Model Driven Solutions

Where Business Meets Technology

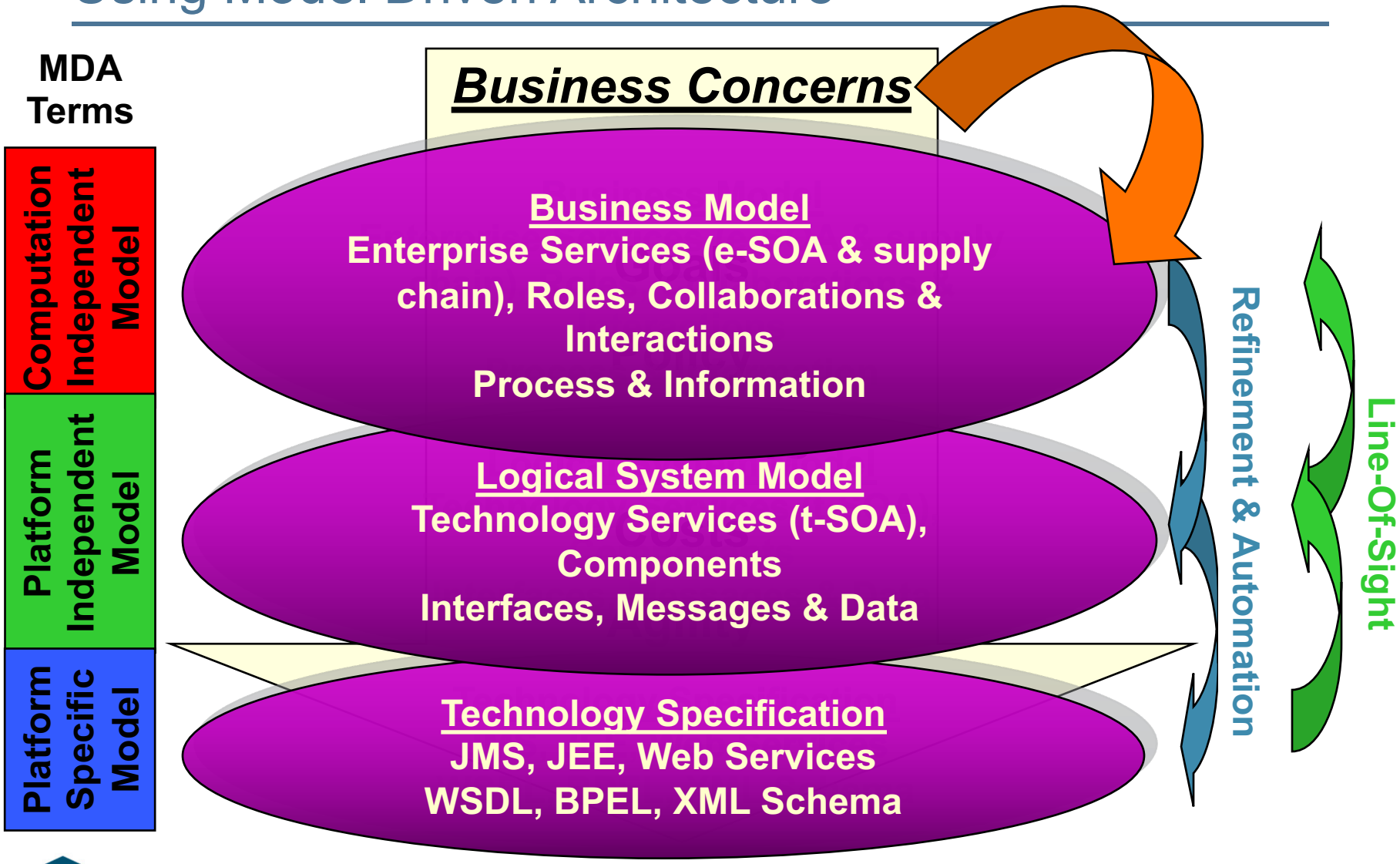
A division of Data Access Technologies, Inc.

Model Driven Service Oriented Architecture

Ed Seidewitz

07 April 2010

Business Focused Solutions
Using Model Driven Architecture



Business Model

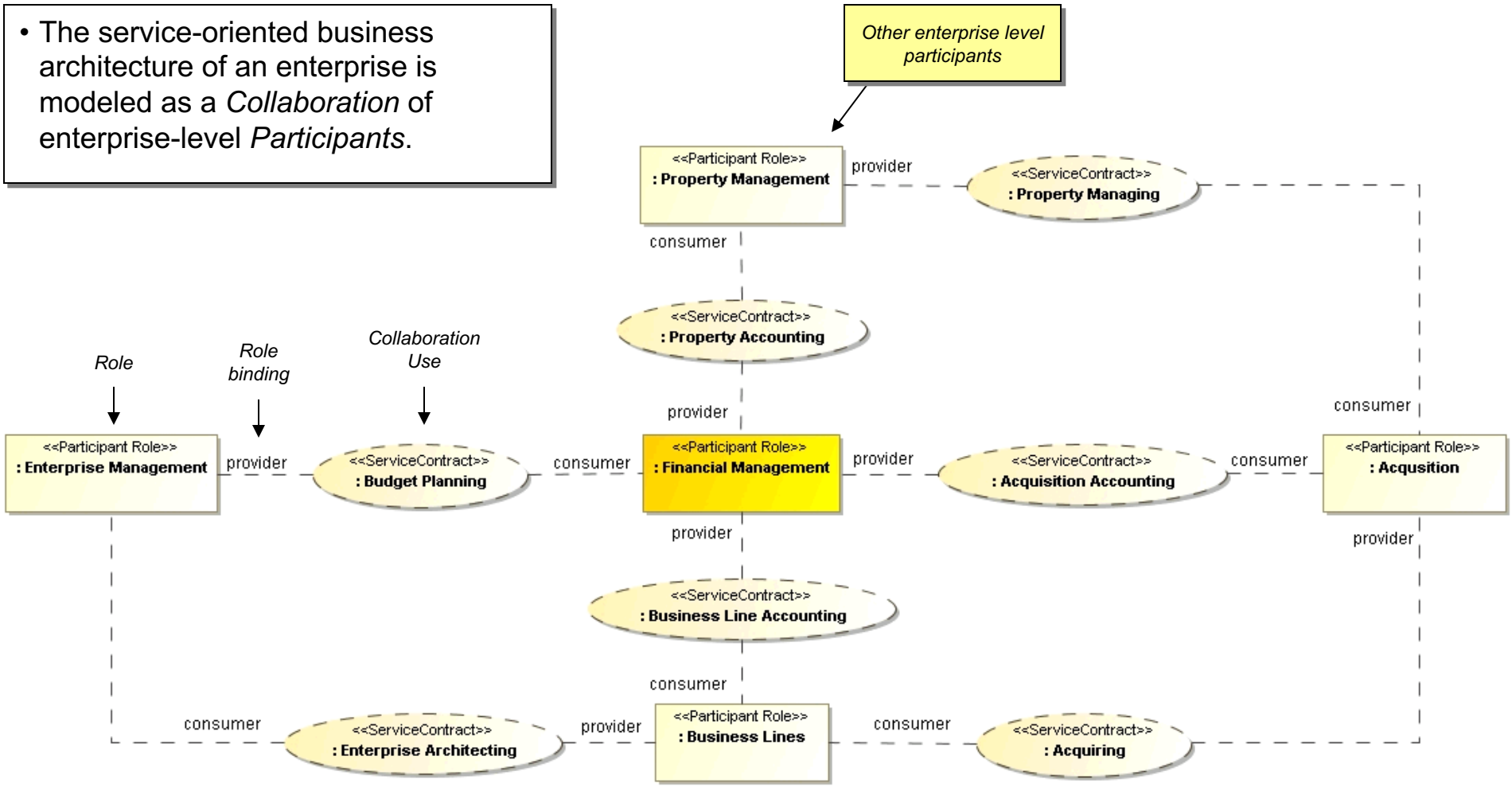
- Example: Financial Management Enterprise Architecture
- Business Architectures
- Service Contracts
- Business Processes
- Information Models

Example: Financial Management Enterprise Architecture

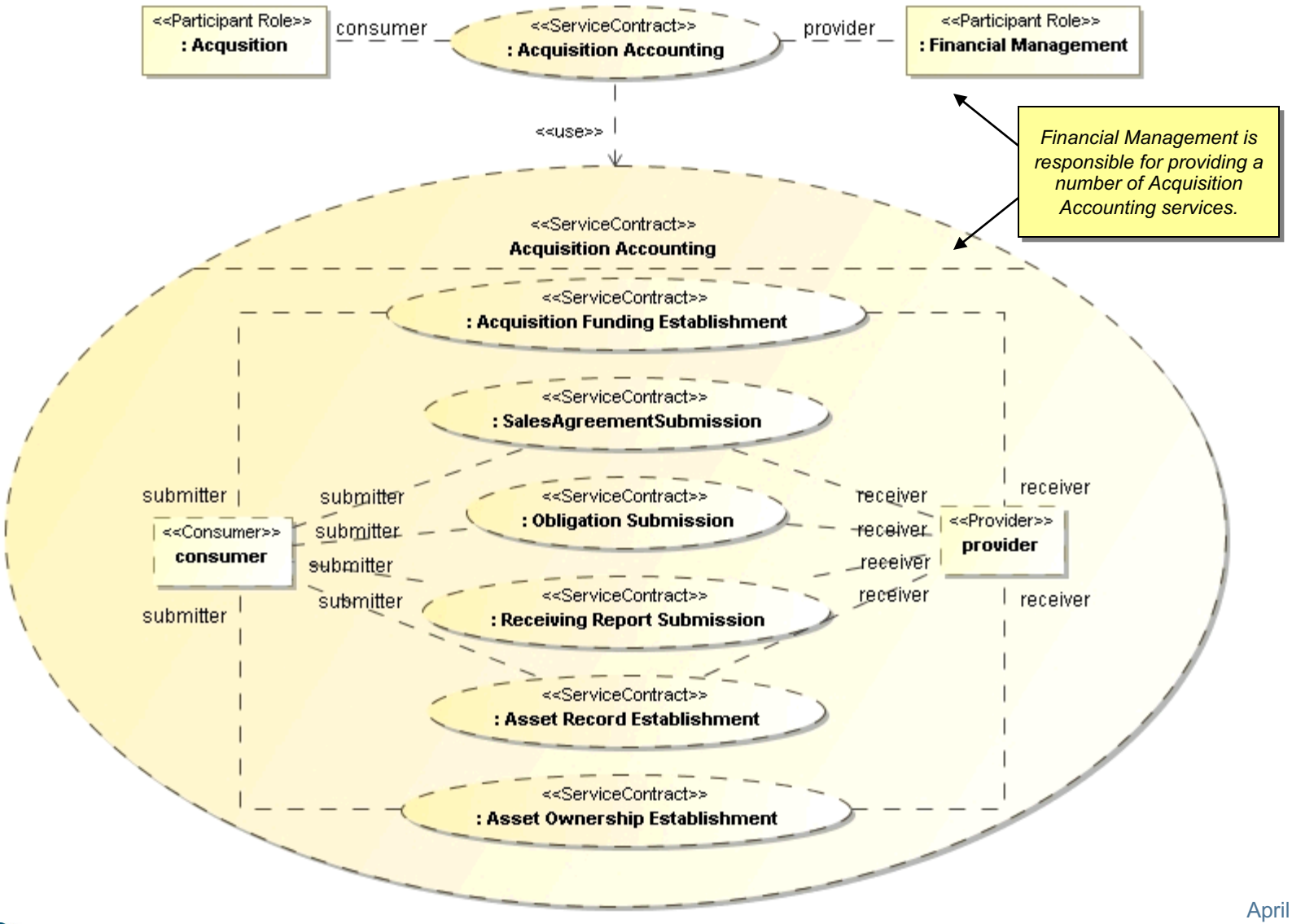
- A simplified Financial Management Enterprise Architecture for a Federal Government agency (also largely applicable to commercial financial management)
- Consistent with the Federal Financial Management Line of Business architecture
- Based on work done for the General Services Administration (GSA) that delivered:
  - A target business architecture for consistent and comprehensive financial management supporting all GSA services and staff offices.
  - A logical system architecture for a cohesive financial management suite supporting the business architecture, particularly in areas in which a transition needed to be made off legacy systems.
  - A set of interface definitions to act as the basis for a standard GSA financial management service-oriented architecture.

Financial Management Enterprise Context

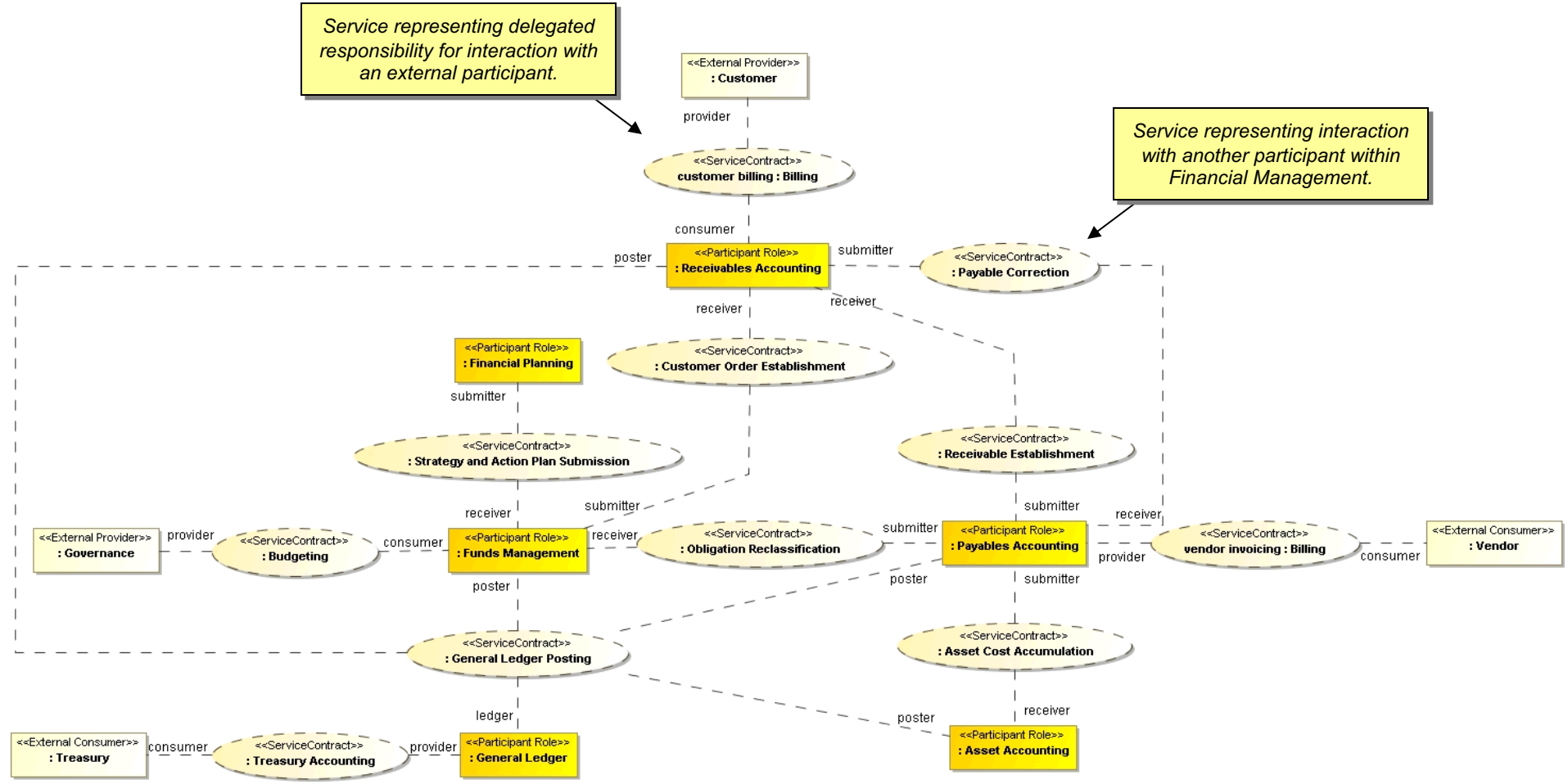
- The service-oriented business architecture of an enterprise is modeled as a *Collaboration* of enterprise-level *Participants*.



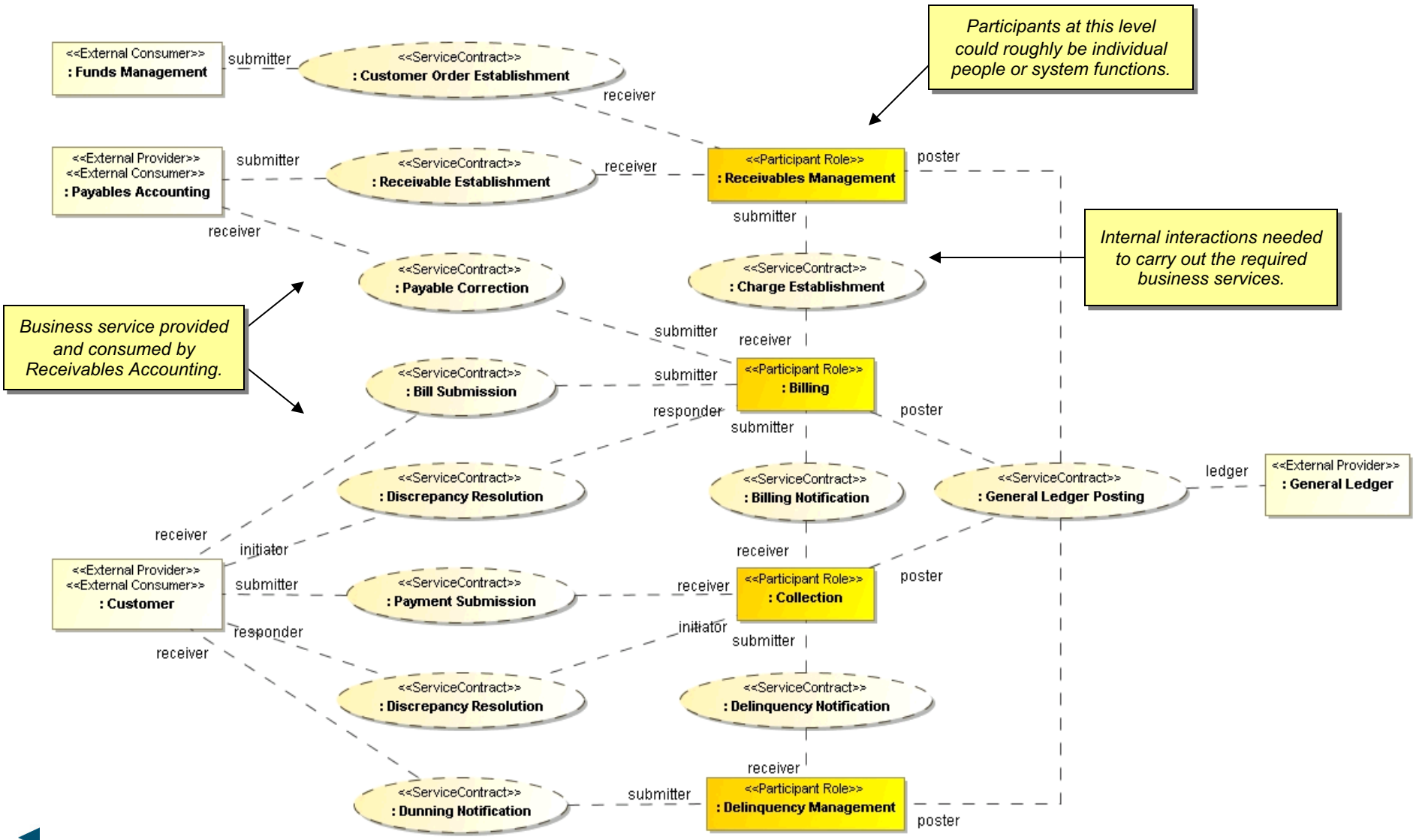
A Composite Service Contract



Financial Management Business Architecture

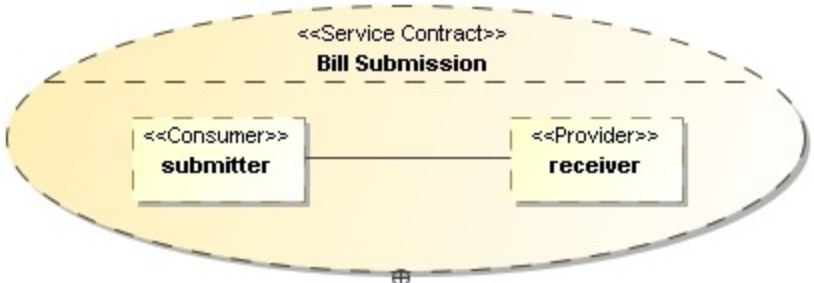


Receivables Accounting Business Architecture



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)

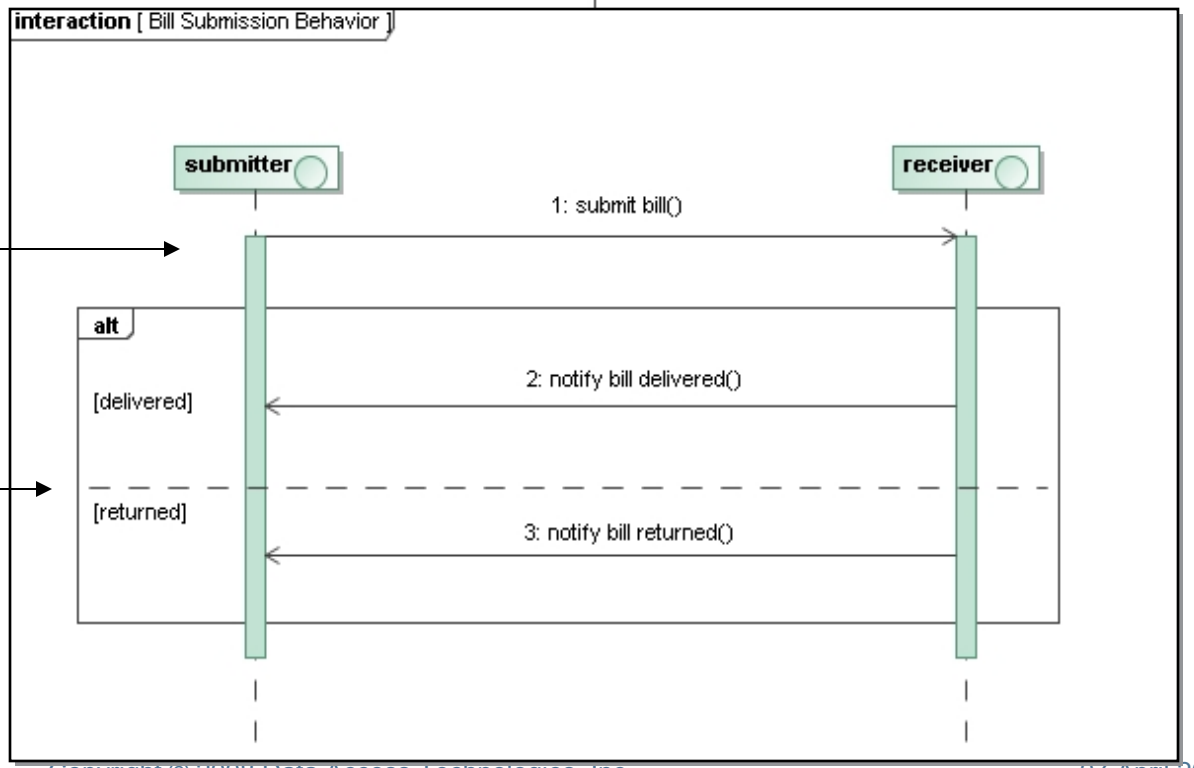


Indicates ownership

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

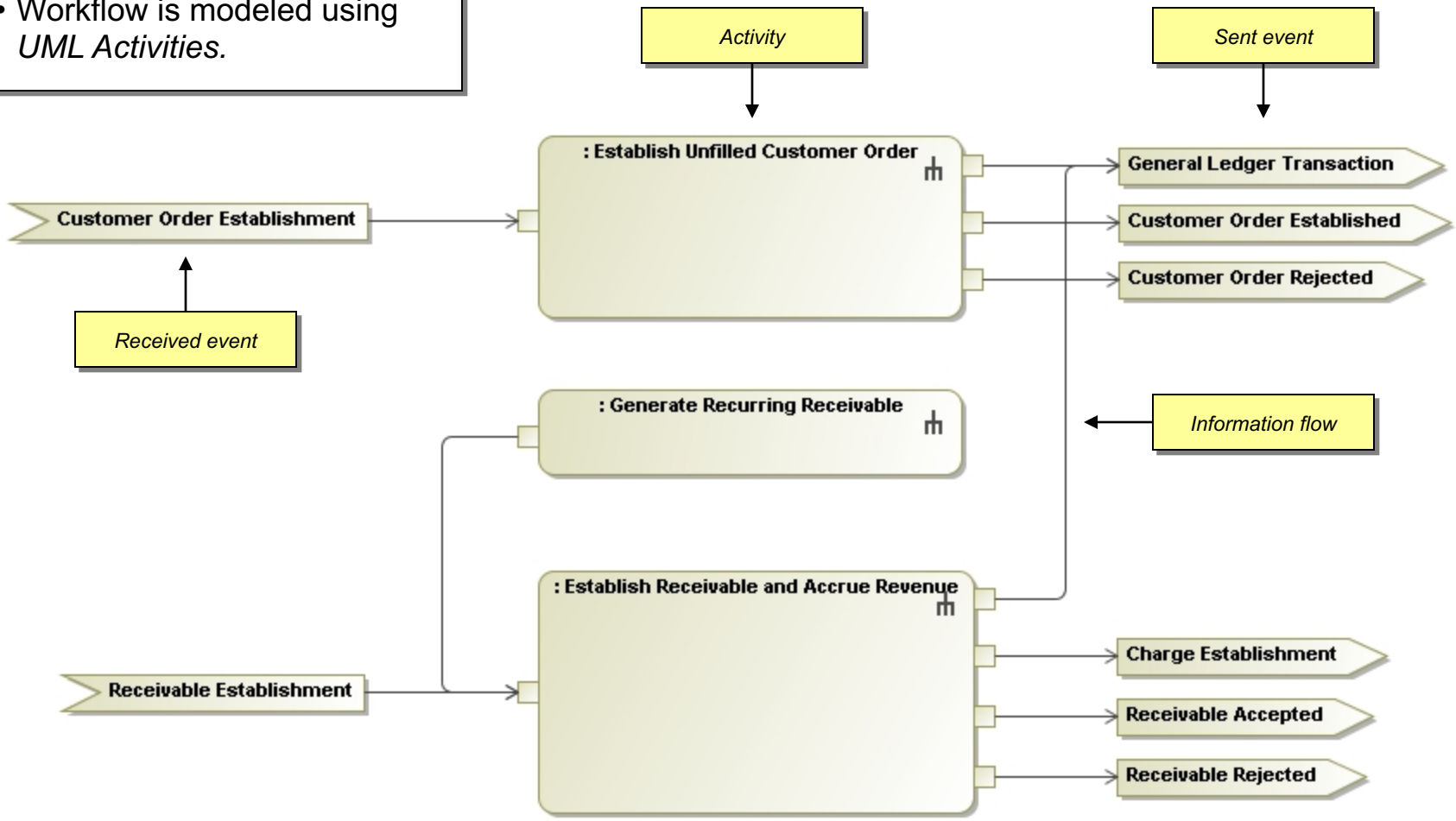
First the submitter submits a bill to the receiver…

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



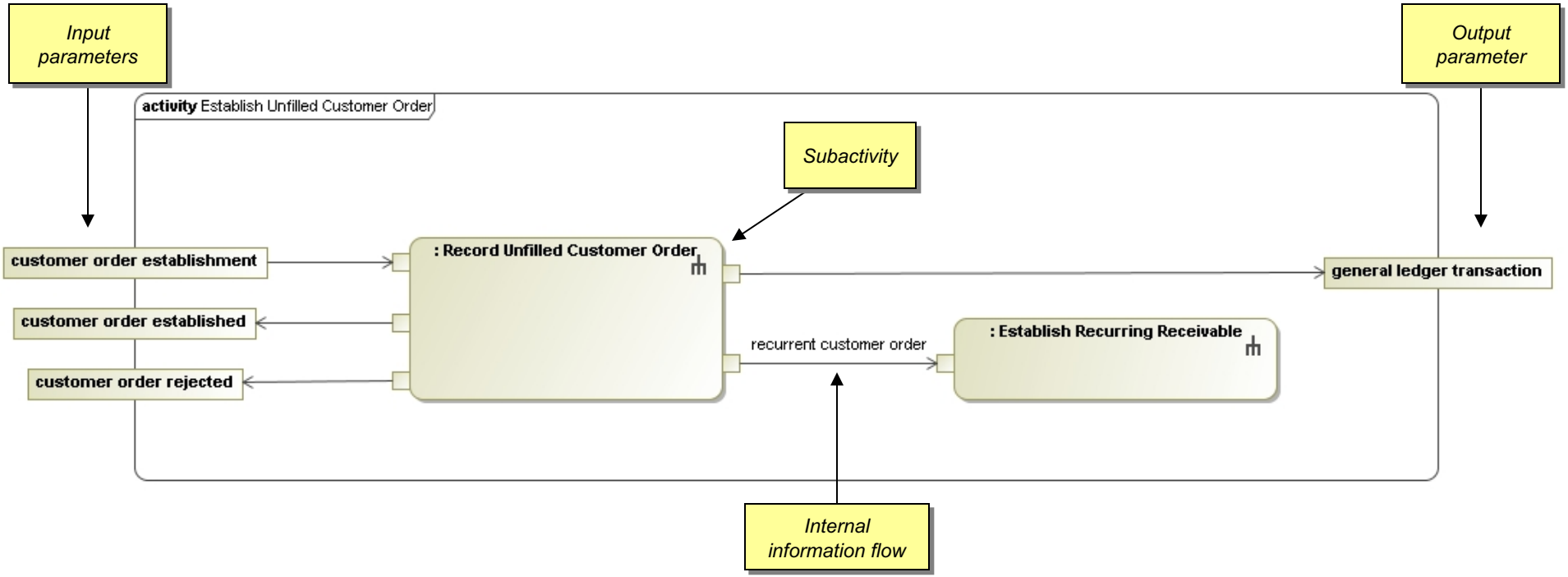
Receivables Management Activities

- Workflow is modeled using *UML Activities.*

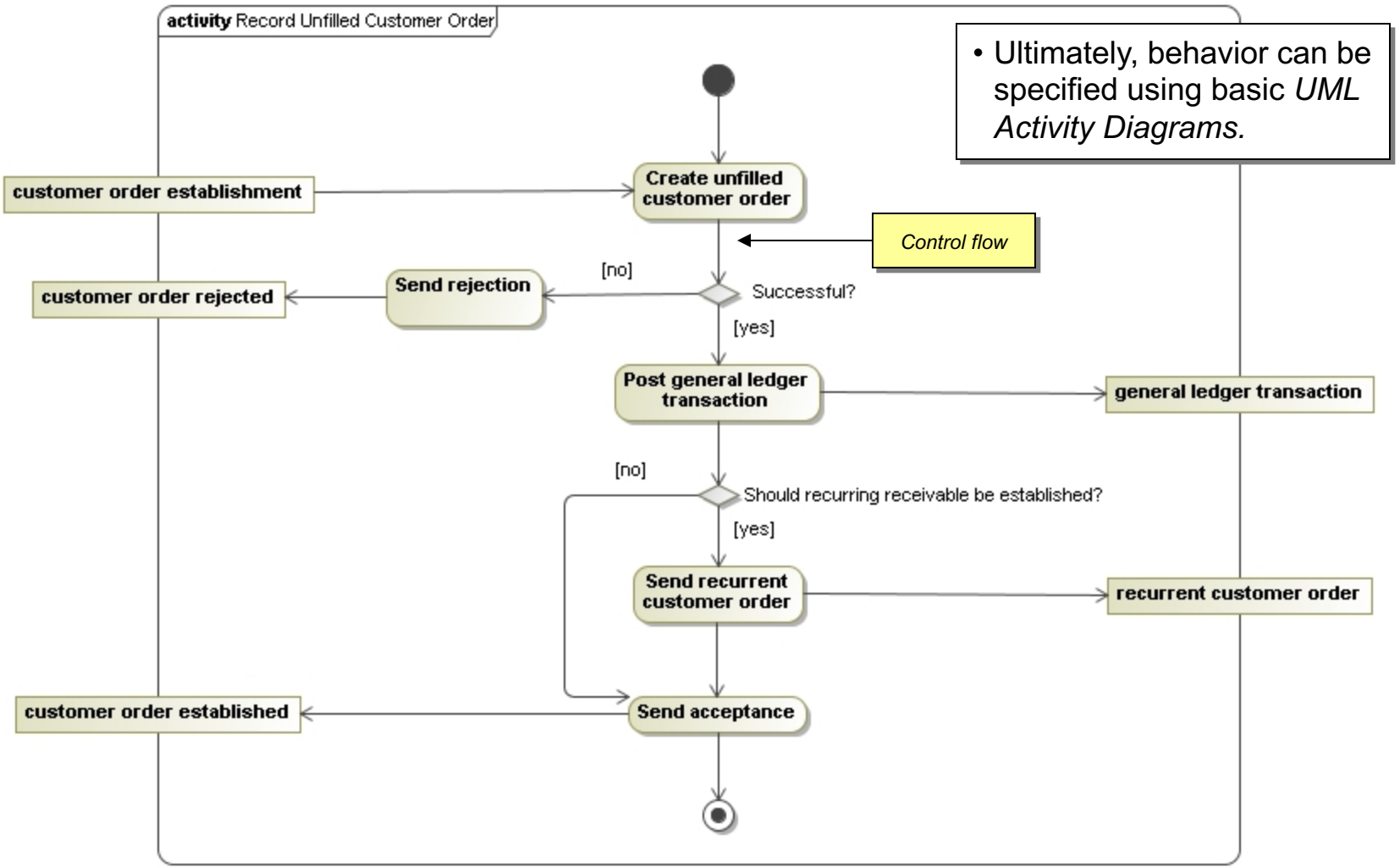


Establish Unfilled Customer Order Subactivities

- Complicated activities may be decomposed into subactivities.

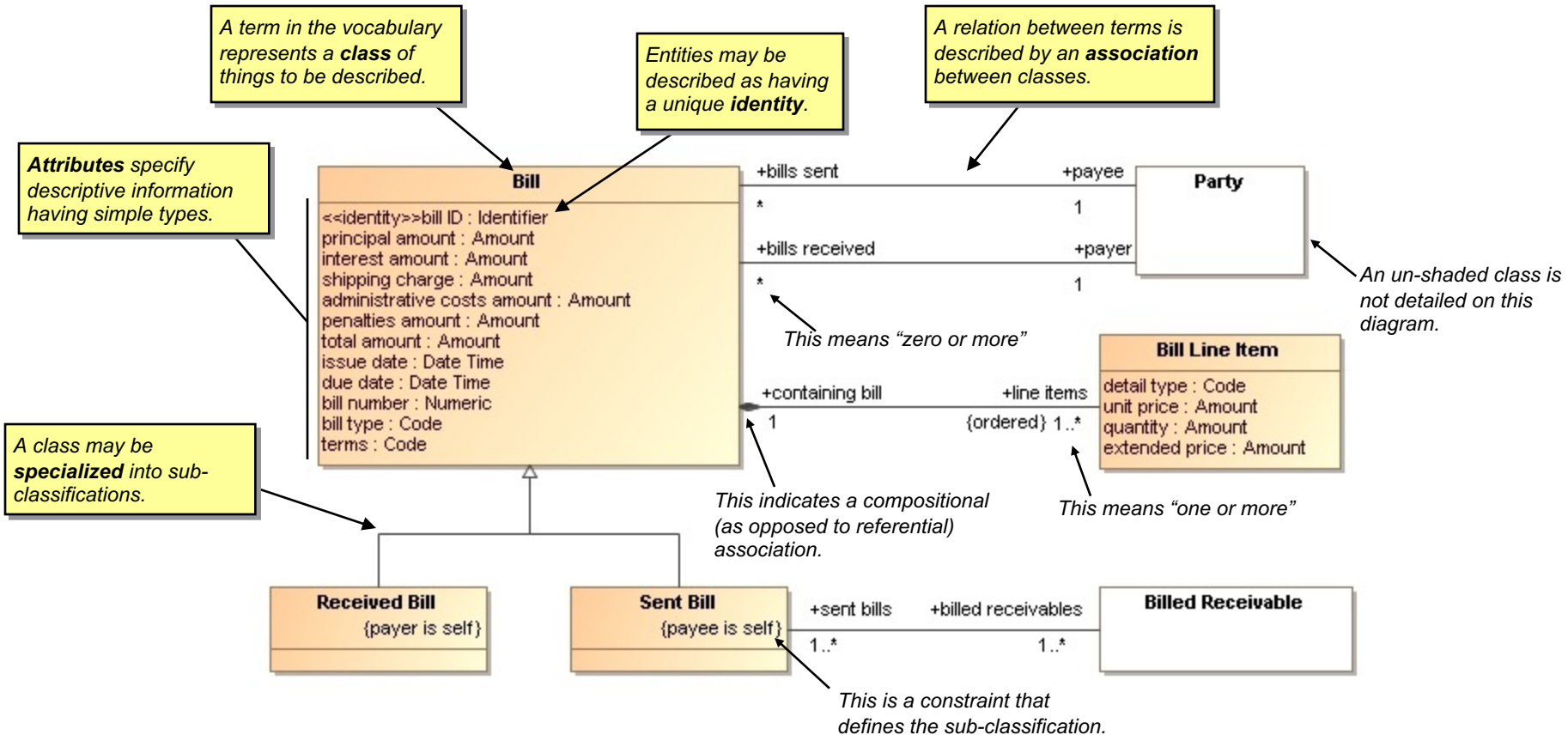


Record Unfilled Customer Order Behavior



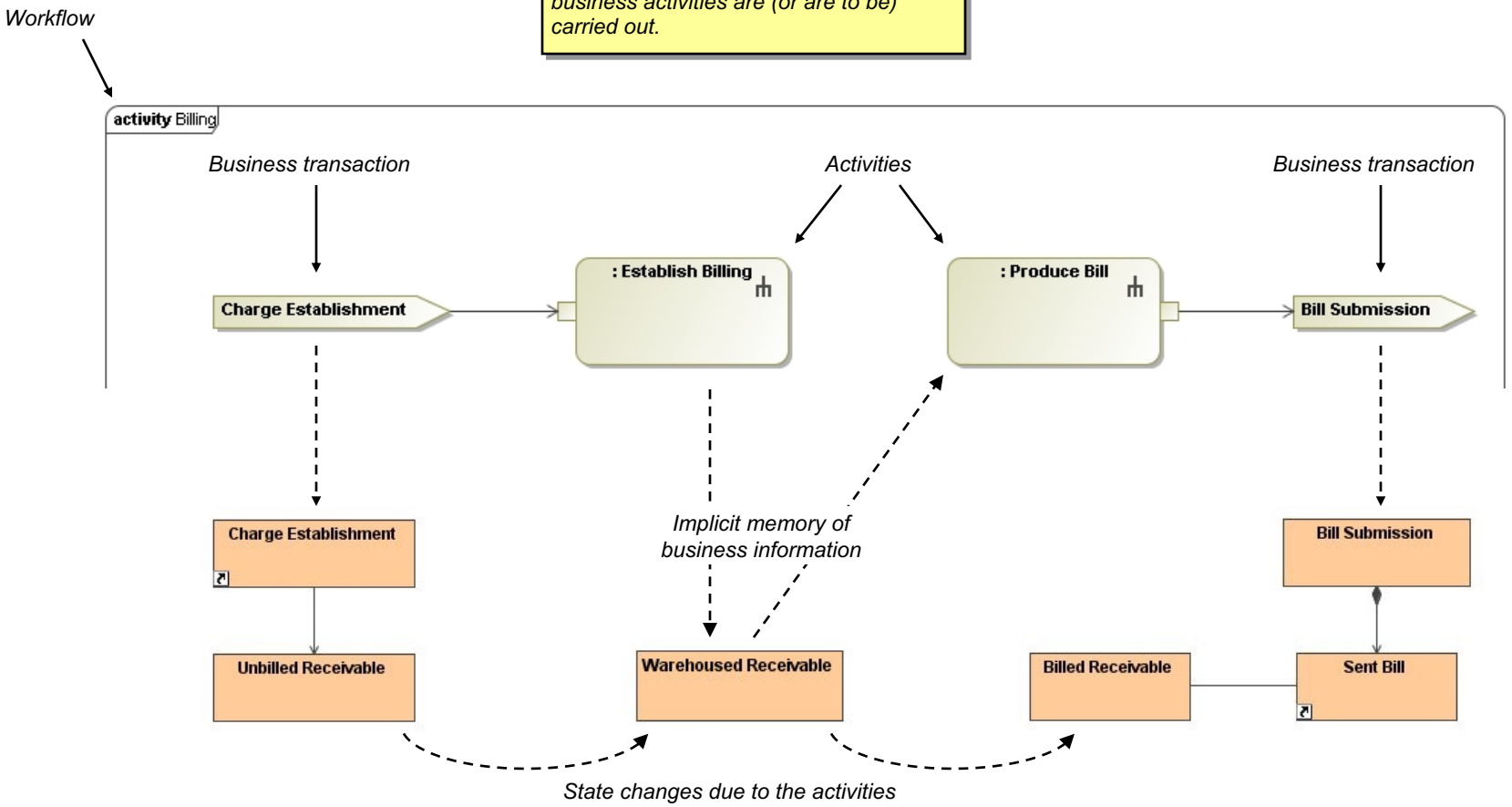
- Ultimately, behavior can be specified using basic *UML Activity Diagrams.*

Information Model



Information Model: What Is It For?

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

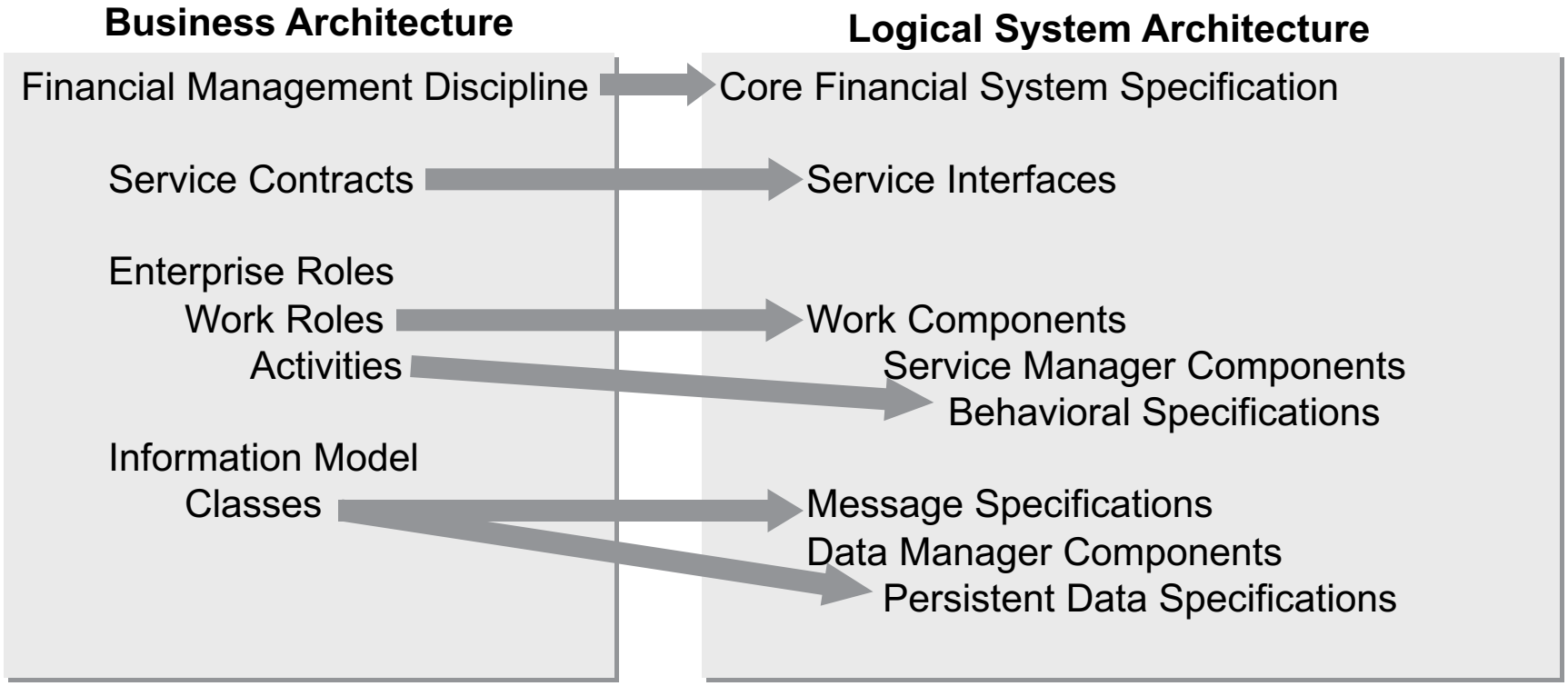


*The **information model** details the vocabulary of the business entities and transactions used in the process model.*

Logical System Architecture

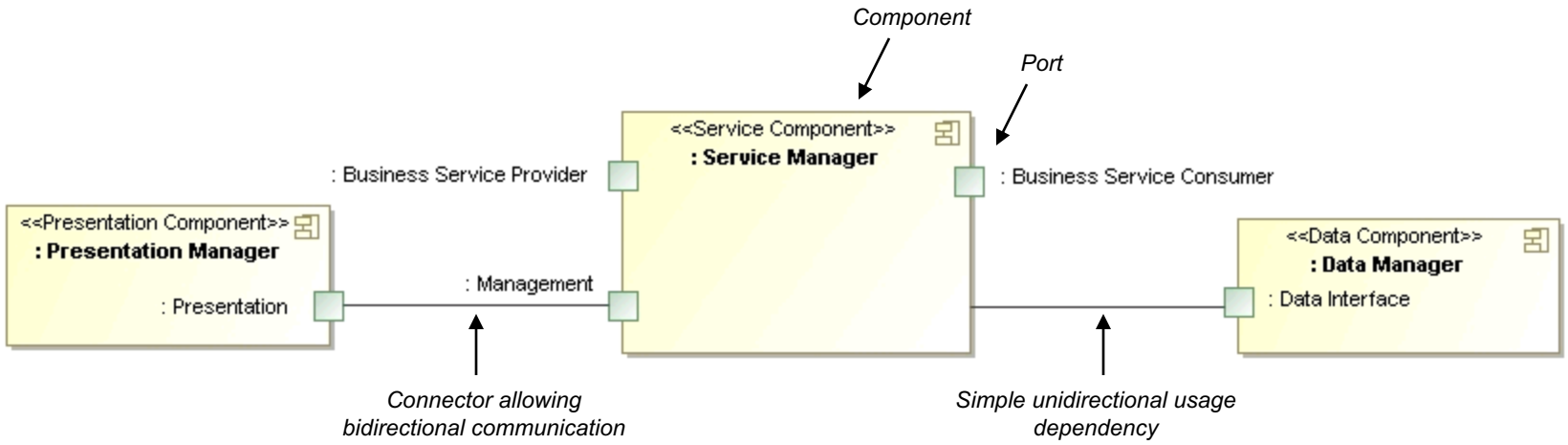
- Example: Core Financial Management System
- Component architectures
- Service Interfaces
- Functional Specifications
- Data Models

From Business Architecture to System Architecture



Three-Tier Component Architecture

- System architecture is modeled using *UML Components.*

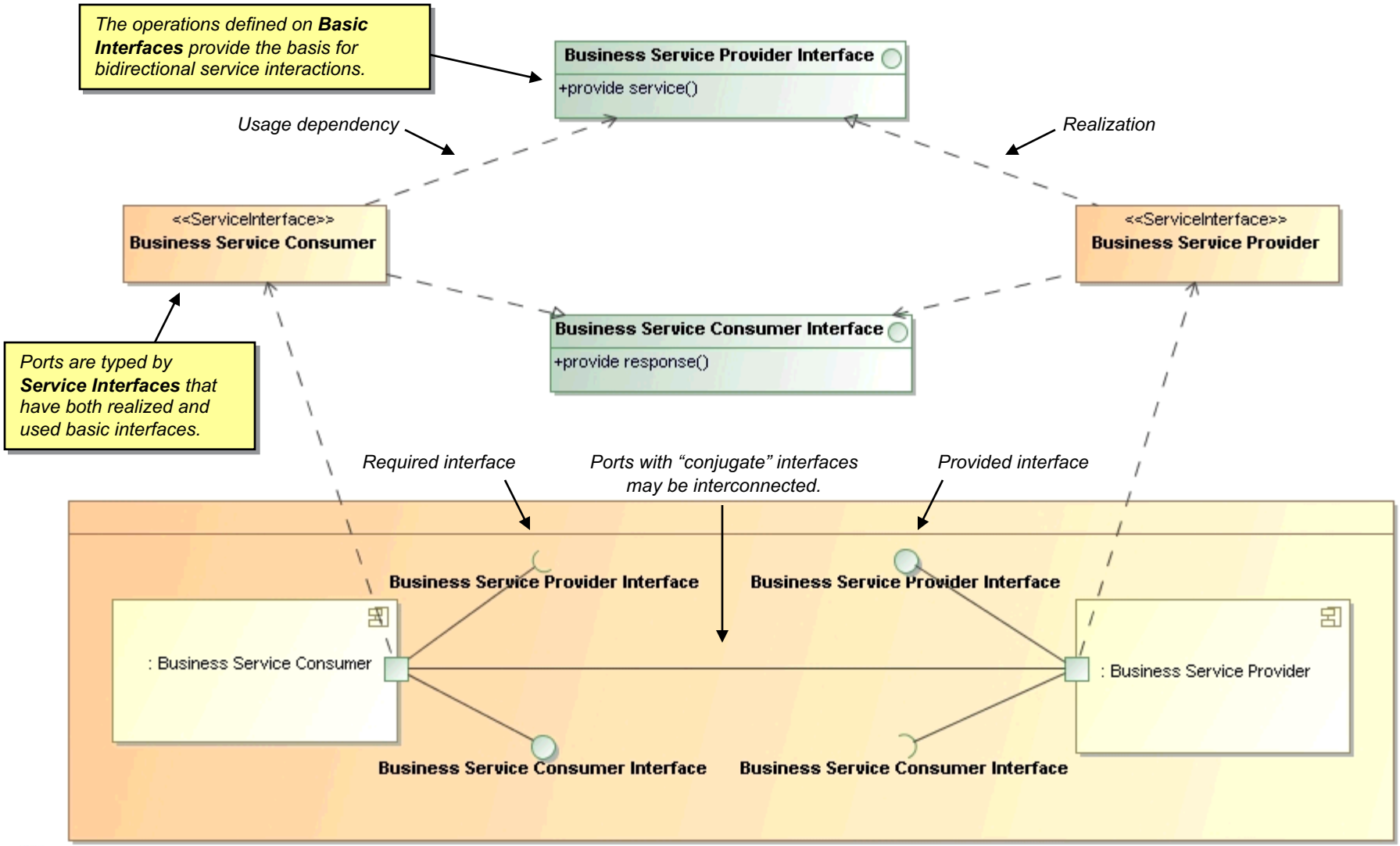


Presentation Components *provide user access to application services.*

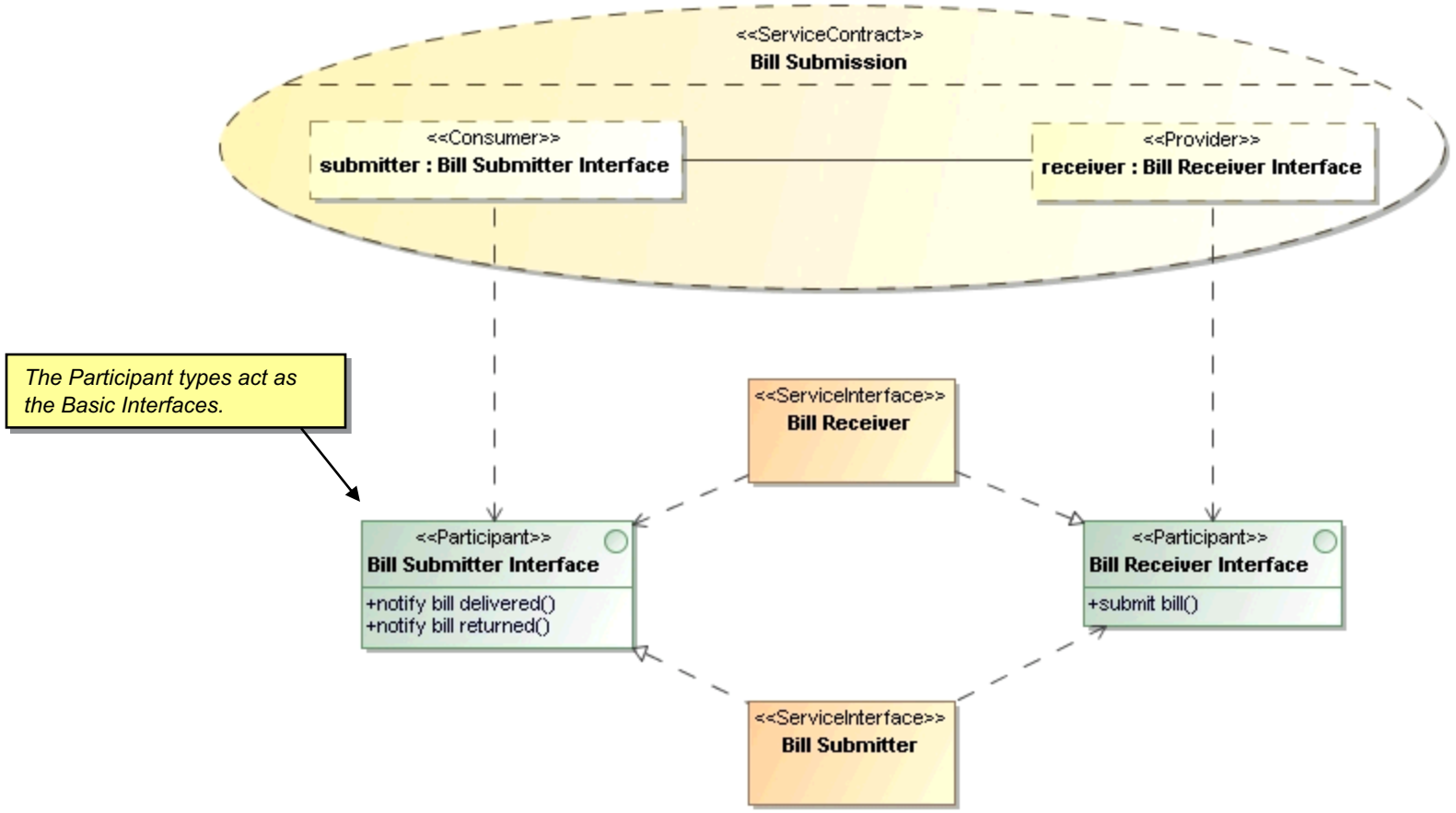
Service Components *provide transactional implementation of application services.*

Data Components *persist data between application transactions.*

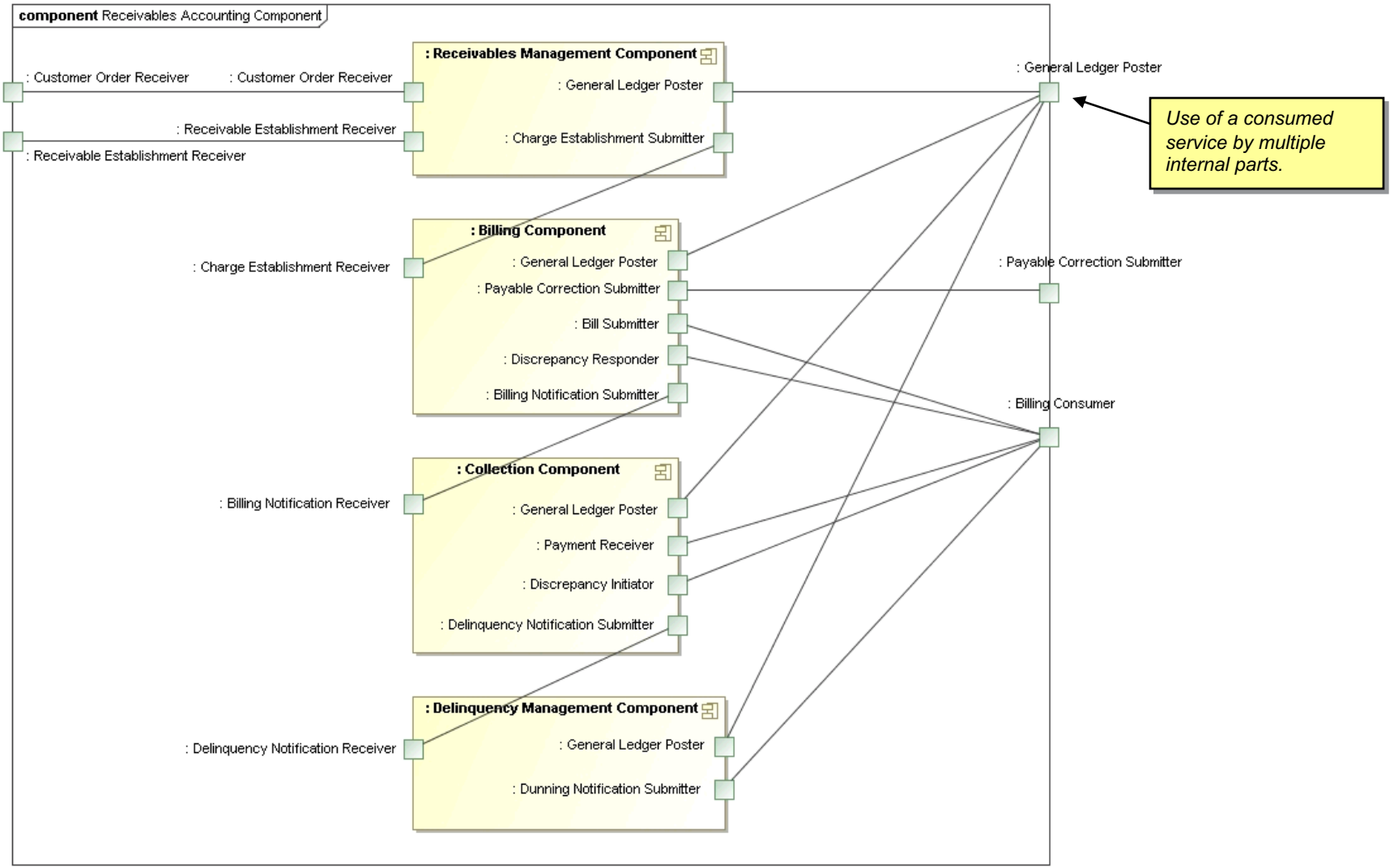
Provided and Required Interfaces



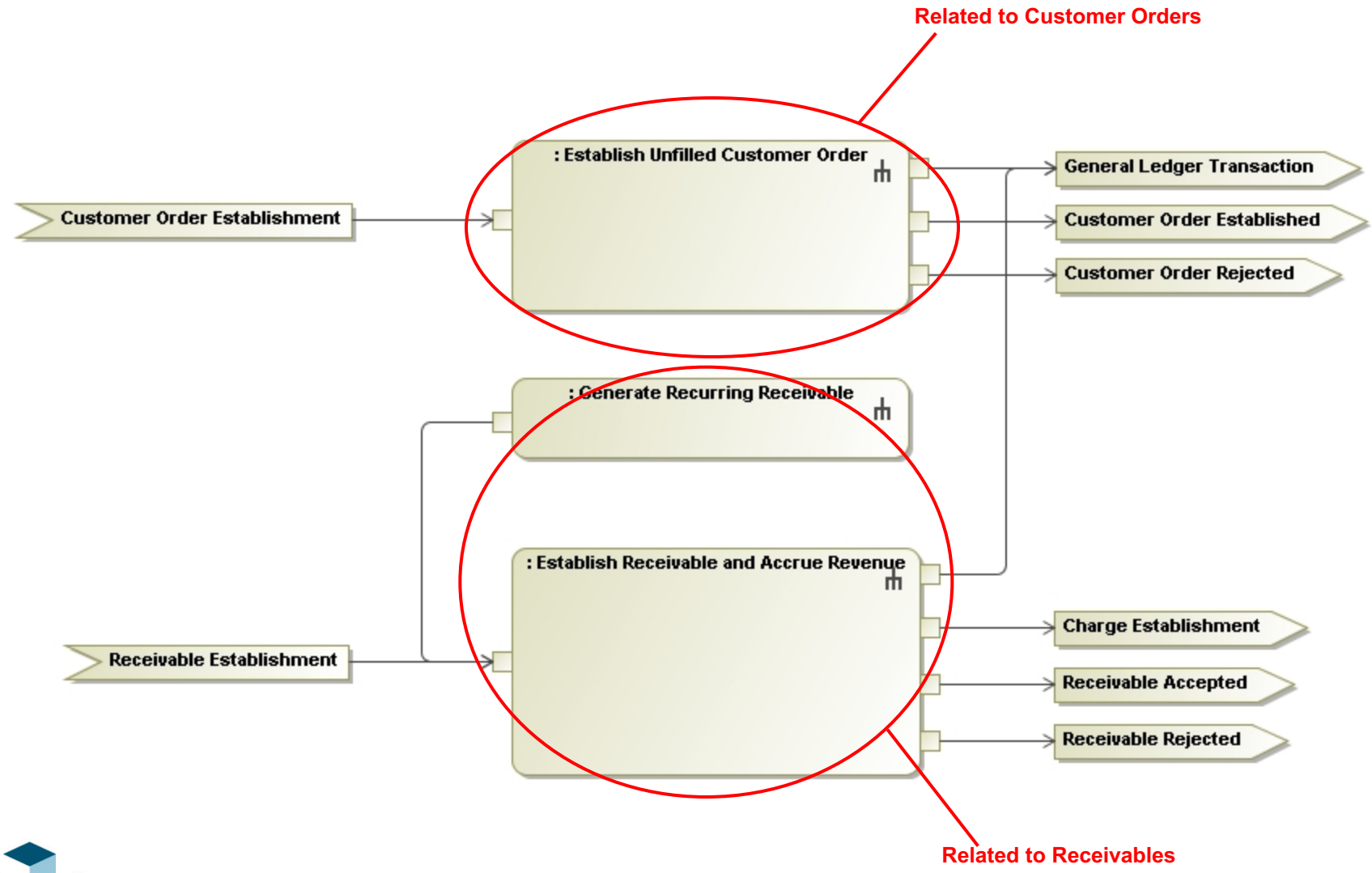
Service Interfaces from Service Contracts



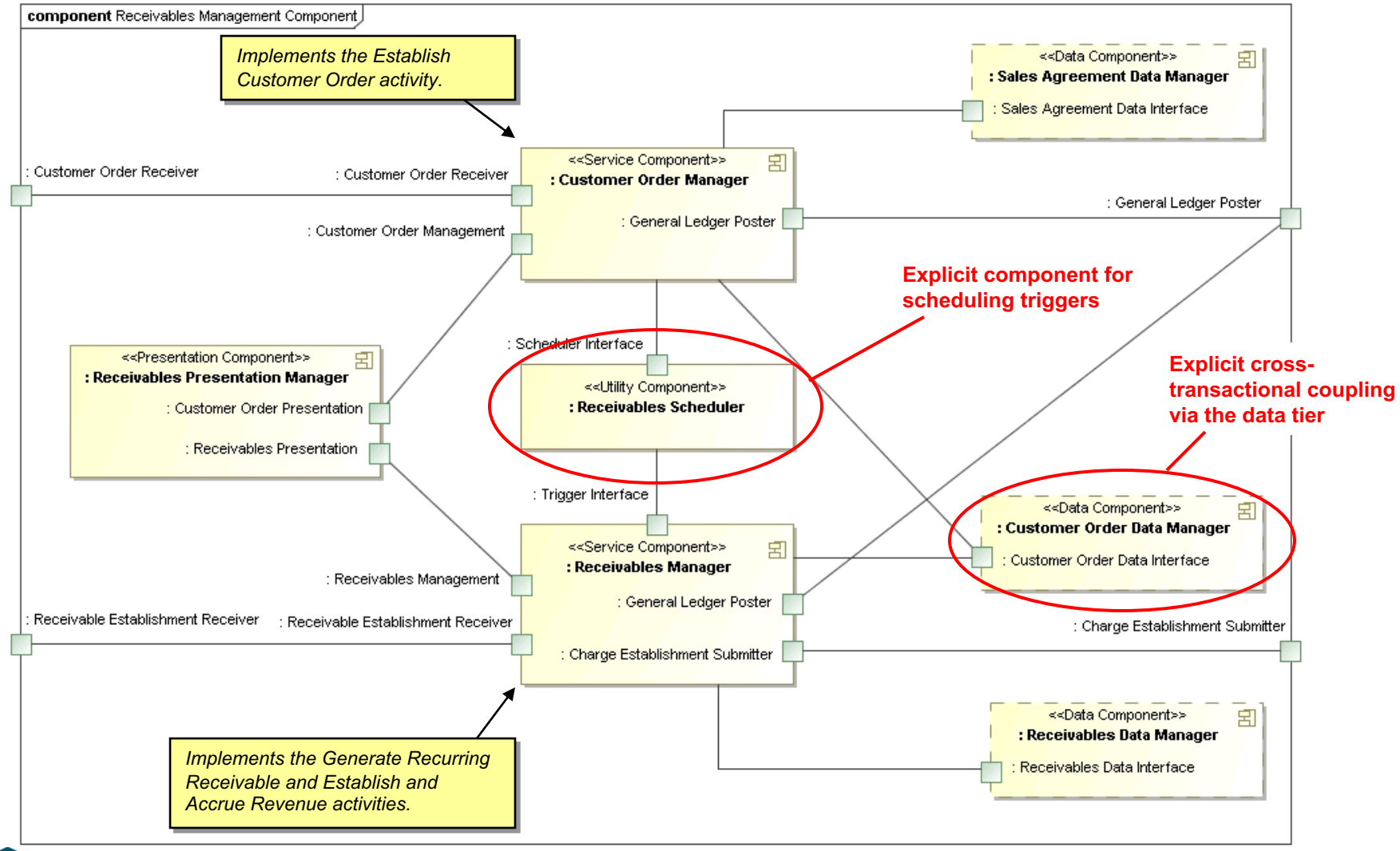
Receivables Accounting Component Architecture



Receivables Management Activities (from Business Model)



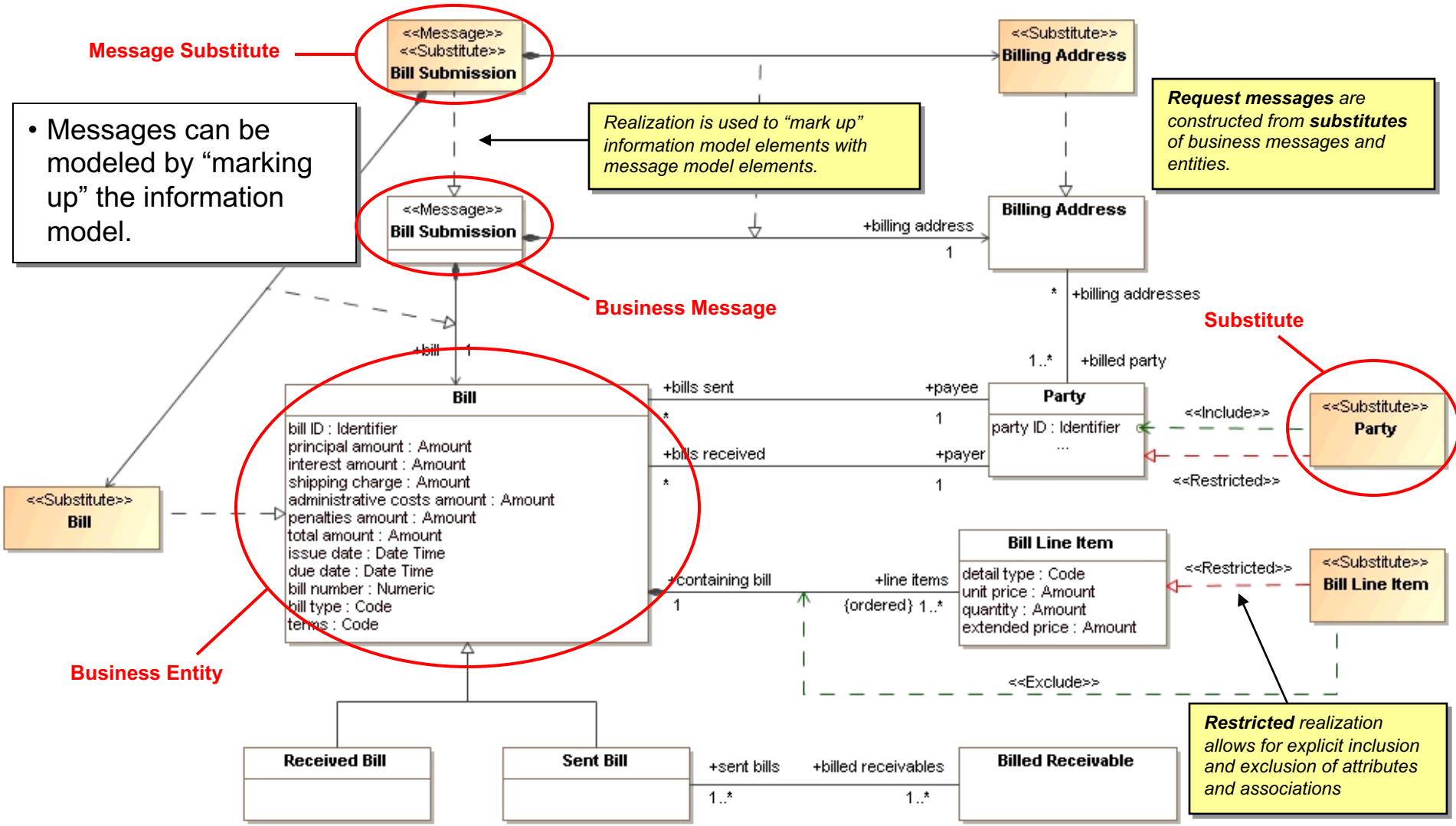
Receivables Management Component Architecture



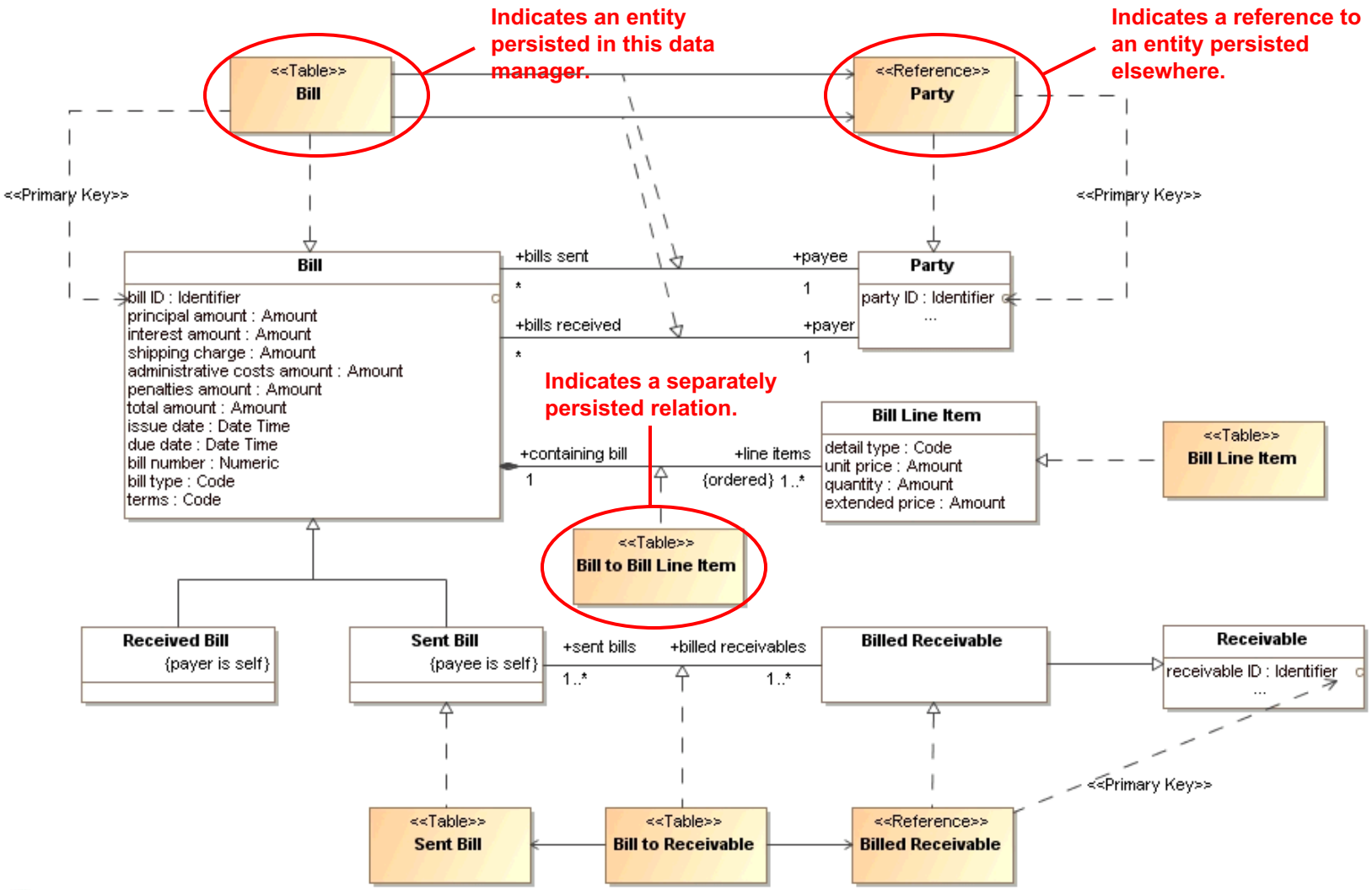
Record Unfilled Customer Order Functional Specification

1. **Receive** CustomerOrderEstablishment
2. **Let** newOrder = CreateCustomerOrder(CustomerOrderEstablishment.newOrder).data
3. **Send** GeneralLedgerTransaction to increase Unfilled Customer Orders and decrease Anticipated Reimbursements
4. **Send** newOrder as RecurrentCustomerOrder
   (**Note**: EstablishRecurringReceivables will check if there are actually any creation triggers.)
5. **Send** CustomerOrderEstablished

Example Request Message Model



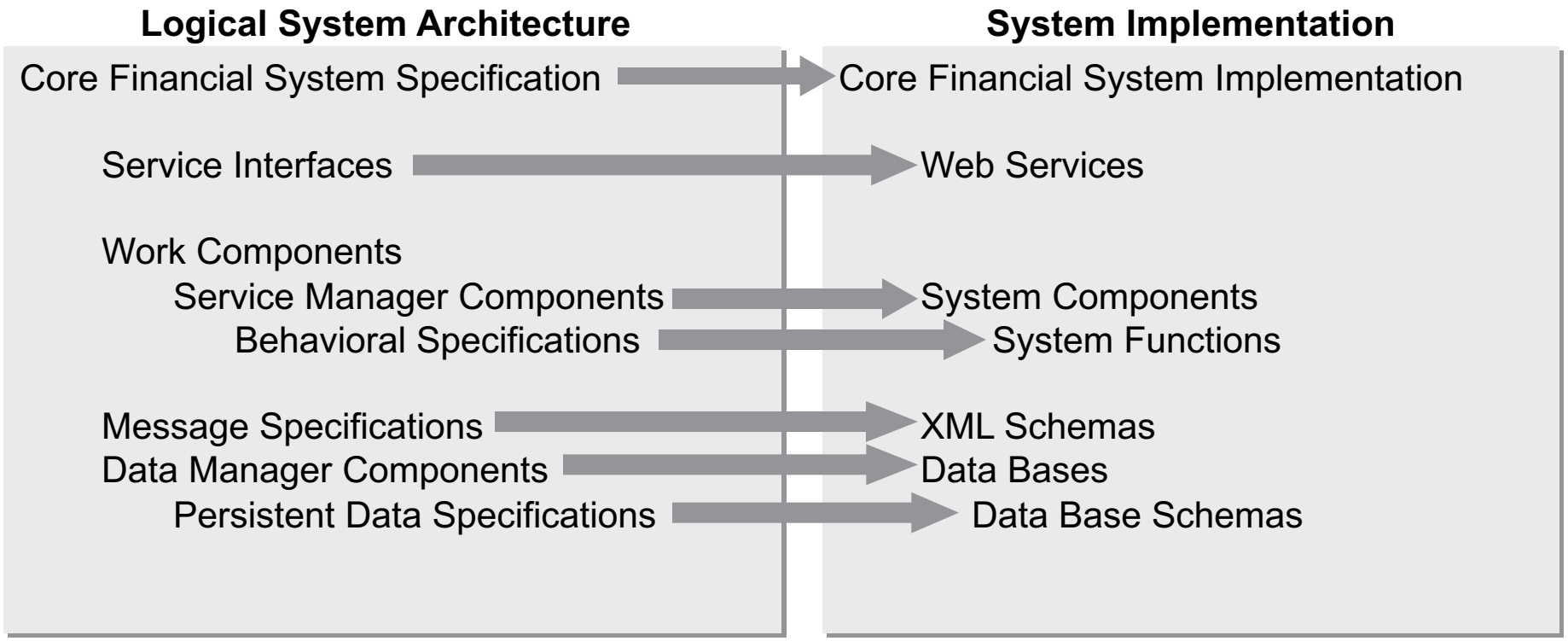
Example Persistence Model



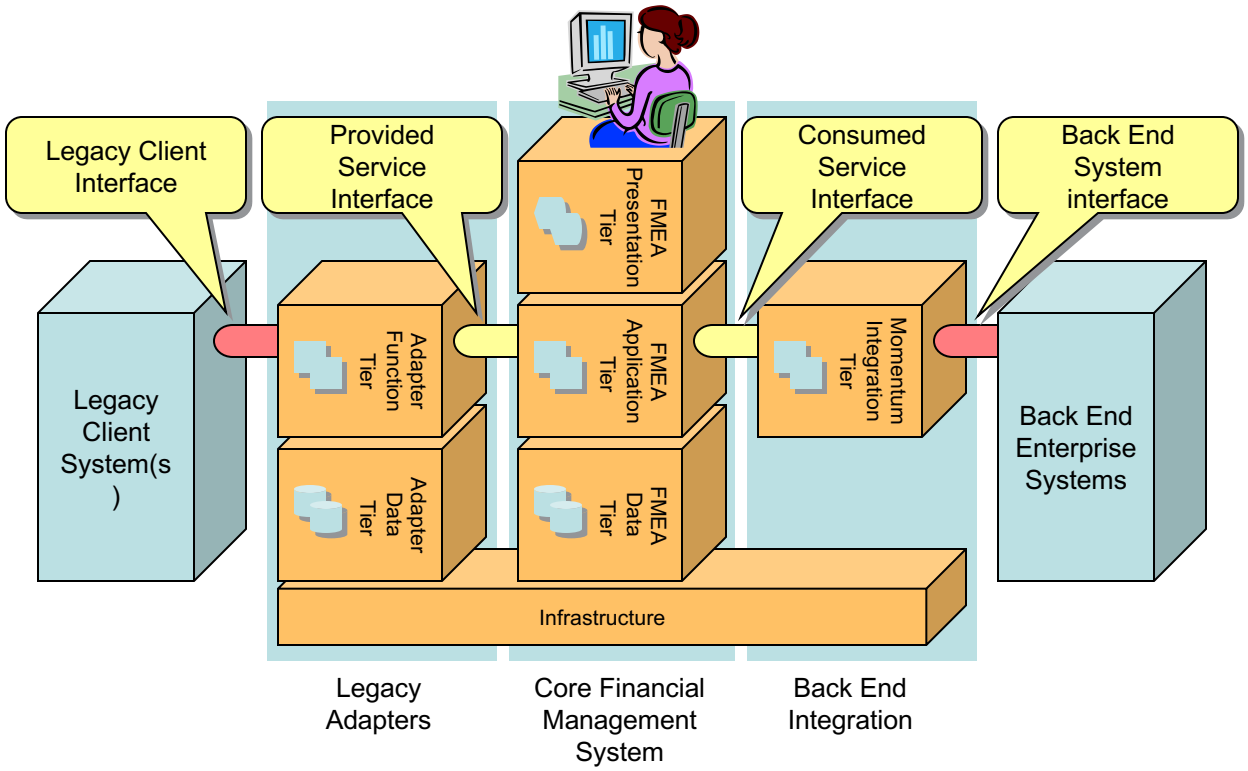
Technology Specification

- Example: Core Financial System Implementation
- Provisioning
- Web Services

From System Architecture to System Implementation



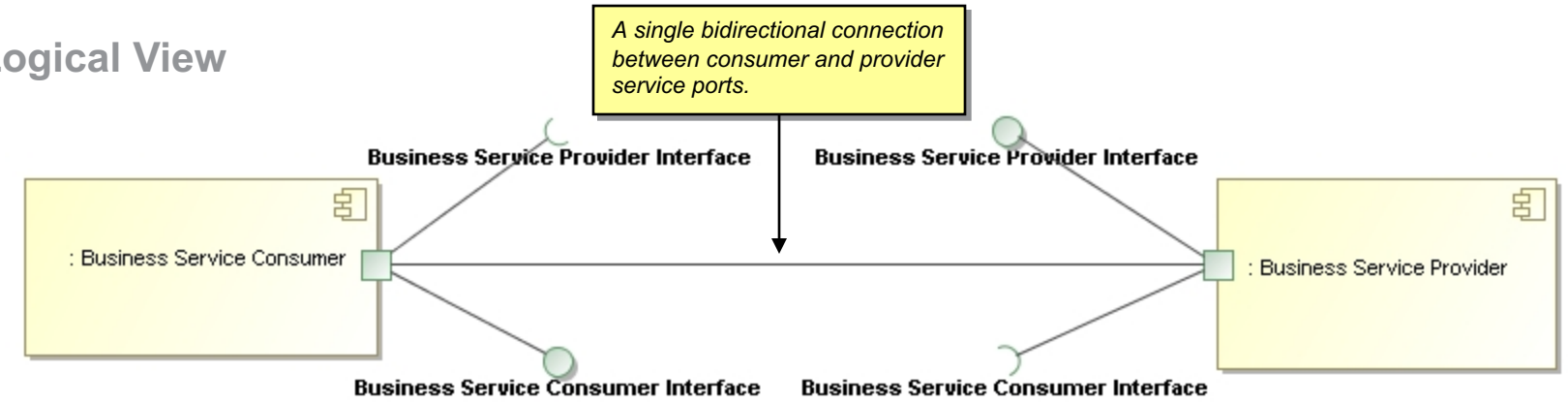
Example Implementation Architecture



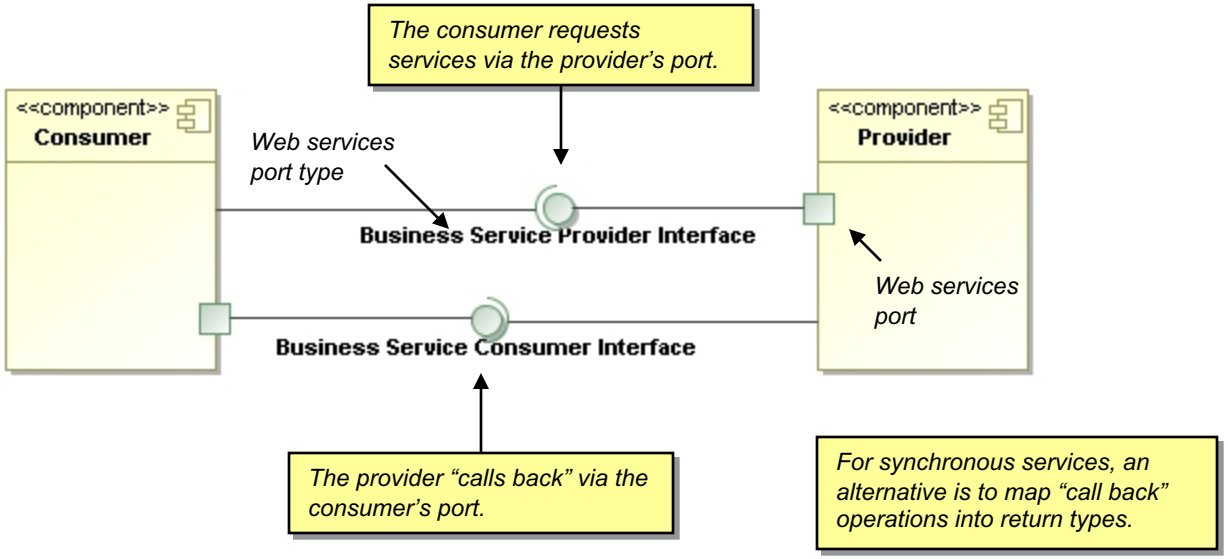
The “top down” solution architecture must be informed by what exists and existing capabilities should be exposed and integrated based on a system of systems architecture

From Service Model to Web Service Implementation

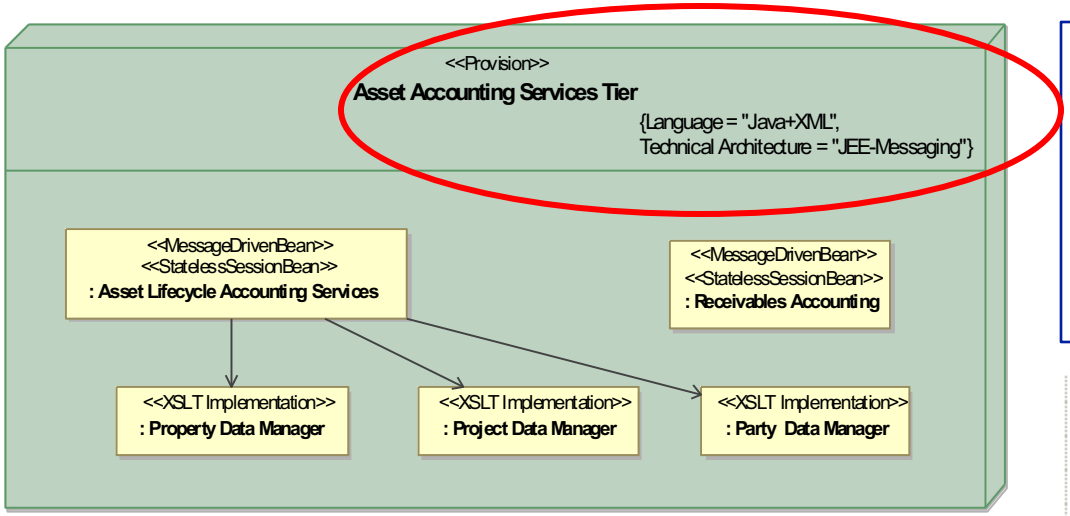
Logical View



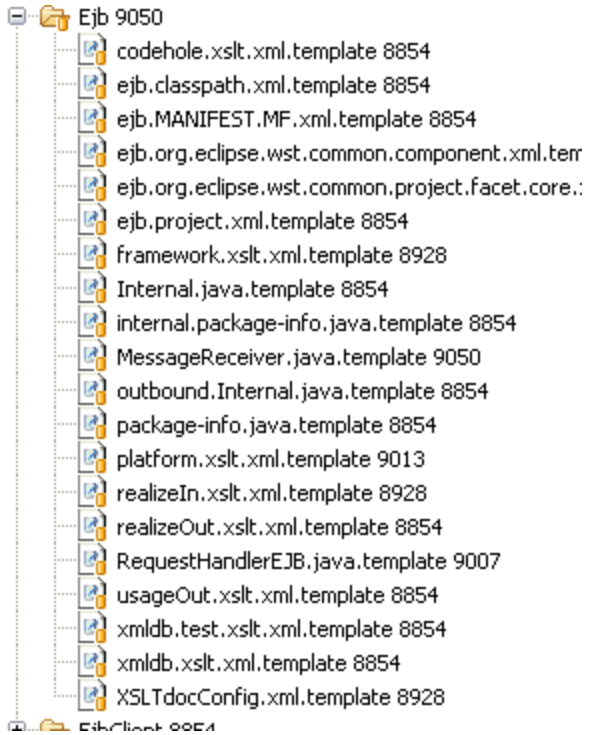
Physical View



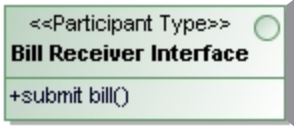
Provisioning the Implementation



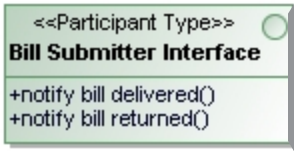

```
<provisioningContext name="service"…>
                <projectRef folder="EjbClient"/>
                <projectRef folder="AppClient"/>
                <projectRef folder="Ejb"/>
                <projectRef folder="ear"/>
                <projectRef folder="JbossConfig"/>
</provisioningContext>
```



Example Web Services Generation




```
<wsdl:portType name="Bill_Receiver_Interface">
   <wsdl:operation name="submit_bill">
     <wsdl:input message="tns:Bill_Submission__Message_Type"
                 name="bill_submission">
     </wsdl:input>
   </wsdl:operation>
 </wsdl:portType>
```




```
<wsdl:portType name="Bill_Submitter_Interface">
   <wsdl:operation name="notify_bill_delivered">
     <wsdl:input message="tns:Bill_Delivered__Message_Type"
                 name="bill_delivered">
     </wsdl:input>
   </wsdl:operation>
   <wsdl:operation name="notify_bill_returned">
     <wsdl:input message="tns:Bill_Returned__Message_Type"
                 name="bill_returned">
     </wsdl:input>
   </wsdl:operation>
 </wsdl:portType>
```

Example Request Message XML Document

```
<BillSubmission__Message identity="…">
     <BillSubmission>
          <bill>
               <Bill>
                    <billID> … </billID>
                    <principleAmount> … </principleAmount>
                    …
                    <payer>
                         <Party identity="…">
                    </payer>
                    …
                    <lineItems>
                         <LineItem__Item>
                              <LineItem> … </LineItem>
                         </LineItem__Item>
                    </lineItems>
               </Bill>
          </bill>
          <billingAddress>
               <Address> … </Address>
               <BillingAddress> … </BillingAddress>
          <billingAddress>
     </BillSubmission>
</BillSubmission__Message>
```