

Lecture 9 - Service Integration, Orchestration and Governance

Integration

Plumbing different software applications/services/systems and forming new software solutions is known as 'Enterprise Integration'.

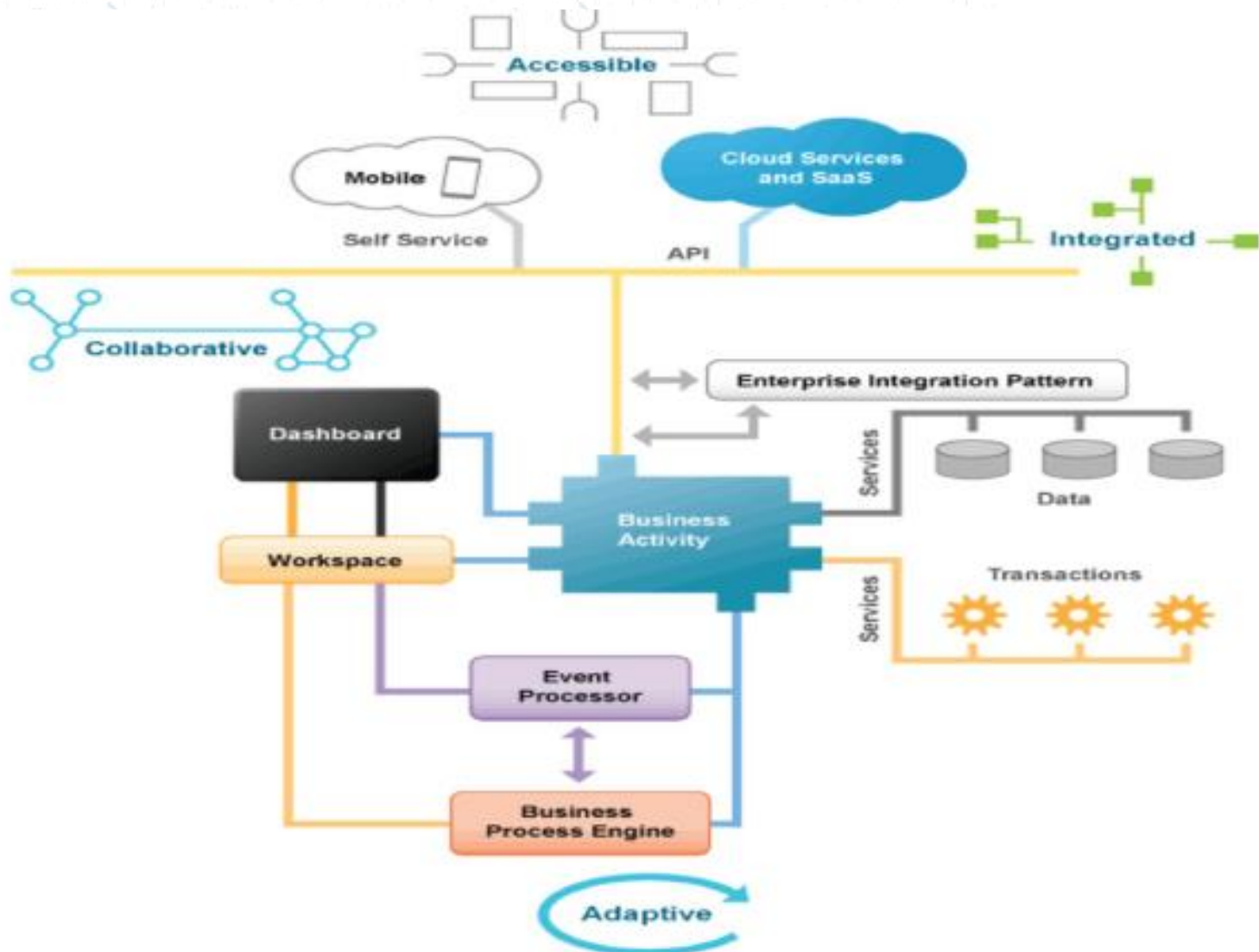
When SOA Integration is Used?

- Build new applications
- Expose a business function
- Reuse services to build new processes
- Business process automation
- Integrate data for analytics

Importance of Integration in SOA

- Enterprises heavily rely on the underlying software systems/services/applications.
- Disparate technologies and platforms
- No single solution or a vendor
- Diverse Business requirements

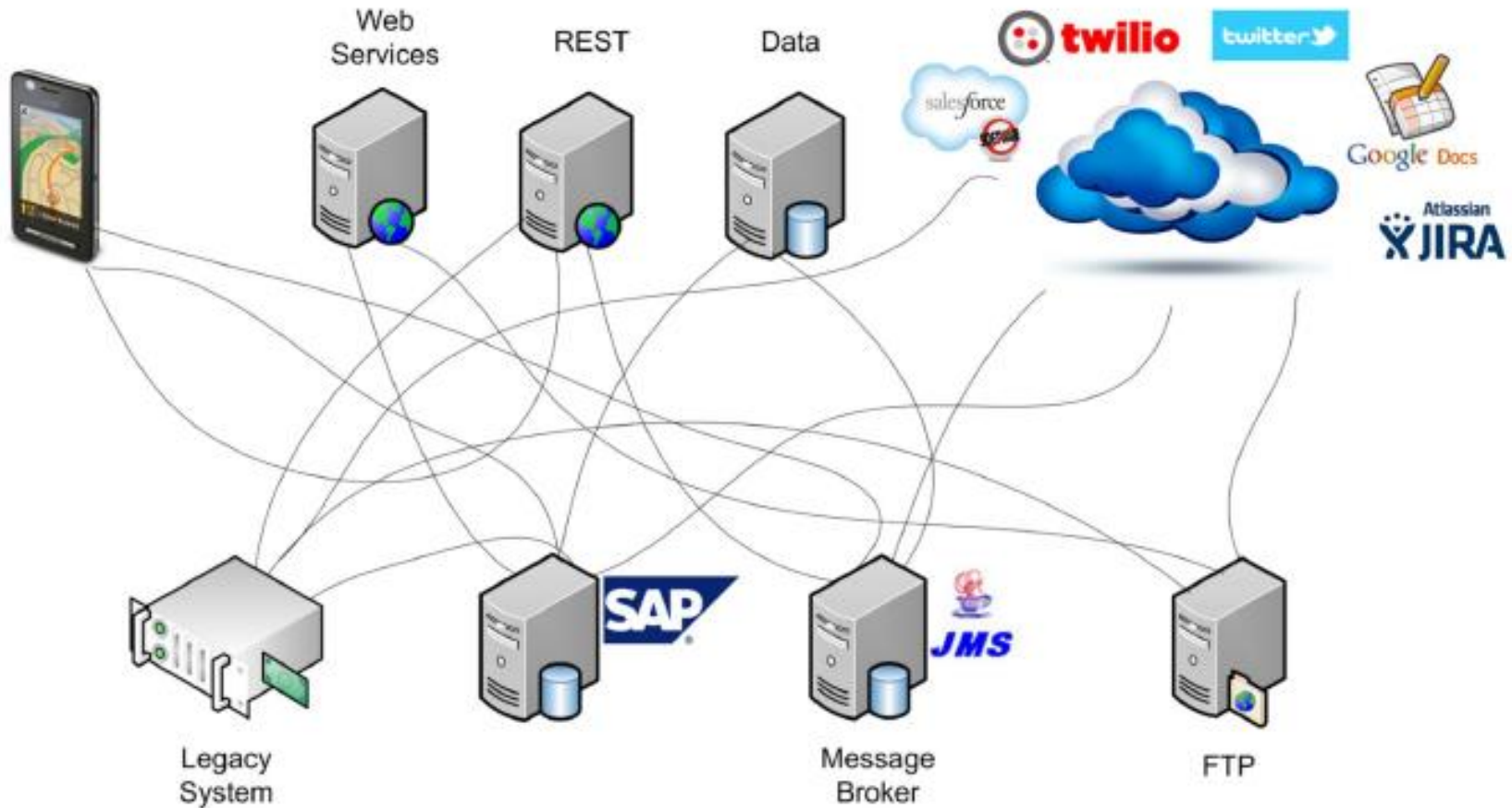
An Example



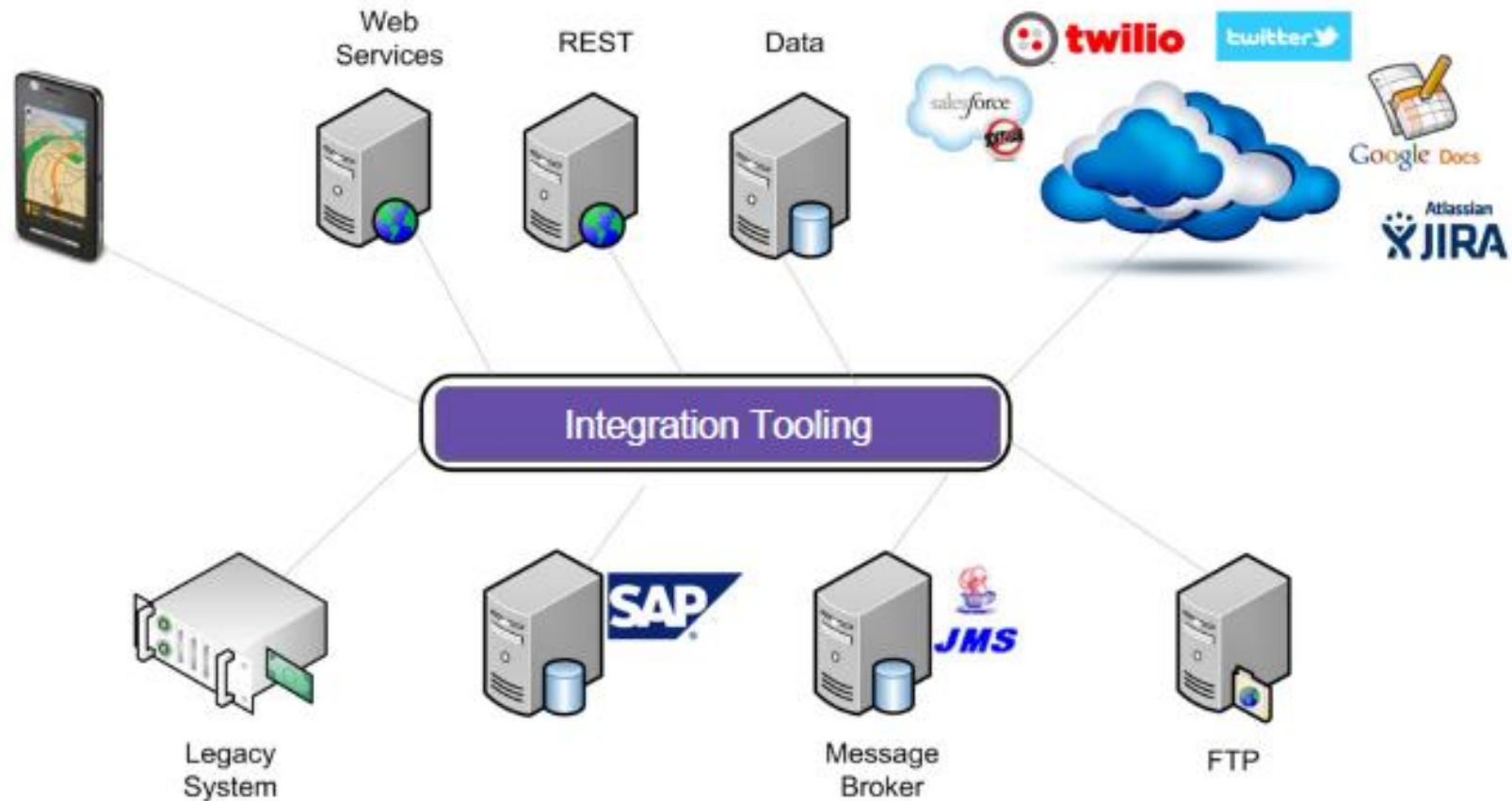
Challenges of Integration

- Heterogeneity: Disparate systems, protocols and standards
- Variety: Legacy systems, SOAP/REST services, Cloud APIs
- Disorganized: Spaghetti architecture, poorly managed
- Costly: Hardly scalable and maintainable
- Unquantifiable: Difficult to measure throughput & productivity

Unplanned Integration



Integration Tooling

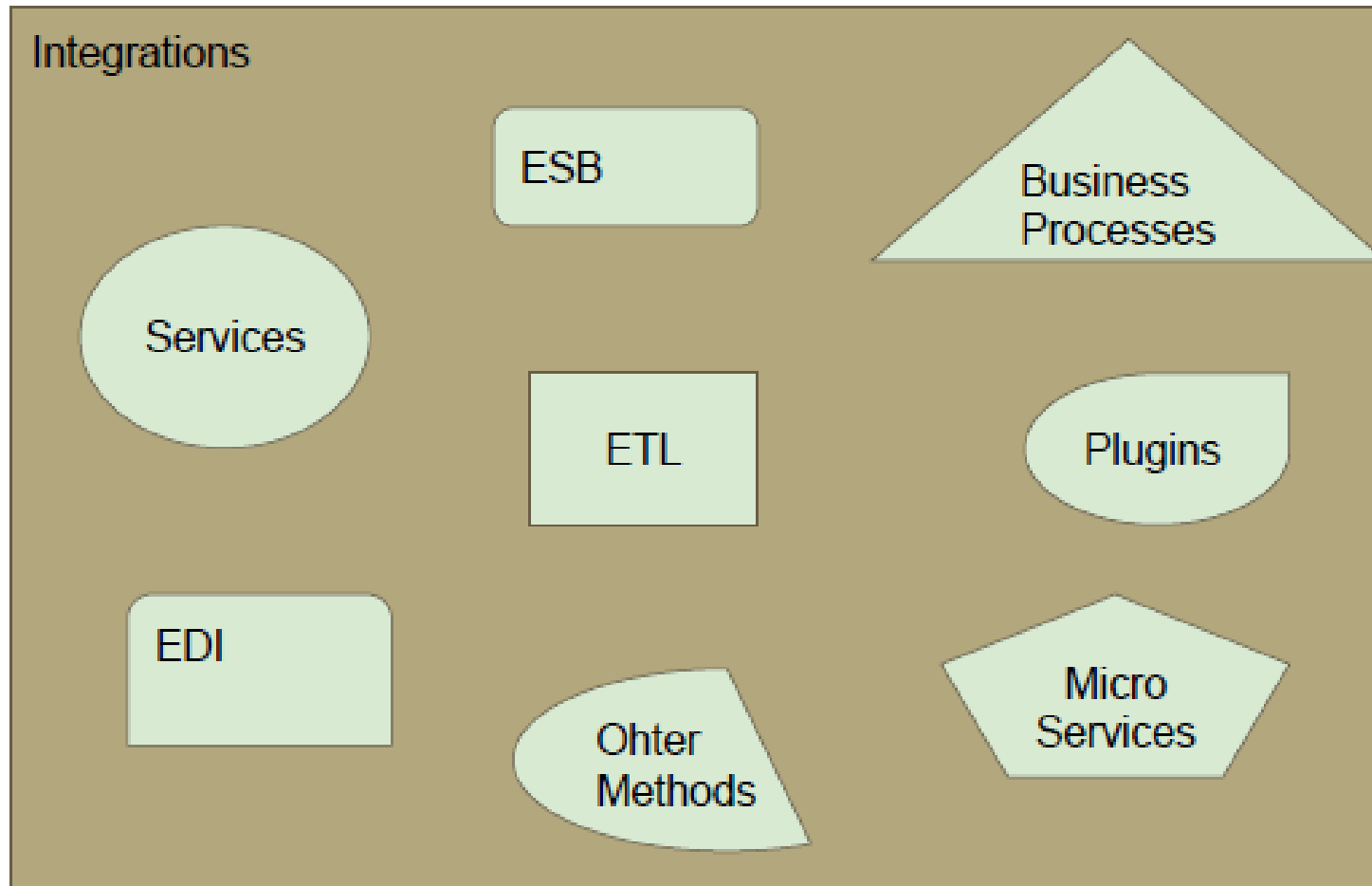


ESB

Implements a communication system between mutually interacting Software applications/services in a service-oriented architecture

- Routes
- Transforms
- Mediation

Integration Tooling - SOA and Non-SOA



ESB - Meets SOA Integration Challenges

- Transports: Support for web (HTTP), files (VFS), e-mail (POP, IMAP) and more..
- Formats/ Protocols: XML, JSON, CSV, EDI, SOAP, REST and more..
- Domain specific apps: Financial Services (FIX), Healthcare (HL7)..
- COTS: SAP, IBM WebSphere MQ, MSMQ and more..
- Cloud apps: Salesforce, Google Apps, Twitter, JIRA and more..
- Custom extensions: Handles proprietary/ non-standard integration cases

Hands On

Using an ESB for Integration

Enterprise Integration Patterns

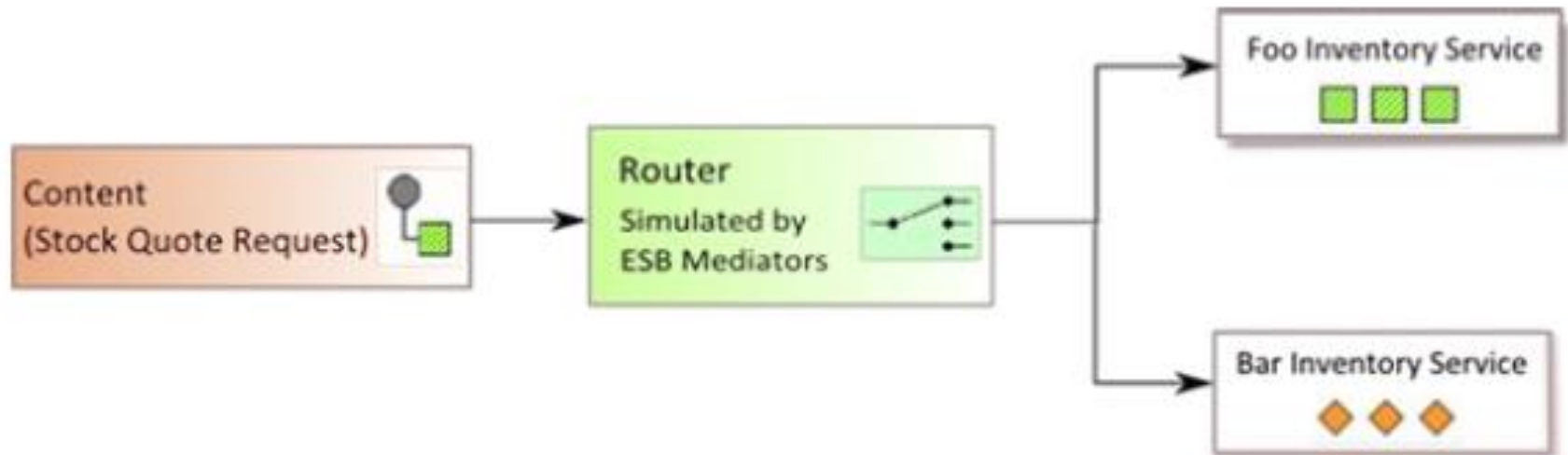
- A design Pattern - A pattern that keeps occurring
- Solution architects over the years found some patterns that kept recurring

<http://www.enterpriseintegrationpatterns.com/books1.html>

- Gregor Hope published a book - Enterprise Integration Patterns Designing, Building, and Deploying Messaging Solutions
- Contains 65 integration patterns

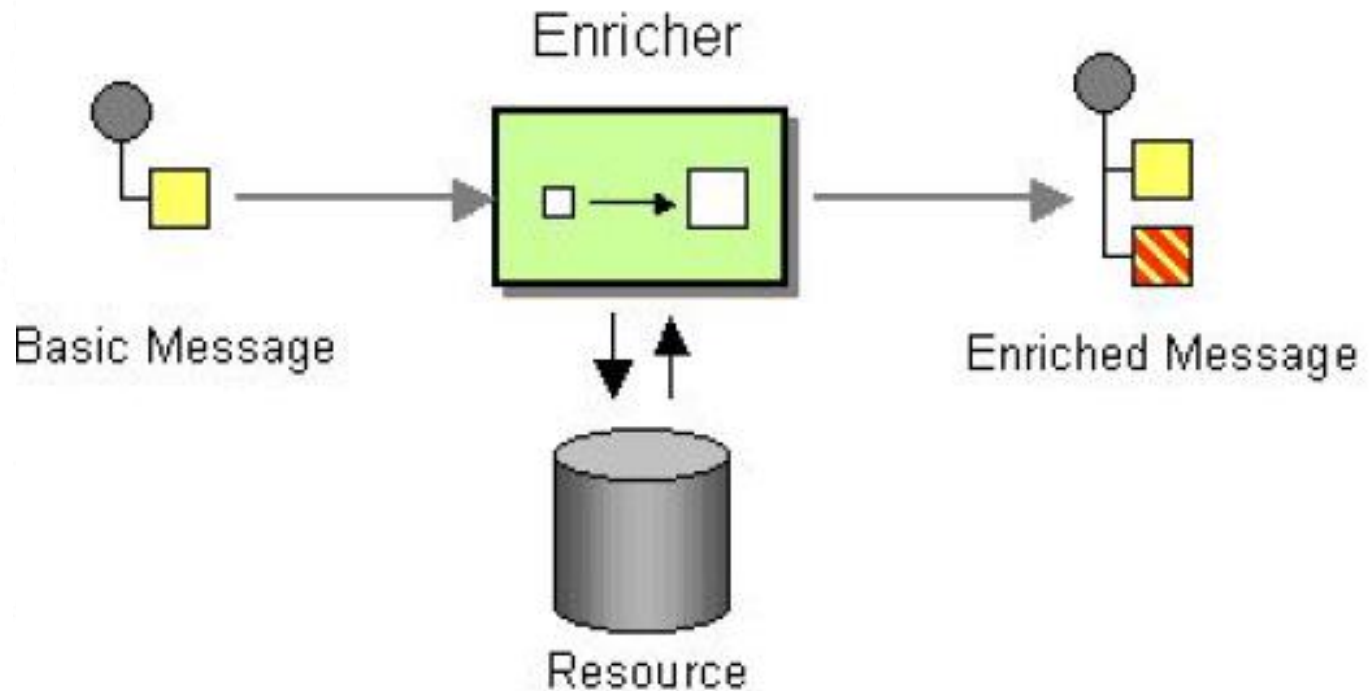
Enterprise Integration Patterns

Content Based Routing



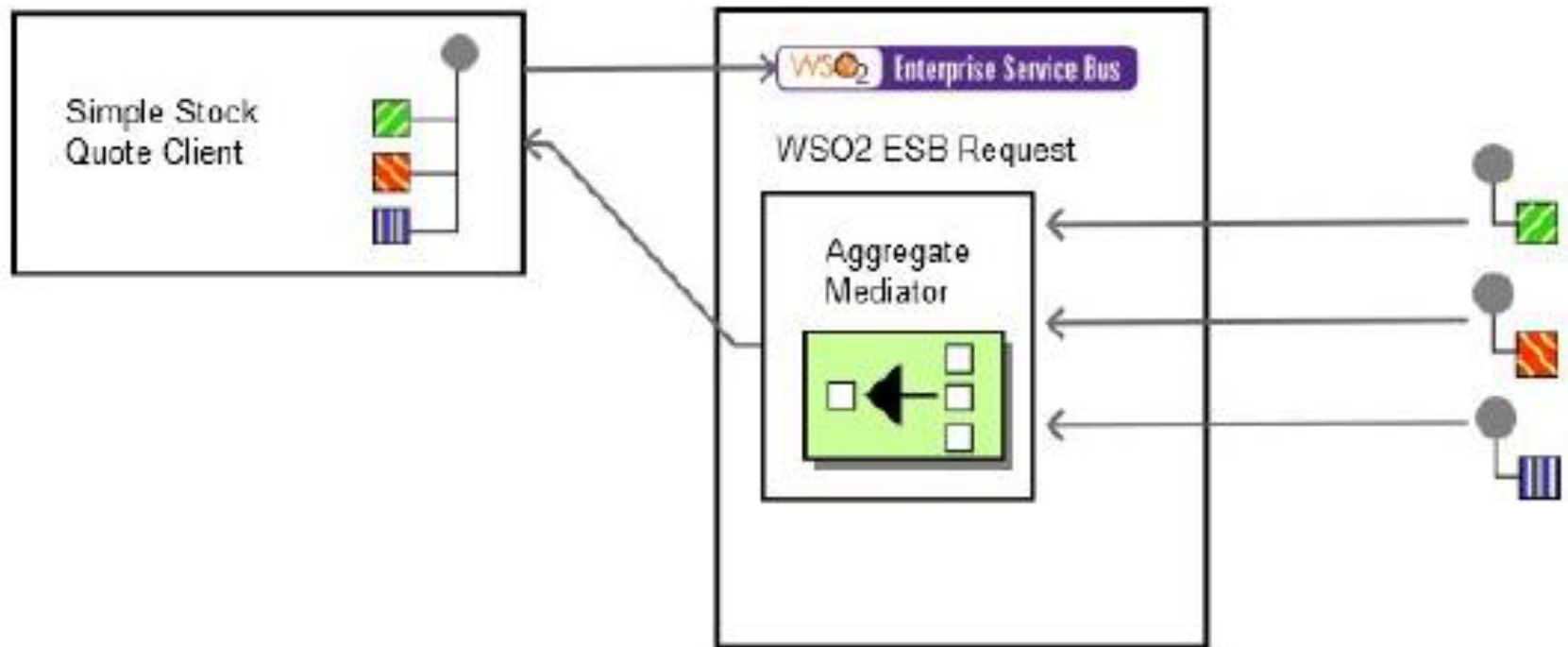
Enterprise Integration Patterns

Enricher



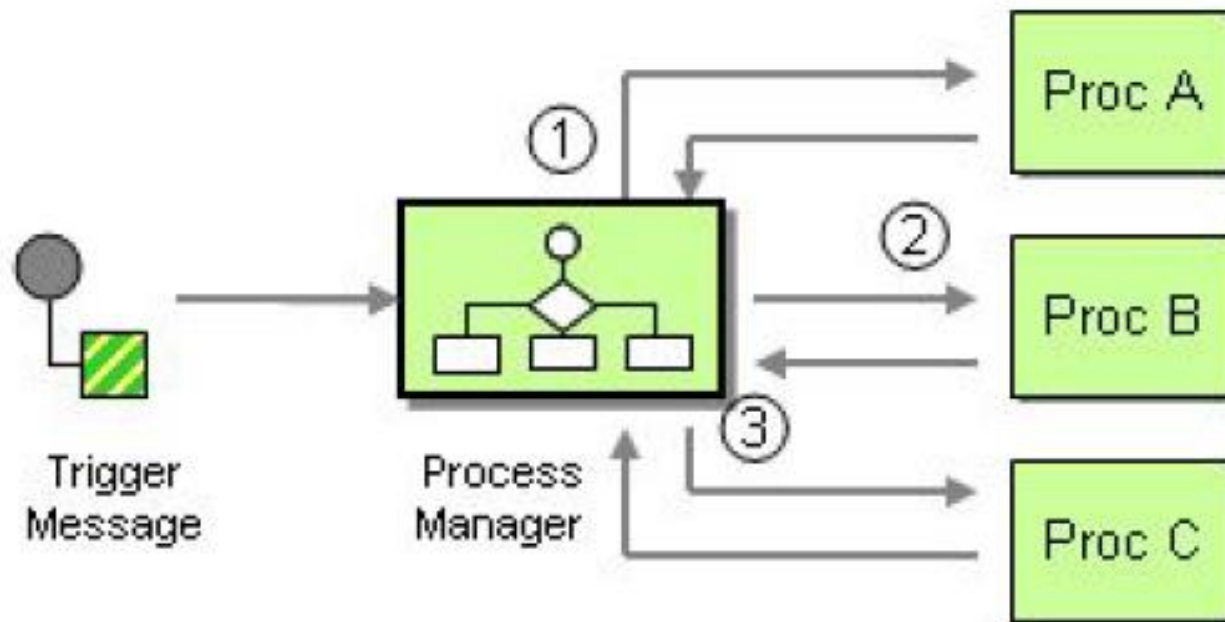
Enterprise Integration Patterns

Aggregator



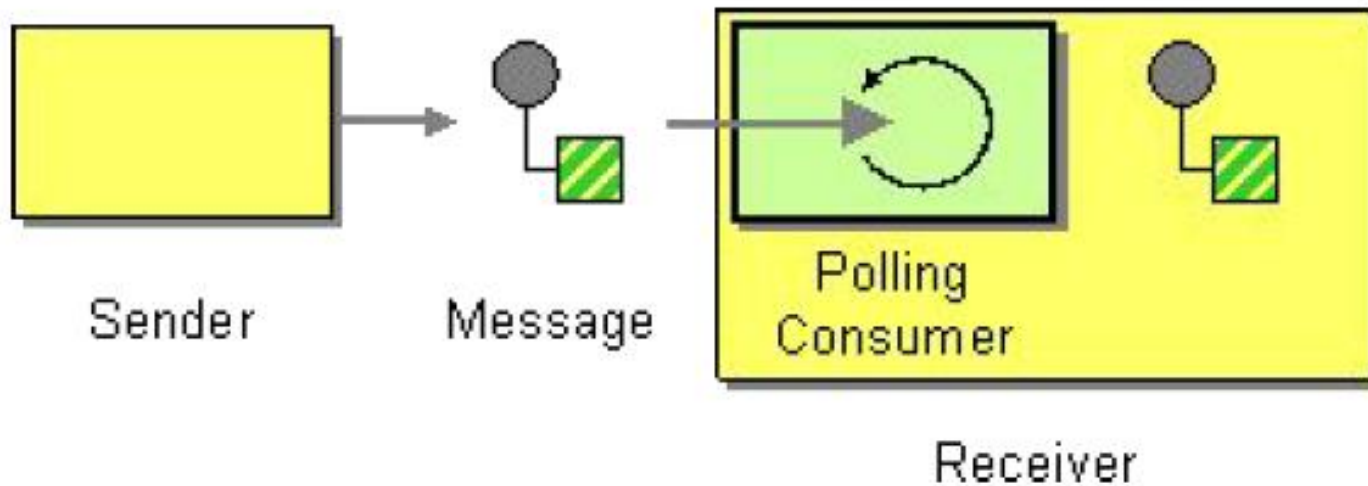
Enterprise Integration Patterns

Process Manager

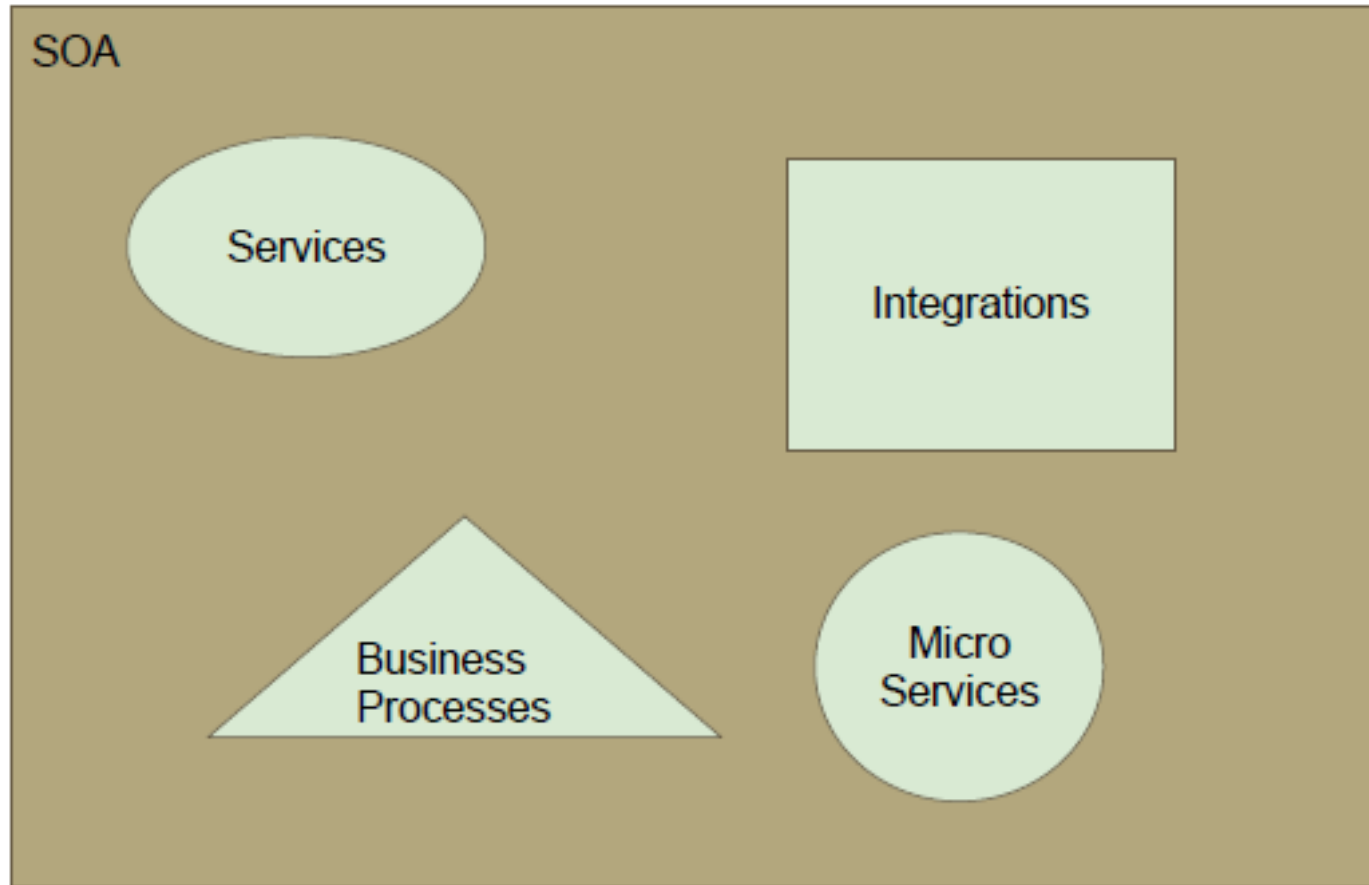


Enterprise Integration Patterns

Polling Consumer



SOA Space



Service Orchestration



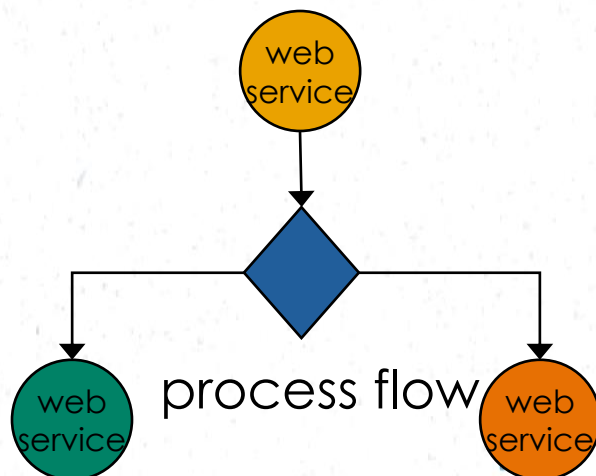
Service Orchestration

- Process Logic module is like a leader in an orchestration
- Services need to be linked and sequenced to form an application
 - This process is known as orchestration.
- Orchestration Models
 - Activity diagram
 - State charts
 - Petri Nets
 - Activity Hierarchy
 - Etc.

Orchestration vs. Choreography

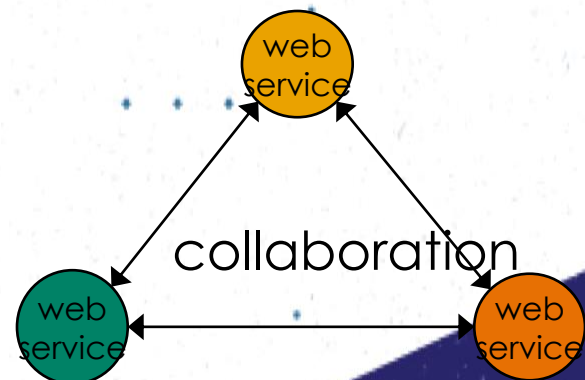
- Orchestration

- An executable business process describing a flow from the perspective and under control of a single endpoint



- Choreography

- The observable public exchange of messages, rules of interaction and agreements between two or more business process endpoints



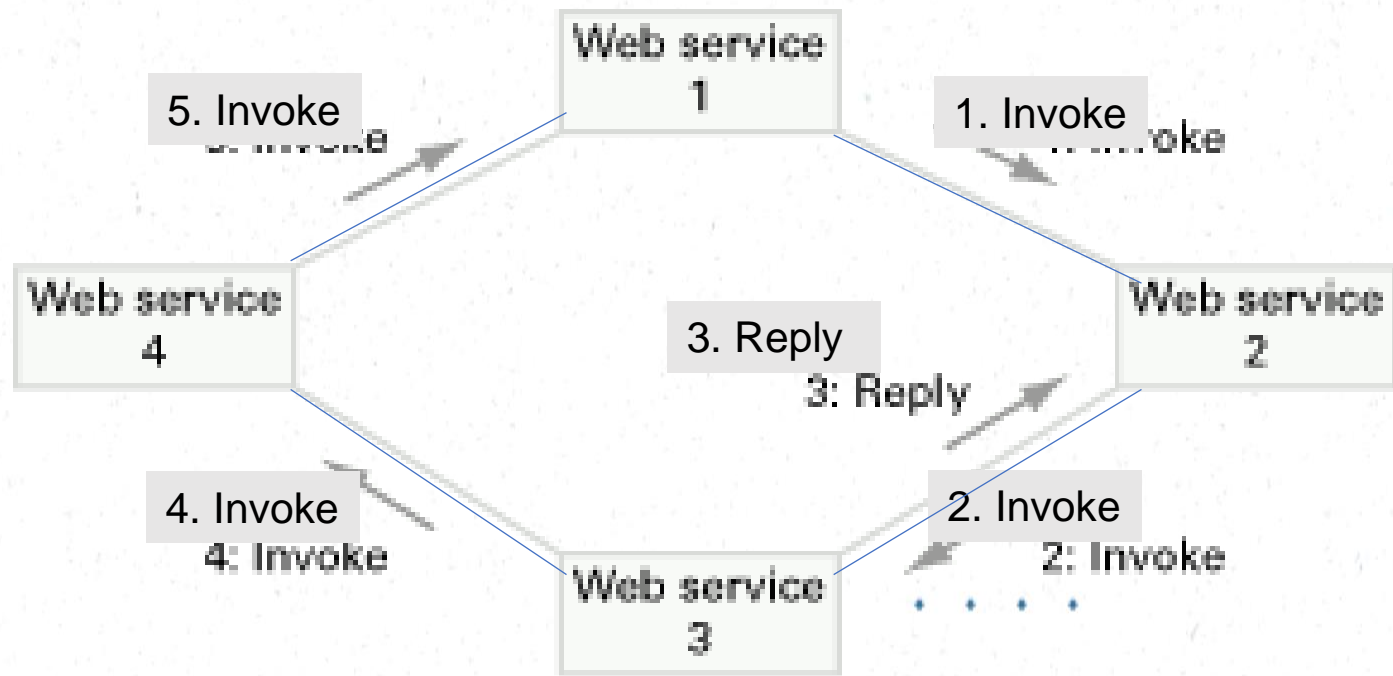
Service Choreography



Choreography

- Does not rely on a central coordinator
- Each Web service involved in the choreography knows exactly when to execute its operations and with whom to interact
- Collaborative effort focusing on the exchange of messages in public business processes
- All participants in the choreography need to be aware of the business process, operations to execute, messages to exchange, and the timing of message exchanges.

Choreography



Orchestration versus Choreography

- From the perspective of composing Web services to execute business processes, orchestration is a more flexible paradigm and has the following advantages over choreography:
 - The coordination of component processes is centrally managed by a known coordinator.
 - Web services can be incorporated without their being aware that they are taking part in a larger business process.
 - Alternative scenarios can be put in place in case faults occur.

Orchestration versus Choreography

- BPEL supports two different ways of describing business processes that support orchestration and choreography:
 - **Executable processes** allow you to specify the exact details of business processes. They follow the orchestration paradigm and can be executed by an orchestration engine.
 - **Abstract business protocols** allow specification of the public message exchange between parties only. They do not include the internal details of process flows and are not executable. They follow the choreography paradigm.

Business Processes & BPEL

- A **business process** is a collection of interrelated tasks, which are designed to deliver a particular result
- A business process can be decomposed into several sub-processes, which have their own attributes, but are aligned with the goal of the overall process
- The analysis of business processes typically includes the mapping of processes and sub-processes down to an activity level
- BPEL: Business Process Execution Language

BPEL

- Basically a tool to create programs using flow diagrams, whose building blocks are individual services
- Really meant for business analysts, not programmers
 - Facilitates orchestration without knowing how to code
 - Programmers would do it in a 'proper' programming language
- WSBPEL is a BPEL implementation for Web Services
 - BPEL could apply to other SOA approaches

BPEL

- BPEL is an XML-based language.
- It is an open standard – not proprietary
- BPEL scope includes:
 - Sequencing of process activities, especially Web Service interactions
 - Correlation of messages and process instances
 - Recovery behavior in case of failures and exceptional conditions
 - Bilateral Web Service based relationships between process roles

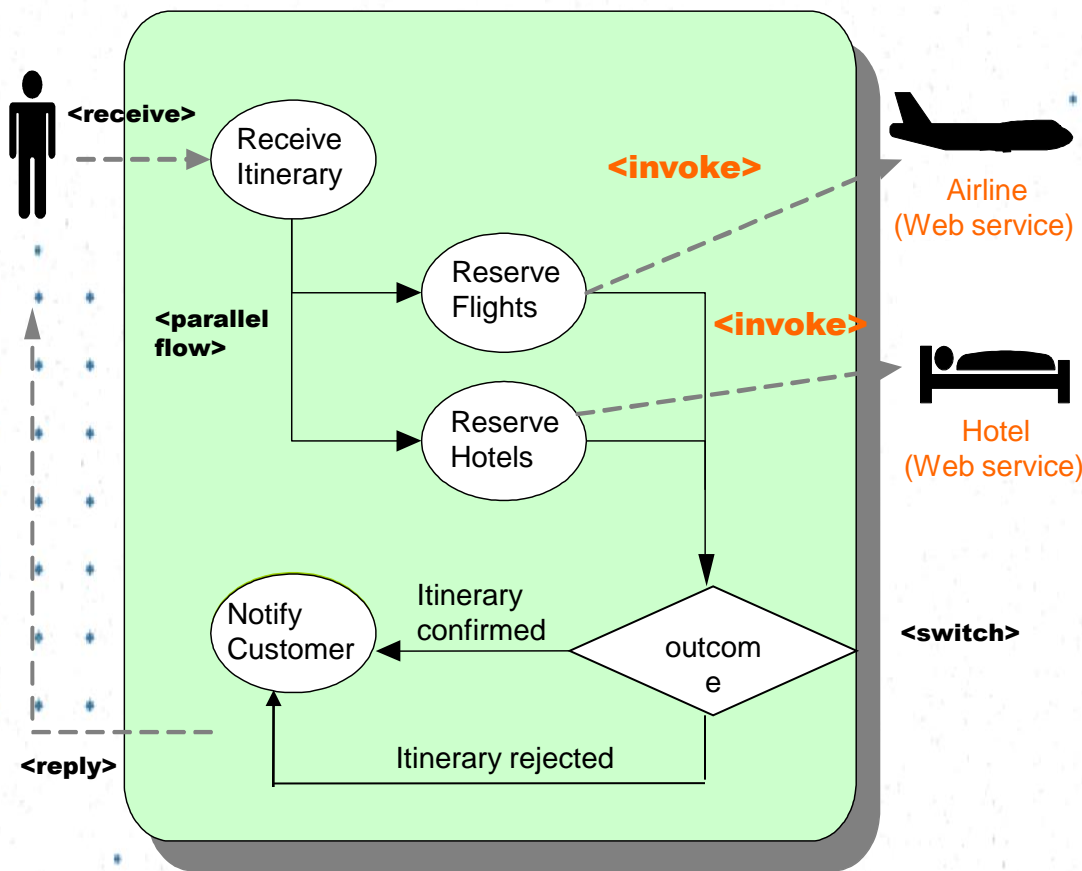
BPEL Activity

- A BPEL process consists of steps. Each step is called an activity.
- BPEL supports primitive and structural activities.
- Primitive activities represent basic constructs and are used for common tasks, such as those listed below:
 - Invoking Web services, using **<invoke>**
 - Waiting for the request, using **<receive>**
 - Manipulating data variables, using **<assign>**
 - Indicating faults and exceptions, using **<throw>**, etc.

BPEL Activity

- We can then combine these activities into more complex algorithms that specify the steps of a business process.
- To combine primitive activities, BPEL supports several structure activities.
- The most important are:
 - Sequence (**<sequence>**) for defining a set of activities that will be invoked in an ordered sequence
 - Flow (**<flow>**) for defining a set of activities that will be invoked in parallel
 - Case-switch construct (**<switch>**) for implementing branches
 - While (**<while>**) for defining loops, etc.

BPEL-Example



- **<receive>** and **<reply>** activities receive messages from and give feedback to customers
- **<invoke>** activities are used to trigger internal and/or external web services
- **<parallel flow>** activity allows tasks to be executed concurrently.
- **<switch>** activity allows conditional behaviours in business process.

A BPEL Process

```

001 <process name="purchaseOrderProcess"
002     targetNamespace="..."
003     xmlns="..."
004     xmlns:ins="...">
...
044 <sequence>
045   <receive partnerLink="purchasing"
046       portType="Ins:purchaseOrderPT"
047       operation="sendPurchaseOrder"
048       variable="PO">
049   </receive>
050   <flow>
051     <links>
052       <link name="ship-to-invoice"/>
053       <link name="ship-to-scheduling"/>
054     </links>
055     <sequence>
056       <assign>
057         <copy>
058           <from variable="PO" part="customerInfo"/>
059           <to variable="shippingRequest"
060               part="customerInfo"/>
061         </copy>
062       </assign>
063       <invoke partnerLink="shipping"
064           portType="Ins:shippingPT"
065           operation="requestShipping"
066           inputVariable="shippingRequest"
067           outputVariable="shippingInfo">
068         <source linkName="ship-to-invoice"/>
069       </invoke>
070       <receive partnerLink="shipping"
071           portType="Ins:shippingCallbackPT"
072           operation="sendSchedule"
073           variable="shippingSchedule">
074         <source linkName="ship-to-scheduling"/>
075       </receive>
076     </sequence>
077   </flow>
078   <sequence>
079     <invoke partnerLink="invoicing"
080         portType="Ins:computePricePT"
081         operation="initiatePriceCalculation"
082         inputVariable="PO">
083     </invoke>
084     <invoke partnerLink="invoicing"
085         portType="Ins:computePricePT"
086         operation="sendShippingPrice"
087         inputVariable="shippingInfo">
088       <target linkName="ship-to-invoice"/>
089     </invoke>
090     <receive partnerLink="invoicing"
091         portType="Ins:invoiceCallbackPT"
092         operation="sendInvoice"
093         variable="Invoice"/>
094   </sequence>
095   <sequence>
096     <invoke partnerLink="scheduling"
097         portType="Ins:schedulingPT"
098         operation="requestProductionScheduling"
099         inputVariable="PO">
100     </invoke>
101     <invoke partnerLink="scheduling"
102         portType="Ins:schedulingPT"
103         operation="sendShippingSchedule"
104         inputVariable="shippingSchedule">
105       <target linkName="ship-to-scheduling"/>
106     </invoke>
107   </sequence>
108   </flow>
109   <reply partnerLink="purchasing"
110       portType="Ins:purchaseOrderPT"
111       operation="sendPurchaseOrder"
112       variable="Invoice"/>
113 </process>

```



Structured Activities

```

001 <process name="purchaseOrderProcess"
002     targetNamespace="..."
003     xmlns="..."
004     xmlns:Ins="...">
...
044 <sequence>
045     <receive partnerLink="purchasing"
046         portType="Ins:purchaseOrderPT"
047         operation="sendPurchaseOrder"
048         variable="PO">
049     </receive>
050     <flow>
051         <links>
052             <link name="ship-to-invoice"/>
053             <link name="ship-to-scheduling"/>
054         </links>
055         <sequence>
056             <assign>
057                 <copy>
058                     <from variable="PO" part="customerInfo"/>
059                     <to variable="shippingRequest"
060                         part="customerInfo"/>
061                 </copy>
062             </assign>
063             <invoke partnerLink="shipping"
064                 portType="Ins:shippingPT"
065                 operation="requestShipping"
066                 inputVariable="shippingRequest"
067                 outputVariable="shippingInfo">
068                 <source linkName="ship-to-invoice"/>
069             </invoke>
070             <receive partnerLink="shipping"
071                 portType="Ins:shippingCallbackPT"
072                 operation="sendSchedule"
073                 variable="shippingSchedule">
074                 <source linkName="ship-to-scheduling"/>
075             </receive>
076         </sequence>

```

```

077 <sequence>
078     <invoke partnerLink="invoicing"
079         portType="Ins:computePricePT"
080         operation="initiatePriceCalculation"
081         inputVariable="PO">
082     </invoke>
083     <invoke partnerLink="invoicing"
084         portType="Ins:computePricePT"
085         operation="sendShippingPrice"
086         inputVariable="shippingInfo">
087         <target linkName="ship-to-invoice"/>
088     </invoke>
089     <receive partnerLink="invoicing"
090         portType="Ins:invoiceCallbackPT"
091         operation="sendInvoice"
092         variable="Invoice"/>
093 </sequence>
094 <sequence>
095     <invoke partnerLink="scheduling"
096         portType="Ins:schedulingPT"
097         operation="requestProductionScheduling"
098         inputVariable="PO">
099     </invoke>
100     <invoke partnerLink="scheduling"
101         portType="Ins:schedulingPT"
102         operation="sendShippingSchedule"
103         inputVariable="shippingSchedule">
104         <target linkName="ship-to-scheduling"/>
105     </invoke>
106 </sequence>
107 </flow>
108 <reply partnerLink="purchasing"
109     portType="Ins:purchaseOrderPT"
110     operation="sendPurchaseOrder"
111     variable="Invoice"/>
112 </sequence>
113 </process>

```

Primitive Activities

```

001 <process name="purchaseOrderProcess"
002     targetNamespace="..."
003     xmlns="..."
004     xmlns:Ins="...">
...
044 <sequence>
045     <receive partnerLink="purchasing"
046         portType="Ins:purchaseOrderPT"
047         operation="sendPurchaseOrder"
048         variable="PO">
049 </receive>
050 <flow>
051     <links>
052         <link name="ship-to-invoice"/>
053         <link name="ship-to-scheduling"/>
054     </links>
055     <sequence>
056         <assign>
057             <copy>
058                 <from variable="PO" part="customerInfo"/>
059                 <to variable="shippingRequest"
060                     part="customerInfo"/>
061             </copy>
062         </assign>
063         <invoke partnerLink="shipping"
064             portType="Ins:shippingPT"
065             operation="requestShipping"
066             inputVariable="shippingRequest"
067             outputVariable="shippingInfo">
068             <source linkName="ship-to-invoice"/>
069         </invoke>
070         <receive partnerLink="shipping"
071             portType="Ins:shippingCallbackPT"
072             operation="sendSchedule"
073             variable="shippingSchedule">
074             <source linkName="ship-to-scheduling"/>
075         </receive>
076     </sequence>

```

```

077 <sequence>
078     <invoke partnerLink="invoicing"
079         portType="Ins:computePricePT"
080         operation="initiatePriceCalculation"
081         inputVariable="PO">
082     </invoke>
083     <invoke partnerLink="invoicing"
084         portType="Ins:computePricePT"
085         operation="sendShippingPrice"
086         inputVariable="shippingInfo">
087         <target linkName="ship-to-invoice"/>
088     </invoke>
089     <receive partnerLink="invoicing"
090         portType="Ins:invoiceCallbackPT"
091         operation="sendInvoice"
092         variable="Invoice"/>
093 </sequence>
094 <sequence>
095     <invoke partnerLink="scheduling"
096         portType="Ins:schedulingPT"
097         operation="requestProductionScheduling"
098         inputVariable="PO">
099     </invoke>
100     <invoke partnerLink="scheduling"
101         portType="Ins:schedulingPT"
102         operation="sendShippingSchedule"
103         inputVariable="shippingSchedule">
104         <target linkName="ship-to-scheduling"/>
105     </invoke>
106 </sequence>
107 </flow>
108 <reply partnerLink="purchasing"
109     portType="Ins:purchaseOrderPT"
110     operation="sendPurchaseOrder"
111     variable="Invoice"/>
112 </sequence>
113 </process>

```


Data Flow

```

001 <process name="purchaseOrderProcess"
002     targetNamespace="..."
003     xmlns="..."
004     xmlns:Ins="...">
...
044 <sequence>
045   <receive partnerLink="purchasing"
046       portType="Ins:purchaseOrderPT"
047       operation="sendPurchaseOrder"
048       variable="PO">
049   </receive>
050   <flow>
051     <links>
052       <link name="ship-to-invoice"/>
053       <link name="ship-to-scheduling"/>
054     </links>
055     <sequence>
056       <assign>
057         <copy>
058           <from variable="PO" part="customerInfo"/>
059           <to variable="shippingRequest"
060               part="customerInfo"/>
061         </copy>
062       </assign>
063       <invoke partnerLink="shipping"
064           portType="Ins:shippingPT"
065           operation="requestShipping"
066           inputVariable="shippingRequest"
067           outputVariable="shippingInfo">
068         <source linkName="ship-to-invoice"/>
069       </invoke>
070       <receive partnerLink="shipping"
071           portType="Ins:shippingCallbackPT"
072           operation="sendSchedule"
073           variable="shippingSchedule">
074         <source linkName="ship-to-scheduling"/>
075       </receive>
076     </sequence>

```

```

077 <sequence>
078   <invoke partnerLink="invoicing"
079       portType="Ins:computePricePT"
080       operation="initiatePriceCalculation"
081       inputVariable="PO">
082   </invoke>
083   <invoke partnerLink="invoicing"
084       portType="Ins:computePricePT"
085       operation="sendShippingPrice"
086       inputVariable="shippingInfo">
087     <target linkName="ship-to-invoice"/>
088   </invoke>
089   <receive partnerLink="invoicing"
090       portType="Ins:invoiceCallbackPT"
091       operation="sendInvoice"
092       variable="Invoice"/>
093 </sequence>
094 <sequence>
095   <invoke partnerLink="scheduling"
096       portType="Ins:schedulingPT"
097       operation="requestProductionScheduling"
098       inputVariable="PO">
099   </invoke>
100   <invoke partnerLink="scheduling"
101       portType="Ins:schedulingPT"
102       operation="sendShippingSchedule"
103       inputVariable="shippingSchedule">
104     <target linkName="ship-to-scheduling"/>
105   </invoke>
106 </sequence>
107 </flow>
108 <reply partnerLink="purchasing"
109     portType="Ins:purchaseOrderPT"
110     operation="sendPurchaseOrder"
111     variable="Invoice"/>
112 </sequence>
113 </process>

```

Partner Links

```

001 <process name="purchaseOrderProcess"
002     targetNamespace="..."
003     xmlns="..."
004     xmlns:Ins="...">
...
044 <sequence>
045   <receive partnerLink="purchasing"
046     portType="Ins:purchaseOrderPT"
047     operation="sendPurchaseOrder"
048     variable="PO">
049   </receive>
050   <flow>
051     <links>
052       <link name="ship-to-invoice"/>
053       <link name="ship-to-scheduling"/>
054     </links>
055     <sequence>
056       <assign>
057         <copy>
058           <from variable="PO" part="customerInfo"/>
059           <to variable="shippingRequest"
060             part="customerInfo"/>
061         </copy>
062       </assign>
063       <invoke partnerLink="shipping"
064         portType="Ins:shippingPT"
065         operation="requestShipping"
066         inputVariable="shippingRequest"
067         outputVariable="shippingInfo"/>
068       <source linkName="ship-to-invoice"/>
069     </invoke>
070     <receive partnerLink="shipping"
071       portType="Ins:shippingCallbackPT"
072       operation="sendSchedule"
073       variable="shippingSchedule">
074     <source linkName="ship-to-scheduling"/>
075   </receive>
076 </sequence>

```

```

077 <sequence>
078   <invoke partnerLink="invoicing"
079     portType="Ins:computePricePT"
080     operation="initiatePriceCalculation"
081     inputVariable="PO">
082   </invoke>
083   <invoke partnerLink="invoicing"
084     portType="Ins:computePricePT"
085     operation="sendShippingPrice"
086     inputVariable="shippingInfo">
087     <target linkName="ship-to-invoice"/>
088   </invoke>
089   <receive partnerLink="invoicing"
090     portType="Ins:invoiceCallbackPT"
091     operation="sendInvoice"
092     variable="Invoice"/>
093 </sequence>
094 <sequence>
095   <invoke partnerLink="scheduling"
096     portType="Ins:schedulingPT"
097     operation="requestProductionScheduling"
098     inputVariable="PO">
099   </invoke>
100   <invoke partnerLink="scheduling"
101     portType="Ins:schedulingPT"
102     operation="sendShippingSchedule"
103     inputVariable="shippingSchedule">
104     <target linkName="ship-to-scheduling"/>
105   </invoke>
106 </sequence>
107 </flow>
108 <reply partnerLink="purchasing"
109   portType="Ins:purchaseOrderPT"
110   operation="sendPurchaseOrder"
111   variable="Invoice"/>
112 </sequence>
113 </process>

```



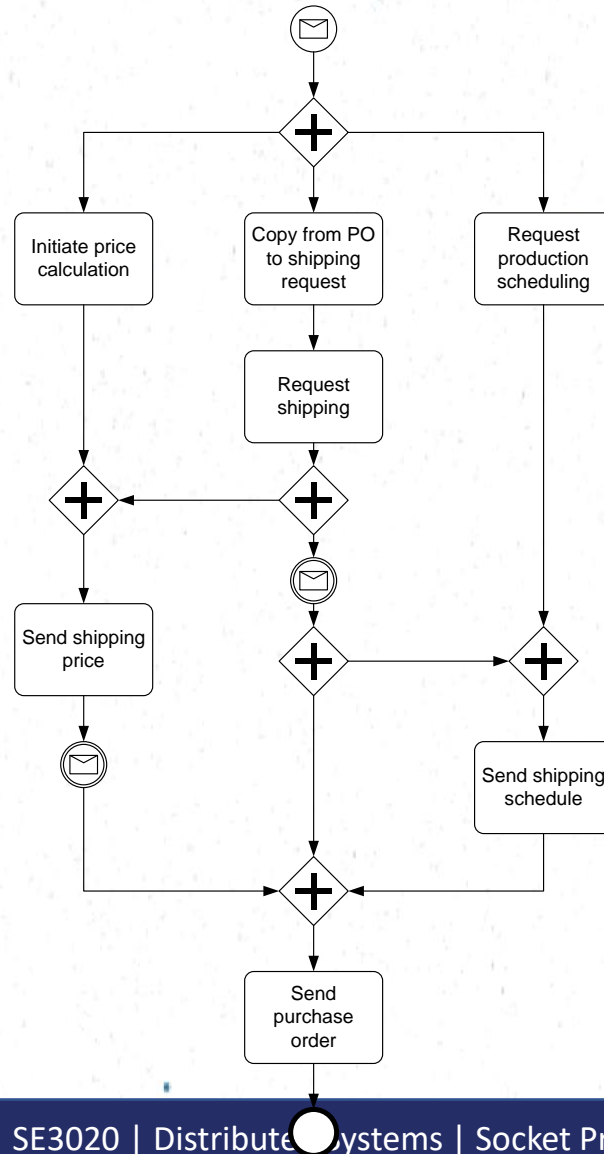
BPMN (Business Process Markup and Notation)

The primary goal of **BPMN** is to provide a notation that is readily understandable by all business users

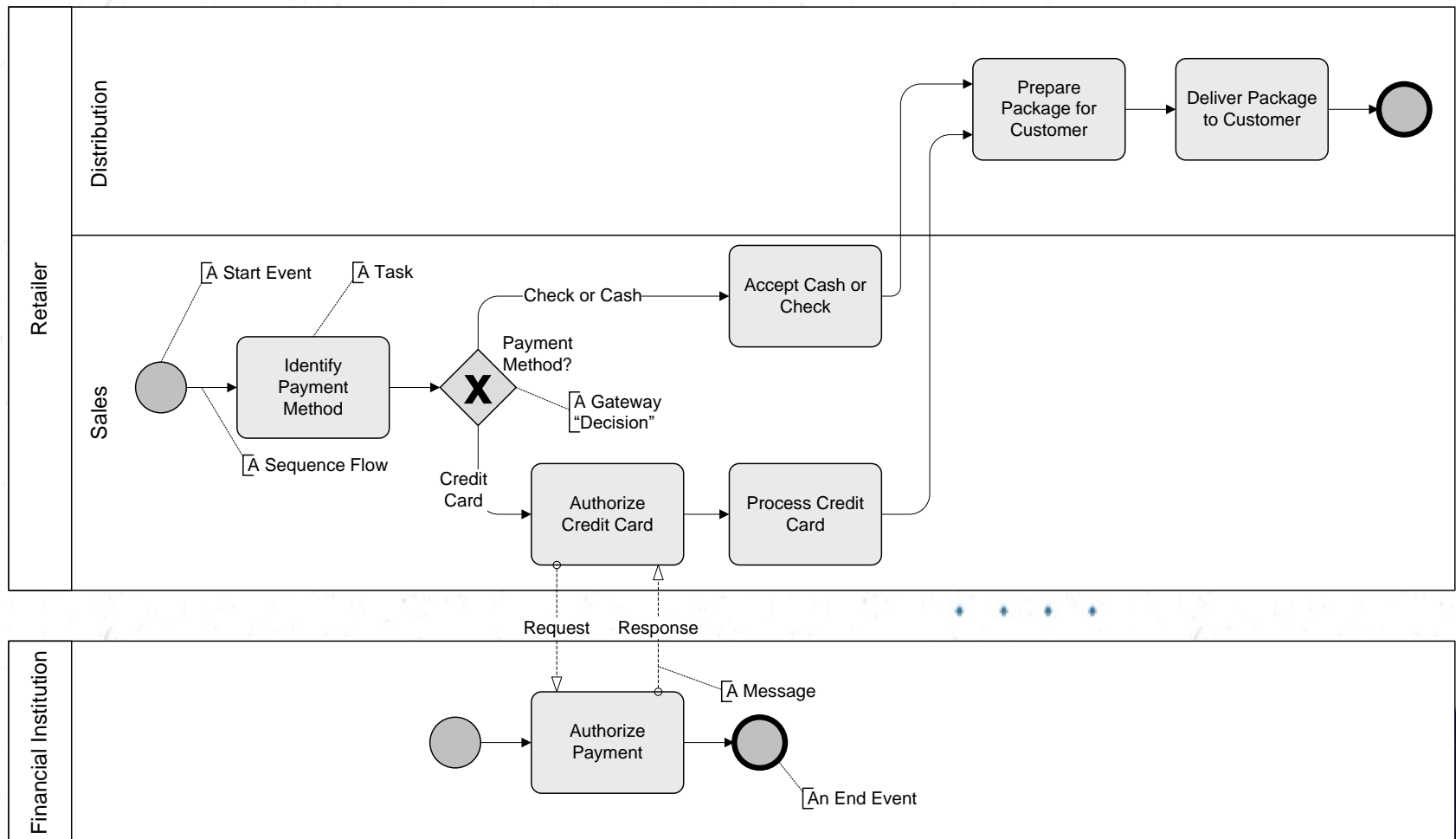
BPMN creates a **standardized bridge** for the gap between the business process design and process implementation.

Another goal, but no less important, is to ensure that XML languages designed for the execution of business processes, such as **BPEL4WS** (Business Process Execution Language for Web Services), can be visualized with a business-oriented notation.

The BPEL Process in BPMN



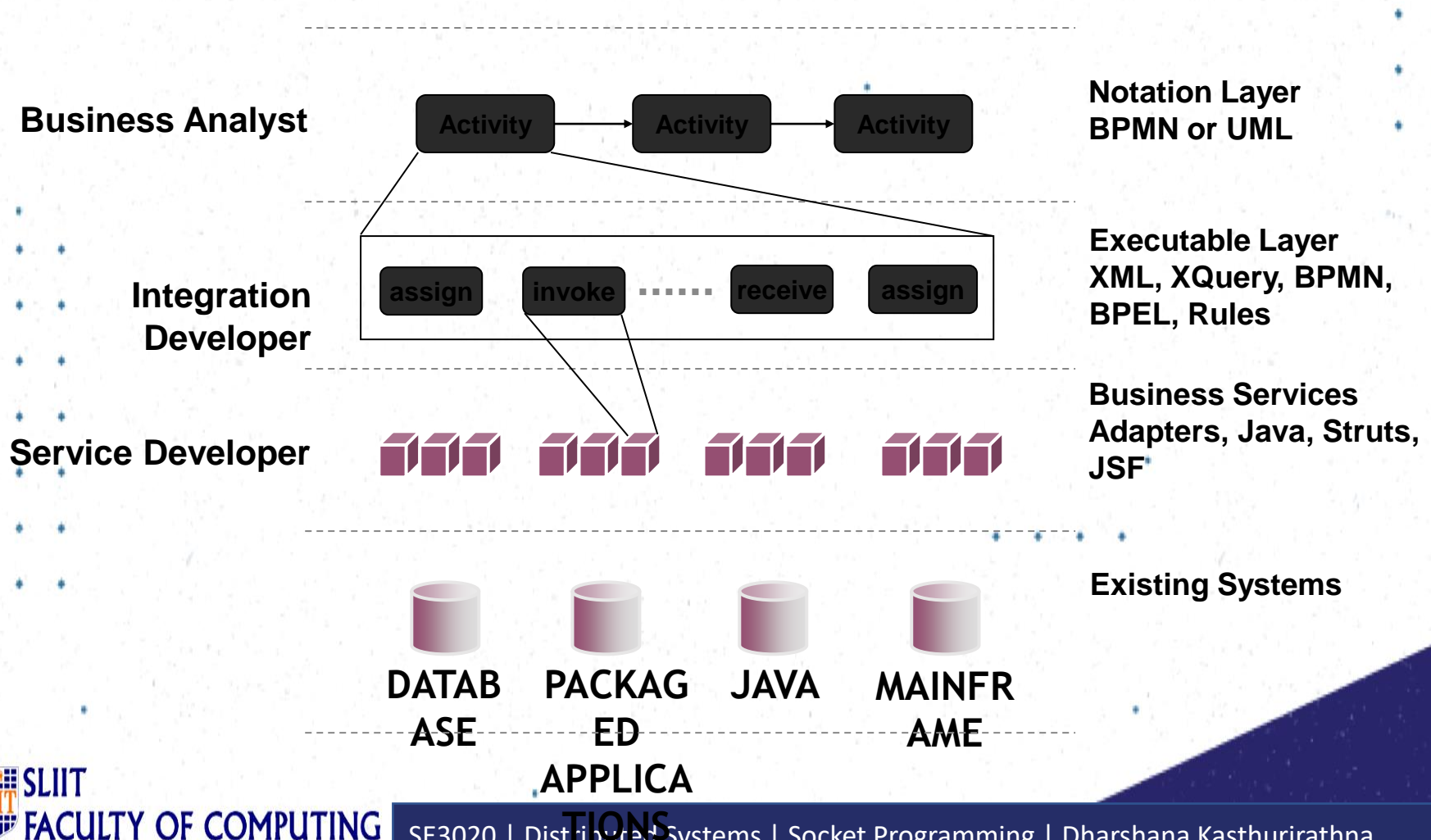
BPMN explained



BPMN Vs BPEL

- BPMN is richer than BPEL
- Transformation from BPEL to BPMN less a problem
- From BPMN to BPEL
 - Loss of information
 - Loss of design considerations
- Potential Solutions
 - Restriction to a subset of BPMN
 - Extension of BPEL

The Top Down Perspective



Securing Web Services

Security Fundamentals

- Authentication
- Authorization
- Integrity
- Confidentiality
- Availability
- Non-Repudiation

Security a Web Service

- WS-Security (SOAP services only)
- SSL with HTTP BasicAuth/HTTP Digest Auth
- SSL with Username Token (OAuth)
- SSL with IDToken (OpenID)

WS Security

- WS Security 1.0 on 2004 and 1.1 on 2006
- Based on
 - PKI
 - X509 Certificates
 - XML Security - XML Encryption and Signature
- Related Standards
 - WS Secure Conversation
 - WS Trust

WS Security

- Username Token
- Message level encryption
- Message level signature
- Message level encryption and signature

WS Security Sepcification Family

- A comprehensive specification
- Provides message level security
- Industry is no longer using it

TLS for HTTP → HTTPS

- TLS - Transport Layer Security
 - Version 1, 1.1, 1.2 and 1.3
- Predecessor of TLS is SSL
- Adds an additional encryption layer over HTTP
- Transport layer provides
 - Transport level confidentiality
 - Transport level integrity

HTTP BasicAuth

- HTTP Header with Username & Password
 - Authorization : Basic <Base64 encoded Username:Password>
- Base64 is not encryption. What is it?
 - Difference between
 - Encoding
 - Encryption
 - Hashing

Hands-on

Create a BasicAuth Header

Demo

HTTP Basic Auth Sample

TLS with BasicAuth

- Provides Authenticity
- Possible to do Authorization
- Transport layer confidentiality and integrity

Http Digest Auth

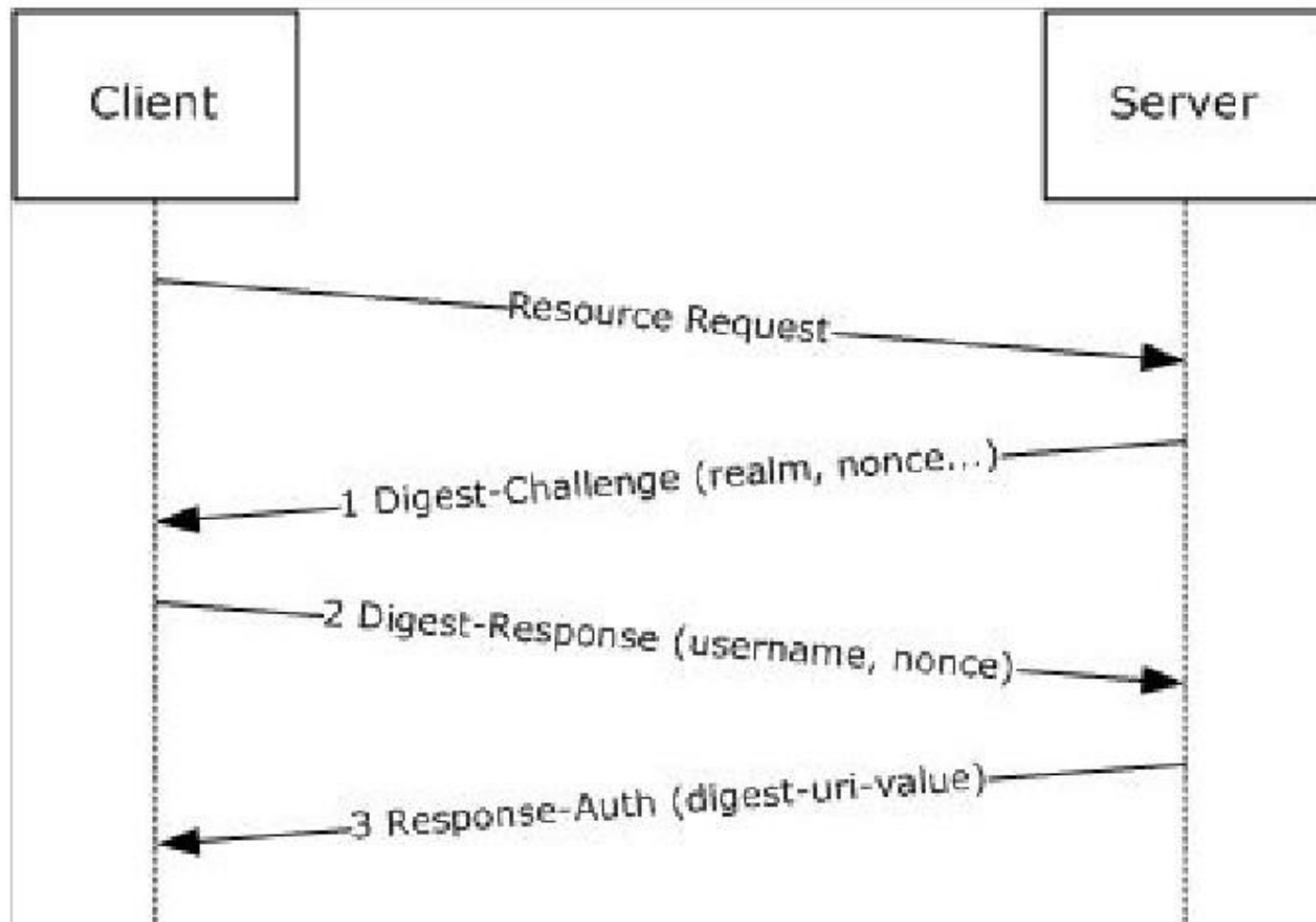
More on Hashing

- Irreversible. Result is called digest.
 - $d = \text{hash}(m1)$
- Deterministic
- Small change → Drastic result in the digest
- Second image resistance. Given $d = \text{hash}(m1)$, it is hard to find $m2$ such that $d = \text{hash}(m2)$
- Collision resistance
 - $\text{hash}(m1) = \text{hash}(m2)$

Cryptographic Nonce

- A number that is used only once
- Random Number
- Adding a Nonce to a message before hashing makes the Digest attacks hard

HTTP DigestAuth



Http DigestAuth

- *STEP 1* : a client sends a request to a server
- *STEP 2* : the server responds with a special code (called a **nonce** i.e. **number used only once**), another string representing the 'realm' and asks the client to authenticate
- *STEP 3* : the client responds with the hashed value of this nonce and the username, password and realm
- *STEP 4* : the server responds with the requested information if the client hash matches their own hash of the nonce, username, password and realm, or an error if not

TLS with HTTP DigestAuth

- Secure than BasicAuth
- Provides Authenticity
- Possible to do Authorization
- Transport layer confidentiality and integrity

OAuth for Security REST Services

- OAuth1.0 & 2.0
 - Open standard for Authorization
- OAuth 2.0 Framework and Bearer Token Usage RFC published on 2012
- OAuth2.0 - Most commonly used standard for security REST Services
 - Facebook Graph API, Google APIs
- Different implementations: e.g. WSO2 Identity Server,

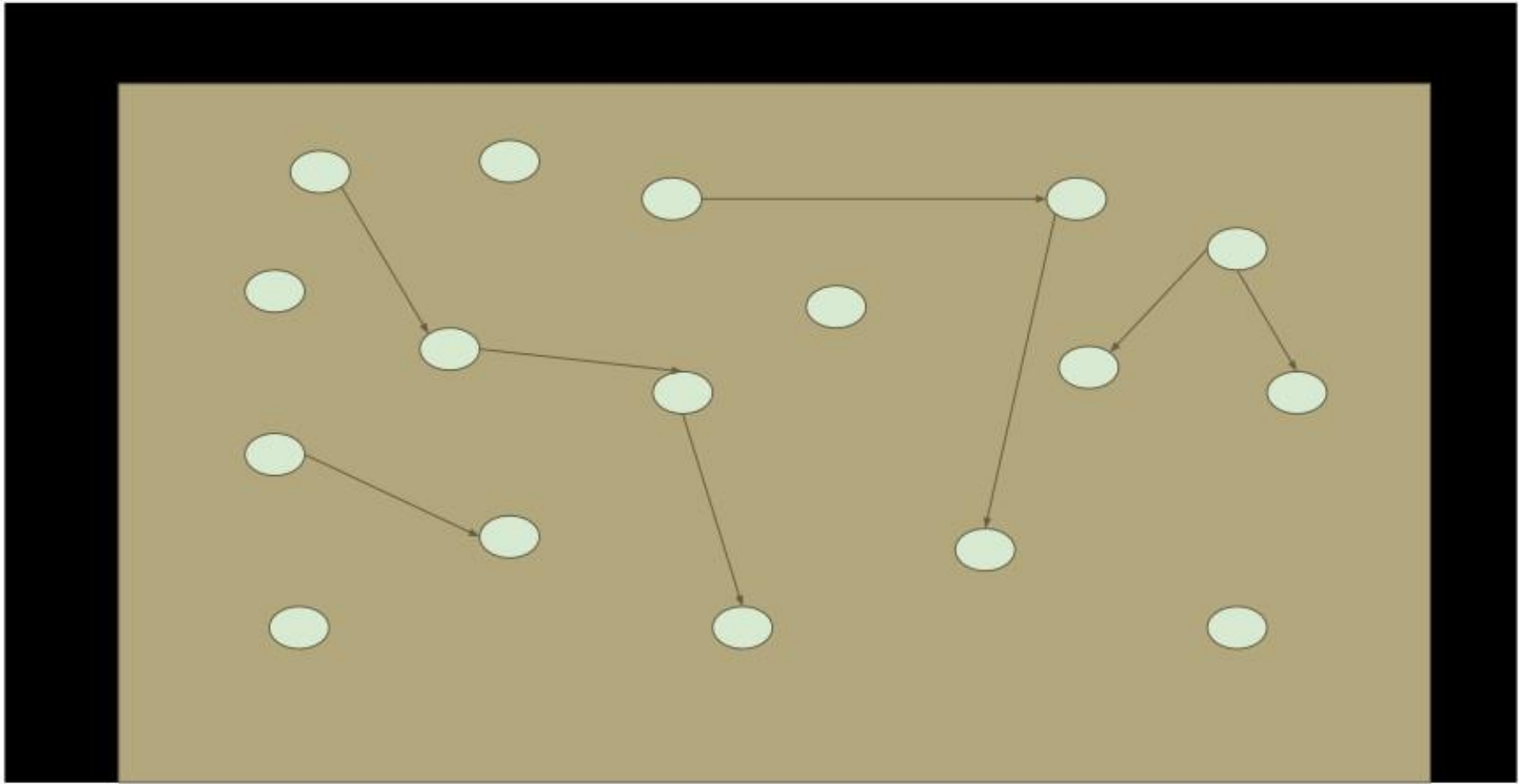
SOA Governance



SOA Governance

- The “course of action” taken to ensure that an effective decision-making process is in place
- A set of processes, responsibilities and tools, which reinforces good behavior and help avoid bad behaviors
- Ensure defined processes and responsibilities are followed through the implementation of proper measurement technique
- It’s all about control!

SOA



Why SOA Governance

- Essential part in making SOA successful
 - Need to reuse services
 - Need to make developing application, automated processes easy
 - Need to manage services

Why SOA Governance

How to promote reuse services

- List services and their descriptions
- Show the technical/business owners of a service
- Analyze inter relationship of service
- Validate the service - Check technical and business rules

Why SOA Governance

Manage Services

- Version the services
- Provide security policies at runtime
- Provide secure access to services
- Track the QoS
- Maintain lifecycle management

When is Governance Required?

- Architecture Governance
- Design-time Governance
- Run-time Governance
- Organizational Governance

How to implement SOA Governance

Service Registry is the central piece of SOA governance

- Service Catalog
- Service Description
- Service Consumption
- Service inter-dependencies
- Service Discovery
- Service Lifecycle
- Service Policies

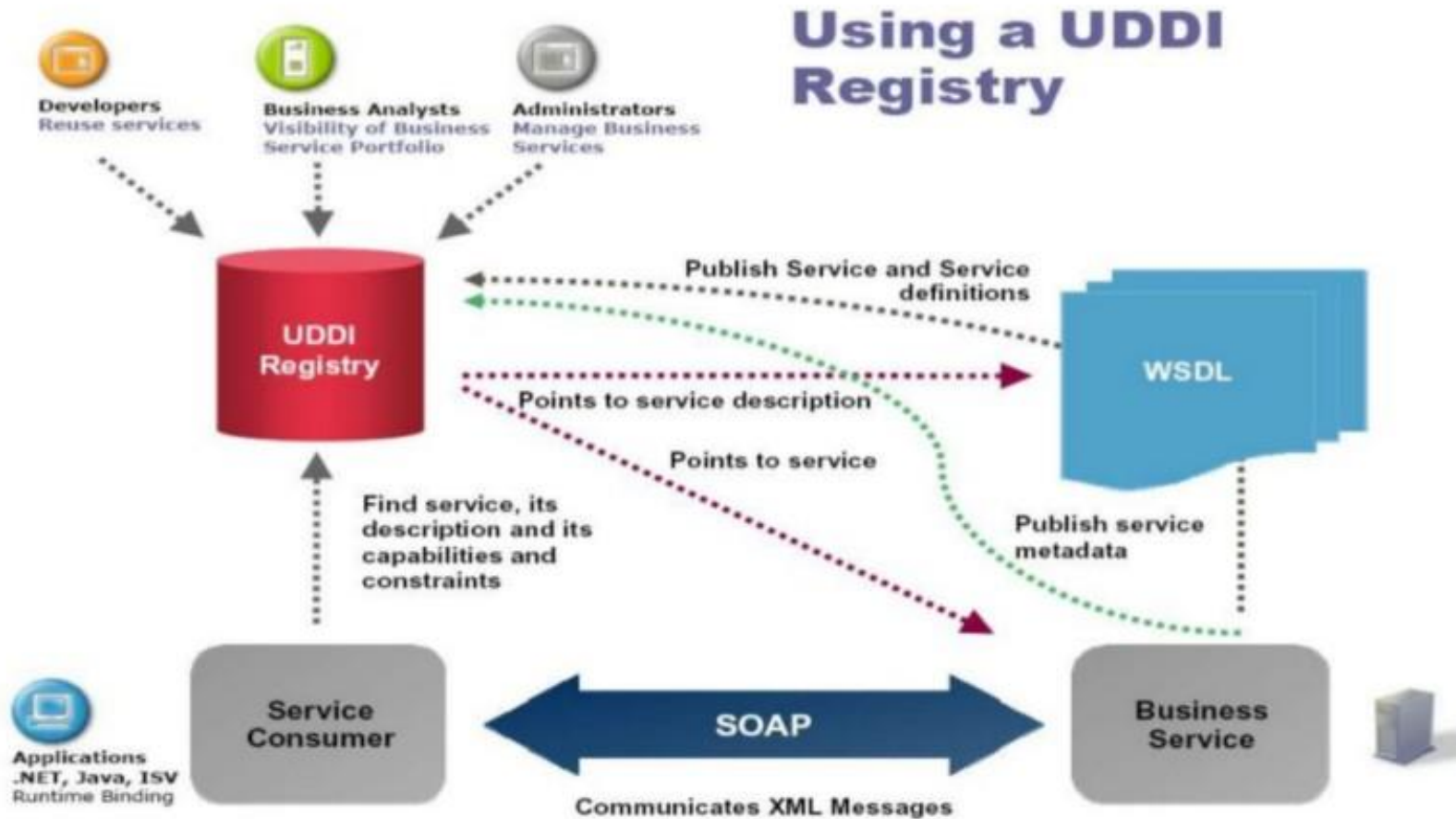
Service Description

| | |
|----------------------------|---|
| Description | Amazon E-Commerce service exposing catalog and commerce function |
| Version | 1.0.00 |
| Owner | Provider Manager |
| Status | Open |
| Creation Date | Jan 08 2007 07:50:17 PM GMT |
| Expiration Date | Jan 08 2010 07:50:17 PM GMT |
| Business Name | Amazon E-Commerce |
| Business Description | Amazon E-Commerce service (ECS) exposes Amazon's product data and e-commerce functionality. |
| Consumer Classes Supported | Internal/External |
| Service Usage Status | Active |
| Lifecycle Status | Production |
| Production Release Date | Jan 10 2007 02:15:00 PM GMT |
| WSDL URL | Amazon E Commerce.wsdl |

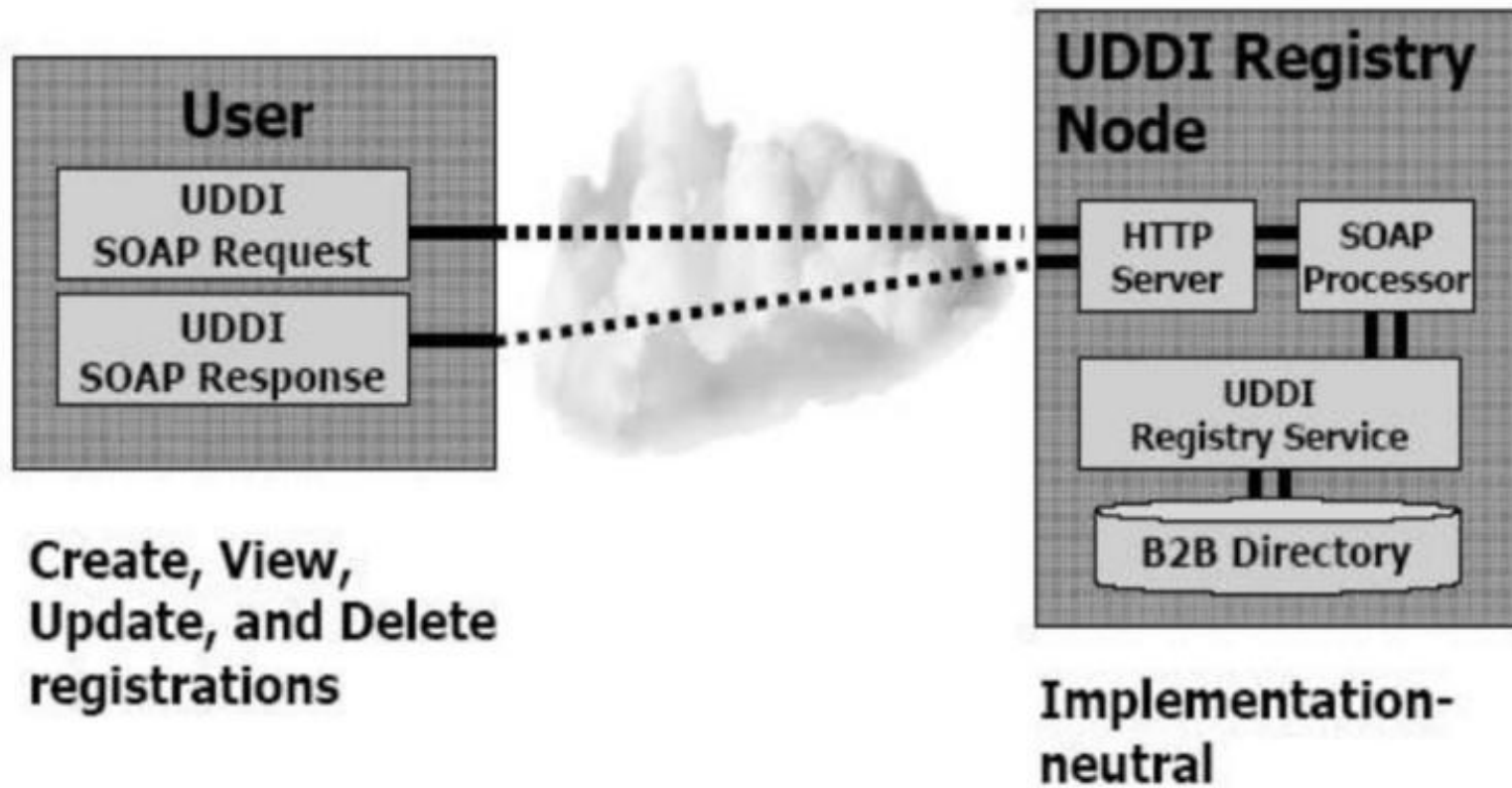
UDDI

- A registry standard - Universal Description, Discovery and Integration
- SOAP 1.1 based standard since 2000
 - Describing services
 - Publishing services
 - Discovering services
- Four components
 - Registry
 - Data, meta-data
 - UDDI specification
 - API for publishing, managing and discovering services

UDDI



UDDI



UDDI - Low Adoption Rate

- Tightly coupled to SOAP/WSDL
- No homogeneous standard in different business
- Look up services at runtime by clients
 - not a common usecase
- Original spec didn't have human friendly formats
- Hard to use (long document required)
(Governance Interoperability Framework)

Current Registry Implementations

- Infravio
- TIBCO Active Matrix
- BEA
- Oracle
- Sun
- Systinet
- WSO2

Modern Governance Products

- Human friendly service catalog
- Has REST interface - No standard
- Can publish services in different formats
- Integrates with developer tooling
- Innovation enabler, not a restrictor
- Notifications to watchers, workflows
- Lifecycle management

Summary

- Orchestration/Choreography to manage the business process
- Service integration to connect services
- Enterprise Service Bus
- Securing web services
 - WS-Security
 - HTTP + BasicAuth/HTTP+ DigestAuth
 - OAuth/OpenID
- SOA Governance