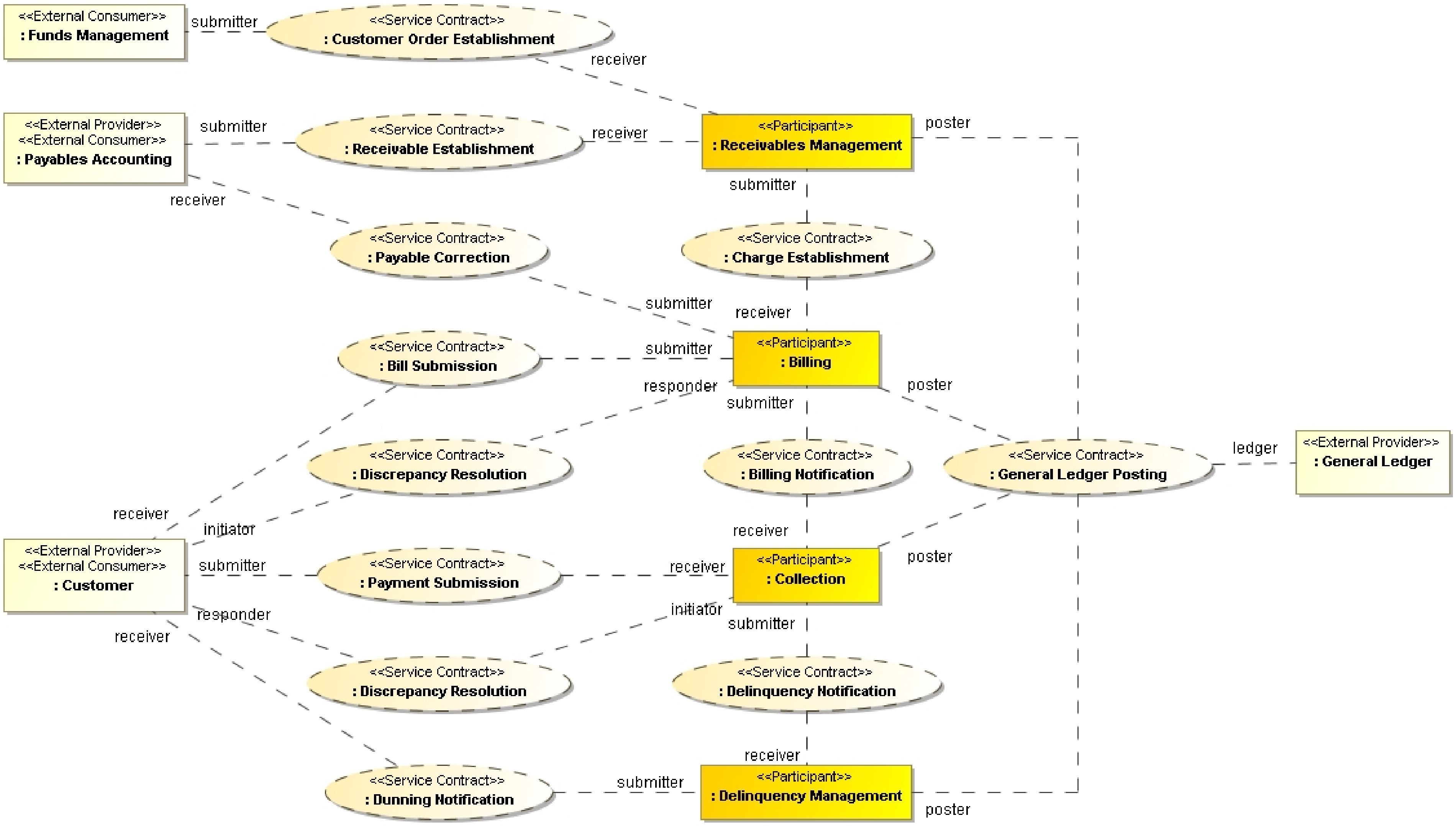
Core Financial System Specification

Service Interfaces

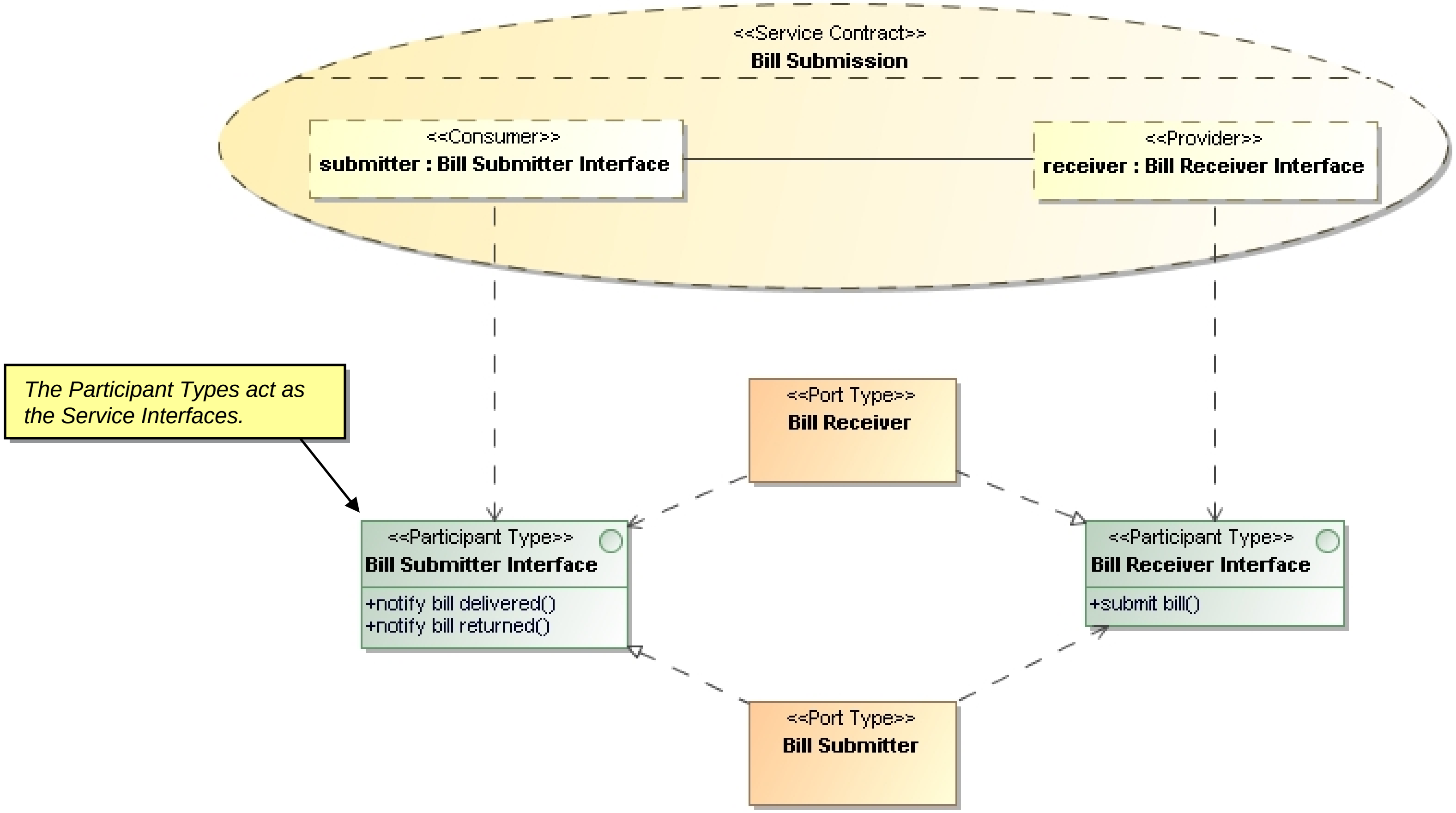
Work Components
Service Manager Components
Behavioral Specifications

Message Specifications
Data Manager Components
Persistent Data Specifications

Receivables Accounting Business Architecture (from Business Model)

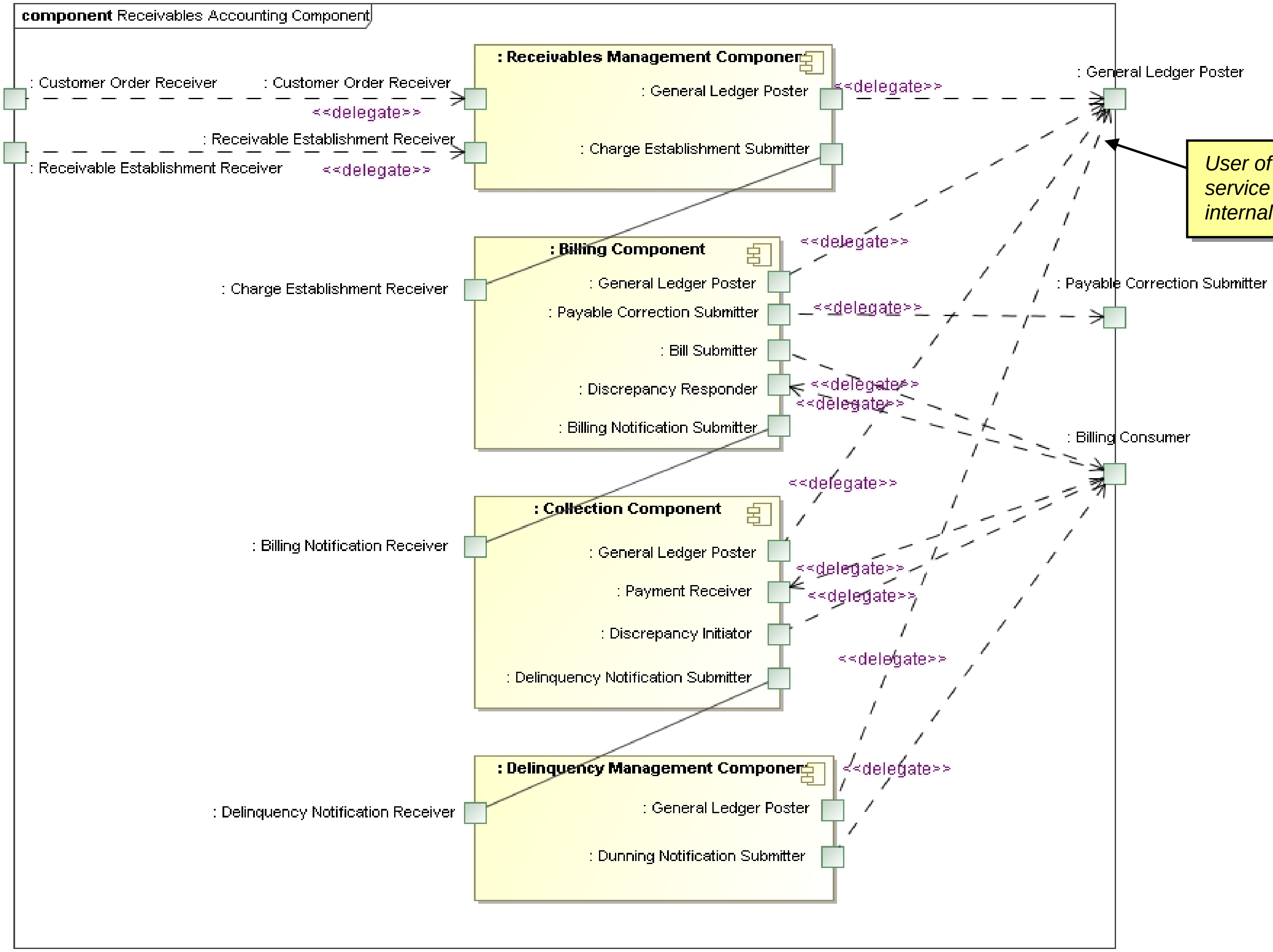


Port Types from Service Contracts



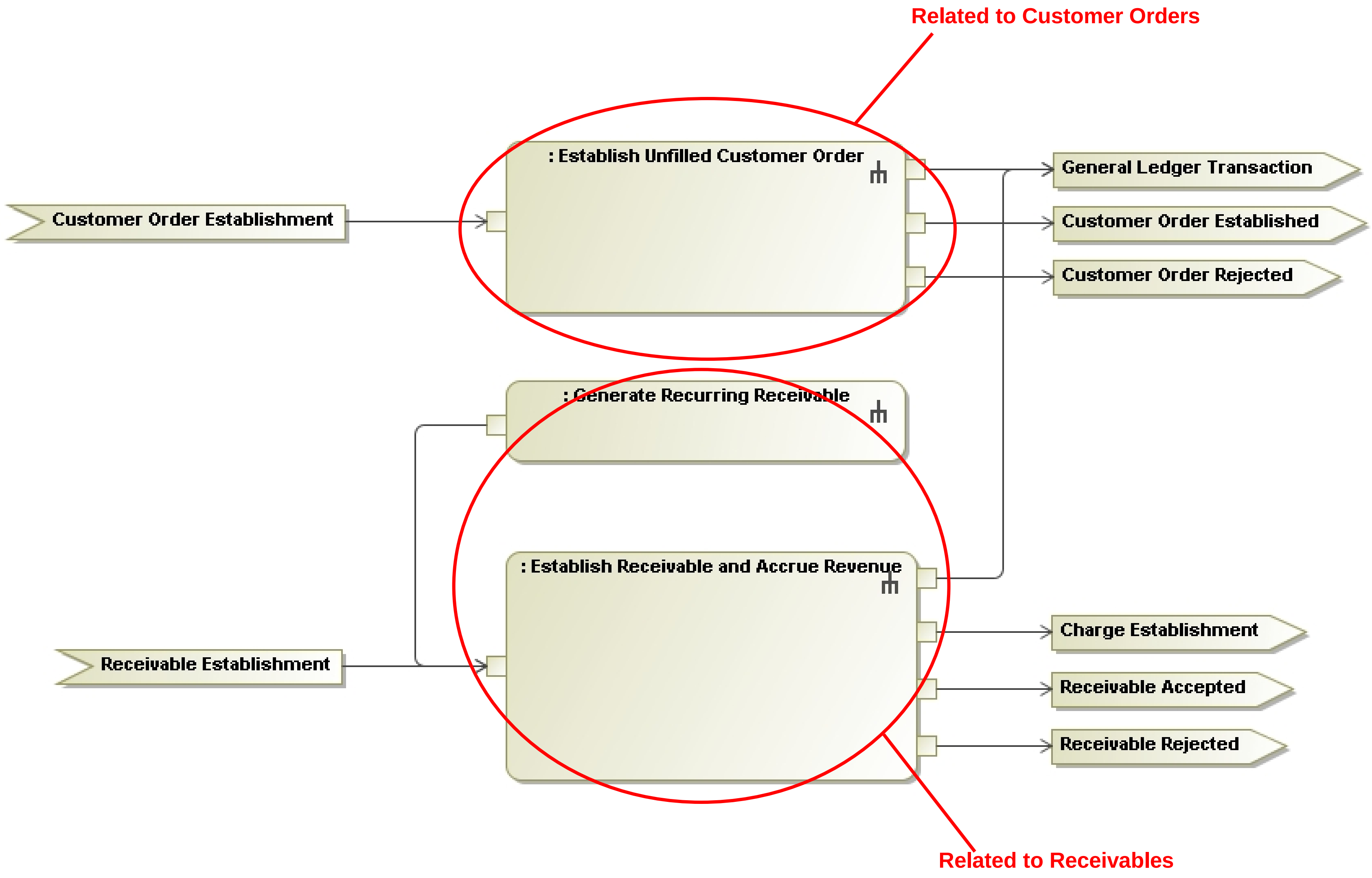
The Participant Types act as the Service Interfaces.

Receivables Accounting Component Architecture

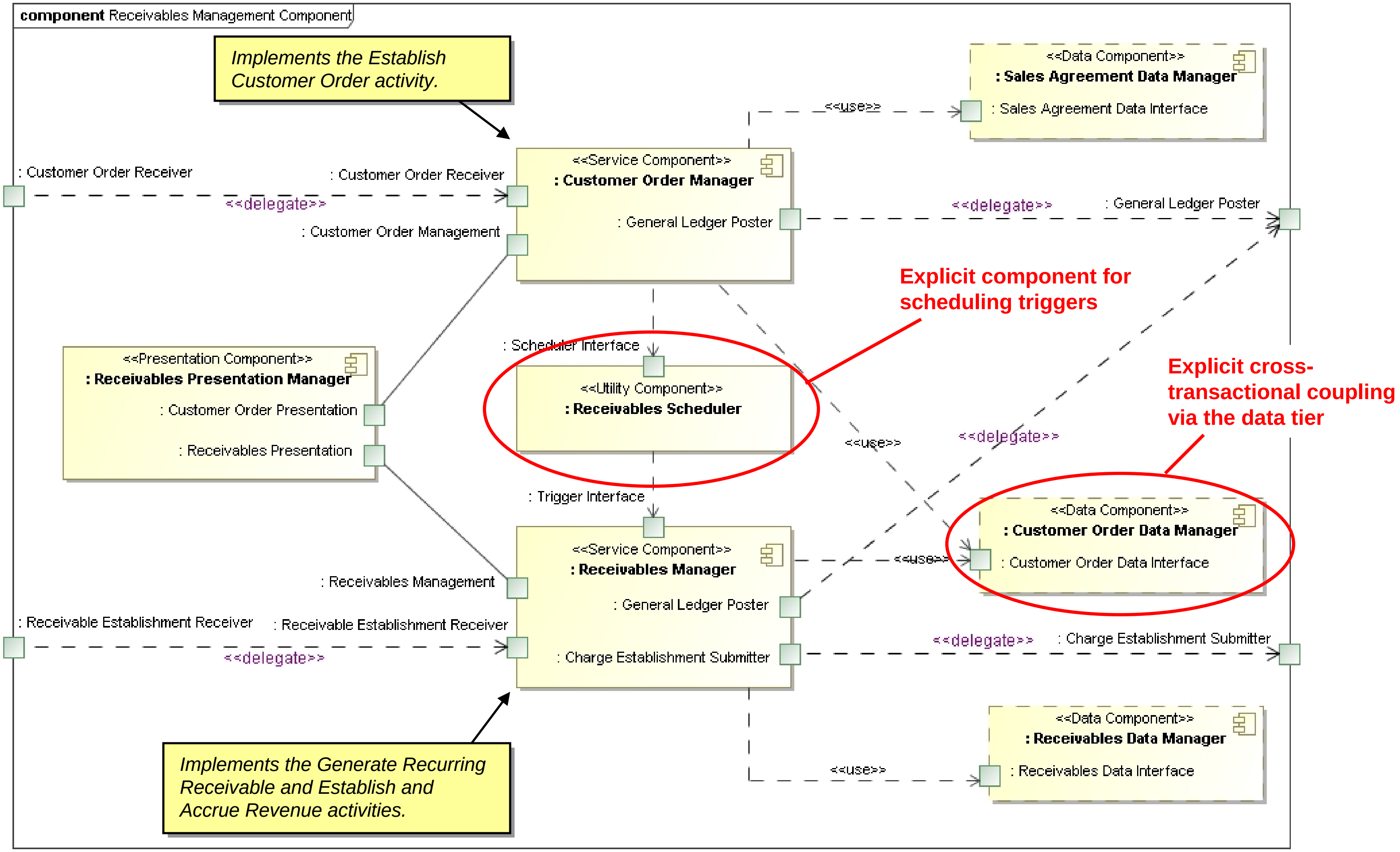

component Receivables Accounting Component
: Receivables Management Component
: Customer Order Receiver
: Customer Order Receiver
<<delegate>>
: Receivable Establishment Receiver
: Receivable Establishment Receiver
<<delegate>>
: General Ledger Poster
: Charge Establishment Submitter
<<delegate>>
: General Ledger Poster
: Billing Component
: Charge Establishment Receiver
: General Ledger Poster
: Payable Correction Submitter
: Bill Submitter
: Discrepancy Responder
: Billing Notification Submitter
<<delegate>>
<<delegate>>
<<delegate>>
<<delegate>>
: Payable Correction Submitter
: Billing Consumer
: Collection Component
: Billing Notification Receiver
: General Ledger Poster
: Payment Receiver
: Discrepancy Initiator
: Delinquency Notification Submitter
<<delegate>>
<<delegate>>
<<delegate>>
<<delegate>>
: Delinquency Management Component
: Delinquency Notification Receiver
: General Ledger Poster
: Dunning Notification Submitter
<<delegate>>

User of a consumed service by multiple internal parts.

Receivables Management Activities (from Business Model)


Related to Customer Orders
: Establish Unfilled Customer Order
Customer Order Establishment
General Ledger Transaction
Customer Order Established
Customer Order Rejected
: Generate Recurring Receivable
: Establish Receivable and Accrue Revenue
Receivable Establishment
Charge Establishment
Receivable Accepted
Receivable Rejected
Related to Receivables

Receivables Management Component Architecture


component Receivables Management Component
<<Data Component>>
: Sales Agreement Data Manager
: Sales Agreement Data Interface
<<use>>
: Customer Order Receiver
: Customer Order Receiver
<<delegate>>
<<Service Component>>
: Customer Order Manager
: General Ledger Poster
<<delegate>>
: General Ledger Poster
: Customer Order Management
<<Presentation Component>>
: Receivables Presentation Manager
: Customer Order Presentation
: Receivables Presentation
: Scheduler Interface
<<Utility Component>>
: Receivables Scheduler
<<use>>
<<delegate>>
: Trigger Interface
<<Data Component>>
: Customer Order Data Manager
: Customer Order Data Interface
<<use>>
<<Service Component>>
: Receivables Manager
: Receivables Management
: General Ledger Poster
: Receivable Establishment Receiver
: Receivable Establishment Receiver
<<delegate>>
: Charge Establishment Submitter
<<delegate>>
: Charge Establishment Submitter
<<Data Component>>
: Receivables Data Manager
: Receivables Data Interface
<<use>>

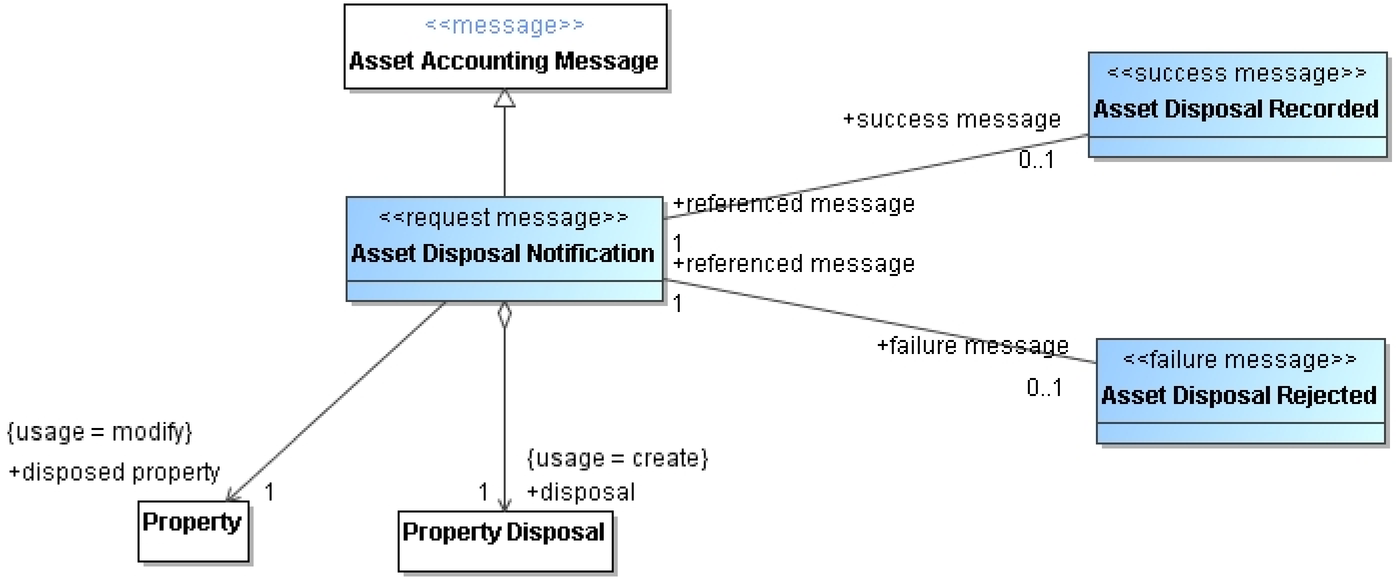
Implements the Establish Customer Order activity.

Explicit component for scheduling triggers

Explicit cross-transactional coupling via the data tier

Implements the Generate Recurring Receivable and Establish and Accrue Revenue activities.

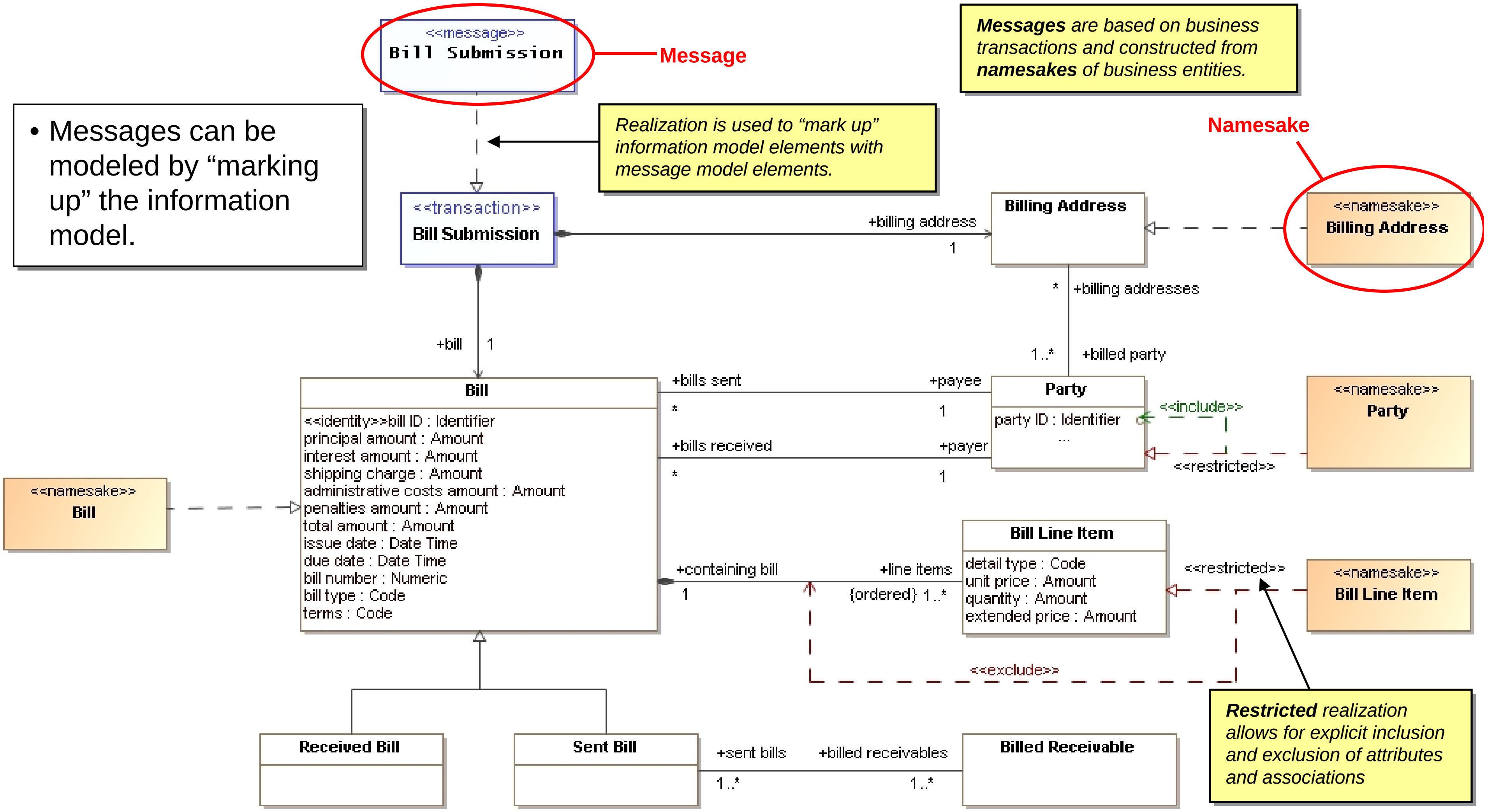
Messages – Transfer Information in Services



(May be a transfer or a write off)

Example Transaction Message Model

- Messages can be modeled by “marking up” the information model.



Messages *are based on business transactions and constructed from* ***namesakes*** *of business entities.*

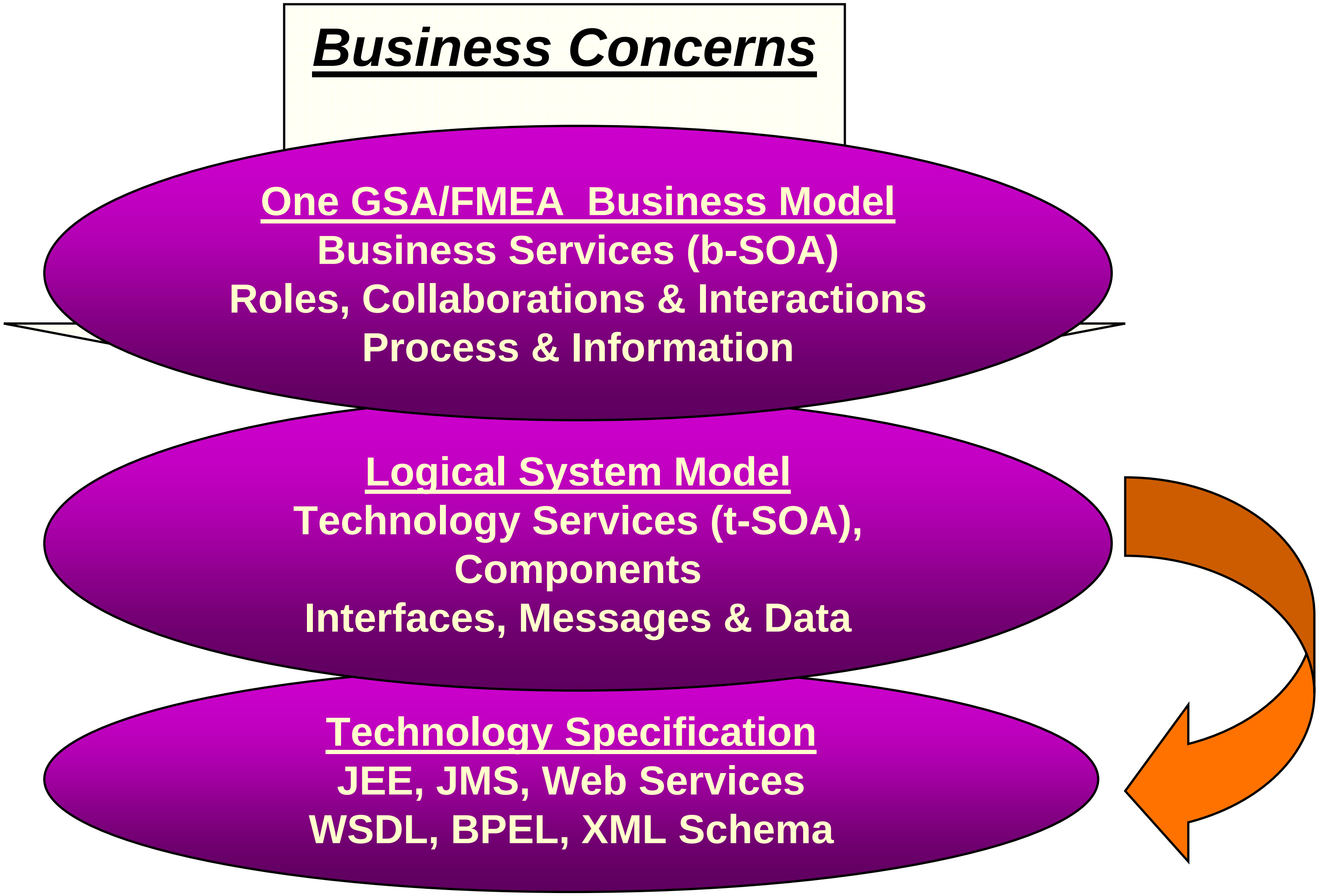
Realization is used to “mark up” information model elements with message model elements.

Restricted *realization allows for explicit inclusion and exclusion of attributes and associations*

Summary

The PIM is a model of the system and how it realizes the business requirements, but the technology decisions and details are abstracted out

Technology Architecture



Business Concerns

One GSA/FMEA Business Model

Business Services (b-SOA)

Roles, Collaborations & Interactions

Process & Information

Logical System Model

Technology Services (t-SOA), Components

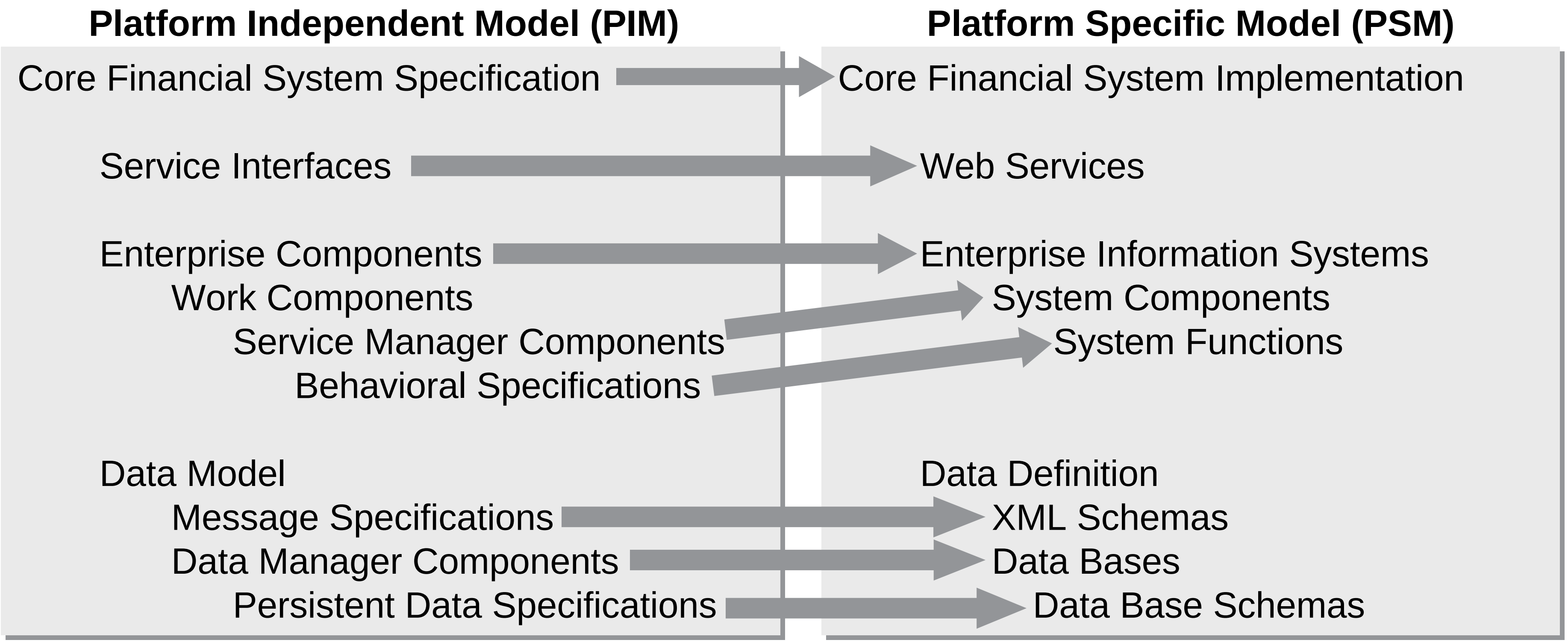
Interfaces, Messages & Data

Technology Specification

JEE, JMS, Web Services

WSDL, BPEL, XML Schema

Platform Specific Model



Platform Independent Model (PIM)

Core Financial System Specification

Service Interfaces

Enterprise Components
Work Components
Service Manager Components
Behavioral Specifications

Data Model
Message Specifications
Data Manager Components
Persistent Data Specifications

Platform Specific Model (PSM)

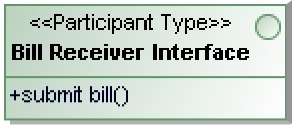
Core Financial System Implementation

Web Services

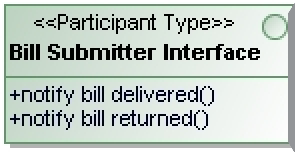
Enterprise Information Systems
System Components
System Functions

Data Definition
XML Schemas
Data Bases
Data Base Schemas

Example Web Services Generation



```
<wsdl:portType name=“BillSubmission.BillSubmissionReceiverInterface">
   <wsdl:operation name=“submitBill">
     <wsdl:input message="tns:BillSubmissionCluster“
                 name=“billSubmission">
     </wsdl:input>
   </wsdl:operation>
 </wsdl:portType>
```

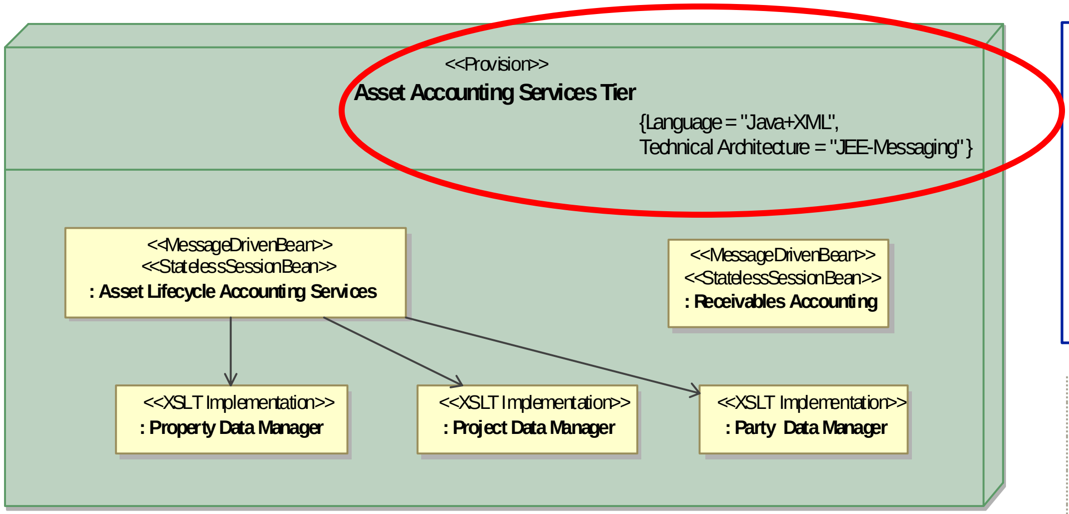


```
<wsdl:portType name=“BillSubmission.BillSubmissionSubmitterInterface">
  <wsdl:operation name=“notifyBillDelivered">
    <wsdl:input message="tns:BillDeliveredCluster“
                name=“billDelivered">
    </wsdl:input>
  </wsdl:operation>
  <wsdl:operation name=“notifyBillReturned">
    <wsdl:input message="tns:BillReturnedCluster“
                name=“billReturned">
    </wsdl:input>
  </wsdl:operation>
</wsdl:portType>
```

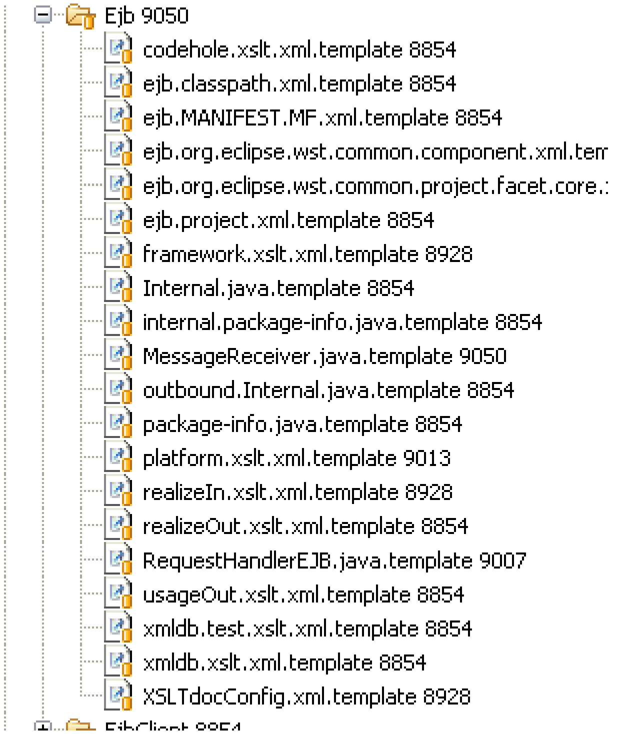
Example Transaction Message XML Document

```
<BillSubmissionCluster>
     <BusinessTransaction>
          <transactionID> … </transactionID>
     </BusinessTransaction>
     <BillSubmission>
          <bill>
               <Bill>
                    <billID> … </billID>
                    <principleAmount> … </principleAmount>
                    …
                    <payer>
                         <Party>
                              <partyID> … </customerID>
                         </Party>
                    </payer>
                    …
                    <lineItems>
                         …
                    </lineItems>
               </Bill>
          </bill>
          <billingAddress>
               <BillingAddressCluster>
                    <Address> … </Address>
                    <BillingAddress> … </BillingAddress>
               </BillingAddressCluster>
          <billingAddress>
     </BillSubmission>
</BillSubmissionCluster>
```

Putting it all together

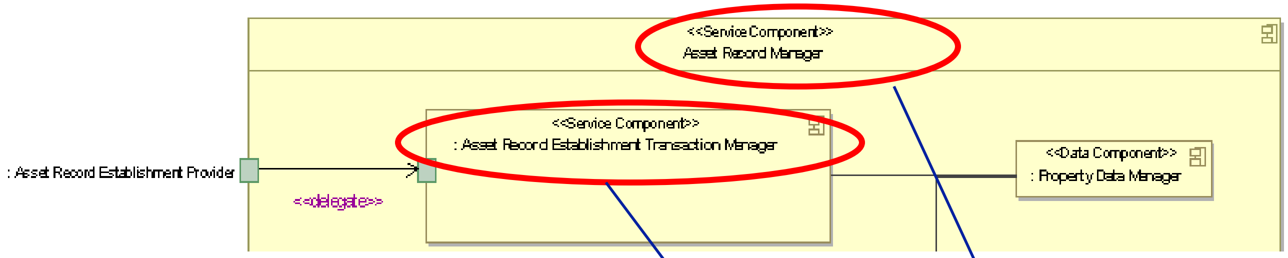


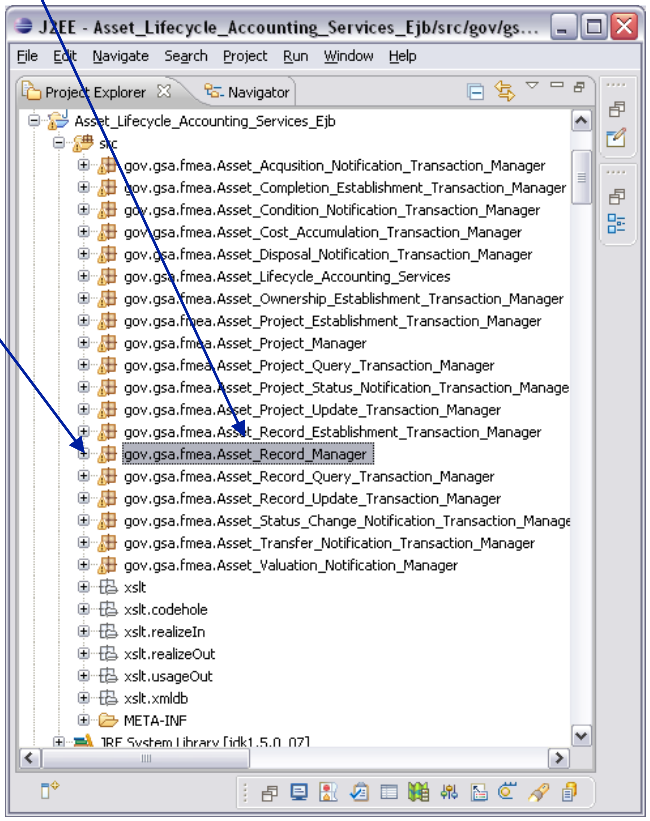
```
<provisioningContext name="service“…>
               <projectRef folder="EjbClient"/>
               <projectRef folder="AppClient"/>
               <projectRef folder="Ejb"/>
               <projectRef folder="ear"/>
               <projectRef folder="JbossConfig"/>
</provisioningContext>
```



```
Ejb 9050
    codehole.xslt.xml.template 8854
    ejb.classpath.xml.template 8854
    ejb.MANIFEST.MF.xml.template 8854
    ejb.org.eclipse.wst.common.component.xml.terr
    ejb.org.eclipse.wst.common.project.facet.core.:
    ejb.project.xml.template 8854
    framework.xslt.xml.template 8928
    Internal.java.template 8854
    internal.package-info.java.template 8854
    MessageReceiver.java.template 9050
    outbound.Internal.java.template 8854
    package-info.java.template 8854
    platform.xslt.xml.template 9013
    realizeIn.xslt.xml.template 8928
    realizeOut.xslt.xml.template 8854
    RequestHandlerEJB.java.template 9007
    usageOut.xslt.xml.template 8854
    xmldb.test.xslt.xml.template 8854
    xmldb.xslt.xml.template 8854
    XSLTdocConfig.xml.template 8928
EjbClient 8854
```

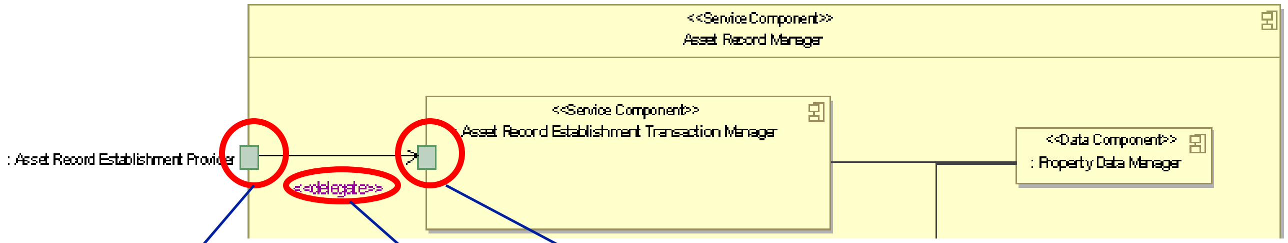
Putting it all together





Asset_Lifecycle_Accounting_Services_Ejb
- src
  - gov.gsa.fmea.Asset_Acqusition_Notification_Transaction_Manager
  - gov.gsa.fmea.Asset_Completion_Establishment_Transaction_Manager
  - gov.gsa.fmea.Asset_Condition_Notification_Transaction_Manager
  - gov.gsa.fmea.Asset_Cost_Accumulation_Transaction_Manager
  - gov.gsa.fmea.Asset_Disposal_Notification_Transaction_Manager
  - gov.gsa.fmea.Asset_Lifecycle_Accounting_Services
  - gov.gsa.fmea.Asset_Ownership_Establishment_Transaction_Manager
  - gov.gsa.fmea.Asset_Project_Establishment_Transaction_Manager
  - gov.gsa.fmea.Asset_Project_Manager
  - gov.gsa.fmea.Asset_Project_Query_Transaction_Manager
  - gov.gsa.fmea.Asset_Project_Status_Notification_Transaction_Manage
  - gov.gsa.fmea.Asset_Project_Update_Transaction_Manager
  - gov.gsa.fmea.Asset_Record_Establishment_Transaction_Manager
  - gov.gsa.fmea.Asset_Record_Manager
  - gov.gsa.fmea.Asset_Record_Query_Transaction_Manager
  - gov.gsa.fmea.Asset_Record_Update_Transaction_Manager
  - gov.gsa.fmea.Asset_Status_Change_Notification_Transaction_Manage
  - gov.gsa.fmea.Asset_Transfer_Notification_Transaction_Manager
  - gov.gsa.fmea.Asset_Valuation_Notification_Manager
  - xslt
  - xslt.codehole
  - xslt.realizeIn
  - xslt.realizeOut
  - xslt.usageOut
  - xslt.xmldb
  - META-INF
- JRE System Library [jdk1.5.0_07]

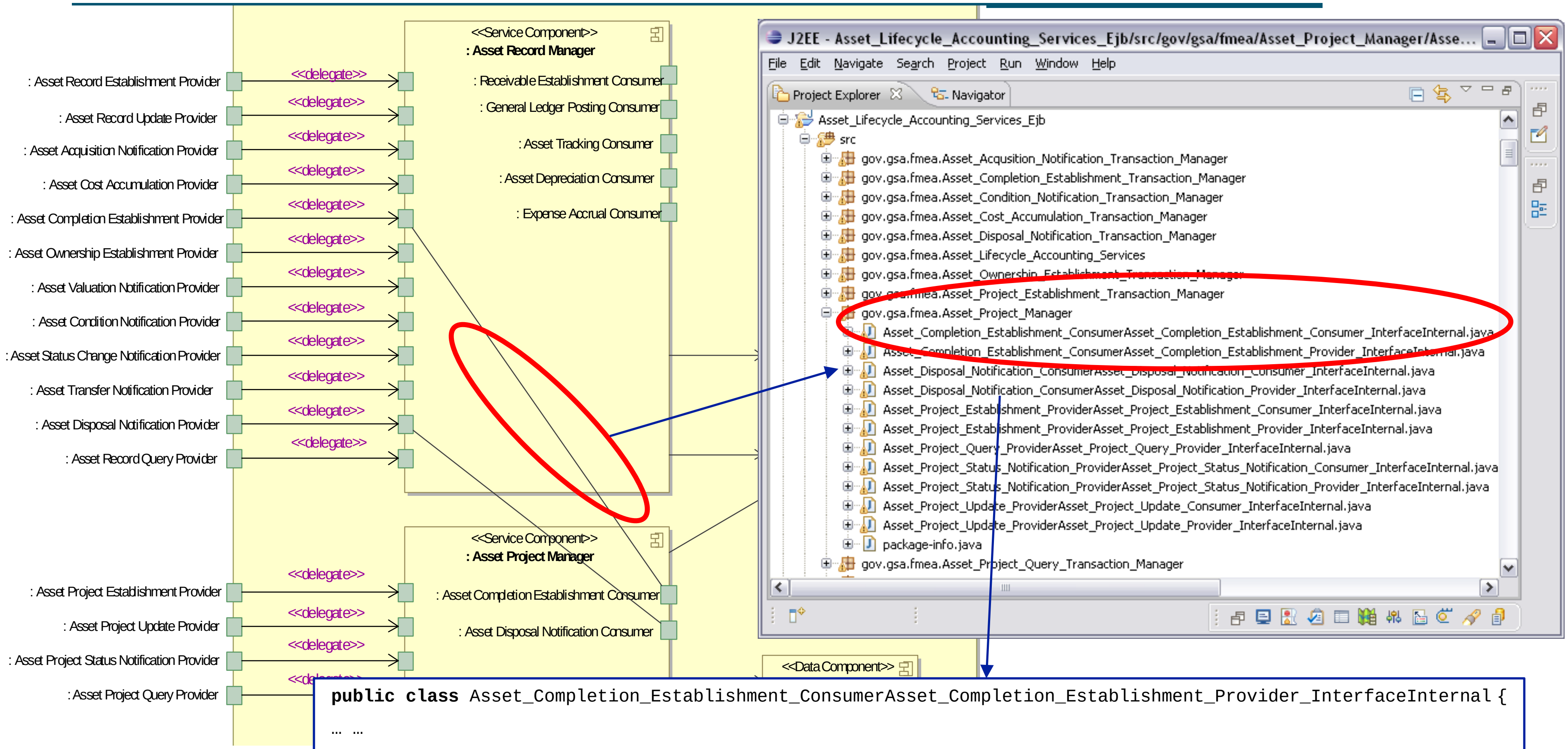
Putting it all together



```
public class Asset_Record_Establishment_ProviderAsset_Record_Establishment_Provider_InterfaceInternal {
……
static public Document establish_asset_record(Document request) throws CheckedException {
… …
// for an inbound operation, determine if we delegate or execute
return
gov.gsa.fmea.Asset_Record_Establishment_Transaction_Manager.
              Asset_Record_Establishment_ProviderAsset_Record_Establishment_Provider_InterfaceInternal.
              establish_asset_record(request);
```

```
public class Asset_Record_Establishment_ProviderAsset_Record_Establishment_Provider_InterfaceInternal{
… …
static public Document establish_asset_record(Document request) throws CheckedException {
… …
// for an inbound operation, determine if we delegate or execute
return ServiceFactory.getPipeline("Asset_Record_Establishment_Transaction_Manager.
              Asset_Record_Establishment_Provider.establish_asset_record.Pipeline").execute(request);
}
```

Putting it all together

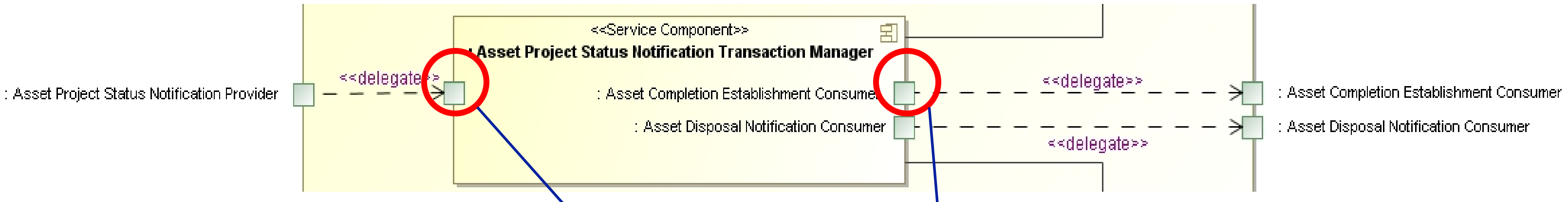


: Asset Record Establishment Provider
: Asset Record Update Provider
: Asset Acquisition Notification Provider
: Asset Cost Accumulation Provider
: Asset Completion Establishment Provider
: Asset Ownership Establishment Provider
: Asset Valuation Notification Provider
: Asset Condition Notification Provider
: Asset Status Change Notification Provider
: Asset Transfer Notification Provider
: Asset Disposal Notification Provider
: Asset Record Query Provider
<<delegate>>
<<Service Component>>
: Asset Record Manager
: Receivable Establishment Consumer
: General Ledger Posting Consumer
: Asset Tracking Consumer
: Asset Depreciation Consumer
: Expense Accrual Consumer
: Asset Project Establishment Provider
: Asset Project Update Provider
: Asset Project Status Notification Provider
: Asset Project Query Provider
<<Service Component>>
: Asset Project Manager
: Asset Completion Establishment Consumer
: Asset Disposal Notification Consumer
<<Data Component>>
J2EE - Asset_Lifecycle_Accounting_Services_Ejb/src/gov/gsa/fmea/Asset_Project_Manager/Asse...
File Edit Navigate Search Project Run Window Help
Project Explorer
Navigator

```
Asset_Lifecycle_Accounting_Services_Ejb
  src
    gov.gsa.fmea.Asset_Acqusition_Notification_Transaction_Manager
    gov.gsa.fmea.Asset_Completion_Establishment_Transaction_Manager
    gov.gsa.fmea.Asset_Condition_Notification_Transaction_Manager
    gov.gsa.fmea.Asset_Cost_Accumulation_Transaction_Manager
    gov.gsa.fmea.Asset_Disposal_Notification_Transaction_Manager
    gov.gsa.fmea.Asset_Lifecycle_Accounting_Services
    gov.gsa.fmea.Asset_Ownership_Establishment_Transaction_Manager
    gov.gsa.fmea.Asset_Project_Establishment_Transaction_Manager
    gov.gsa.fmea.Asset_Project_Manager
      Asset_Completion_Establishment_ConsumerAsset_Completion_Establishment_Consumer_InterfaceInternal.java
      Asset_Completion_Establishment_ConsumerAsset_Completion_Establishment_Provider_InterfaceInternal.java
      Asset_Disposal_Notification_ConsumerAsset_Disposal_Notification_Consumer_InterfaceInternal.java
      Asset_Disposal_Notification_ConsumerAsset_Disposal_Notification_Provider_InterfaceInternal.java
      Asset_Project_Establishment_ProviderAsset_Project_Establishment_Consumer_InterfaceInternal.java
      Asset_Project_Establishment_ProviderAsset_Project_Establishment_Provider_InterfaceInternal.java
      Asset_Project_Query_ProviderAsset_Project_Query_Provider_InterfaceInternal.java
      Asset_Project_Status_Notification_ProviderAsset_Project_Status_Notification_Consumer_InterfaceInternal.java
      Asset_Project_Status_Notification_ProviderAsset_Project_Status_Notification_Provider_InterfaceInternal.java
      Asset_Project_Update_ProviderAsset_Project_Update_Consumer_InterfaceInternal.java
      Asset_Project_Update_ProviderAsset_Project_Update_Provider_InterfaceInternal.java
      package-info.java
    gov.gsa.fmea.Asset_Project_Query_Transaction_Manager
```

```
public class Asset_Completion_Establishment_ConsumerAsset_Completion_Establishment_Provider_InterfaceInternal {
… …
static public Document establish_asset_completion(Document request) throws CheckedException {
……
return
gov.gsa.fmea.Asset_Record_Manager.
Asset_Completion_Establishment_ProviderAsset_Completion_Establishment_Provider_InterfaceInternal.
establish_asset_completion(request);
}
```

Putting it all together


: Asset Project Status Notification Provider
<<delegate>>
<<Service Component>>
: Asset Project Status Notification Transaction Manager
: Asset Completion Establishment Consumer
: Asset Disposal Notification Consumer
<<delegate>>
<<delegate>>
: Asset Completion Establishment Consumer
: Asset Disposal Notification Consumer

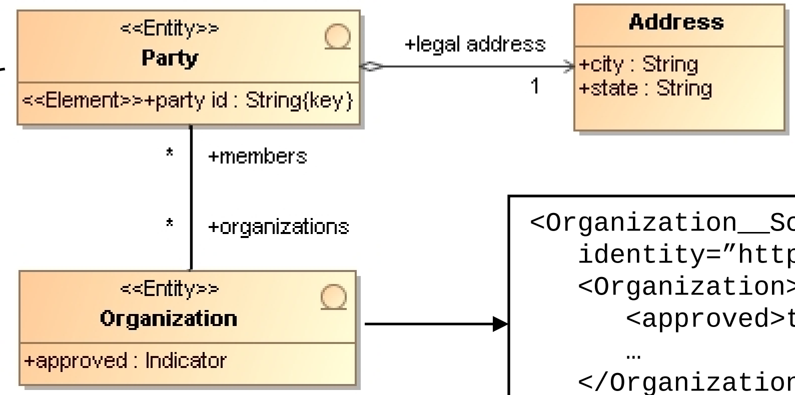
s_Notification_Transaction_Manager.Asset_Project_Status_Notification_Provider.notify_asset_project_status..xslt

```
h: send results to outgoing operations-->
:t="mdf:Asset_Project_Status_Notification_Transaction_Manager.Asset_Completion_Establishment_Consumer.establish_asset_completion.UsageOut.codehole($chan
```

Asset_Project_Status_Notification_Transaction_Manager.Asset_Completion_Establishment_Consumer.establish_asset_completion..xslt

```
… …
 <xsl:function name="mdf:Asset_Project_Status_Notification_Transaction_Manager.Asset_Completion_Establishment_Consumer.establish_asset_completion
                as="node() ?">
     <xsl:param name="documentFragment" as="node() ?"/>
     <xsl:variable name="document.out">
        <xsl:apply-templates select="$documentFragment" mode="mdf.schema.copy">
           <xsl:with-param name="namespace"
                           select="'http://www.modeldriven.org/xsd/FMEA_Asset_Accounting_Implementation_Model.uml/Asset_Completion_Establishment
        </xsl:apply-templates>
     </xsl:variable>
     <xsl:sequence select="Asset_Completion_Establishment_ConsumerAsset_Completion_Establishment_Provider_InterfaceInternal:establish_asset_comp
  </xsl:function>
… …
```

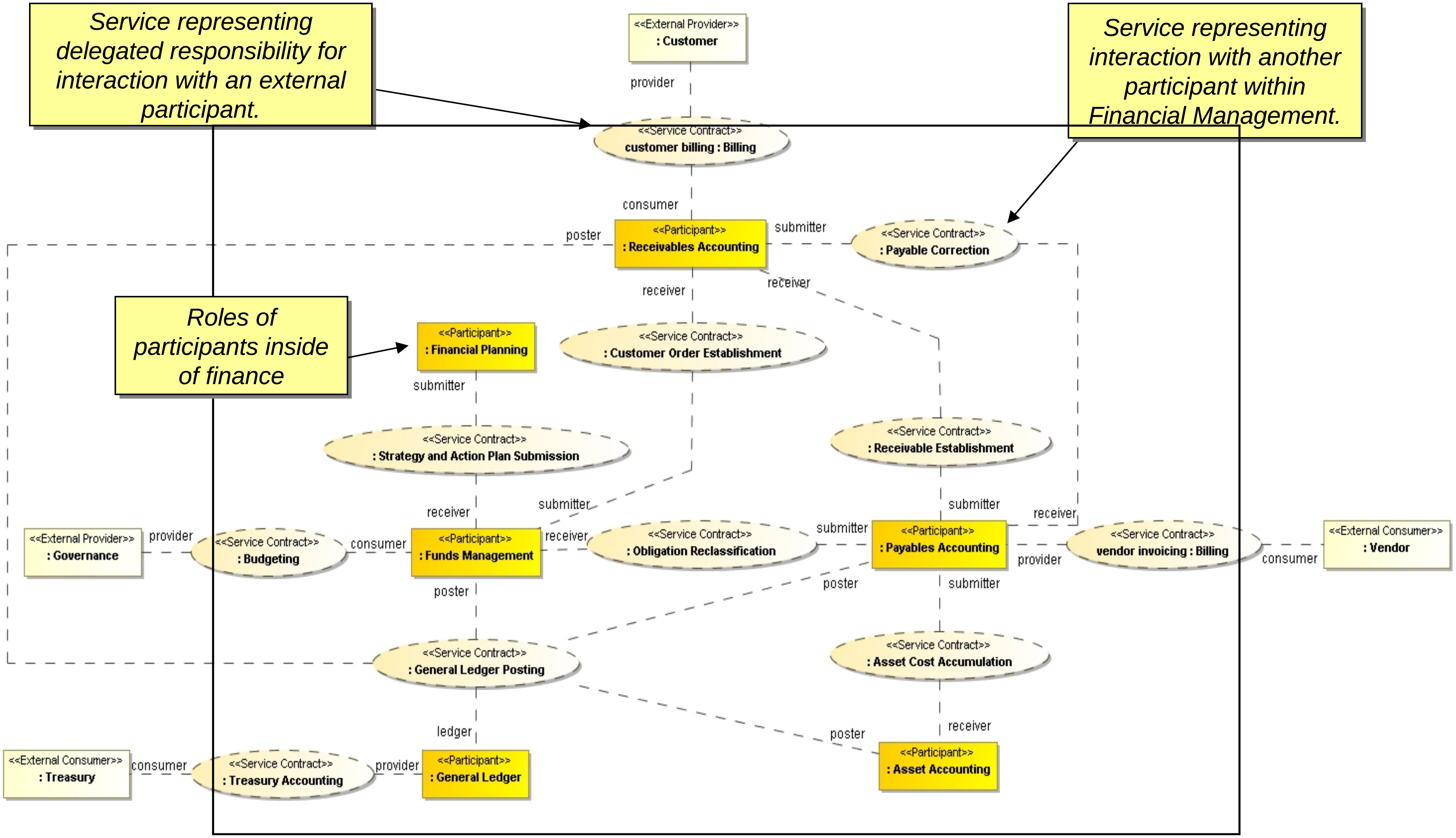
Default Mapping – Classes



```
<Organization__Schema
   identity=”http://ocfo.gsa.gov/fmea/organization/1”>
   <Organization>
      <approved>true</approved>
      …
   </Organization>
</Party__Schema>
```

```
<Party__Schema identity=”URI”>
   <Party>
      <party_id>ABC123<party_id>
         <legal_address>
            <Address>
               <city>Vienna</city>
               <state>VA</state>
            </Address>
         </legal_address>
         <organizations>
            <Organization__Item identity=”http://ocfo.gsa.gov/fmea/organization/1”/>
            <Organization__Item identity=”…”/>
            …
         </organizations>
   </Party>
</Party__Schema>
```

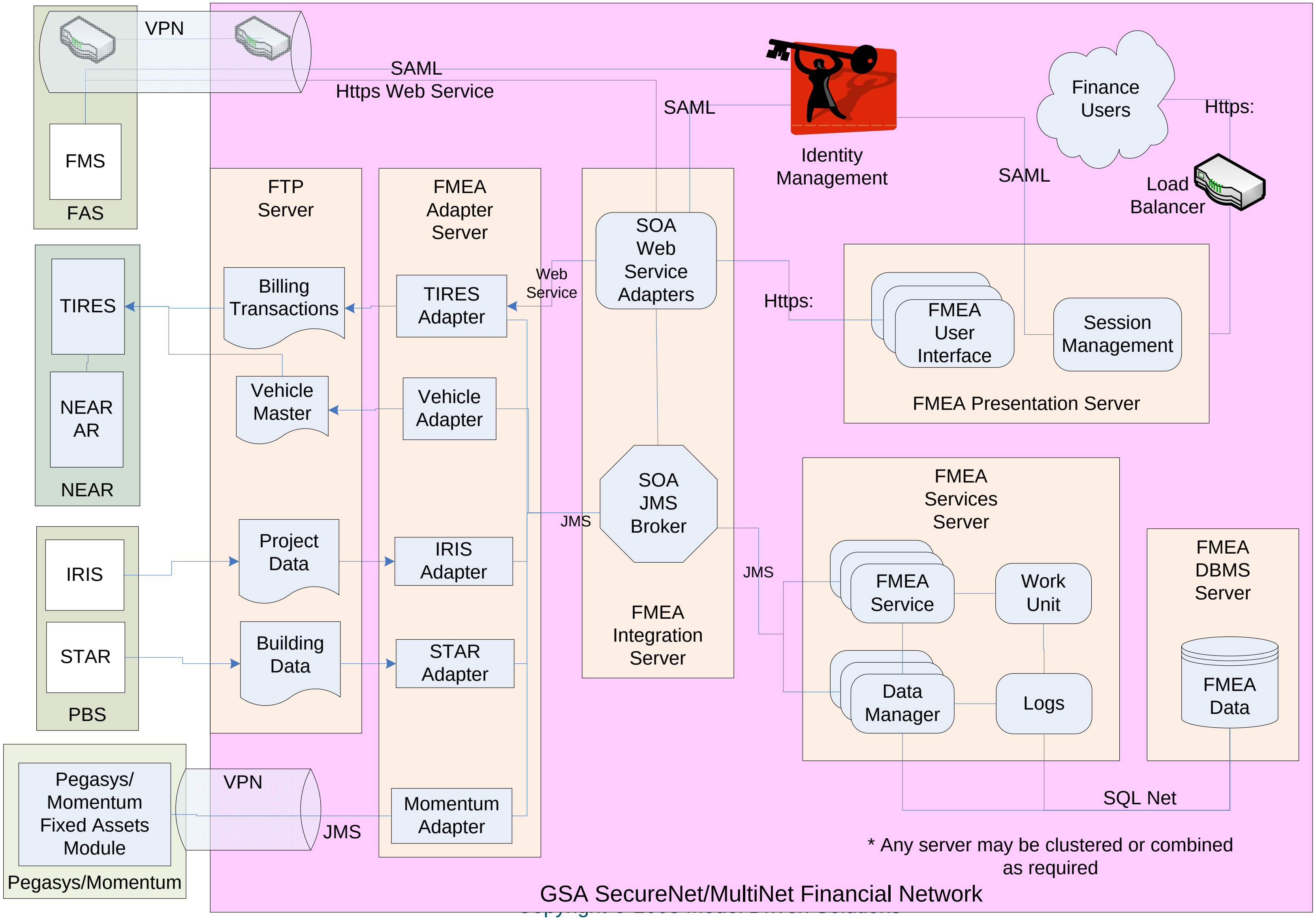
End result – this executes


Service representing delegated responsibility for interaction with an external participant.

Service representing interaction with another participant within Financial Management.

Roles of participants inside of finance

On this infrastructure



Late breaking news

- 2/3 of the way through this process, JSA decides to make a commitment to JBI on Sun Glassfish
- For 10% for funds, the same application is provisioned to a new technical architecture
- Change takes about 6 weeks, now components can be deployed to either infrastructure with no change.
- Most of the time is spent just debugging glassfish
- How would this change have been possible without MDA?

Conclusion

- FMEA is a general architecture of the federal financial services domain, done for GSA by Model Driven Solutions.
- It supports both internal GSA needs as well as the “line of business”.
- It uses MDA, SOA and BPM to provide a business centric architecture, drilling down to technology models.
- Artifacts can be generated for model based acquisition, the FEA, testing, service interfaces, data management, workflow and components.
- FMEA is entering the next phase of acquisition and implementation.
- **All assets to produce this application are being donated to ModelDriven.org as open source.**