Web Service Process

Lin Zuoquan

Information Science Department

Peking University

lz@is.pku.edu.cn

http://www.is.pku.edu.cn/~lz/teaching/stm/saswws.html

Courtesy some graphic slides from online

Outline

- Web Services Process
- WS Languages

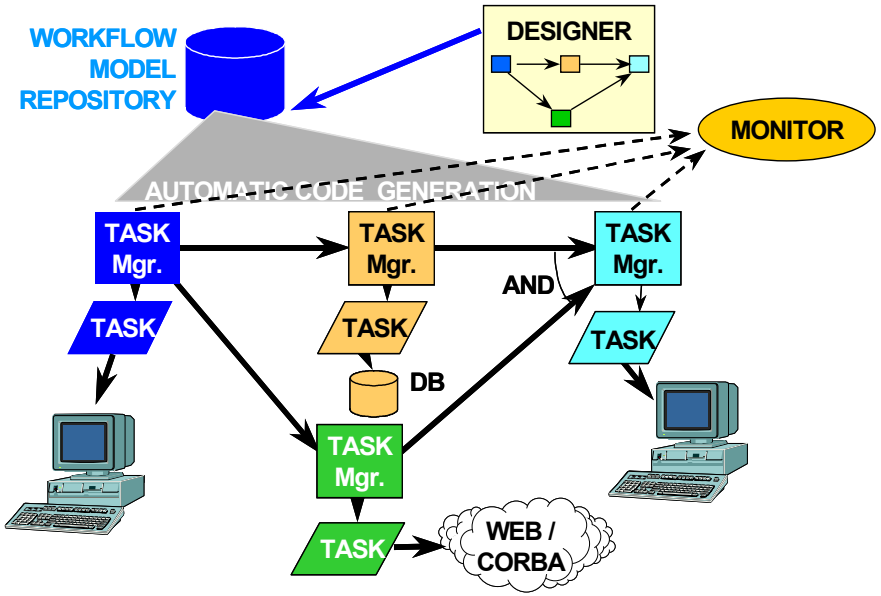
Web Services Process

Some Issues on WSP

- Service Composition
- Service Discovery
- Service Coordination and Management
- Uniform Information Exchange Infrastructure
- Fault Tolerance and Scalability
- Adaptiveness
- Reliability and Transactions
- Security
- Accountability
- Testing …

Web Services and Workflow Systems

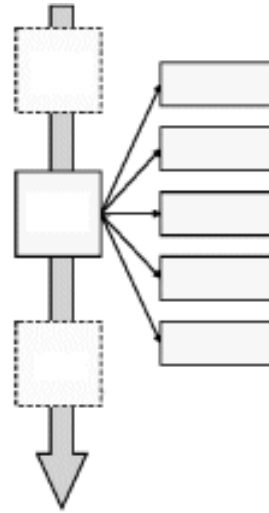
- Web Services can be orchestrated with hard-coded applications or by using workflows.



Workflow management systems (WfMS) are capable of integrating business objects for setting up e-services (Web Services) in an amazingly short time and with impressively little cost.

Workflow

"A process means the treatment of objects. Thus, it consists of activities and objects"

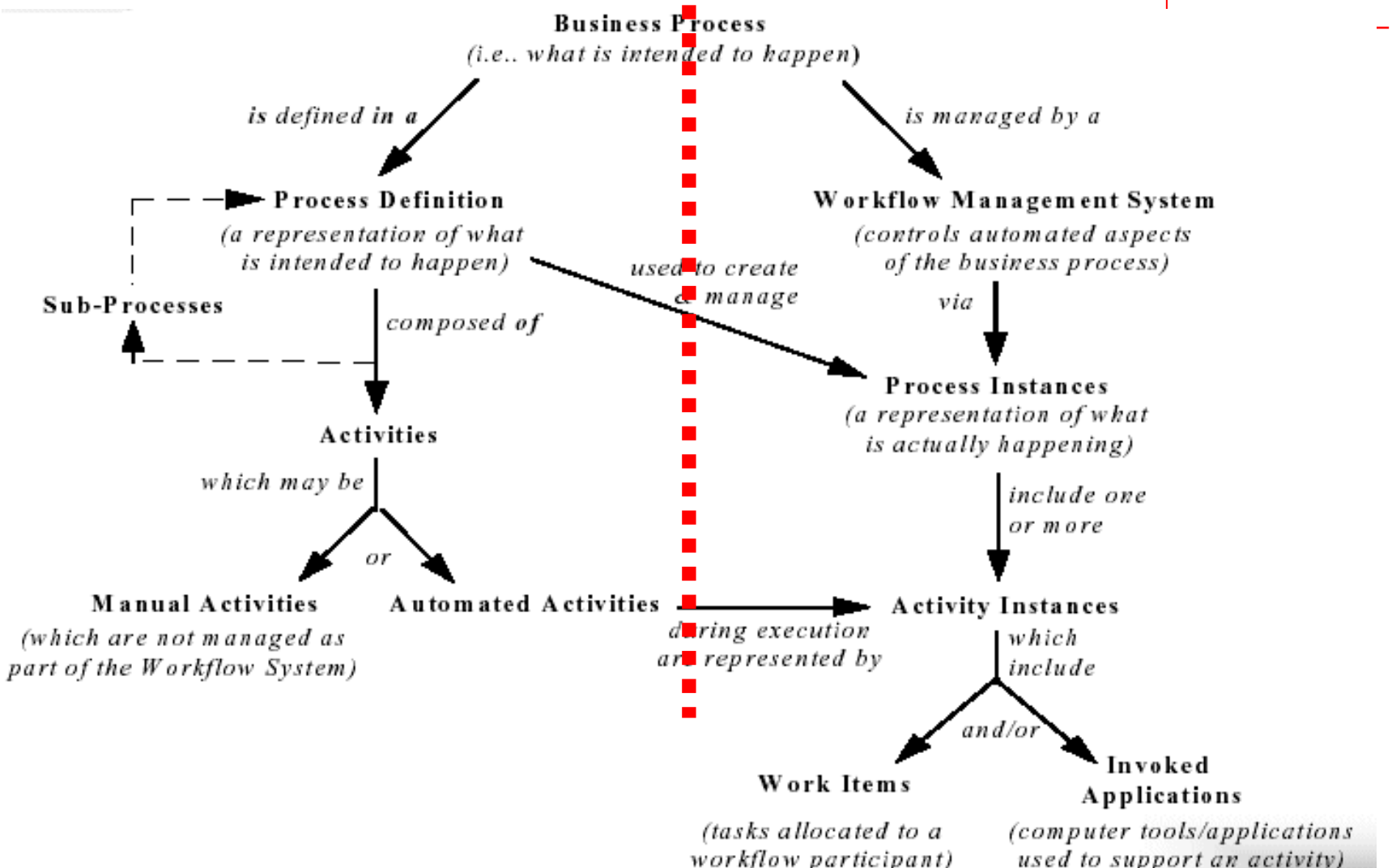


Workflow = Automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedure rules (WFMC)

Interoperability of workflow products

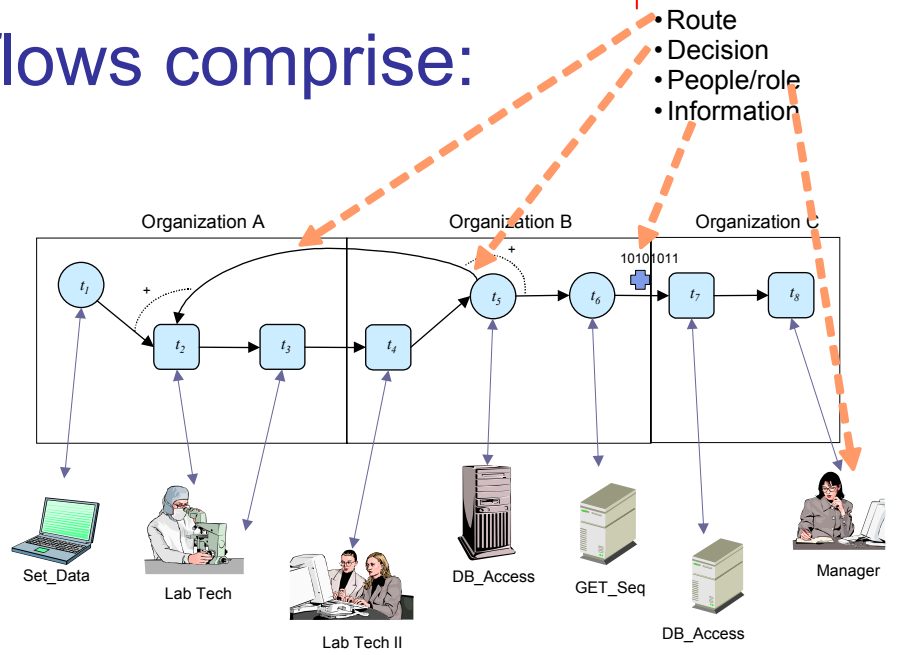
http://wfmc.org

Relationships among Workflow concepts



WS Processes and Workflows

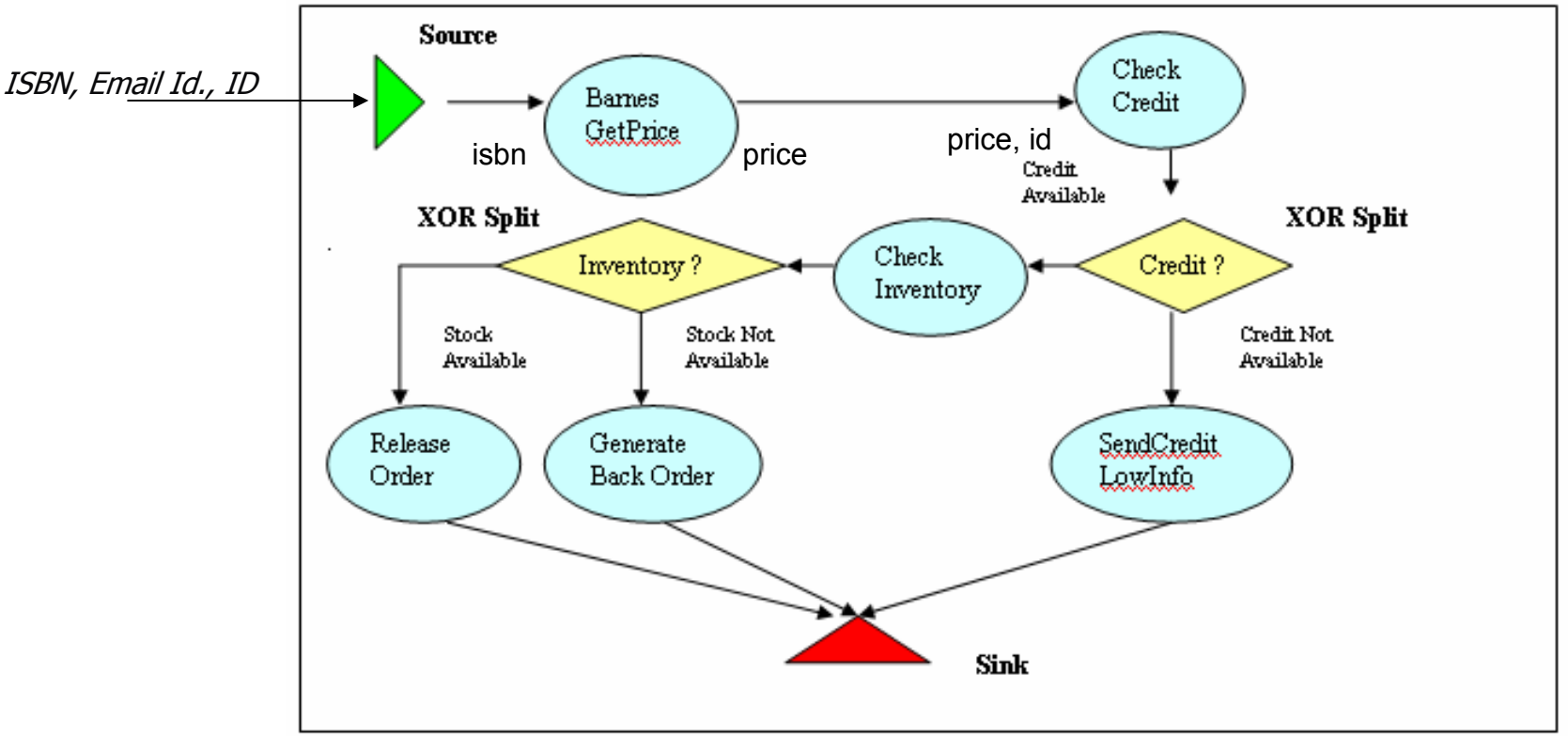
- Web Processes/Workflows comprise:
  - Web Services/Tasks,
  - Routing rules,
  - Decisions,
  - Participants and,
  - Information



A WfMS is a system or set of tools that completely defines, manages, and executes a workflow or Web Process.

Processes
A simple example

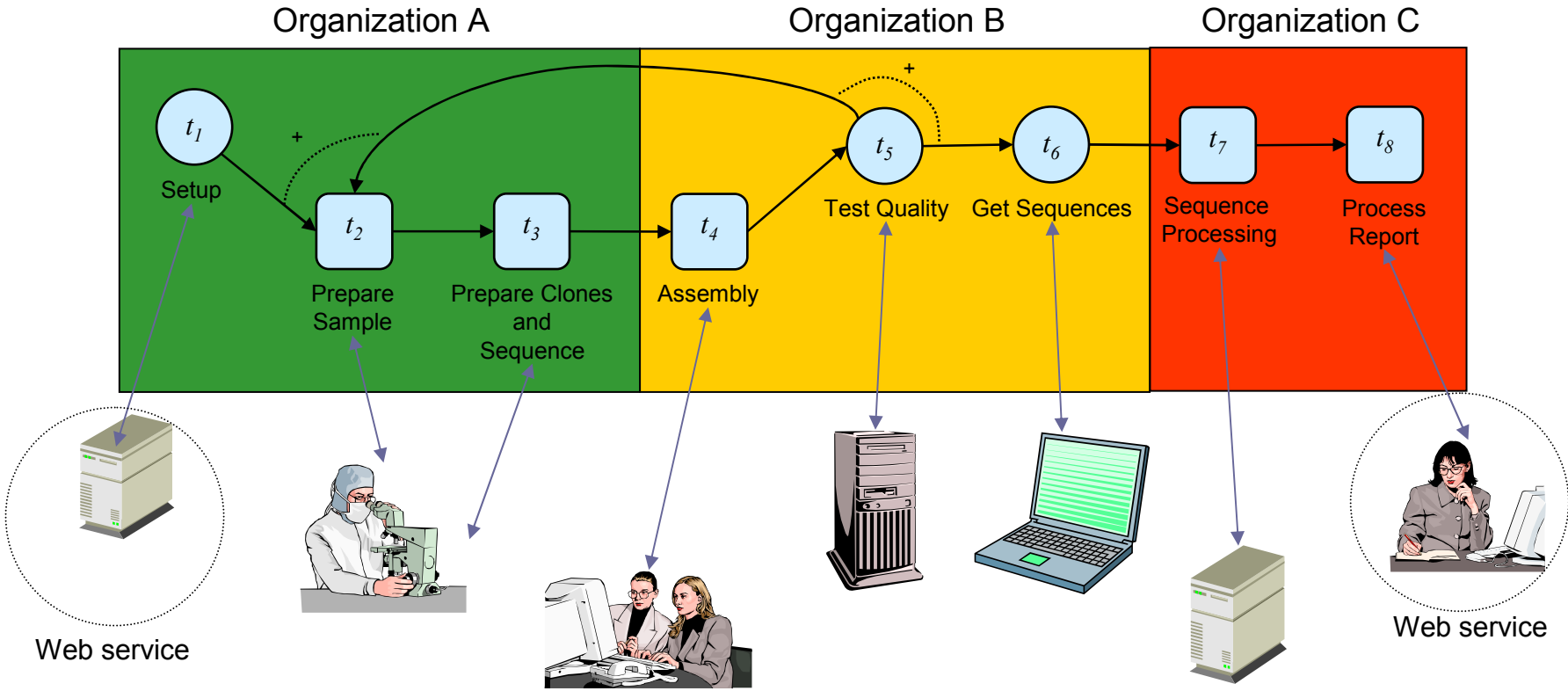
- A process is an abstract representation of a business process.



The BarnesBookPurchase process

Processes
A more complex example

A Web process can be viewed as a workflow for which the tasks are represented with Web services



Processes
Execution

- Once the design of a process is completed, it can be executed.
- Processes can be executed with hard-coded applications or by using workflows.
- Workflows are enacted with Workflow Management System (WfMS) or other process orchestration technology.

Process Composition

The composition of cross-organizational Internet-based processes requires new technological developments which include:

- **Discovery of Web Services**
- **Integration of Web Services**
- **End-to-End Process Analysis**
  - **Correctness/validation, performance**

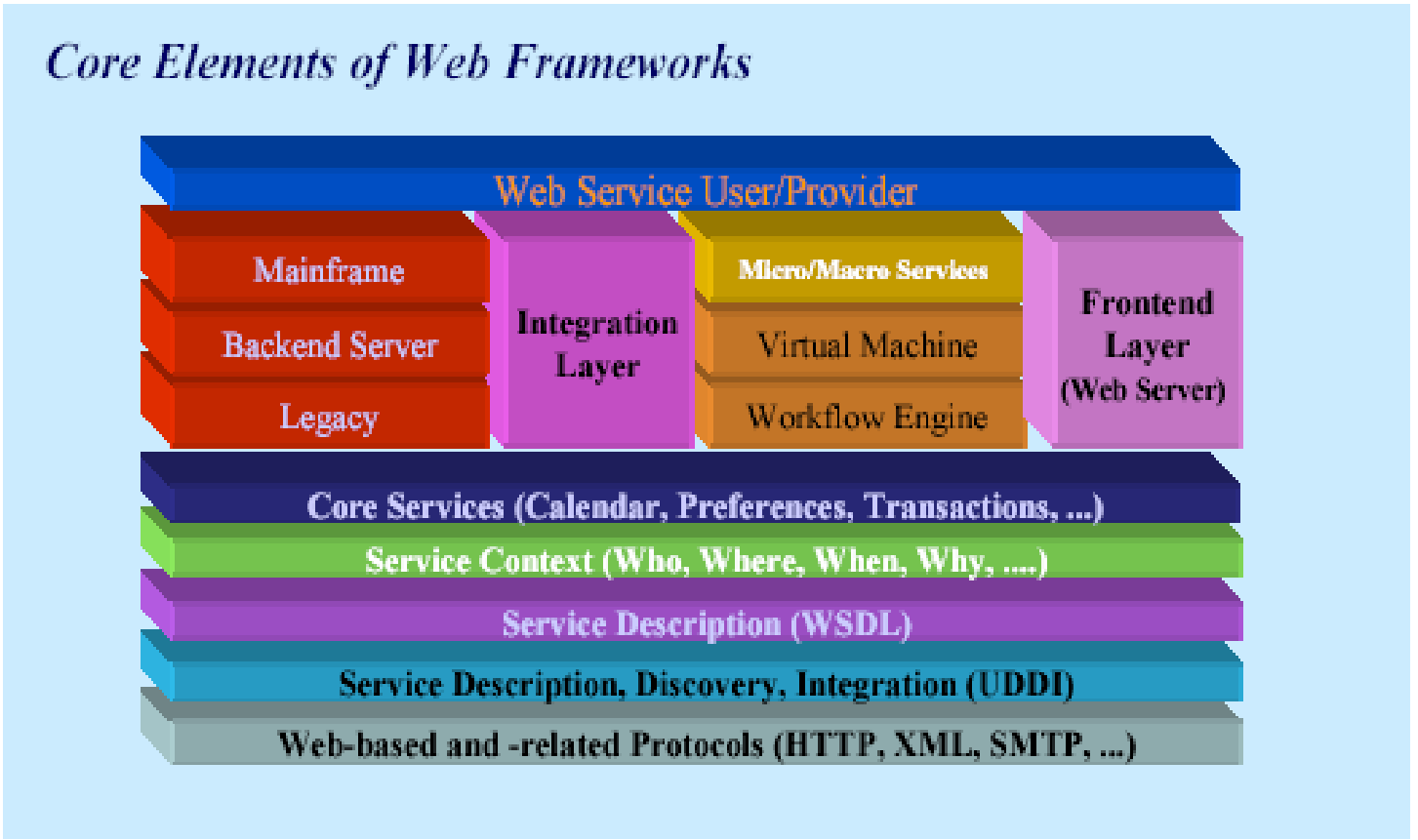
Composition
Classification

- Proactive Composition & Reactive Composition
  - Proactive: offline composition of available services
    - When: services are stable and always running
    - Example: ticket reservation service
  - Reactive: dynamically creating a composite service.
    - When: composite service not often used and service processes not stable.
    - Example: tour manager where the itinerary is not predefined

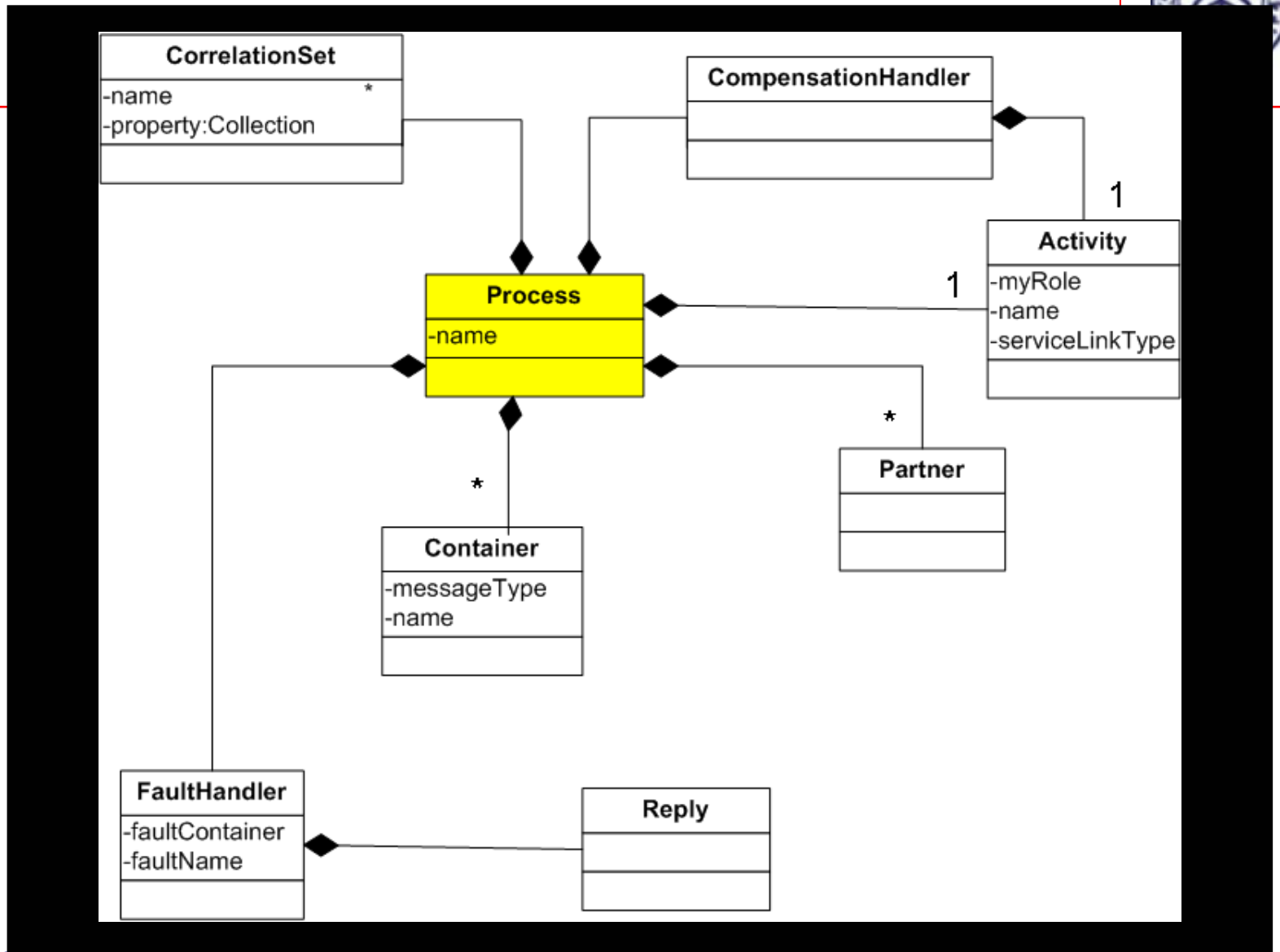
Classification(contd)

- Mandatory & Optional-Composite Services
  - Mandatory: all subcomponents must participate to yield a result
    - Example: service that calculates the averages of stock values for a company.
  - Optional: subcomponents are not obligated to participate for a successful execution.
    - Example: services that include a subcomponent that is an optimizer.

Web Frameworks



Metamodel


CorrelationSet
-name
-property:Collection
*
CompensationHandler
1
Activity
-myRole
-name
-serviceLinkType
1
Process
-name
*
Partner
*
Container
-messageType
-name
FaultHandler
-faultContainer
-faultName
Reply

WS Languages

- WSFL (Web Services Flow Language)
- BPML (Business Process Modeling Language)
- ebXMl BPSS (Business Process Specification Schema)
- XLANG
- BPEL4WS (Business Process Execution Language for Web Services)

WSFL in brief

- WSFL is an XML language for the description of WS **Composition**
- It is possible to produce web services by composing web services
  - Intra-enterprise web services might collaborate to present a single Web Service Interface to the public
  - Web services from different enterprises might collaborate to perform business to business transactions
- Service Flow described how service-to-service communications, collaborations and flows are performed.

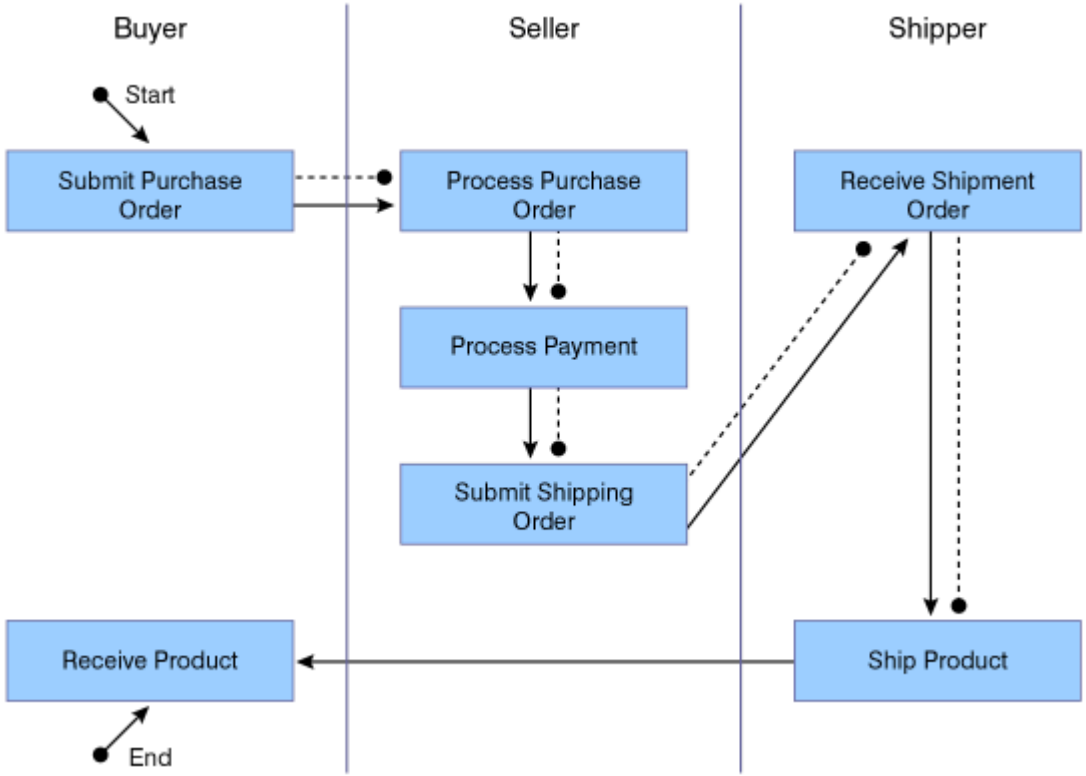
WSFL

- Composition of Service
  - Flow Model – execution sequence of activities
  - Global Model – interactions of services
  - Recursive Composition – scalability
- It is layered on the top of WSDL

Workflow
Business Process Modeling

- **(Business) Process**: any collection of activities that when combined accomplish a given business objective. E.g., processing an order.
- **Flow Model**: the actual XML representation of the **Directed Graph** that models the business process and use to compose Web services as defined by WSDL (synonyms : flow composition, orchestration, and choreography).
- **Global Model**: a set of the necessary links that specify how messages can be sent between the Web services in the flow model as the flow is executed.
- **Recursive Composition**: recursively compose WSFL business processes within existing WSFL business processes (flexibility and fine granularity).
- **Service Provider**: the party responsible for performing a particular activity within a process (using WSDL).
- **Service Provider Type**: the type defined by a *Web Service Interface* (using WSDL).
- **Control Link**: a control link is the WSFL equivalent to the directed edge; i.e., the workflow processor walks through each of the activities in the process.
- **Data Link**: the data link is the mechanism that the workflow processor uses to control the flow of data through the process.
- **Transition Conditions**: a transition condition is a true or false statement that the processor may use to determine the current state of any particular activity.
- **Lifecycle Interface**: the WSDL-defined Web service interface that describes the basic set of operations that all WSFL Web services support within a particular Web services application.

A business process



Getting into the flow

```
  &lt;activity name="submitPO"&gt;
  &lt;performedBy serviceProvider="buyer"/&gt;
  &lt;implement&gt;
     &lt;export&gt;
        &lt;target portType="totalSupplyPT“ operation="submitPO"/&gt;
     &lt;/export&gt;
  &lt;/implement&gt;
&lt;/activity&gt;
&lt;activity name="processPO"&gt;
  &lt;performedBy serviceProvider="seller"/&gt;
  &lt;implement&gt;
    &lt;export&gt;
      &lt;target portType="receivePO“ operation="receivePO"/&gt;
    &lt;/export&gt;
  &lt;/implement&gt;
&lt;/activity&gt;
&lt;controlLink source="submitPO" target="processPO"/&gt;
```

BPML in brief

- BPML (http://www.bpmi.org)
  - BPML is a workflow definition language with no references to web services or their composition
  - Data format is XML since process language contains XPATH expressions
  - Very elaborate process model that includes concepts for
    - Inter-workflow communication (message exchange between ongoing workflow instances)
    - Participants
    - Closed and open transactions
    - Compensation
    - recovery

Example

```
 <process name = “TrackTrouble”>
<supports abstract = “Customer”/>
<message name = “troubleReportinput” type = “request”>
        <xsd:element name = “service” type = “Service”/>
        <xsd:element name = “trouble” type = “xsd:string”/>
</message>
<message name = “troubleReportoutput” type = “response”>
        <xsd:element name = “cookie” type = “TrackTrouble”/>
</message>
<message name = “getStatusinput” type = “request”/>
<message name = “getStatusOutput” type = “response”>…</message>
<sequence name = “reportAndTrack”>
        <operation name = “report Trouble”>
                <participant name = “reportTroubleForm”/>
                <input name = “trobleReportinput”/>
                <output name = “trobleReportOnput”>
                        <assigned target = “cookie” select = “TrackTrouble/text()”/>
                </output>
        </operation>
```

Example (contd.)

```
        <operation name = “findProvider”>
                <participant select = “troublereportinput/service”/>
                <output message = “getproviderinput”/>
                <input message = “getproviderOutput”/>
        </operation>
        <operation name = “createticket”>
                <participant select = “getProviderOutput/provider”/>
                <output message = “openTicketinput”>
                        <assign select = “troubleReportinput/trouble”/>
                        <assign select = “trackTrouble/text()” target = “customer”/>
                </output>
                <input message = “openTicketOutput”/>
        </operation>
        <consume name = “notifyCustomer”>
                <input message = “ticketClosed”/>
        </consume>
</sequence>
</process>
```

ebXML in brief

- ebXML BPSS (http://www.ebxml.org)
  - Look under “Specifications”
  - ebXML BPSS is a Process Specification Language
  - Specific emphasis on document exchange
    - Business data messages
    - Acknowledgement messages
  - Message activities
    - Non-repudiation
    - Confidential
    - Encrypted

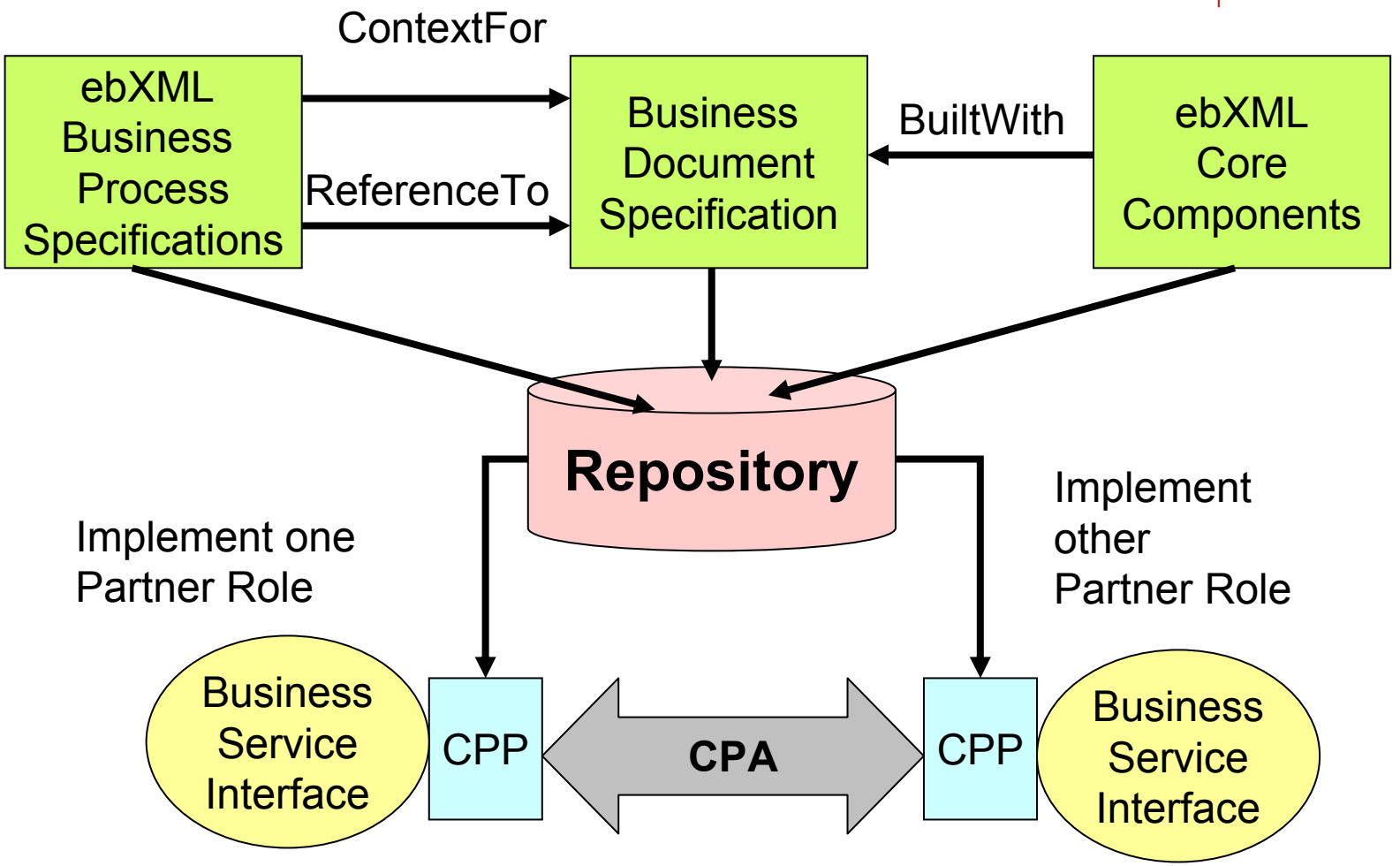
ebXML

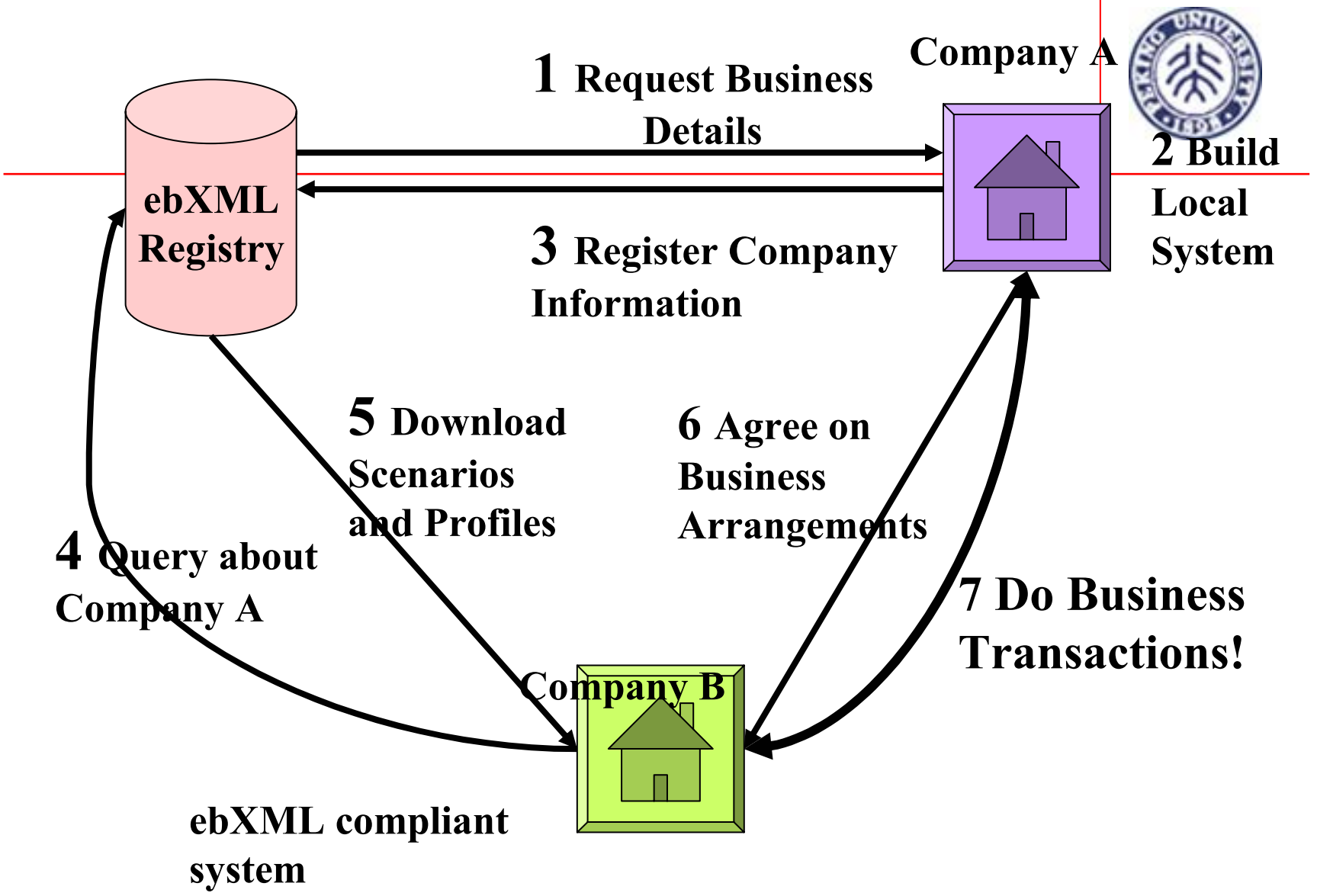
- A joint global initiative by
  - **UN/ CEFACT** – United Nations Center For Trade Facilitation And Electronic Business
  - **OASIS** – Organization for the Advancement of Structured Information Standards
- United Nations Center for Trade Facilitation and Electronic Business
  - Sets worldwide policy and technical development in trade facilitation and electronic business
  - Developed international EDI standard, UN/ EDIFACT
- The initiative leverages from the success of EDI in large businesses, and intends reaching small and medium enterprises

ebXML Architecture

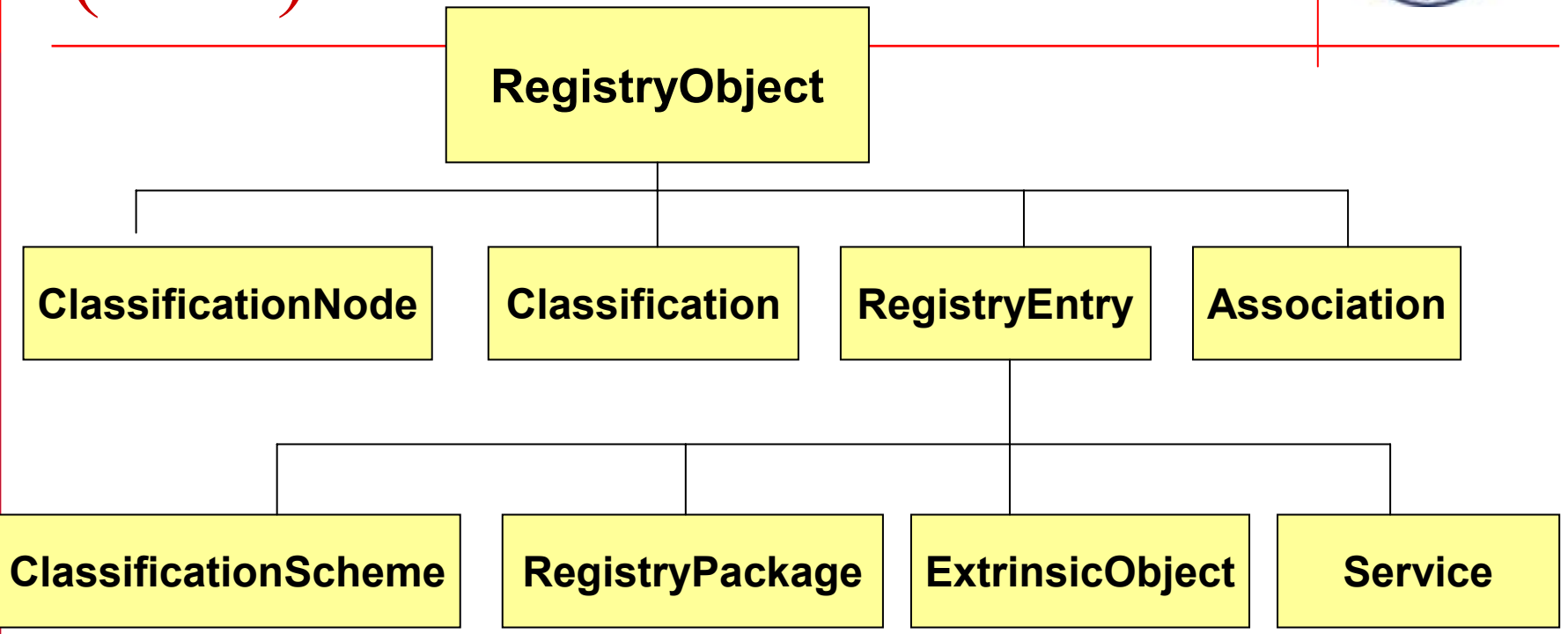
- **Business Process Specification Schema (BPSS)** Business processes are not fixed!
- **Trading Partner Information**
  - **Collaboration Protocol Profile CPP**
- **Trading Partner Agreement**
  - **Collaboration Protocol Agreement CPA**
- **Registry/Repository**
- **Messaging Service**
- **Core Components**

An Overview of Functional Components in ebXML




Company A
1 Request Business Details
2 Build Local System
ebXML Registry
3 Register Company Information
5 Download Scenarios and Profiles
6 Agree on Business Arrangements
4 Query about Company A
7 Do Business Transactions!
Company B
ebXML compliant system

ebXML Registry Information Model (RIM)



XLANG in brief

- XLANG (www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.htm)
  - Extension of WSDL for behavior definition
  - Main constructs (block structured)
    - Activation operation, i.e WSDL operation that starts the behavior
    - Operation, delayFor, delayUntil, raise
    - Empty, Sequence, Switch, While, All, Pick
    - Correlation
    - Context

Example

```
<?xml version = “1.0”?>
    <definitions name=“StockQuoteProvider”..>
    <service>
      <xlang:behaviour>
                <xlang:body>
                        <Xlang:sequence>
                        <xlang:action operation=“AskLastTradePrice”
                                port=“pGetRequest” activation=“true”/>
                        <xlang:action operation=“SendLastTradePrice”
                                port=“pSendRespnse”/>
                         </Xlang:sequence>
                </xlang:body>
      </xlang:behaviour>
    </service>
    </definitions>
```

BPEL4WS

- BPEL4WS enables:
  - Defining business processes as coordinated sets of Web service interactions.
  - Define both abstract and executable processes.
    - Abstract processes are for e-commerce specifications.
    - Executable processes provide a model to integrating enterprise applications.
  - BPEL enables the creation of compositions of Web services
    - Composition based on abstract descriptions

- Where it comes from:
  - Strong roots in traditional flow models.
  - Plus many concepts from structured programming languages.
  - All laid on top of WSDL and core XML specifications.
  - Merges WSFL and XLANG concepts.

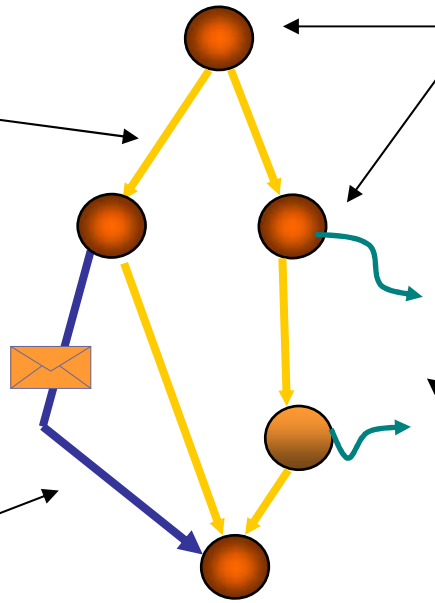
Structure of a BPEL4WS Process

```
<process ...>

   <partners> ... </partners>
        <!-- Web services the process interacts with -->
   <containers> ... </containers>
        <!– Data used by the process -->
   <correlationSets> ... </correlationSets>
        <!– Used to support asynchronous interactions -->
   <faultHandlers> ... </faultHandlers>
        <!–Alternate execution path to deal with faulty conditions -->
   <compensationHandlers> ... </compensationHandlers>
        <!–Code to execute when “undoing” an action -->
   (activities)*
        <!–  What the process actually does -->

</process>
```

Traditional Flow Models



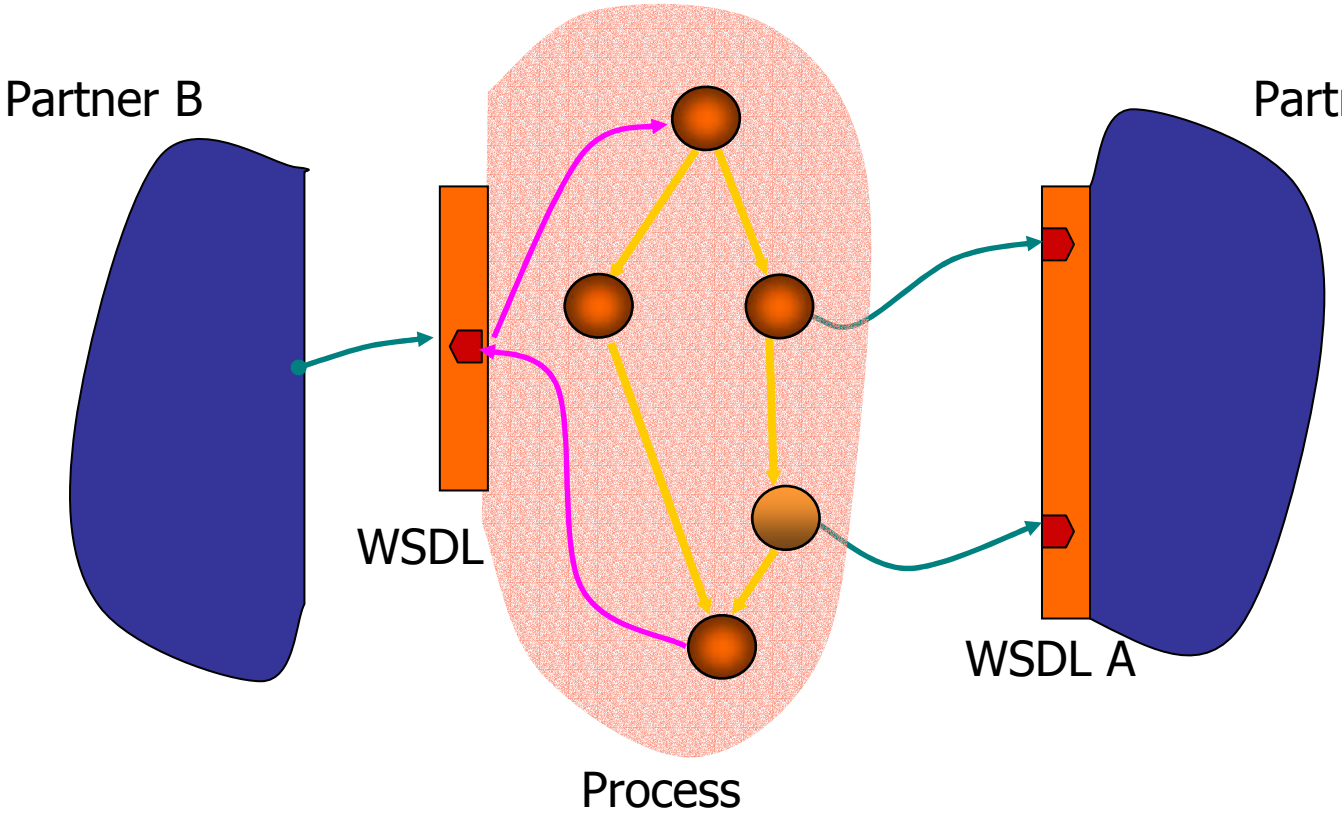
Control links define execution flow as a directed acyclic graph

Activities represent units of processing.

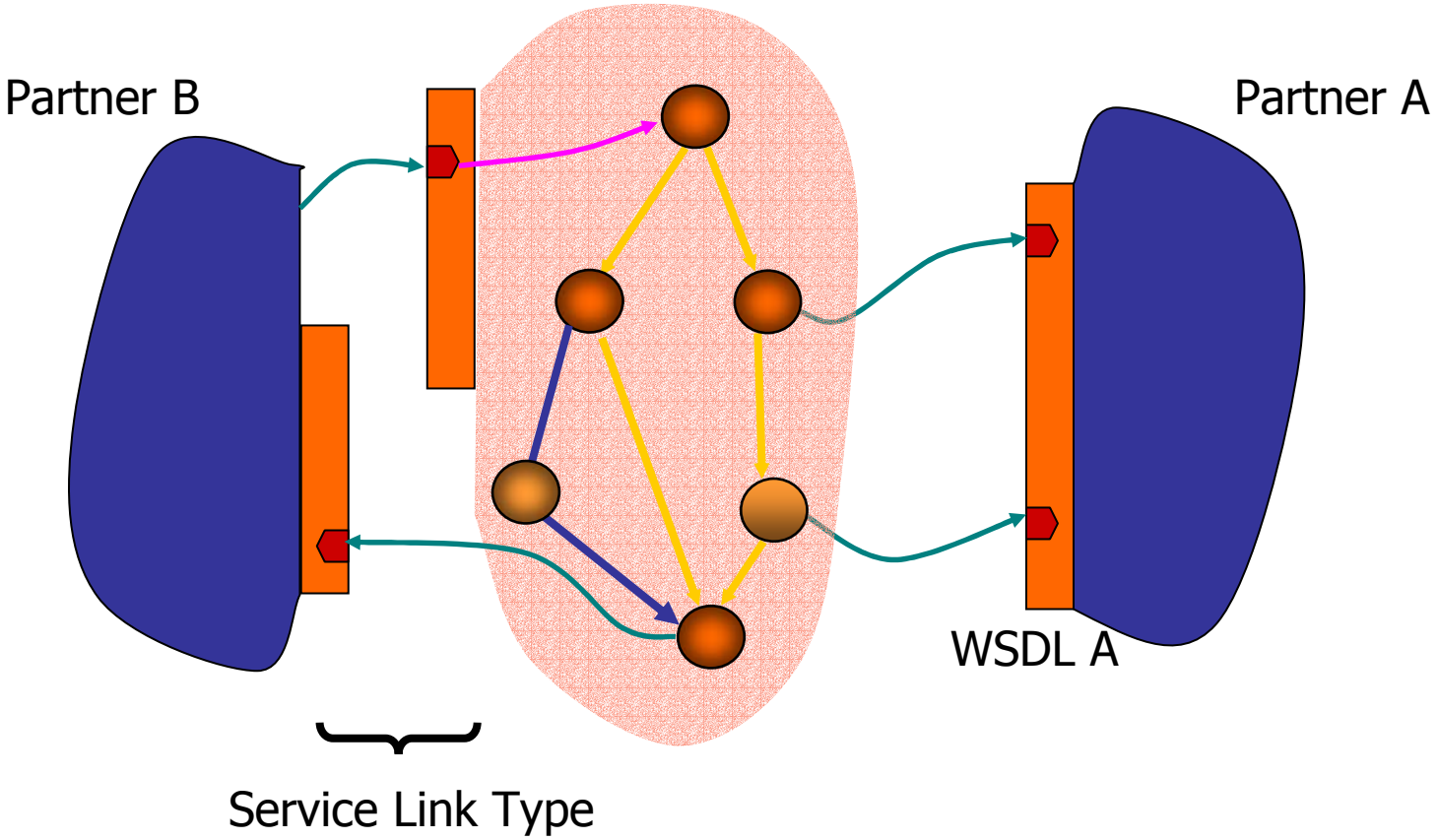
Flow of data is explicitly modeled through data links.

Activities are mapped to application invocations or human actions

BPEL and WSDL - Partners



BPEL and WSDL - Partners



Partner Definitions and Links

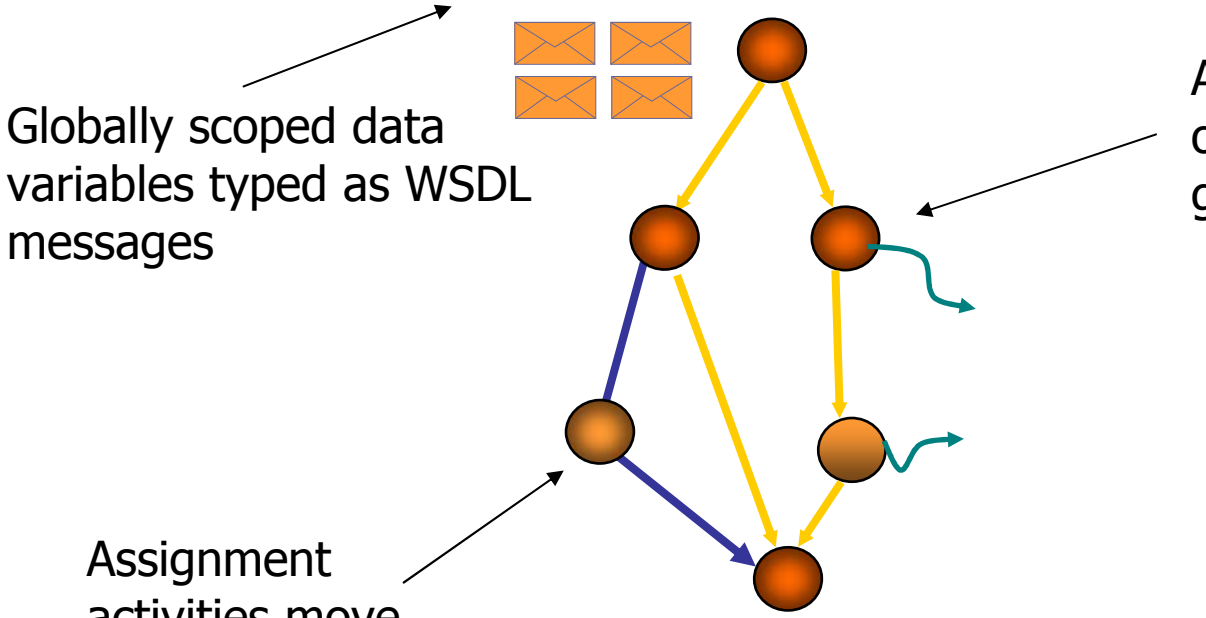
```
<partner name="..." serviceLinkType="..."
         partnerRole="..." myRole="..."/>
```

<!– A partner is accessed over a WS “channel”, defined by a service link type -->

```
<serviceLinkType name="...">
  <role name="...">
    <portType name="..."/>*
  </role>
  <role name="...">
    <portType name="..."/>*
  </role>
</serviceLinkType>
```

<!– A SLT defines two roles and the portTypes that each role needs to support -->

BPEL Data Model



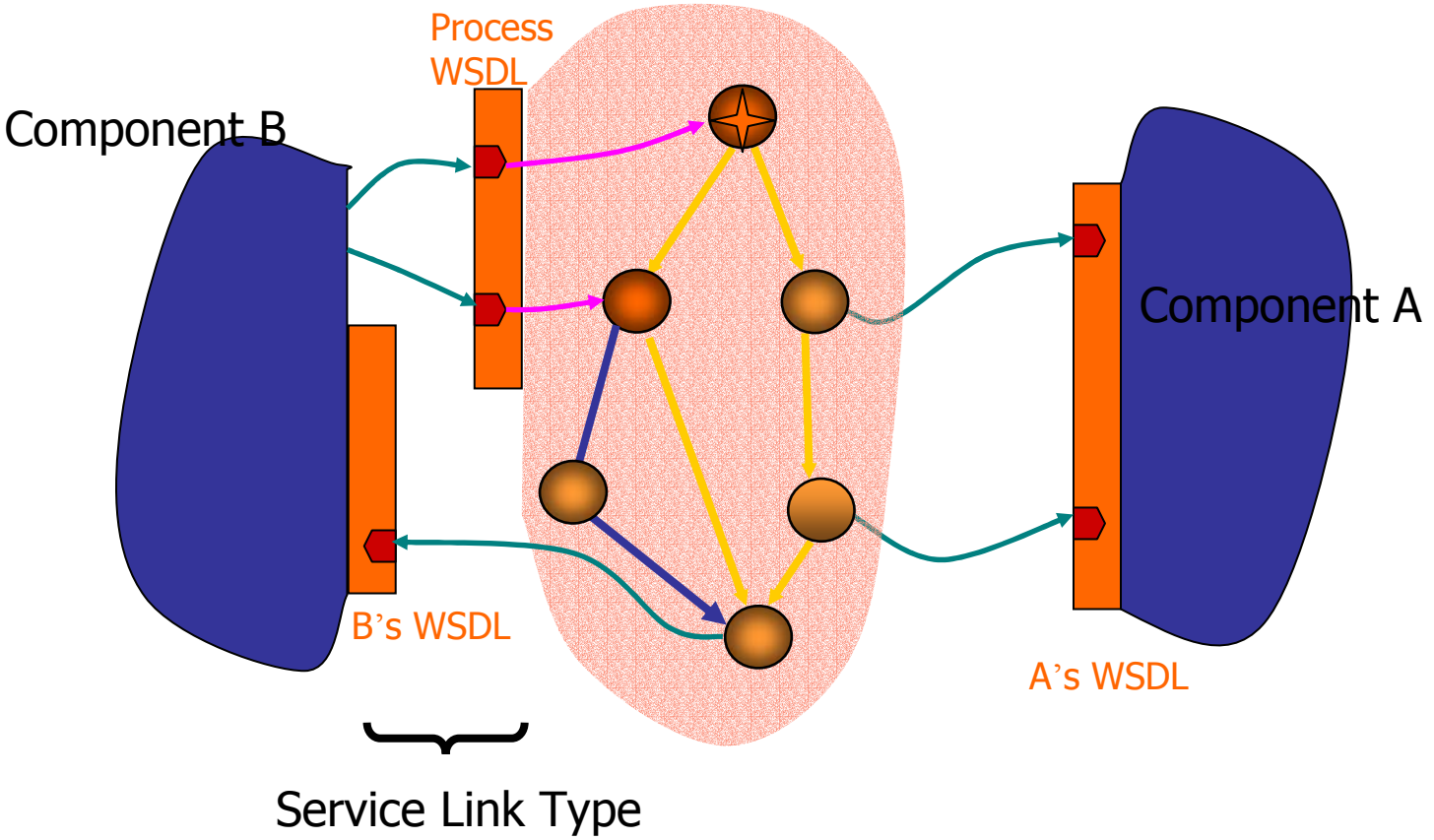

```
<container name="..." message="..."/>*
```

BPEL Basic Activities

```
<invoke partner="..." portType="..." operation="..."
        inputContainer="..." outputContainer="..."/>
   <!-- process invokes an operation on a partner:      -->

<receive partner="..." portType="..." operation="..."
         container="..." [createInstance="..."]/>
   <!-- process receives invocation from a partner:      -->
<reply partner="..." portType="..." operation="..."
       container="..."/>
   <!-- process send reply message in partner invocation:     -->
<assign>
   <copy>
     <from container="..."/> <to container="..."/>
   </copy>+
</assign>
   <!– Data assignment between containers:      -->
```

BPEL Composition of Web Services



More Basic Activities

```
<throw faultName="..." faultContainer="..."/>
```

<!-- process detects processing error and switches into fault processing mode -->

```
<terminate/>
```

<!– pull the plug -->

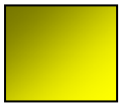
```
<wait for="..."? until="..."?/>
```

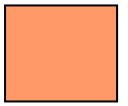
<!-- process execution stops for a specified amount of time-->

```
<empty>
```

<!– Do nothing; a convenience element -->

BPEL Structured Activities





```
<sequence>
```
<!– execute activities sequentially-->

```
<flow>
```
<!– execute activities in parallel-->

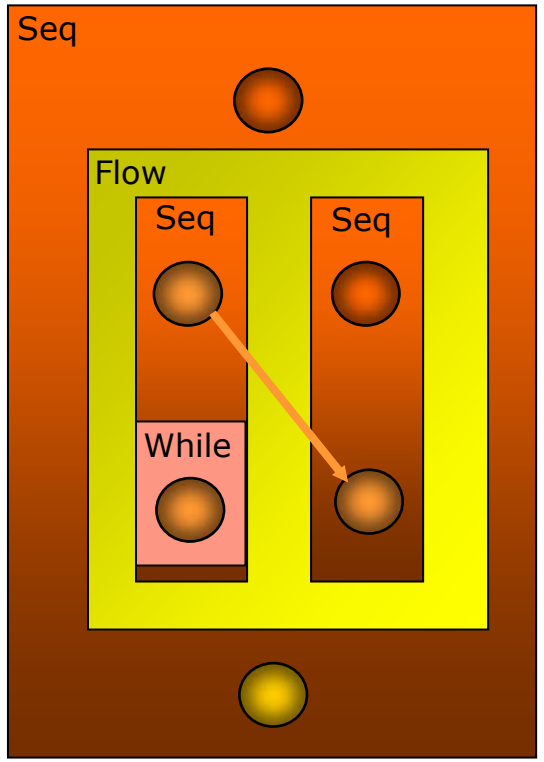
```
<while>
```
<!– iterate execution of activities until condition is violated-->

```
<pick>
```
<!– several event activities (receive message, timer event) scheduled for execution in parallel; first one is selected and corresponding code executed. -->

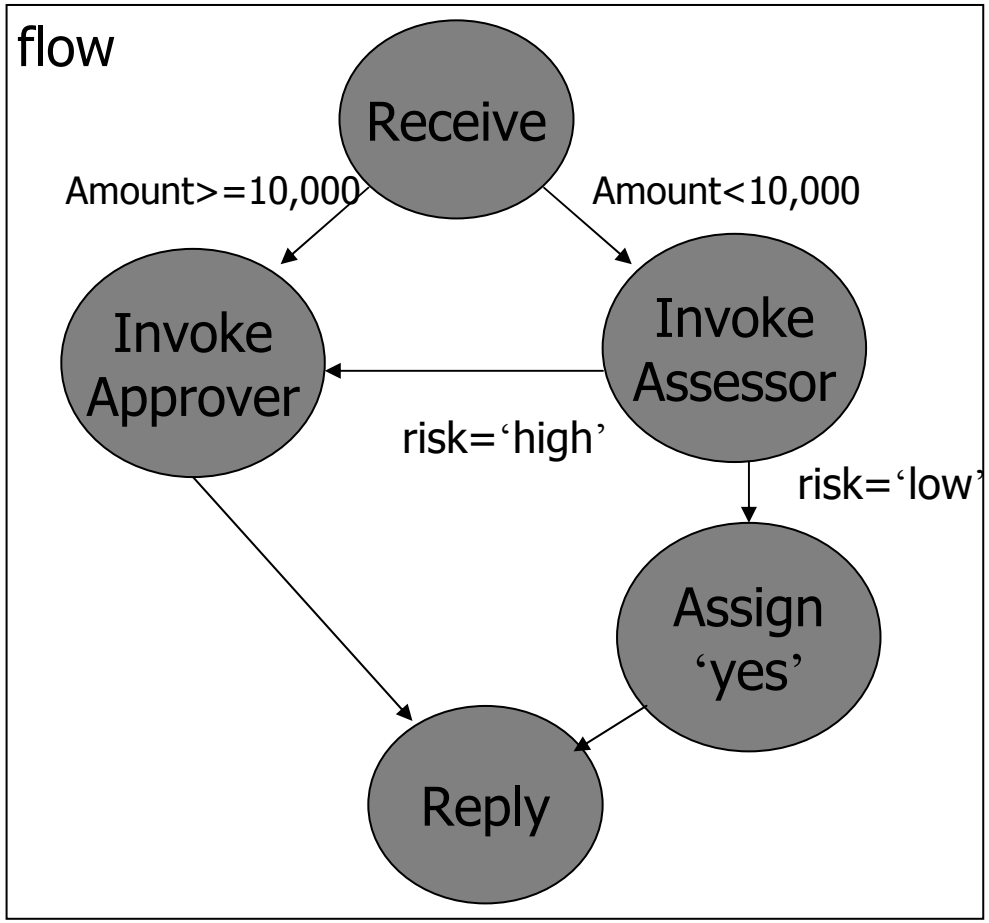
```
<link ...>
```
<!– defines a control dependency between a source activity and a target -->

Nesting Structured Activities. Example

```
<sequence>
   <receive .../>
   <flow>
      <sequence>
         <invoke .../>
         <while ... >
            <assign> ... </assign>
         </while>
      </sequence>
      <sequence>
         <receive .../>
         <invoke ... >
      </sequence>
   </flow>
   <reply>
</sequence>
```



"Flow Oriented" Authoring Style

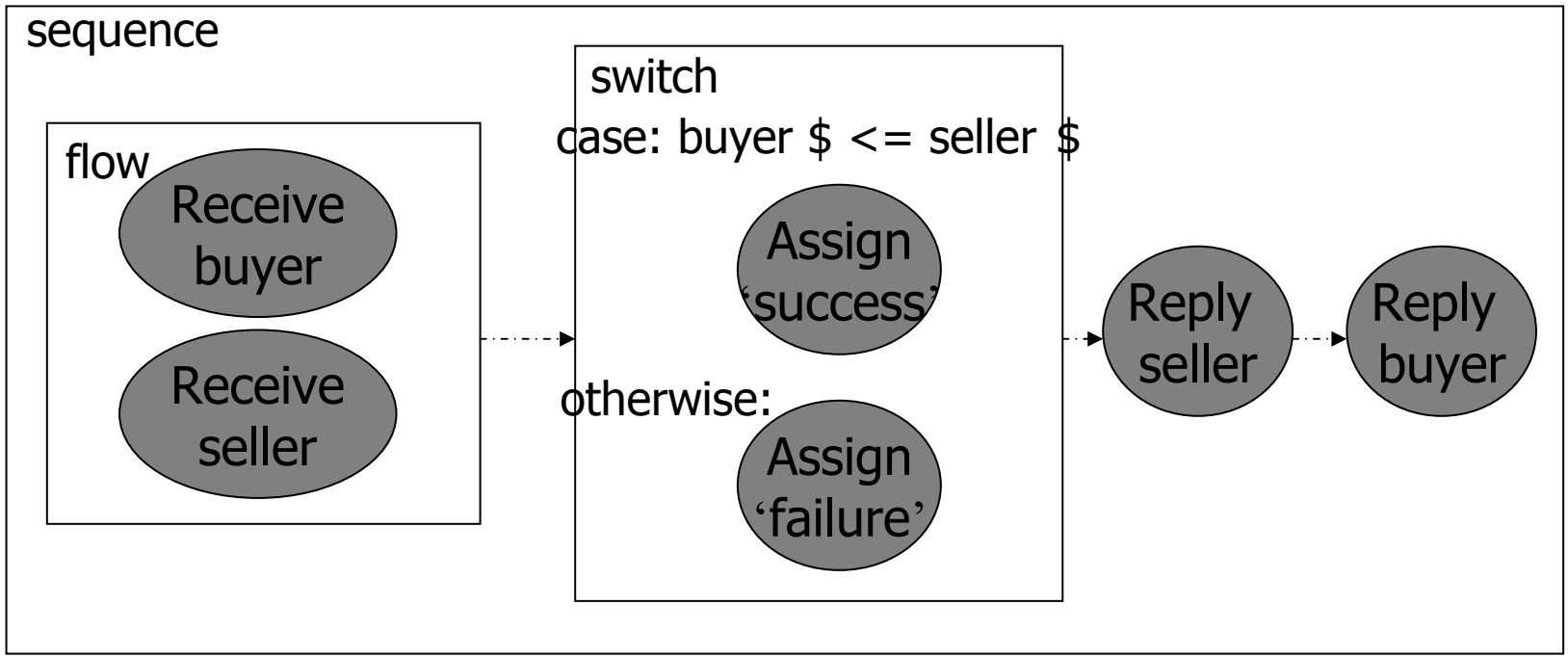


Customer asks for a loan, giving name and amount info. Two services are involved:

-A risk assessor which can approve the loan if the risk is low
-A loan approver which checks the name and decides whether to approve the loan.

The reply goes back to the customer.

“Structured” Authoring Style

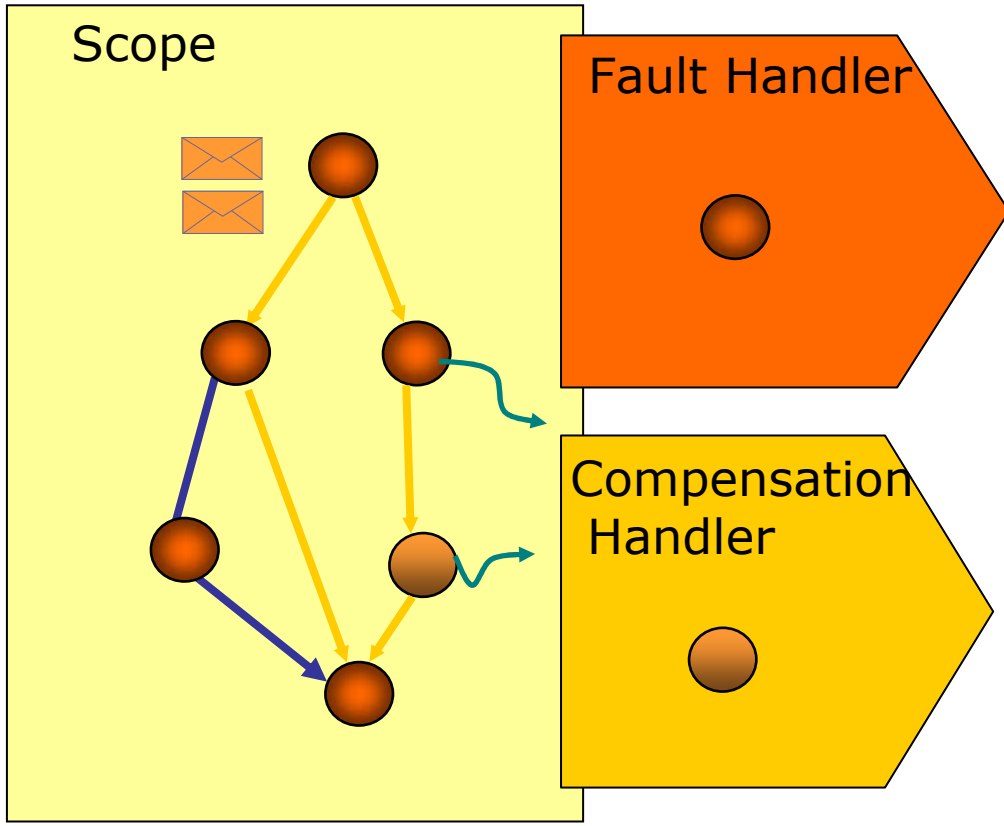


BPEL Handlers and Scopes

A **scope** is a set of (basic or structured) activities.

Each scope can have two types of **handlers** associated:

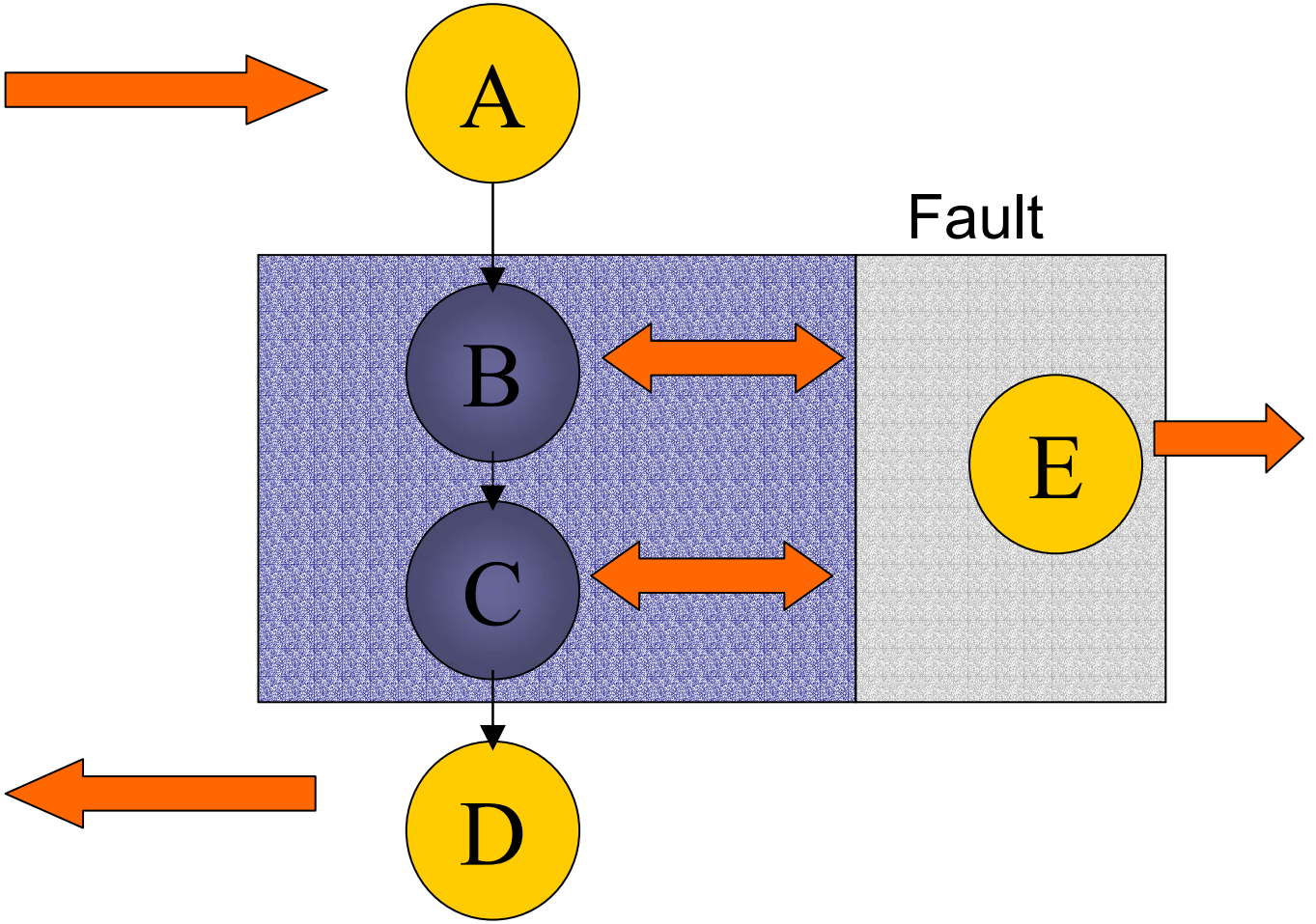
- **Fault handlers.** Many can be attached, for different fault types.
- **Compensation handlers.** A single compensation handler per scope.



How Handlers Work

- A compensation handler is used to reverse the work performed by an already completed scope
  - A compensation handler can only be invoked by the fault handler or compensation handler of its immediate enclosing scope
- A fault handler defines alternate execution paths when a fault occurs within the scope.
- Typical scenario:
  1. Fault is thrown (retuned by invoke or explicitly by process)
  2. Execution of scope is terminated
  3. Appropriate fault handler located (with usual propagation semantics)
  4. Main execution is compensated to “undo” business effects of unfinished work.

Scope and Fault Example



What is Correlation?

- BPEL4WS can model many types of interactions:
  - simple stateless interactions
  - Stateful, long running, asynchronous interactions.
- Correlation sets (CSs) provide support for the latter:
  - CSs represent the data that is used to maintain the state of the interaction (a “conversation”).
  - At the process end of the interaction, CSs allow incoming messages to reach the right process instance.
- What is a correlation set?
  - A set of business data fields that capture the state of the interaction (“correlating business data”). For example: a “purchase order number”, a “customer id”, etc.
  - Each set is initialized once
  - Its values do not change in the course of the interaction.

Defining Correlation Sets

```
<correlationSet name="..." properties="..."/>
```

<!– A CS is a named set of properties. Properties are defined as WSDL extensibility elements: -->

```
<bpws:property name="..." type="..."/>
```

<!– A property has a simple XSD type and a global name (Qname) -->

```
<bpws:propertyAlias propertyName="..."
                    messageType="..." part="..."
                    query="..."/>
```

<!– A property is “mapped” to a field in a WSDL message type. The property can thus be found in the messages actually exchanged. Typically a property will be mapped to several different message types and carried on many interactions, across operations and portTypes -->

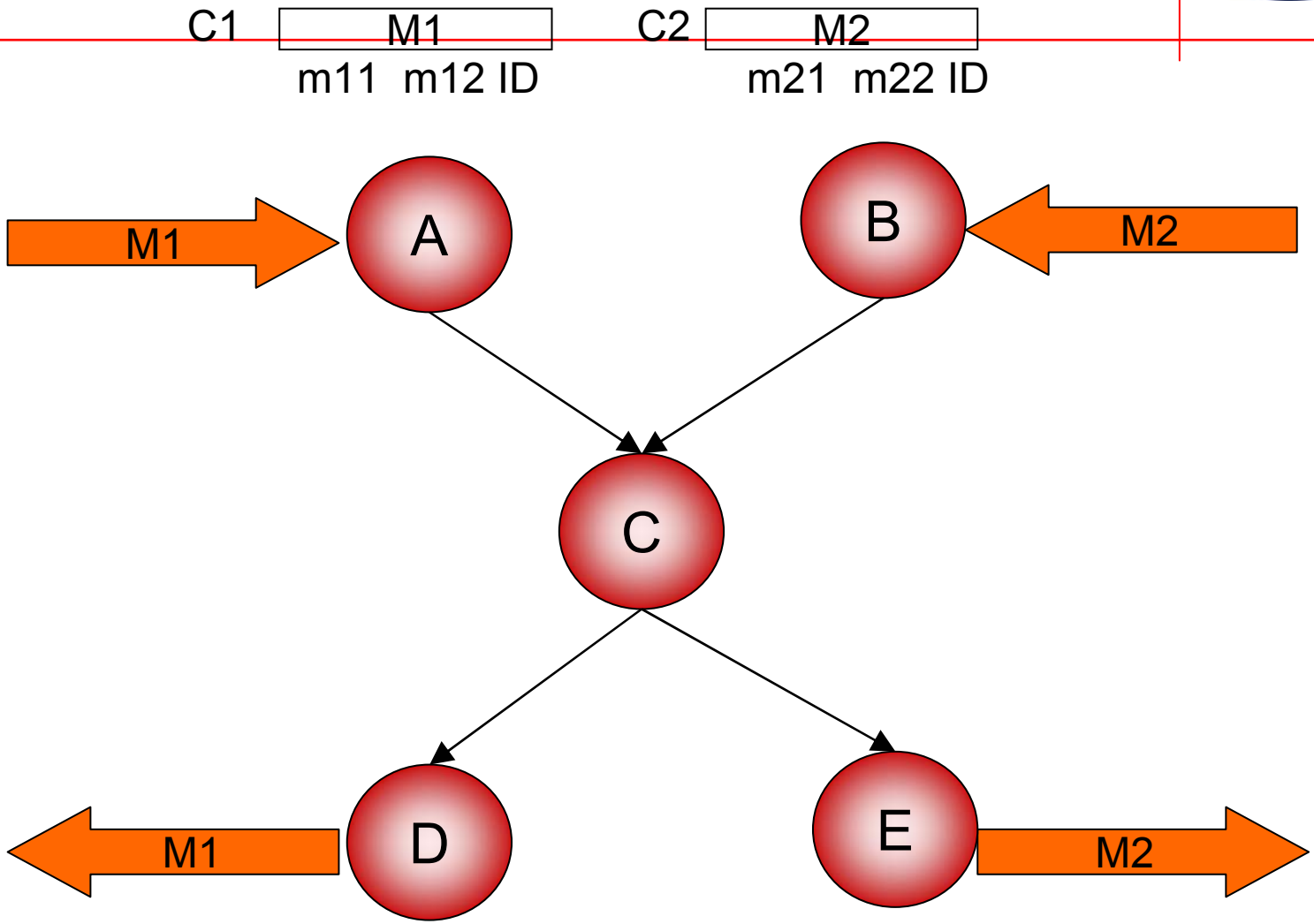
Using Correlation

```
<receive partner="..." operation="..." portType="..."
         container="...">
    <correlations>
         <correlation set="PurchaseOrder"
                      initiation="yes"/>
     </correlations>
</receive>
```

<!– An input or output operation identifies which correlation sets apply to the messages received or sent. That CS will be used to assure that the message is related to the appropriate stateful interaction.

<!– A CS is initialized once, in an interaction where the set appears with the "initiation" attribute set to "yes". Its value may never be changed afterward -->

Multiple Start Correlation



BPEL4WS Status

- Published August 10, 2002 by BEA, IBM, and Microsoft.
- To be submitted to a standards body.
- Several Java implementations available

Tools Requirements

BPEL4J Engine and Editor

Platforms:

- Linux
- Windows 2000

Java Tools:

- JDK1.2 or above

Other Tools: one of the following

- Websphere Application Server4.0
- Apache Tomcat 4.0.1

Browsers:

- Netscape
- IE

BPEL4WS Resources

- BPEL4WS 1.0:
  - http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/
- BPWS4J Java Implementations:
  - http://www.alphaworks.ibm.com/tech/bpws4j
  - http://www.collaxa.com/
- Two introductions to BPEL:
  - http://www.ibm.com/developerworks/webservices/library/ws-bpelcol1
  - http://www.ibm.com/developerworks/webservices/library/ws-bpelwp/