BPEL
(Business Process Execution Language)

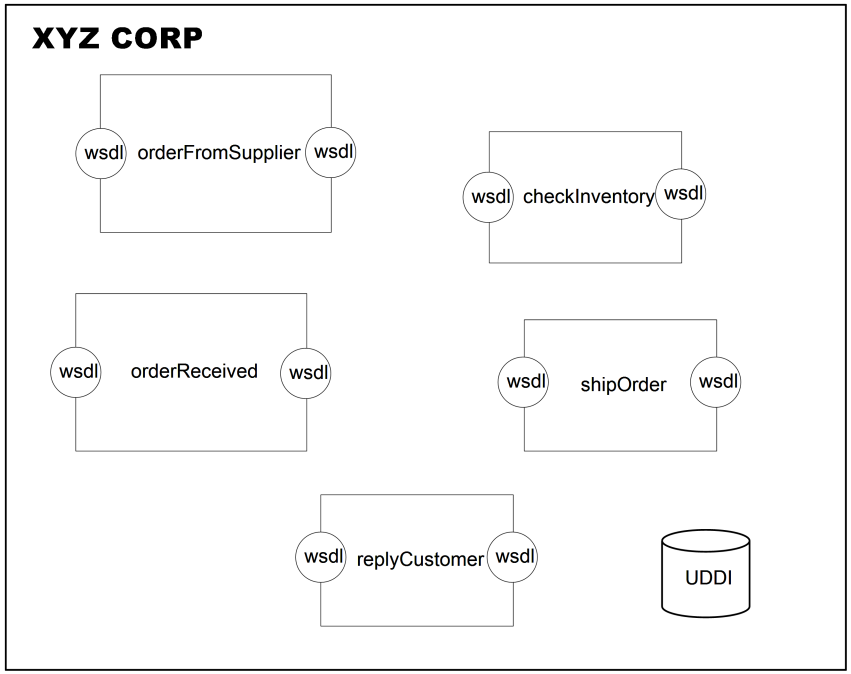
Prof. (Dr.) Vipul K. Dabhi
Associate Professor,
Department of Information Technology,
D. D. University

Business Process (BP)

- A BP consists of both internal computations and invocations of operations exported by web service partners
- The operations it exports constitute its interface to its partners
- The sequence of invocations it executes is referred to as a protocol and
  - is data dependent
  - responds to exceptional conditions

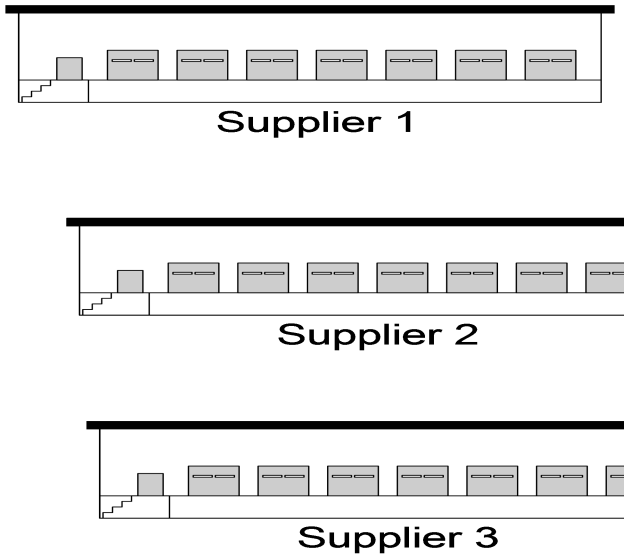
Web Services

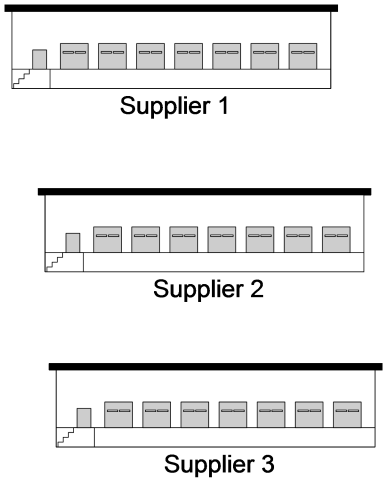
- Consider the following set of web services



Web Services (2)

- What about between organizations?





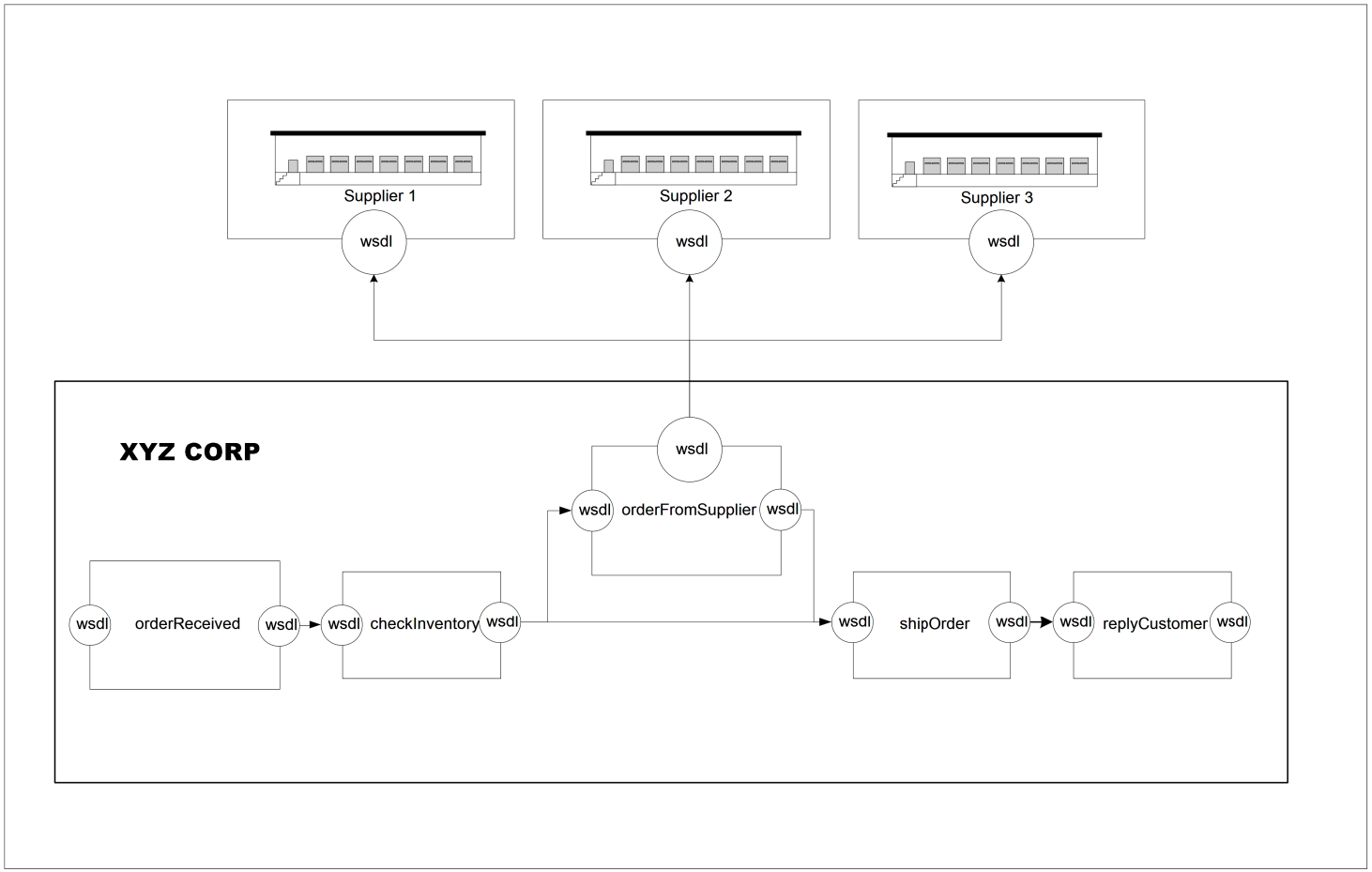
Overview

- Businesses today requires to **quickly** adapt to customer needs and market conditions
  - EAI and B2B interactions (through web services)
- Needs to be flexible **internally** and **externally**
- Without a common set of standard, each organization is left to build their own set of **proprietary** business protocols
  - Leaving little flexibility for true web services collaboration

Web Service Composition

- Definition:
  - Provides an **open**, **standards-based** approach for connecting web services together to create **higher-level** business processes.
  - Standards are designed to **reduce the complexity** required to compose web services, hence **reducing time and costs**, and **increase overall efficiency** in businesses

Web Service Composition (2)



New Issues

- Must be able to communicate with other Web Services
- Must be able to access and modify data received in messages
  - Use XPath to extract information from messages
- Must have control constructs
  - sequence, switch (if), flow (concurrency), while, link (synchronize concurrent processes), invoke, etc
- Must be able to handle faults

Basic Requirements

- Ability to invoke services in a **asynchronous** manner
  - Achieve **reliability**, **scalability**, and **adaptability** required by Today's IT environment
- Manage **exception** and **transactional integrity**
  - Studies shown **nearly 80%** of the time spent in building business processes are spent in exception management
- Provide **dynamic**, **flexible**, and **adaptable** framework
  - Provide a clear **separation** between the process logic and the web services used
  - Able to **compose higher-level services** from existing processes

Standards

- **BPEL4WS** (a.k.a. BPEL) – Business Process Execution Language for Web Services
  - IBM and Microsoft
- **WSCI** – Web Services Choreography Interface
  - Sun, SAP, BEA, and Intalio
- **BPML** – Business Process Management Language
  - BPMI.org (chartered by Intarlio, Sterling Commerce, Sun, CSC, and others)

Orchestration vs. Choreography

Orchestration

A Single Director in Control

Choreography

Defines Interaction. WS-Choreography describes Publicly Visible Message Exchange.

Orchestration Vs Choreography

- Orchestration
  - A central process takes control over the involved web services and coordinates the execution.
  - The involved web services do not know that they are involved into a composition and that they are a part of higher business process. Only central coordinator knows this.
- Choreography
  - Does not rely on a central coordinator. Each web service in choreography knows exactly when to execute its operation and whom to interact with.
  - Choreography is a collaborative effort focused on exchange of messages in public business processes. All participants of choreography need to be aware of business process, operations to execute, messages to exchange and timing of message exchanges

BPEL - Process Models

- Provides support for two business process models
  - **Executable**
    - Models the behavior of participants in a specific business interaction, a **private workflow**
  - **Abstract**
    - Business protocols in BPEL, specify the public **message exchanges** between parties

Executable and Abstract Processes

- We can specify exact details of business processes. Such processes are called executable business processes and follow the orchestration paradigm. They can be executed by an orchestration engine.

- We can specify public message exchange between parties only. Such processes are called abstract business processes. They follow the choreography paradigm.

Abstract Vs. Executable BPs

- **Executable BP** – complete description of BP (including all computations)
- **Abstract BP** – contains only externally visible (communication related) behavior of BP
  - Not executable
  - Internal decision making algorithm and data manipulation not described
- Languages for describing abstract and executable BPs share a common core, but differ primarily in data handling capabilities
- BPEL4WS is used to specify both abstract and executable BPs

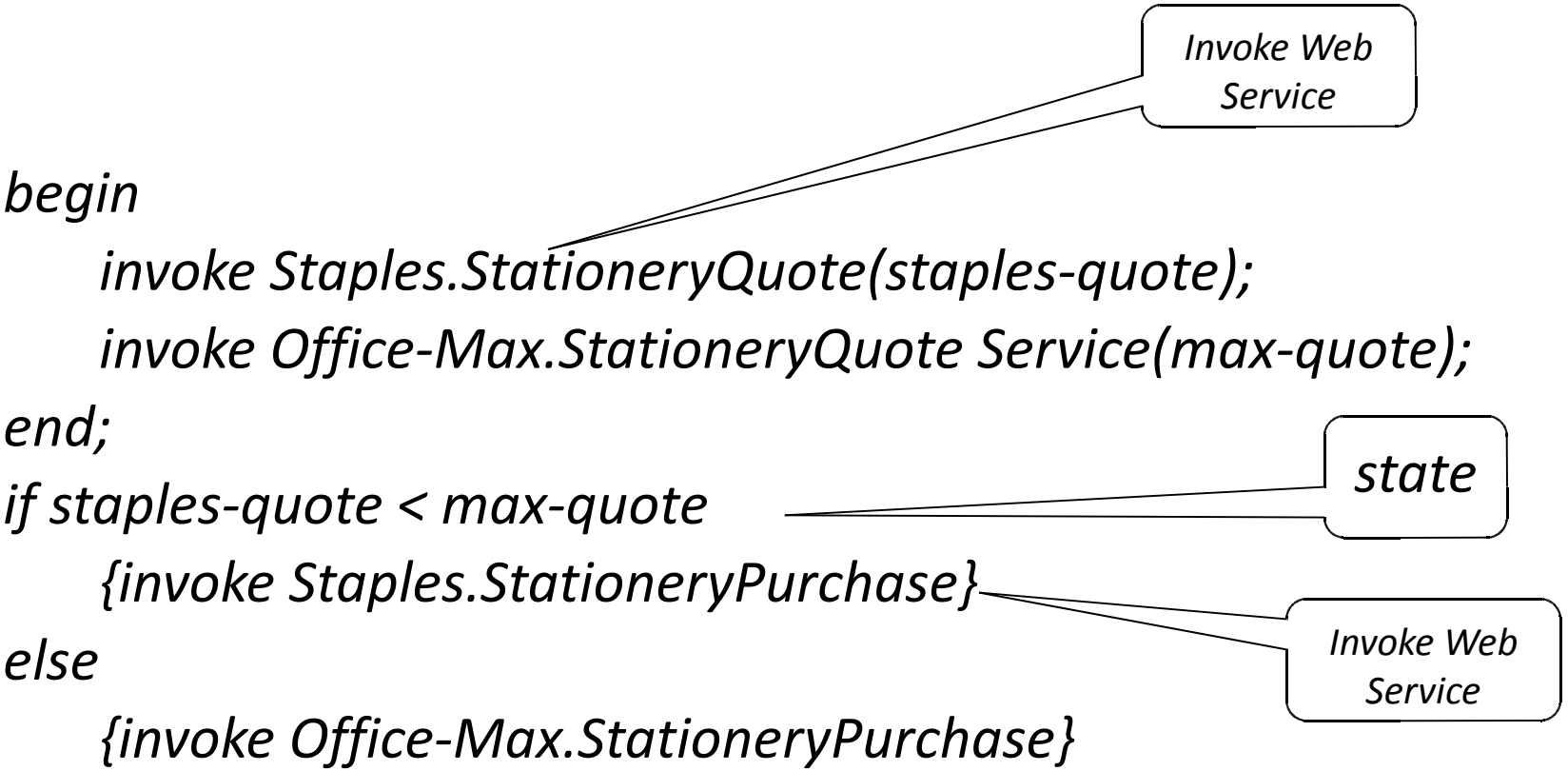
BPEL Goals

- Platform independence – XML based language (Java, .Net implementations available)
- Services and Messages are first order citizens
- Provide an asynchronous programming model
- Leverage runtime metadata based interface like WSDL
- ‘enterprise ready’ (security, transactional, reliable, scalable, etc.)

BPEL vs. WSDL

- WSDL supports a stateless model which describes operations supported by web servers
  - One or two messages needed for client/server communication
  - No mechanism for describing state between operations
- A business process (BP) typically characterized by long-running, stateful sequence of operations with one or more web services (business partners).

Simple Example: Ordering Stationery



BPEL4WS

- XML-Based language
- It describes the **control logic** for web services coordination in a business process
- **Interpreted** and **executed** by a BPEL engine

BPEL Servers / Engines

- BPEL servers provide a run-time environment for executing BPEL business processes.
- BPEL servers exist for J2EE, .NET and other platforms
- Most important commercial servers are:
  - Oracle BPEL Process Manager
  - Microsoft BizTalk
  - ActiveBPEL Engine
  - Apache Agila
  - IBM Web Sphere Studio Application Developer

BPEL - Overview

- Use Web Services Standard as a base
  1. Every BPEL is **exposed as a web service** using WSDL. And the WSDL describes the public entry and exit points of the process
  2. **Interacts through WSDL interfaces** with external web services
  3. **WSDL data types are used** to describe information flow within the BPEL process

BPEL and Web Services

- As BPEL processes are exposed as web services, we need a WSDL for the BPEL process.
- BPEL introduces WSDL extensions, which enable us to accurately specify relations between several web services in business process. These relations are called partner links.

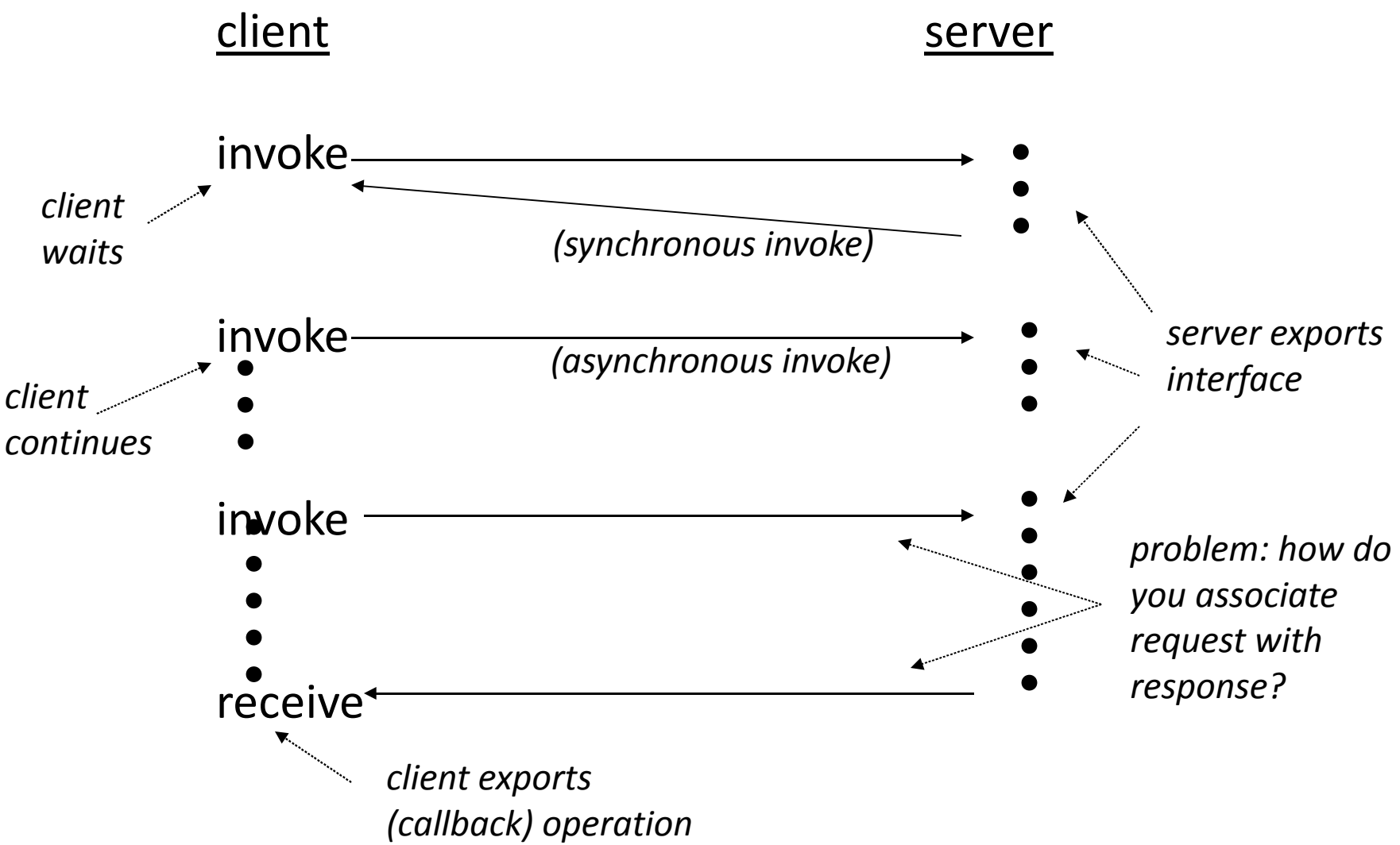
Communication – Client Side

- Invoking an operation of an interface (specified in WSDL) exported by server
  - Client assigns message to operation's input variable
  - Client executes *invoke* on operation
    - Asynchronous (in-only WSDL pattern):
      - Client resumes execution immediately
    - Synchronous (in-out WSDL pattern):
      - Client waits for response and then resumes execution
      - Client can access response message in operation's output variable

Communication – Client Side

- Receiving an operation (*e.g.*, an asynchronous response to a prior invocation) on a (callback) operation exported by client
  - Client executes *receive* on operation
    - Client waits for message
    - Client can access message in variable associated with operation and resume execution

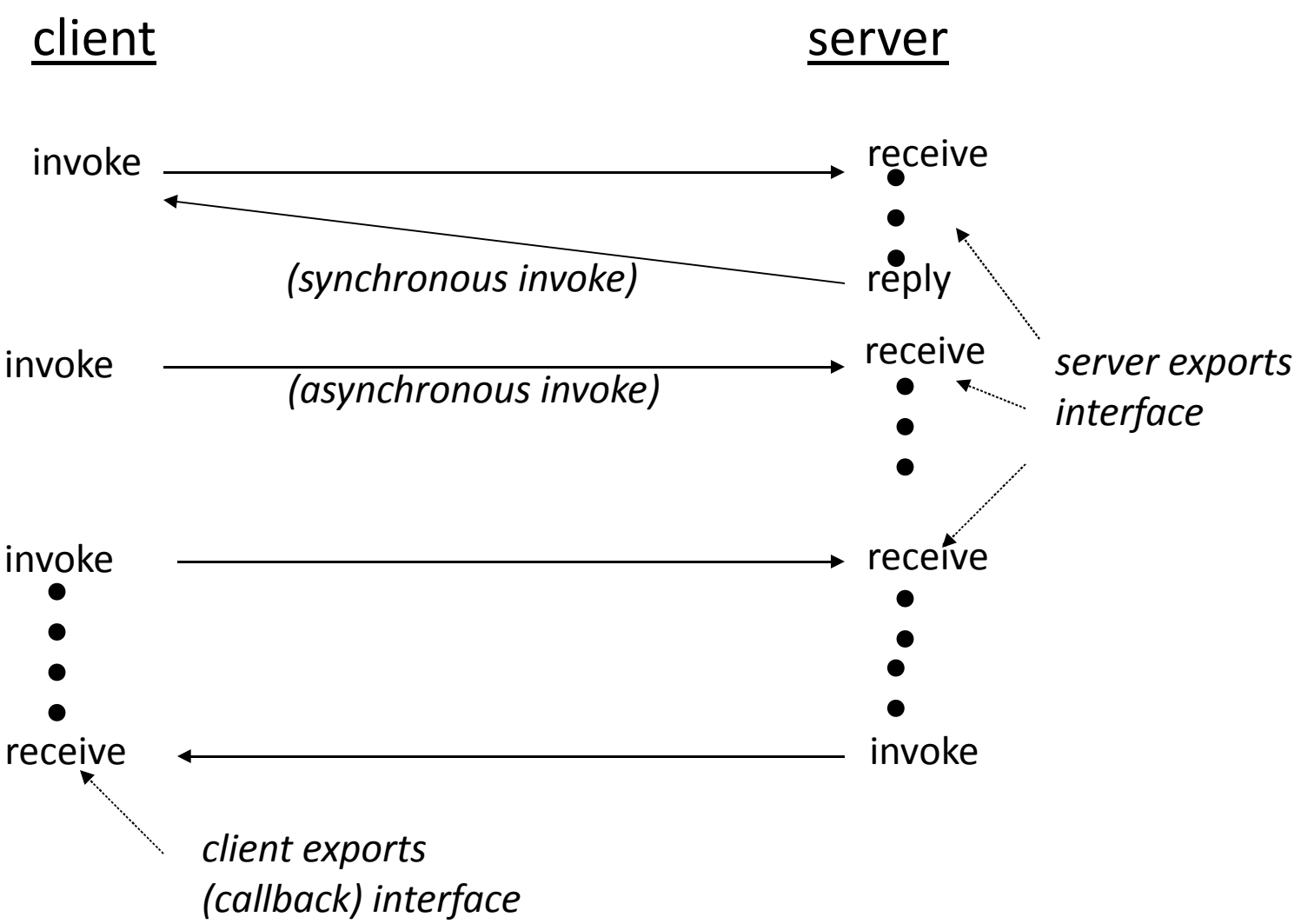
Communication – Client Side



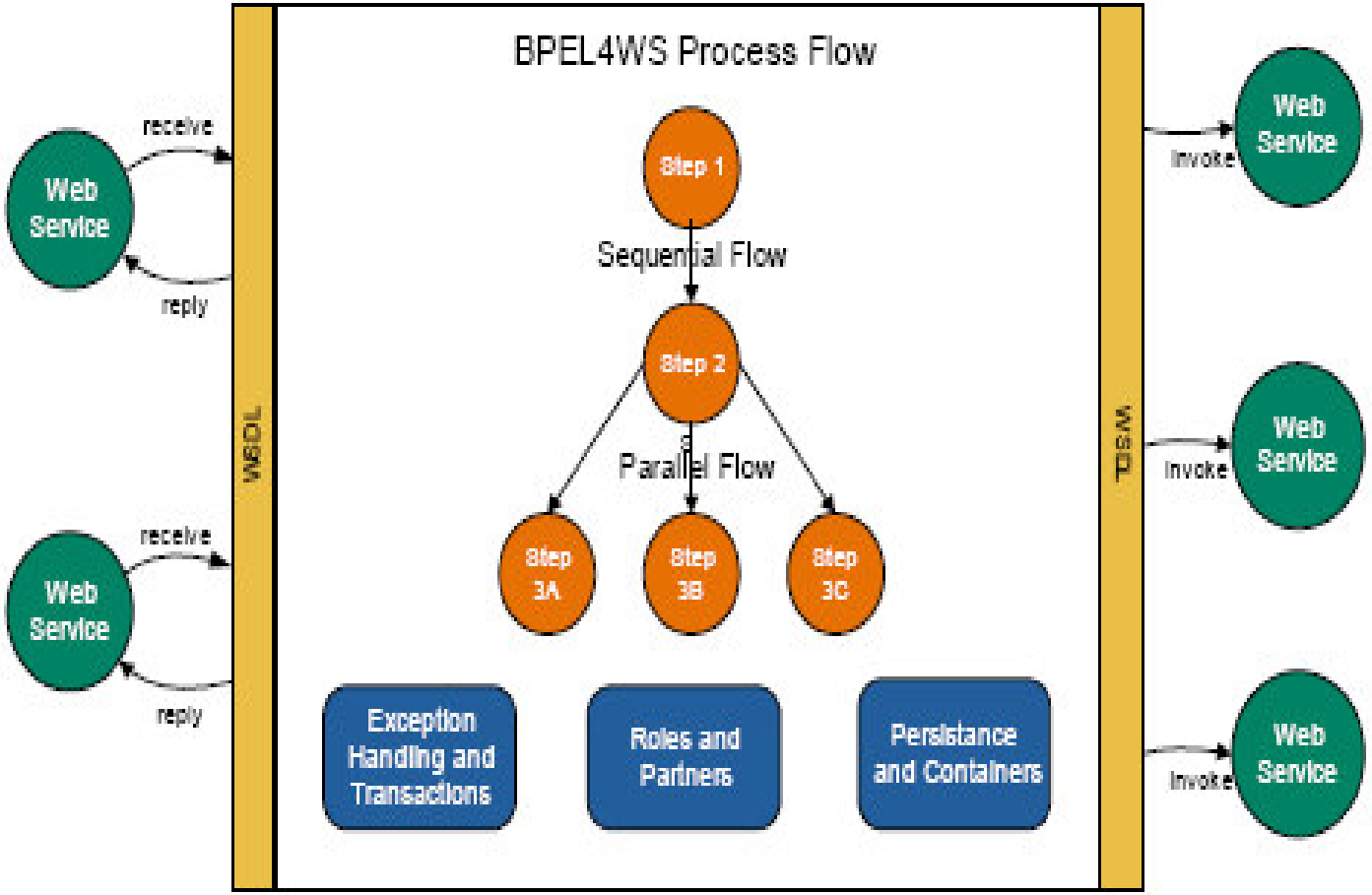
Communication – Server Side

- Accepting an operation invocation on an (exported) interface (specified in WSDL)
  - Server executes *receive* on operation and waits
- Responding to a synchronous operation invocation
  - Server executes *reply* on operation (rpc)
- Invoking a client's exported (callback) operation
  - Server executes *invoke* on operation

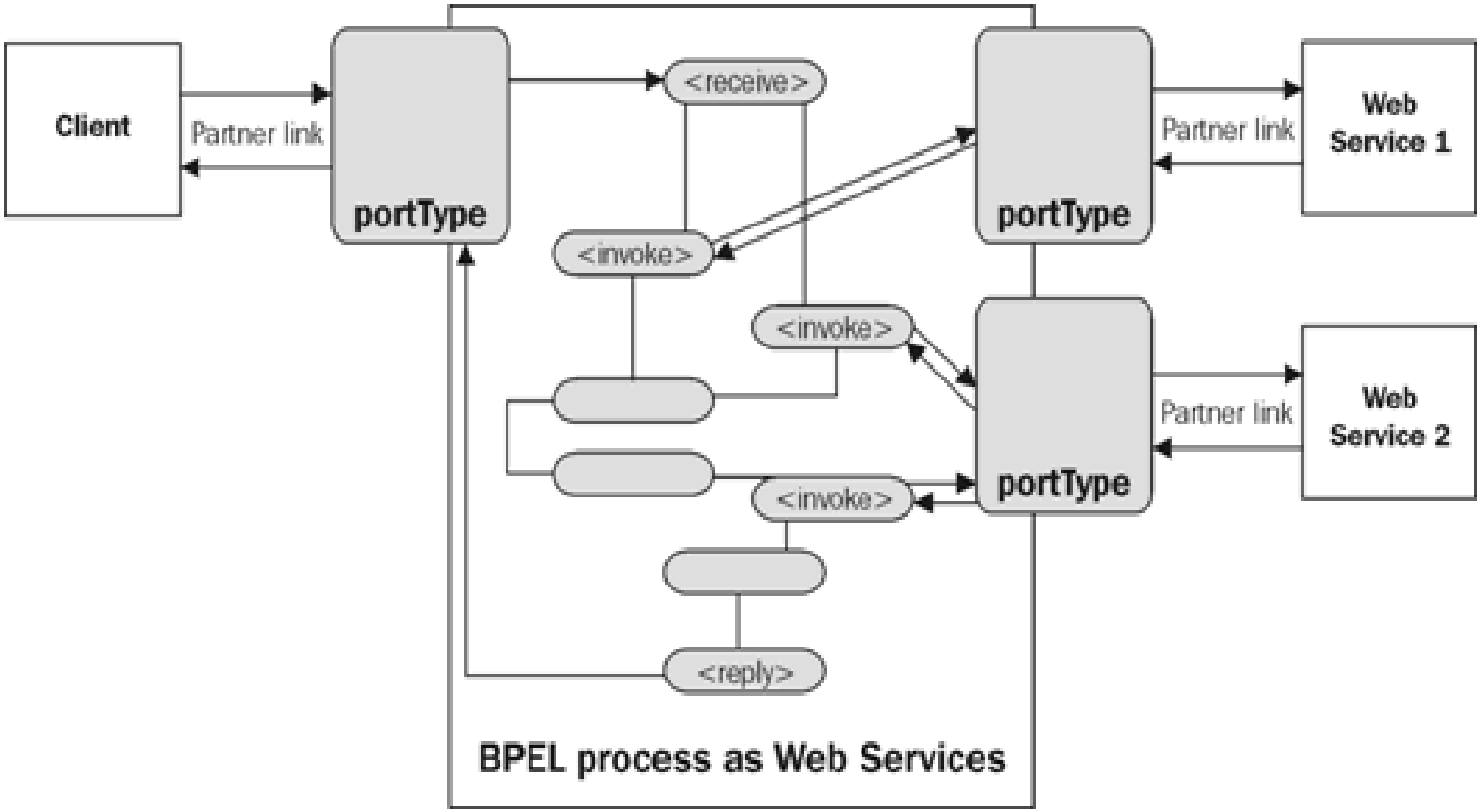
Communication – Server Side



BPEL - Process Overview



BPEL – Example Process



CORE Concepts

- A BPEL process consists of steps. Each step is called an activity.
- BPEL supports basic and structured activities.

BPEL - Activities

- **Basic** Activities:
  - Interacts with external services
    - <invoke>, <receive>, and <reply>
- **Structured** Activities:
  - Internal process control flow
    - sequential flow, conditional branching, looping etc.

BPEL Basic Activities

- <invoke> - Invoking other web services.
- <receive> - Waiting for client to invoke business process through sending a message using <receive>
- <reply> - Generating a response for synchronous operations
- <assign> - Manipulating data variables
- <throw> - Indicating faults and exceptions
- <terminate> - Terminating the entire process.

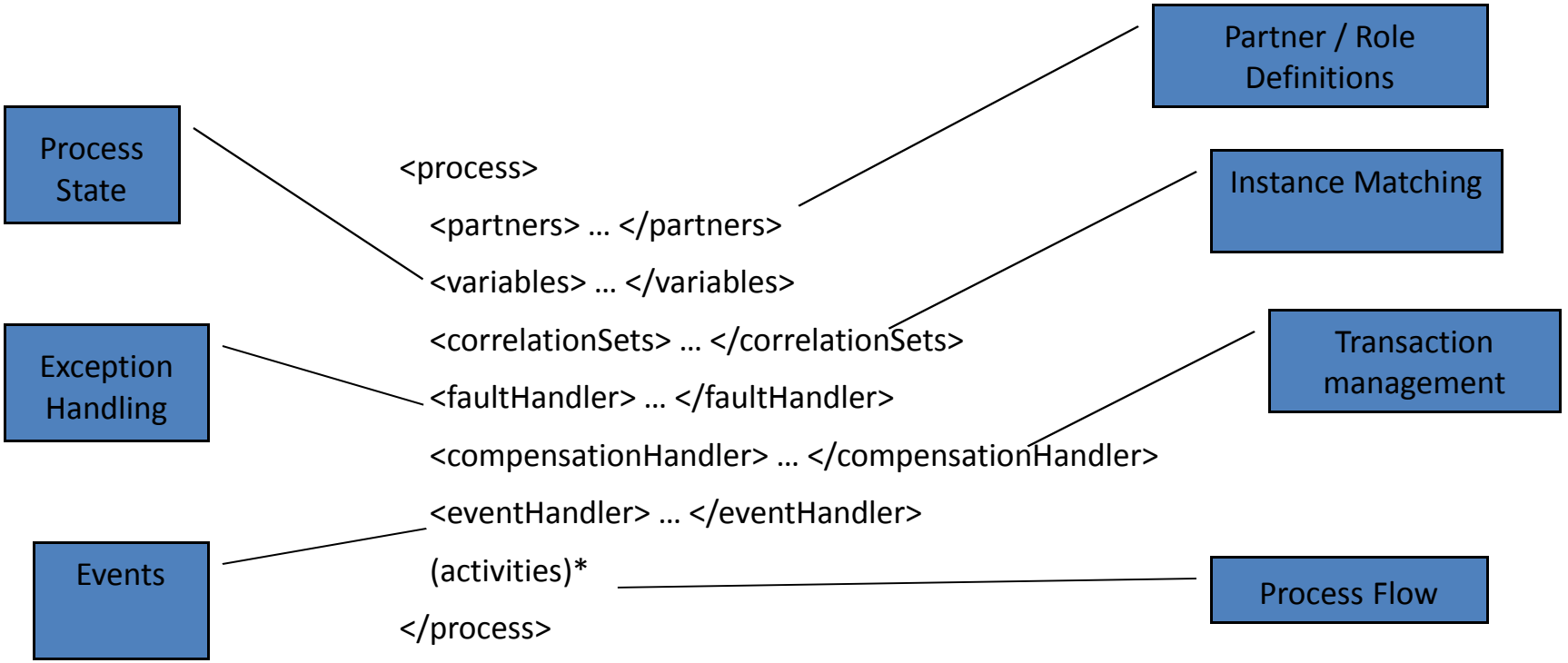
BPEL Structured Activities

- <sequence> - for defining a set of activities that will be invoked in an order sequence.
- <flow> - for defining a set of activities that will be invoked in parallel.
- <switch> - Case-switch construct for implementing branches
- <while> - for defining loops
- <pick> - to select one of a number of alternative paths.

BPEL Activities

- Each BPEL process will also define partner links, using <partnerLink> and declare variables using <variable>

BPEL Structure Overview



activities = <receive>, <reply>, <invoke>, <assign>, <throw>,
<terminate>, <wait>, <empty>, <sequence>, <switch>,
<while>, <pick>, <flow>, <scope>, <compensation>

BPEL 1.1 Syntax

BPEL - Containers and Partners

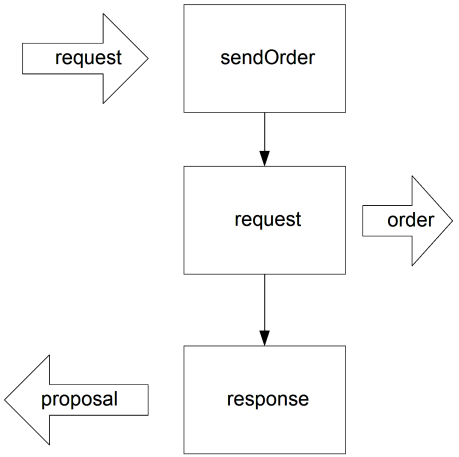
- **Containers**
  - Data exchanges in the message flow
    - e.g. WSDL messageType
- **Partners**
  - Any services that the process invokes OR any services that the invokes the process

```
<partners>
        <partner name=”buyer” … myRole=”agent”/>
        <partner name=”supplier” … myRole=”requestor” partnerRole=”supplier”/>
</partners>
<containers>
        <container name=”request” messageType=”tns:orderRequest”/>
        <container name=”response” messageType=”tns:orderResponse”/>
</containers>
```

BPEL - Code

- A sequence

```
<sequence>
    <receive partner=”buyer” … operation=”sendOrder” container=”request”/>
    <invoke partner=”supplier” … operation=”request” container=”order”/>
    <reply partner=”buyer” … operation=”response” container=”proposal”/>
</sequence>
```



BPEL - Others

- Transactions and Exceptions
  - Building on top of **WS-Coordination** and **WS-Transaction** specifications
  - Transaction
    - A set of activities can be grouped in a single transaction through the **<scope>** tag
      - Can specify **compensation handlers** (rollback) if there is an error
  - Exception Handling
    - Through the use of **throw and catch** (similar to Java)