أكاديمية الحكومة الإلكترونية الفلسطينية

The Palestinian eGovernment Academy

www.egovacademy.ps

Tutorial III:

Process Integration and Service Oriented Architectures

Session 13
BPEL

Prepared By

Mohammed Aldasht

European Commission

TEMPUS

About

This tutorial is part of the PalGov project, funded by the TEMPUS IV program of the Commission of the European Communities, grant agreement 511159-TEMPUS-1-2010-1-PS-TEMPUS-JPHES. The project website: www.egovacademy.ps

Project Consortium:

- Birzeit University, Palestine (**Coordinator**)
- Palestine Polytechnic University, Palestine
- Palestine Technical University, Palestine
- Ministry of Telecom and IT, Palestine
- Ministry of Interior, Palestine
- Ministry of Local Government, Palestine
- University of Trento, Italy
- Vrije Universiteit Brussel, Belgium
- Université de Savoie, France
- University of Namur, Belgium
- TrueTrust, UK

Coordinator:
Dr. Mustafa Jarrar
Birzeit University, P.O.Box 14- Birzeit, Palestine
Telfax:+972 2 2982935 mjarrar@birzeit.edu

© Copyright Notes

Everyone is encouraged to use this material, or part of it, but should properly cite the project (logo and website), and the author of that part.

No part of this tutorial may be reproduced or modified in any form or by any means, without prior written permission from the project, who have the full copyrights on the material.

Attribution-NonCommercial-ShareAlike
CC-BY-NC-SA

This license lets others remix, tweak, and build upon your work non-commercially, as long as they credit you and license their new creations under the identical terms.

Tutorial Map

Intended Learning Objectives
A: Knowledge and Understanding
3a1: Demonstrate knowledge of the fundamentals of middleware.
3a2: Describe the concept behind web service protocols.
3a3: Explain the concept of service oriented architecture.
3a4: Explain the concept of enterprise service bus.
3a5: Understanding WSDL service interfaces in UDDI.
B: Intellectual Skills
3b1: Design, develop, and deploy applications based on Service Oriented Architecture (SOA).
3b2: use Business Process Execution Language (BPEL).
3b3: using WSDL to describe web services.
C: Professional and Practical Skills
3c1: setup, Invoke, and deploy web services using integrated development environment.
3c2: construct and use REST and SOAP messages for web services communication.
D: General and Transferable Skills
d1: Working with team.
d2: Presenting and defending ideas.
d3: Use of creativity and innovation in problem solving.
d4: Develop communication skills and logical reasoning abilities.

Title	T	Name
Session0: Syllabus and overview	0	Aldasht
Sesson1: Introduction to SOA	2	Aldasht
Session2: XML namespaces & XML schema	2	Aldasht
Session 3: Xpath & Xquery	4	Romi
Session4: REST web services	3	M. Melhem
Session5: Lab2: Practice on REST	3	M. Melhem
Session 6: SOAP	2	Aldasht
Session 7: WSDL	**3**	Aldasht
Session8: Lab 3: WSDL practice	3	Aldasht
Session9: ESB	4	Aldasht
Session10: Lab4: Practice on ESB	4	Aldasht
Session11: integration patterns	4	M. Melhem
Session12: Lab5: integration patterns	4	M. Melhem
Session13: BPEL	3	Aldasht
Session14: Lab6: Practice on BPEL	2	Aldasht
Session15: UDDI	3	Aldasht

Session 13: Component based service development/ WS composition (BPEL).

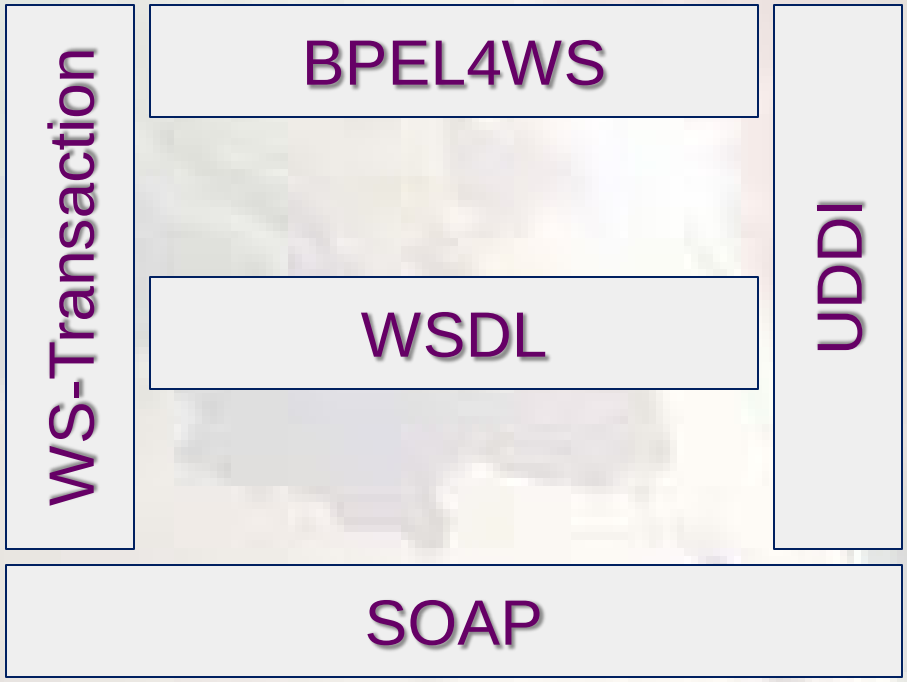
Session ILOs

After completing this module students will be able to use business process execution language BPEL.

Session Outlines

- **Introduction**
- Partners
- BPEL: Important features
- BPEL for Service Composition
- Developing BP with BPEL
- BPEL process example

Overview



Relationship among standards, Source, [2]

Introduction

- A *business process* specifies the potential execution order of operations from [2]:
  - A collection of web services
  - The data shared between these web services
  - Other issues involving which partners are involved, how multiple services and organizations participate
- This especially allows:
  - Specifying long-running transactions between web services.
  - Increasing consistency and reliability for web services applications.

Introduction, cont.

- To enable users to connect different components even across organizational boundaries in a platform- and language-independent manner, standards used:
  - WSDL to describe the Web service.
  - UDDI to advertise and organize Web services.
  - SOAP to communicate with Web service.

Introduction, cont.

- BPEL allows specifying business processes and how they relate to web services [2].
  - Business processes specified in BPEL are fully executable and portable.
  - A BPEL business process interoperates with the web services of its partners.
  - BPEL supports the specification of business protocols between partners.

General structure of a BPEL for web service 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>
```

Session Outlines

- Introduction
- **Partners**
- BPEL: Important features
- BPEL for Service Composition
- Developing BP with BPEL
- BPEL process example

Partners

- The partners of business processes involving web services often interact with.
- Partners are connected to a process in a bilateral manner *partner links* [2].
- Process can use the partner link to speak to a specific web service.

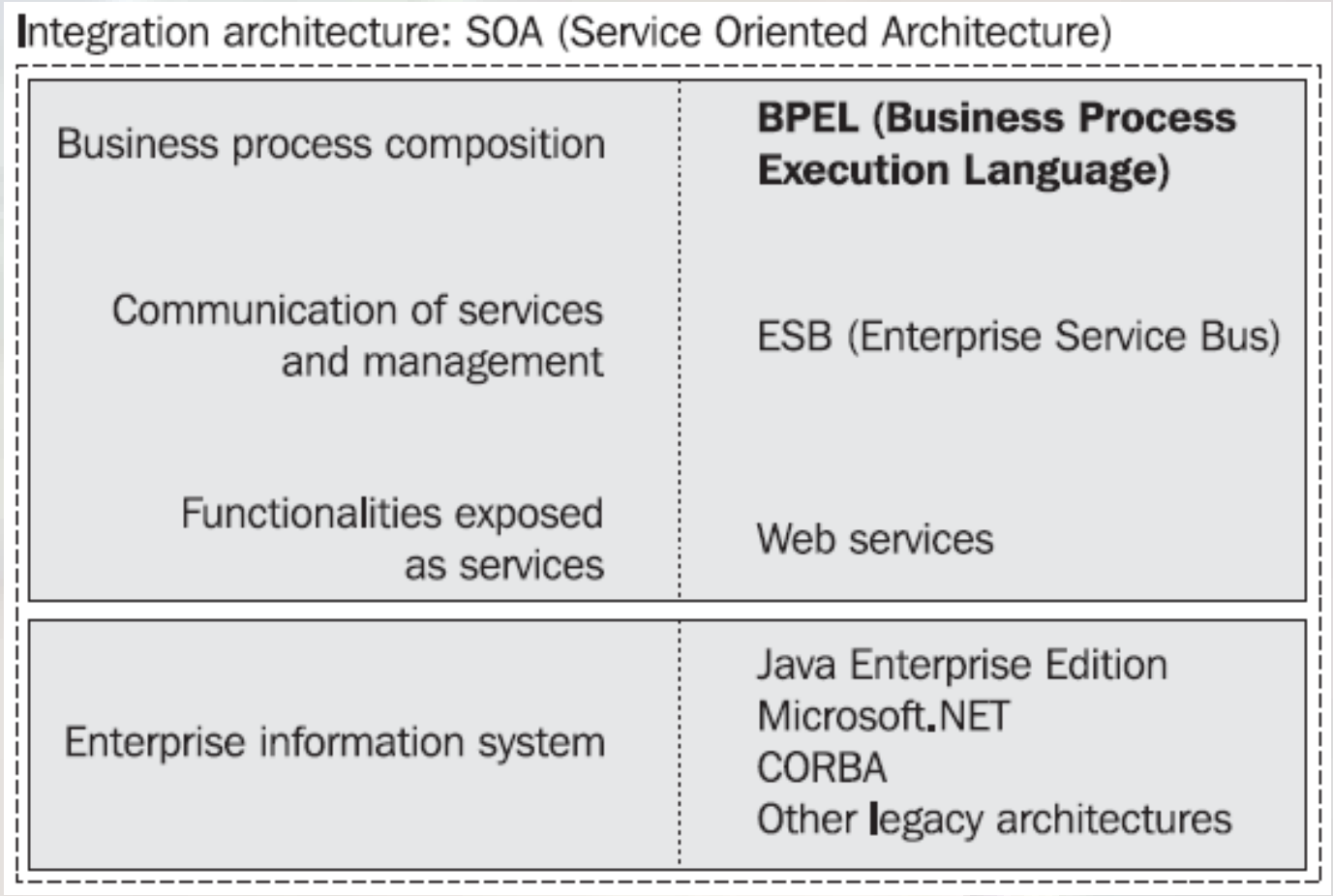
Partners, cont.

- Partners are one of the following [1]:
  - Services that the process invokes only.
  - Services that invoke the process only.
  - Or services that the process invokes and invoke the process

Automation of Business Processes

- For efficient automation of business processes through IT we need to:
  - Provide a **standardized way** to expose and access the functionality of applications as services.
  - Provide an **enterprise bus** infrastructure for communication and management of services.
  - Provide **integration architecture** between the various services and existing and newly developed applications used in business processes.
  - Provide a **specialized language** for composition of applications into business processes.

Relation between SOA, web services, ESB, and BPEL

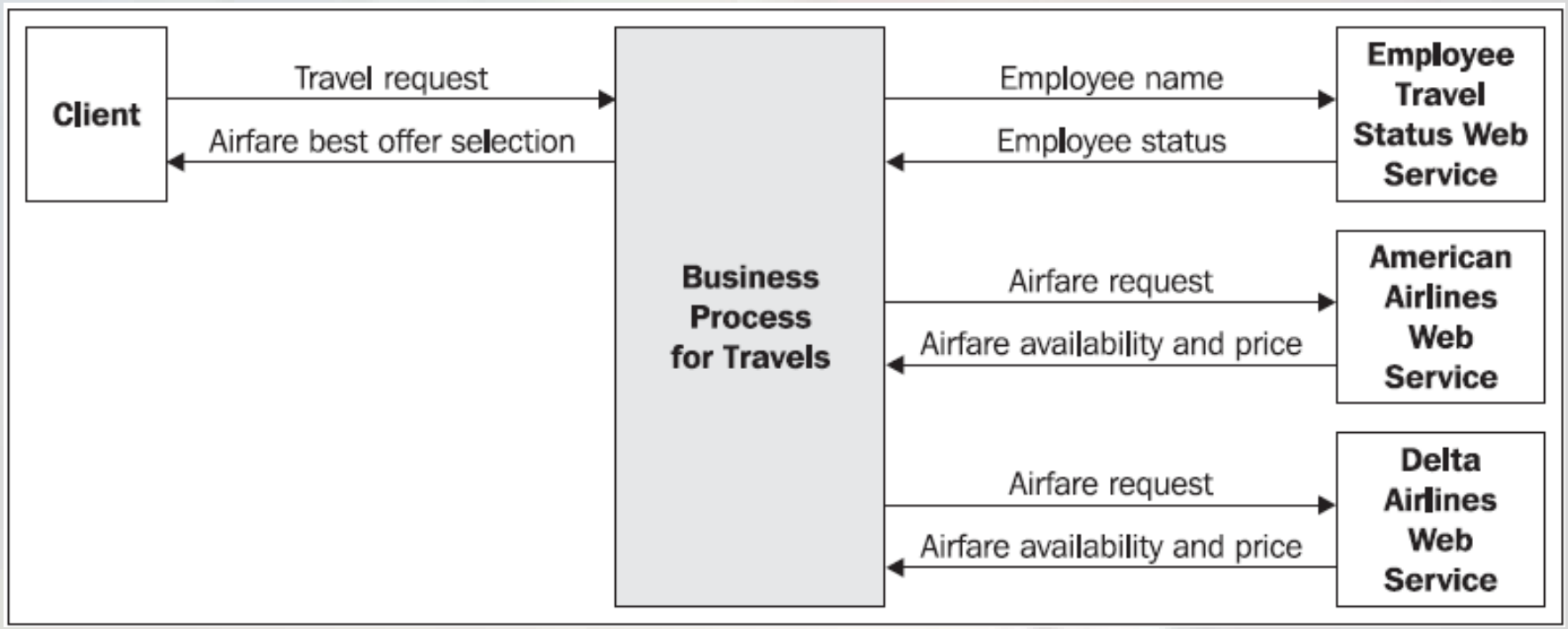


Source, [3]

Web service context

- Web services are a distributed architecture:
  - The distributed computing paradigm started with DCE (Distributed Computing Environment) RPC (Remote Procedure Call), and messaging systems.
  - Then the following emerged, distributed objects and ORBs (Object Request Brokers), such as CORBA (Common Object Request Broker Architecture), DCOM (Distributed Component Object Model), and RMI (Remote Method Invocation).
  - Based on them component models, such as EJB (Enterprise Java Beans), COM+ (Component Object Model), .NET Enterprise Services, and CCM (CORBA Component Model) have been developed.

BP example



Source, [3]

Session Outlines

- Introduction
- Partners
- **BPEL: Important features**
- BPEL for Service Composition
- Developing BP with BPEL
- BPEL process example

BPEL: Important features

- Describe the logic of business processes through composition of services
- Compose larger business processes out of smaller processes and services
- Handle synchronous and asynchronous (often long-running) operation invocations on services, and manage callbacks that occur at later times
- Invoke service operations in sequence or parallel

BPEL: Important features, cont.

- Selectively compensate completed activities in case of failures
- Maintain multiple long-running transactional activities, which are also interruptible
- Resume interrupted or failed activities to minimize work to be redone
- Route incoming messages to the appropriate processes and activities

BPEL: Important features, cont.

- Schedule activities based on the execution time and define their order of execution
- Execute activities in parallel and define how parallel flows merge based on synchronization conditions

Session Outlines

- Introduction
- Partners
- BPEL: Important features
- **BPEL for Service Composition**
- Developing BP with BPEL
- BPEL process example

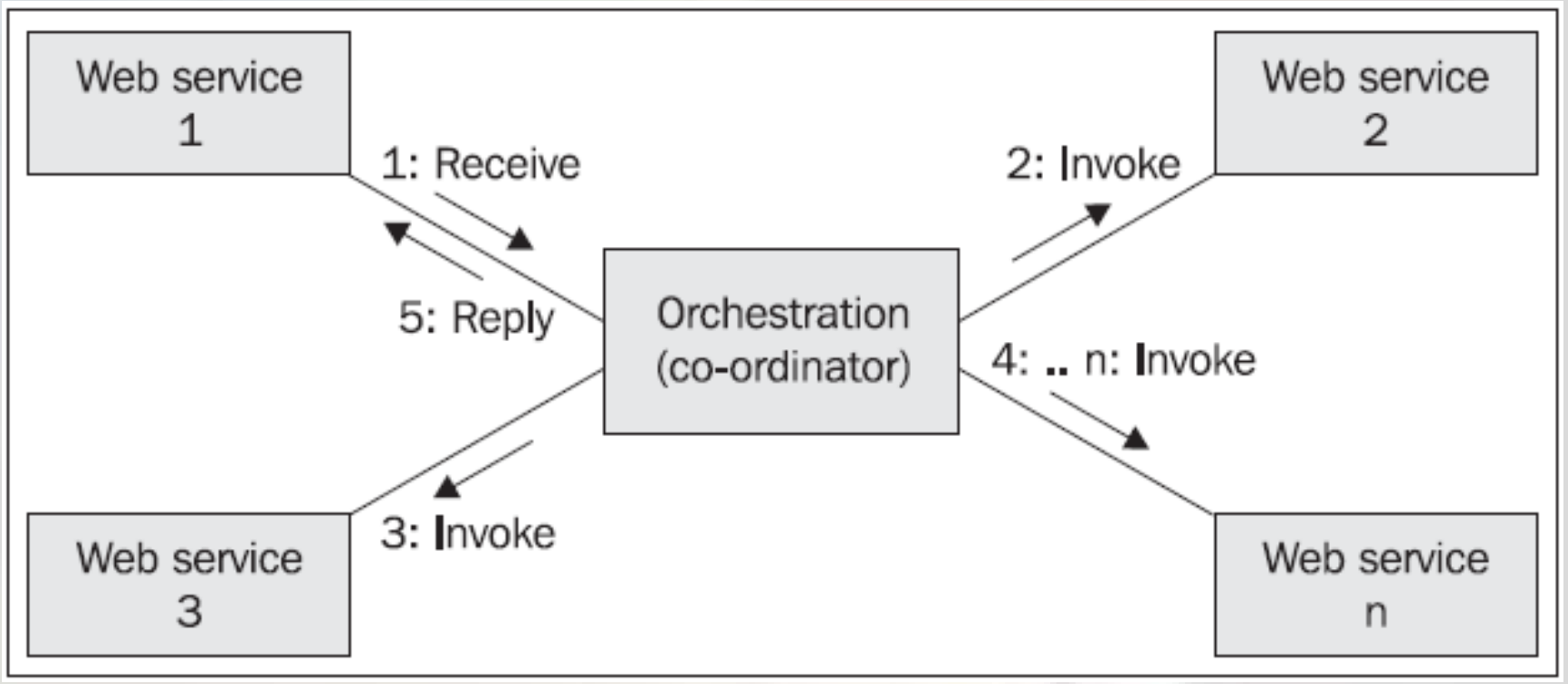
BPEL for Service Composition

- Business process automation requires standards
- Also, requires a specialized language for composing services into business processes
- Composition of business processes should support many process instances, long-running processes, etc.
- The main goal of BPEL is to standardize the process of automation between web services.

Composition of services in private processes, Orchestration

- A central process (which can be another web service):
  - Takes control over the involved web services
  - Coordinates the execution of different operations on the web services involved in the operation.

Orchestration

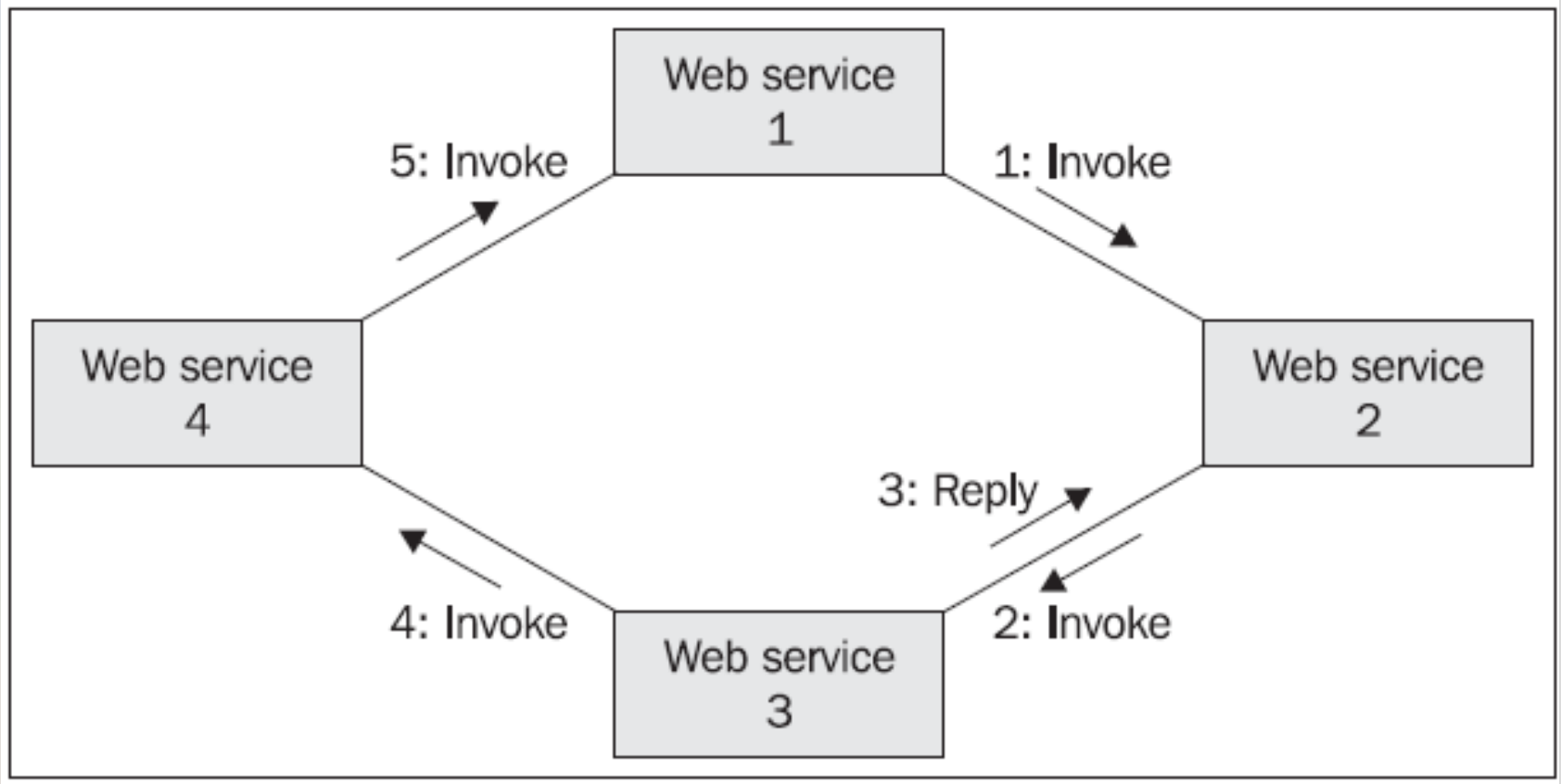


Source, [3]

Composition of services in public processes, Choreography

- Each web service involved in the choreography knows exactly when to execute its operations and whom to interact with.
- Choreography is a collaborative effort focused on exchange of messages in public business processes

Choreography



Source, [3]

Session Outlines

- Introduction
- Partners
- BPEL: Important features
- BPEL for Service Composition
- **Developing BP with BPEL**
- BPEL process example

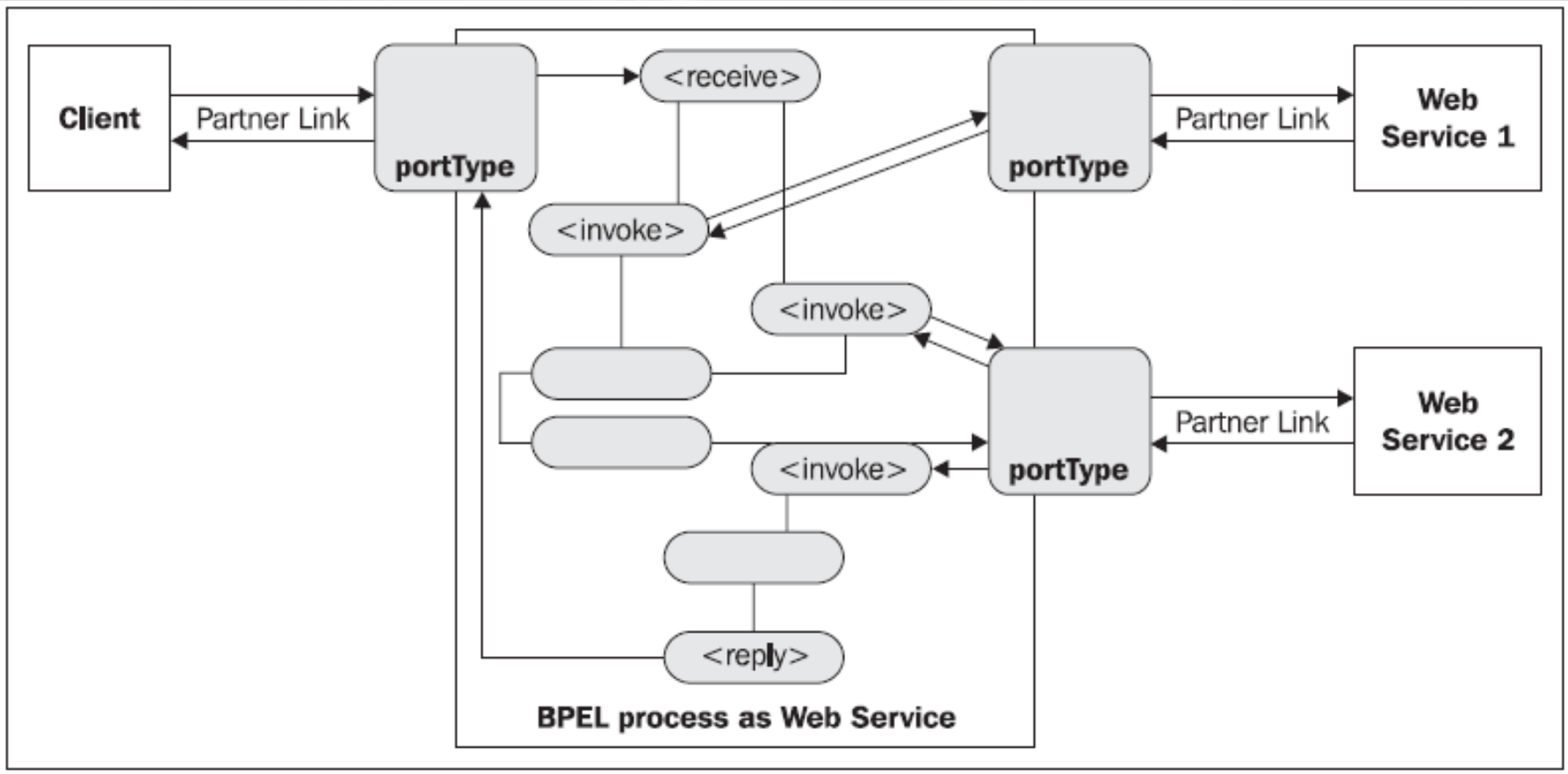
Developing BP with BPEL

- BPEL uses an XML-based vocabulary that allows us to specify and describe business processes.
- With BPEL, we can describe business processes in two distinct ways:
  - Executable business processes specify the exact details of business processes and can be executed by a BPEL engine.
  - Abstract business processes specify only the public message exchange between parties, without specific details of process flows.

Partner Links

- When we describe a business process in BPEL, we actually define a new web service that is a composition of existing services.
- Developing BPEL processes requires a good understanding of WSDL and other related technologies.
- BPEL introduces WSDL extensions, which enable us to accurately specify relations (**partner links**) between several web services in the business process.

BPEL process and its partner links



Source, [3]

BPEL Core Concepts

- A BPEL process consists of steps “activities”,
- Two types: **basic** and **structured** activities.
- Seven basic activities:
  1. Invoking other web services, using `<invoke>`
  2. Waiting for the client to invoke the business process through sending a message, using `<receive>` (receiving a request)

Basic activities, cont.

- Cont.:
  3. Generating a response for synchronous operations, using `<reply>`
  4. Manipulating data variables, using `<assign>`
  5. Indicating faults and exceptions, using `<throw>`
  6. Waiting for some time, using `<wait>`
  7. Terminating the entire process, using `<terminate>`

BPEL structured activities

1. Sequence (`<sequence>`) for defining a set of activities that will be invoked in an ordered sequence
2. Flow (`<flow>`) for defining a set of activities that will be invoked in parallel
3. Case-switch construct (`<switch>`) for implementing branches
4. While (`<while>`) for defining loops
5. The ability to select one of a number of alternative paths, using `<pick>`

Session Outlines

- Introduction
- Partners
- BPEL: Important features
- BPEL for Service Composition
- Developing BP with BPEL
- **BPEL process example**

Simple BPEL process example

- Select the best insurance offer from several
- We first declare the partner links to the BPEL process client (called `client`) and two insurance web services (called `insuranceA` and `insuranceB`):

Example, Cont.

```
<?xml version="1.0" encoding="utf-8"?>
<process name="InsuranceSelectionProcess"
  targetNamespace="http://companyx.com/bpel/example/"
  xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
  xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
  xmlns:ins="http://companyx.com/bpel/insurance/"
  xmlns:com="http://companyx.com/bpel/company/" >
```

Example, Cont.

- The two insurance web services (called `insuranceA` and `insuranceB`):

```
<partnerLinks>
      <partnerLink name="client"
            partnerLinkType="com:selectionLT"
            myRole="insuranceSelectionService"/>
      <partnerLink name="insuranceA"
            partnerLinkType="ins:insuranceLT“
            myRole="insuranceRequester"
            partnerRole="insuranceService"/>
      <partnerLink name="insuranceB"
            partnerLinkType="ins:insuranceLT"
            myRole="insuranceRequester"
            partnerRole="insuranceService"/>
  </partnerLinks>
```

Example, Cont.

- Declare variables for:
  - The insurance request (`InsuranceRequest`),
  - Insurance A and B responses (`InsuranceAResponse`, `InsuranceBResponse`),
  - The final selection (`InsuranceSelectionResponse`)

Example, Cont.

```
...
   <variables>
       <!-- input for BPEL process -->
       <variable name="InsuranceRequest"
              messageType="ins:InsuranceRequestMessage"/>
       <!-- output from insurance A -->
       <variable name="InsuranceAResponse"
              messageType="ins:InsuranceResponseMessage"/>
       <!-- output from insurance B -->
       <variable name="InsuranceBResponse"
              messageType="ins:InsuranceResponseMessage"/>
       <!-- output from BPEL process -->
       <variable name="InsuranceSelectionResponse"
              messageType="ins:InsuranceResponseMessage"/>
   </variables>
...
```

Example, Cont.

- Finally, we specify the process steps:
  - First we wait for the initial request message from the client (`<receive>`)
  - Then we invoke both insurance web services (`<invoke>`) in parallel using the `<flow>` activity.
  - The insurance web services return the insurance premium.
  - Then we select the lower amount (`<switch>/<case>`) and return the result to the client (the caller of the BPEL process) using the `<reply>` activity.

Example, Cont.

```
...
  <sequence>
  <!-- Receive the initial request from client -->
  <receive partnerLink="client"
      portType="com:InsuranceSelectionPT"
      operation="SelectInsurance"
      variable="InsuranceRequest"
      createInstance="yes" />
```

Example, Cont.

```
<!-- Make concurrent invocations to Insurance A and B -->
<flow>
    <!-- Invoke Insurance A web service -->
    <invoke partnerLink="insuranceA"
            portType="ins:ComputeInsurancePremiumPT"
            operation="ComputeInsurancePremium"
            inputVariable="InsuranceRequest"
            outputVariable="InsuranceAResponse" />
    <!-- Invoke Insurance B web service -->
    <invoke partnerLink="insuranceB"
            portType="ins:ComputeInsurancePremiumPT"
            operation="ComputeInsurancePremium"
            inputVariable="InsuranceRequest"
            outputVariable="InsuranceBResponse" />
</flow>
```

Example, Cont.

```
<!-- Select the best offer and construct the response -->
   <switch>
   <case condition = "bpws:getVariableData
                      ('InsuranceAResponse',
                      'confirmationData',
                      '/confirmationData/ins:Amount')
                      &lt;= bpws: getVariableData
                      ('InsuranceBResponse',
                      'confirmationData',
                      '/confirmationData/ins:Amount')">
        <!-- Select Insurance A -->
        <assign>
          <copy>
               <from variable="InsuranceAResponse" />
               <to variable="InsuranceSelectionResponse" />
          </copy>
        </assign>
   </case>
```

Example, Cont.

```
      <otherwise>
      <!-- Select Insurance B -->
        <assign>
        <copy>
              <from variable="InsuranceBResponse" />
              <to variable="InsuranceSelectionResponse" />
        </copy>
        </assign>
      </otherwise>
    </switch>
    <!-- Send a response to the client -->
    <reply partnerLink="client"
        portType="com:InsuranceSelectionPT"
        operation="SelectInsurance"
        variable="InsuranceSelectionResponse"/>
    </sequence>
</process>
```

Summary

During this module we have explained with examples how to use business process execution language BPEL; the following subject have been covered:

1. Partners
2. BPEL: Important features
3. BPEL for Service Composition
4. Developing BP with BPEL
5. BPEL process example

Next module will cover the UDDI

References

1. Sanjiva Weerawarana, Francisco Curbera, “Business Process with BPEL4WS: Understanding BPEL4WS”, IBM TJ Watson Research Center, 2002
2. Frank Leymann, Dieter Roller, “Business processes in a web services world”, IBM Software Group, 2002
3. Matjaz B. Juric, Benny Mathew and Poornachandra Sarang, Business Process Execution Language for Web Services , Second Edition, Packt Publishing , 2006.

Thanks

Mohammed Aldasht