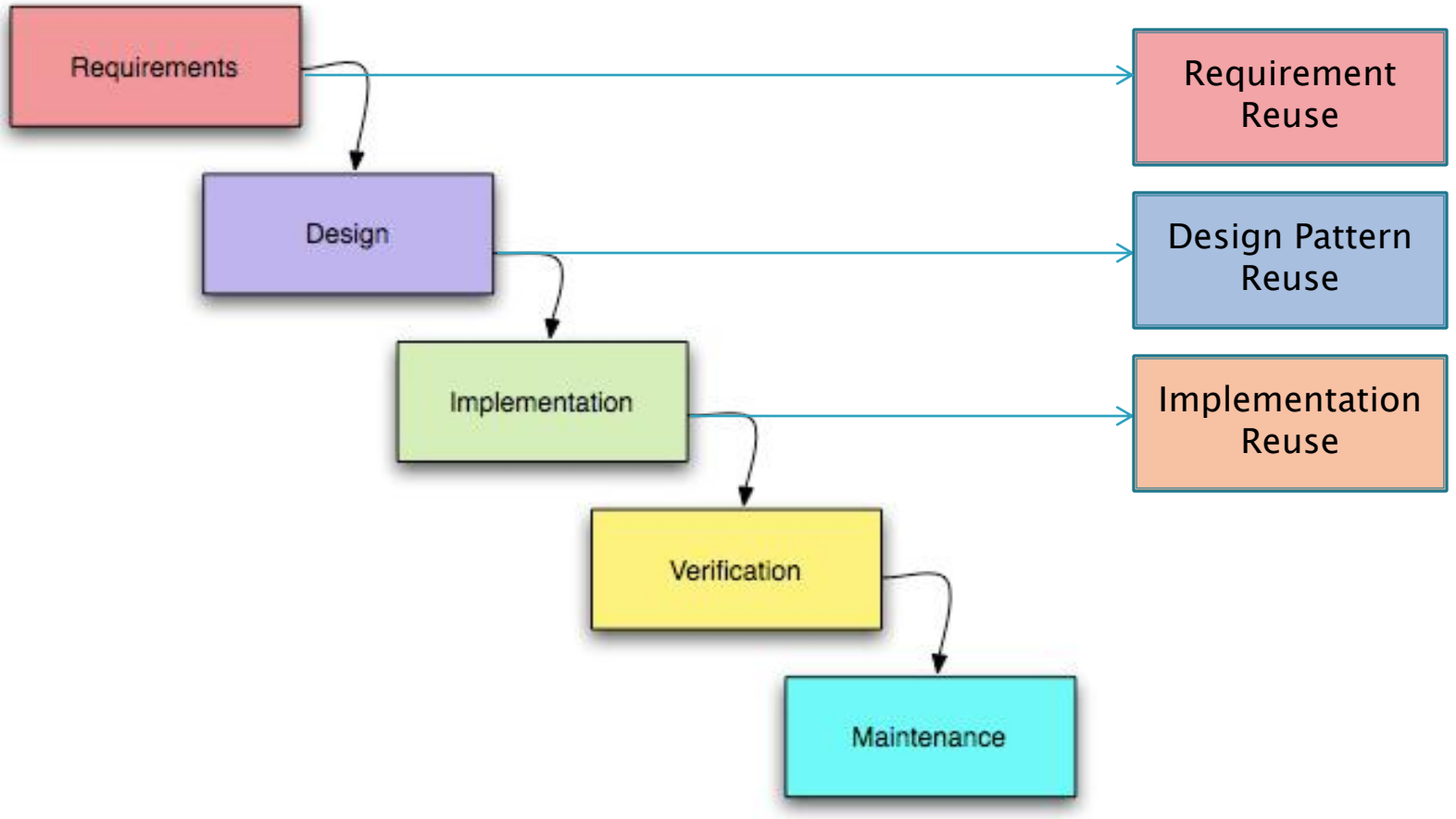
Module 4 Implementation Reuse (Part 4)

Dr. Shiping Chen

Where Are We?



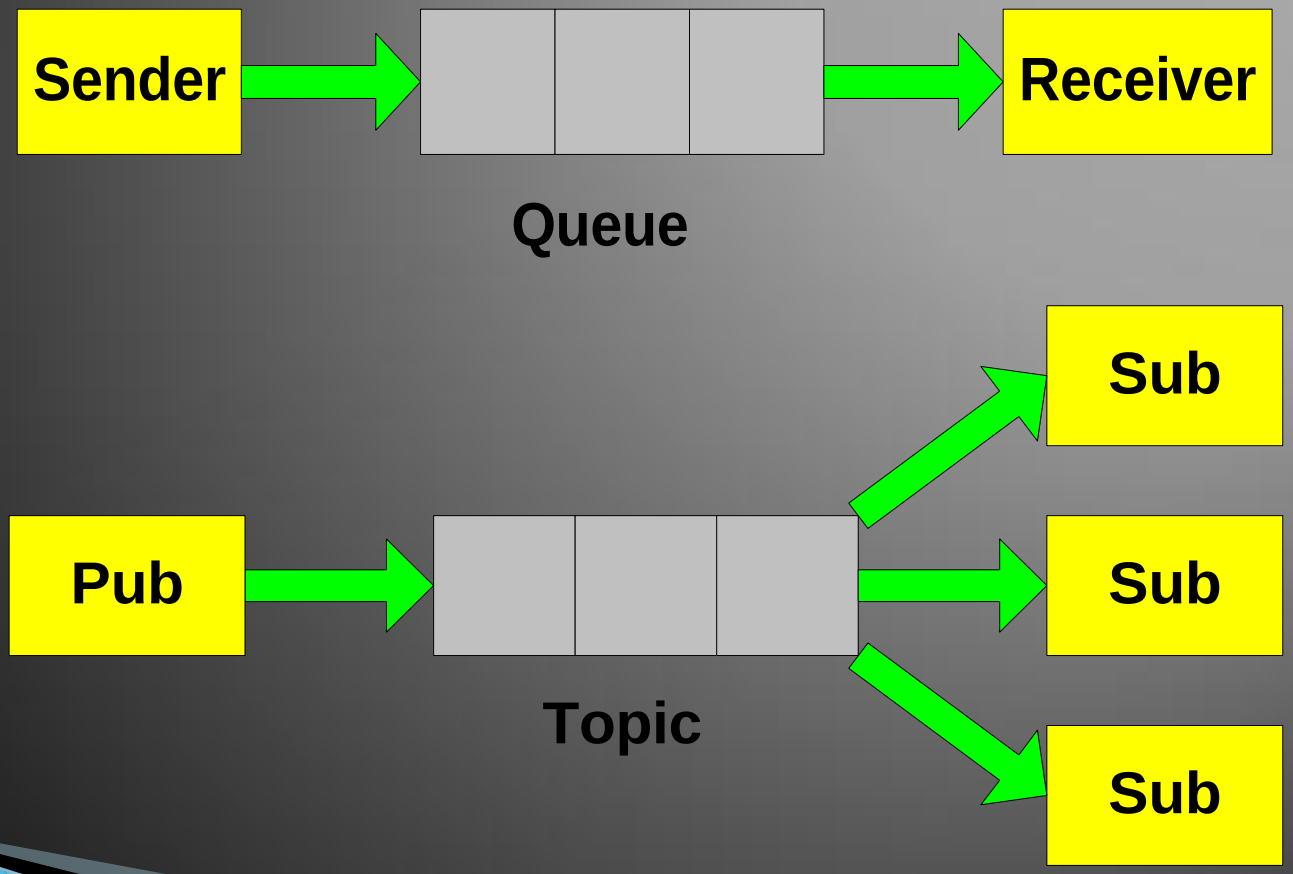
Agenda

- Messaging Technology
  - Overview
  - Basic Concepts and paradigms
  - Advices for programming
- Workflow for reuse
  - Motivations
  - Basic Concepts
  - Current Workflow Products
  - Overview of BizTalk
- SOA/SOC for reuse
  - Basic Concepts
  - SOA Example
  - Overview of Web Service Stack
  - BPEL: Business Process Execution Language

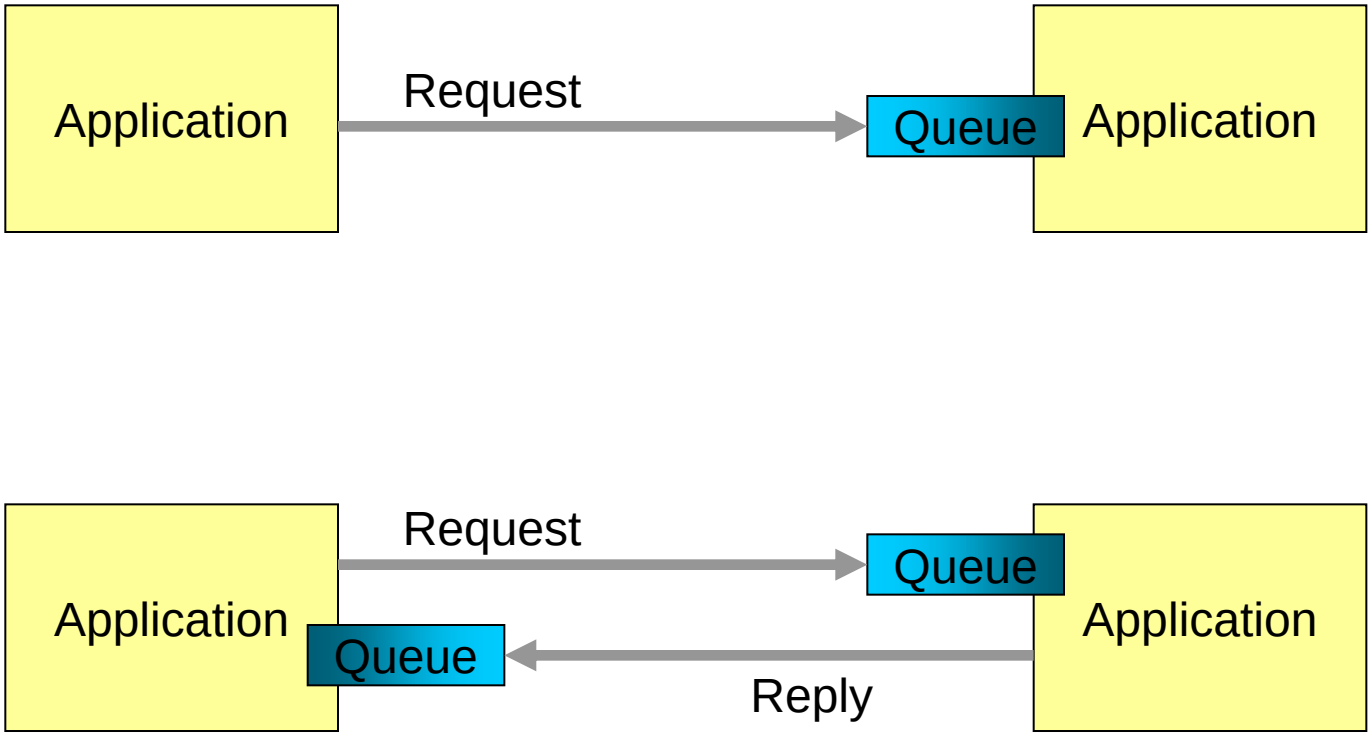
MOM – Messaging Oriented Middleware

- **A distributed technology/infrastructure that increases:**
  - Flexibility: loosely coupling
  - Efficiency/Performance: asynchronous sending/call
  - Robustness/Reliability: tolerant for system/network failure

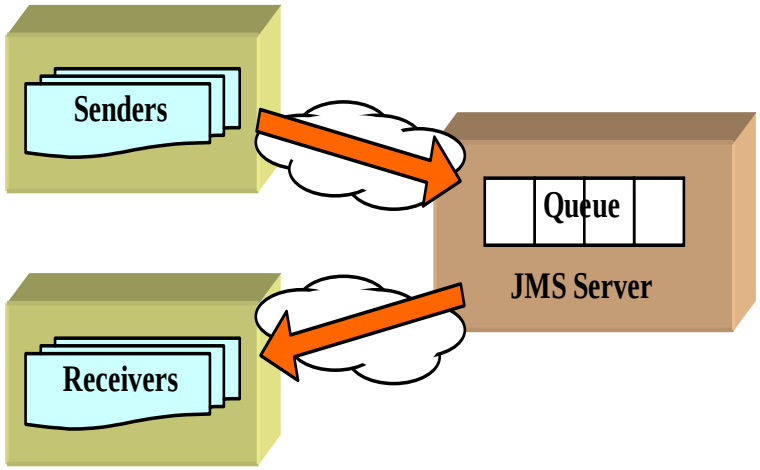
Message Consuming: P2P vs. Pub/Sub



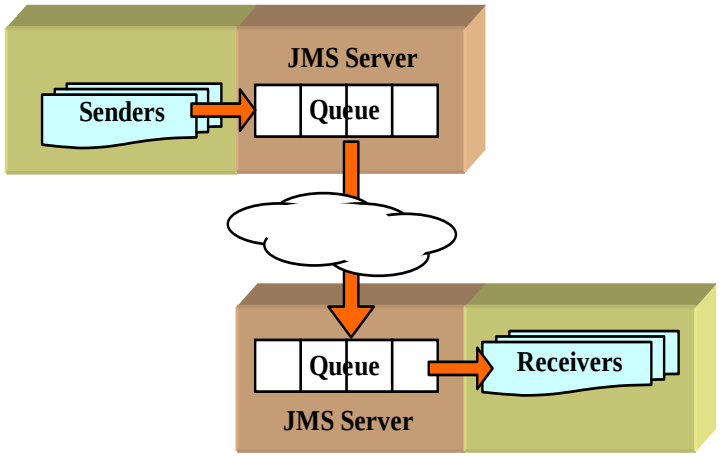
Message Transport: Asynchronous vs. Synchronous



Queue application: 1 Queue vs. 2 Queues



1 Queue



2 Queues

QoS of Messaging

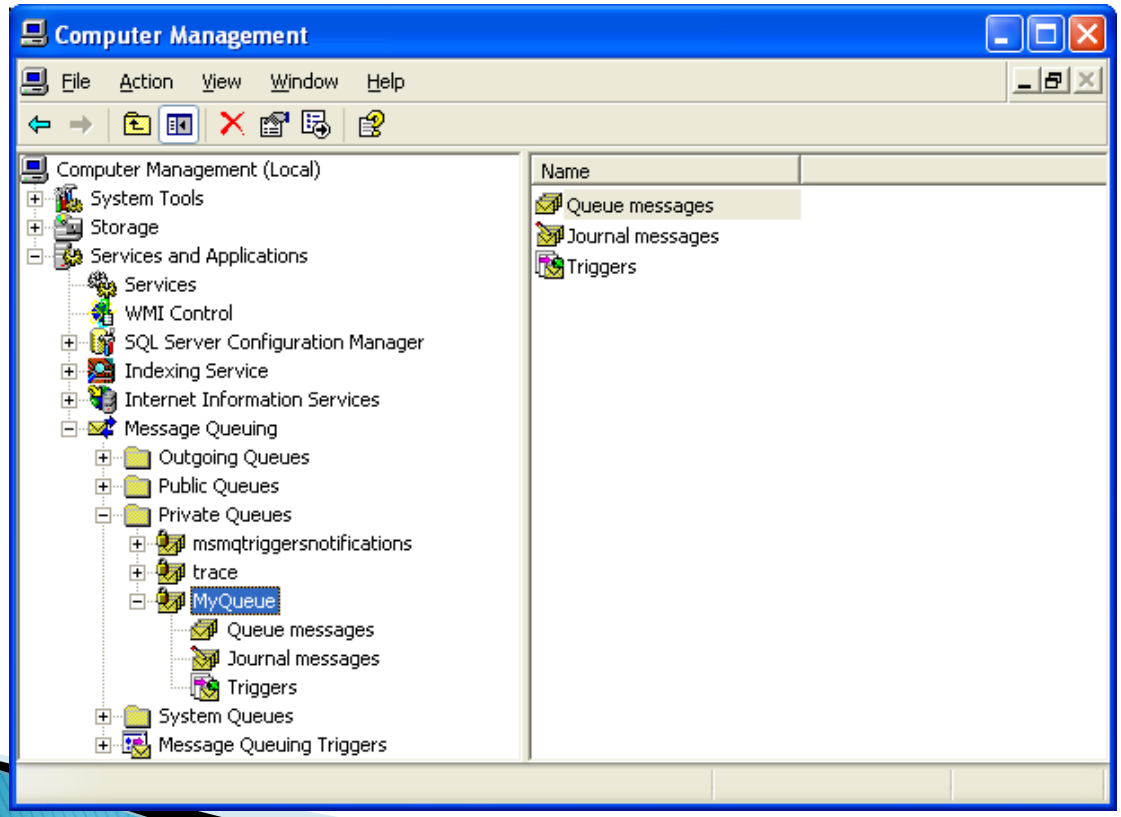
- Persistent
  - no messages in queues lost even on system/network failure
- Durable
  - Subscribers will receive all messages since subscribing no matter if the subscribers are running or off-line.
- Transcational
  - A group of messages are either all sent, or not sent any
  - A group of messages sending and a group of database updating are either successful, or nothing happening

Major MOM Players

- Vendors & Products
  - MicroSoft MSMQ
  - IBM Web Sphere MQ (Prevosuly called IBM MQ Series)
  - TIBCO www.tibco.com
  - Sonic MQ www.sonicsoftware.com
- Open Sources
  - JBossMQ (now owned by RedHat)
  - ActiveMQ www.logicblaze.com (now owned by IONA)
  - SwiftMQ www.swiftmq.com
  - OpenJMS http://openjms.sourceforge.net

MSMQ: MicroSoft Messaging Queues

- MicroSoft's Solution to Messaging
- Built into Windows XP Prof & Server Editions
- Programmable in C#, VB, C/C++



MSMQ C# Code Snapshot

```
using System.Messaging;
….

    // open a queue
    MessageQueue mq = null;
    mq = new MessageQueue();
    mq.Path = mqPath;

    // To send a string
    string msg = "Hello MSMQ";
    mq.Send(msg);

    // release
    mq.Close();
….
```

(a) Sender

```
using System.Messaging;
….

    // open a queue
    MessageQueue mq = null;
    mq = new MessageQueue();
    mq.Path = mqPath;

    // To receive a msg
    Message msg = mq.Receive();
    Console.WriteLine (msg.Body.ToString());

    // release
    mq.Close();
….
```

(b) Receiver

JMS: Java Messaging Services

A Java interface specification from Sun, which provides a standard way for Java applications to access enterprise MOM infrastructure in teams of:

- Unified Message Format
- Unified API
- Portable – Programming once, apply everywhere.

JMS Java Code Snapshot

```
...
// Common steps
cf = (QueueConnnectionFactory)
          ctx.lookup(cfName);
queue = (Queue) ctx.lookup(queueName);
con = cf.createQueueConnection();
session = con.createQueueSession(...);
con.start();

// Steps for a sender
s = session.createSender(queue);
msg = session.createTextMessage();
(To construct the message, i.e. msg)
...
s.send(msg);
s.close();
...
```

(a) Sender

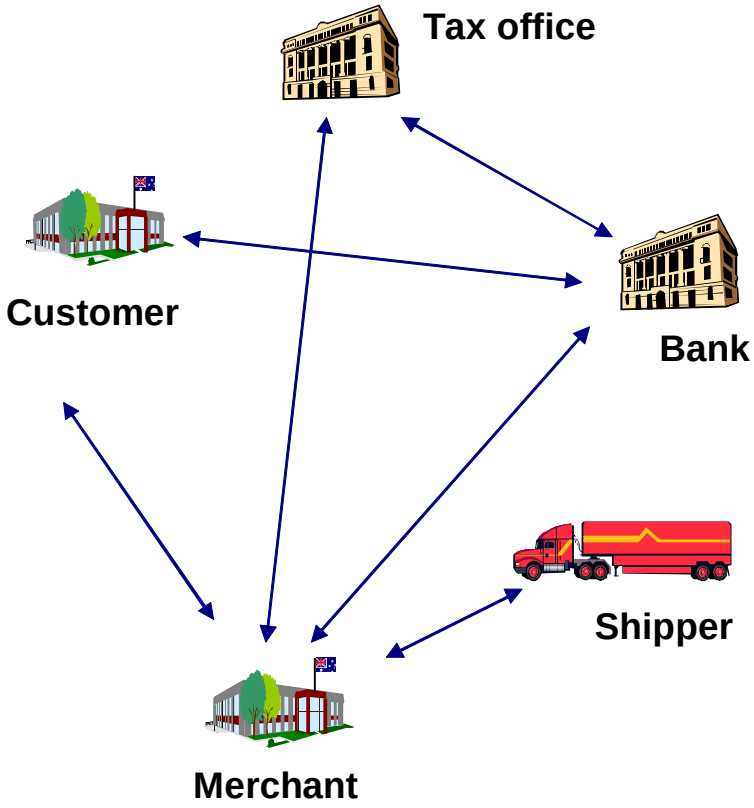
```
...
// Common Steps
cf = (QueueConnnectionFactory)
          ctx.lookup(cfName);
queue = (Queue) ctx.lookup(queueName);
con = cf.createQueueConnection();
session = con.createQueueSession(...);
con.start();

// Steps for a receiver
r = session.createReceiver(queue);
msg = r.receive(msg);
(To process the message, i.e. msg)
...
r.close();
...
```

(b) Receiver

SOA: Services Oriented Architecture

- Set a vision…
  - A peaceful and harmonious world
  - Applications everywhere working together through shared, open services
  - Seamless, easy integration between applications, across departments, across organisations, across the world
  - A single WWW ‘application’ – “there is only one application, and it’s still being written”…



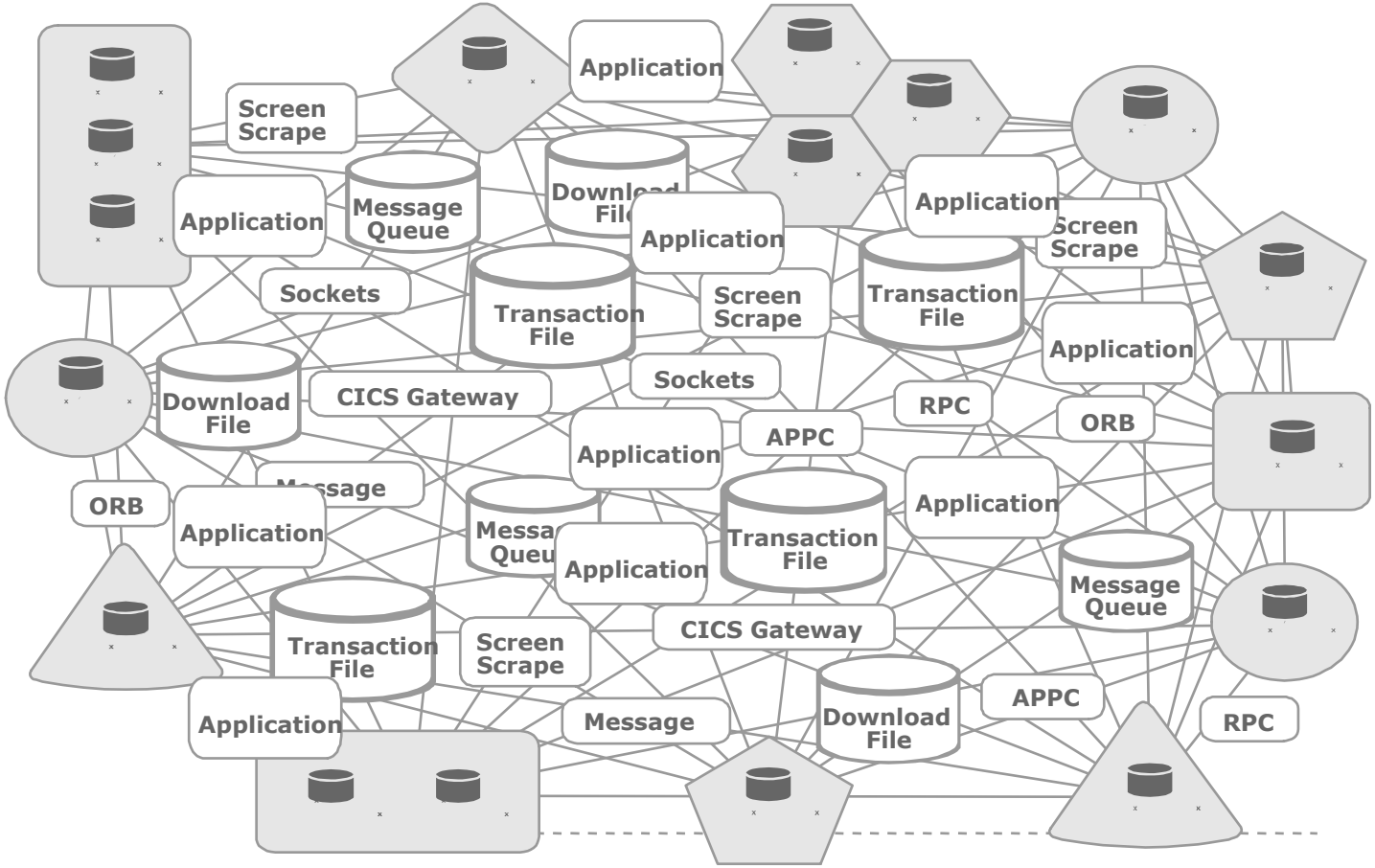
SOA Definition

- There are many, but no widely-agreed upon definitions. Here is one:
  - The service interface is *independent of the implementation*. Application developers or system integrators can build applications by *composing one or more services* without knowing the services' underlying implementations" (JavaWord)

SOA: Key Concepts

- Services are opaque
  - Only known the interface of its services, nothing else
- Services are autonomous
  - Controlled by others; can fail/change independently
- Services are loosely coupled
  - Independently developed/deployed, interact with minimal level of common knowledge between the consumer and services provider
- Services can be composed and federated
  - Composed to deliver a specific service, and federated for a specific business model
- Services should have business relevant granularity, i.e. Business-Driven

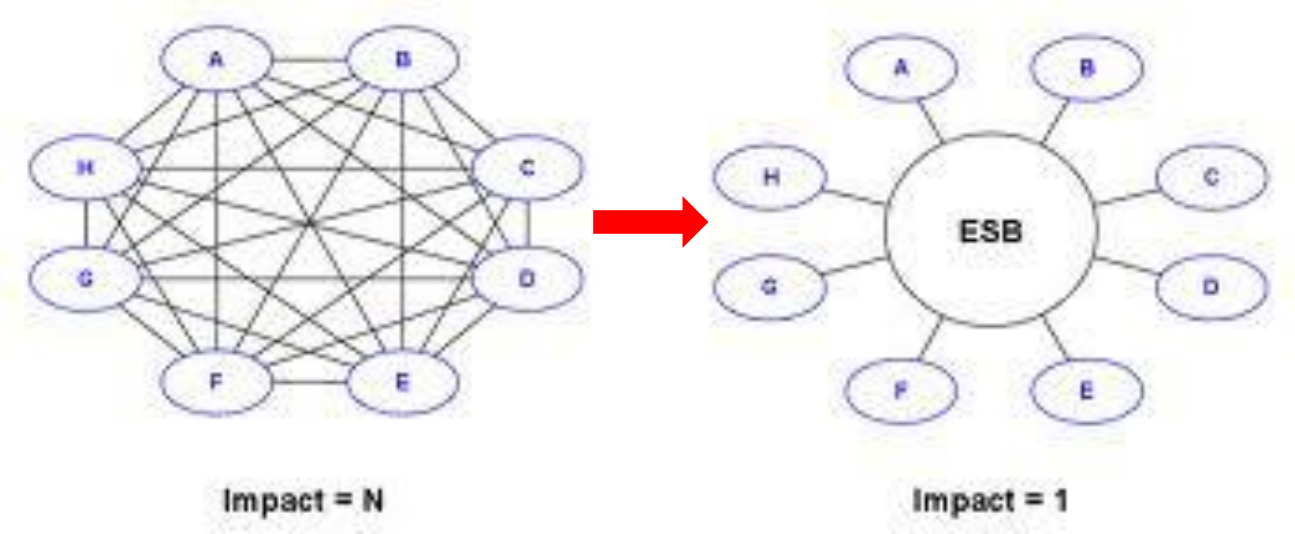
Why SOA? – The Cruel Reality



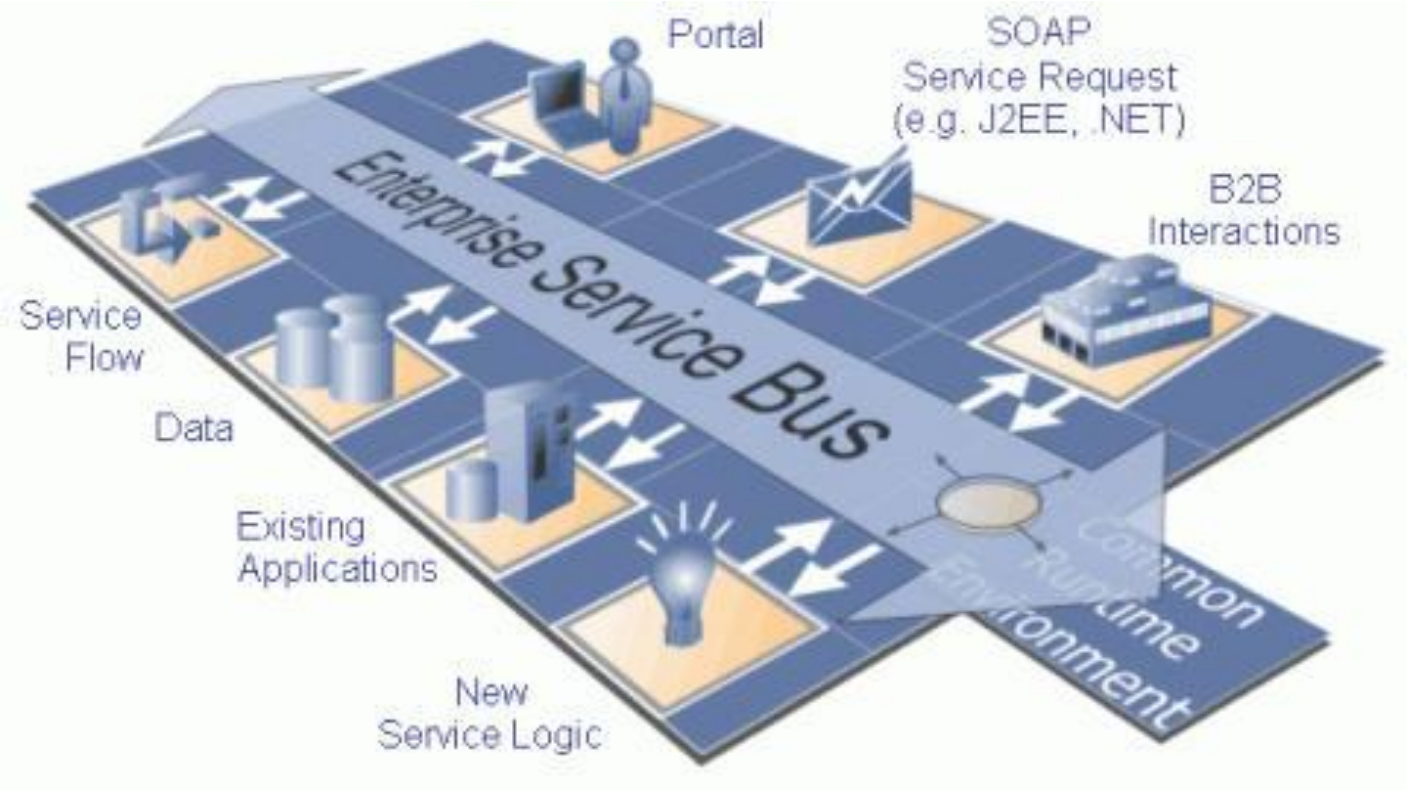
Source: Gartner report

Why SOA? – The Solution

- Respond to business changes
- Address new needs with existing applications
- Unlock existing application investments
- Support new channels & complex interactions
- Support organic business

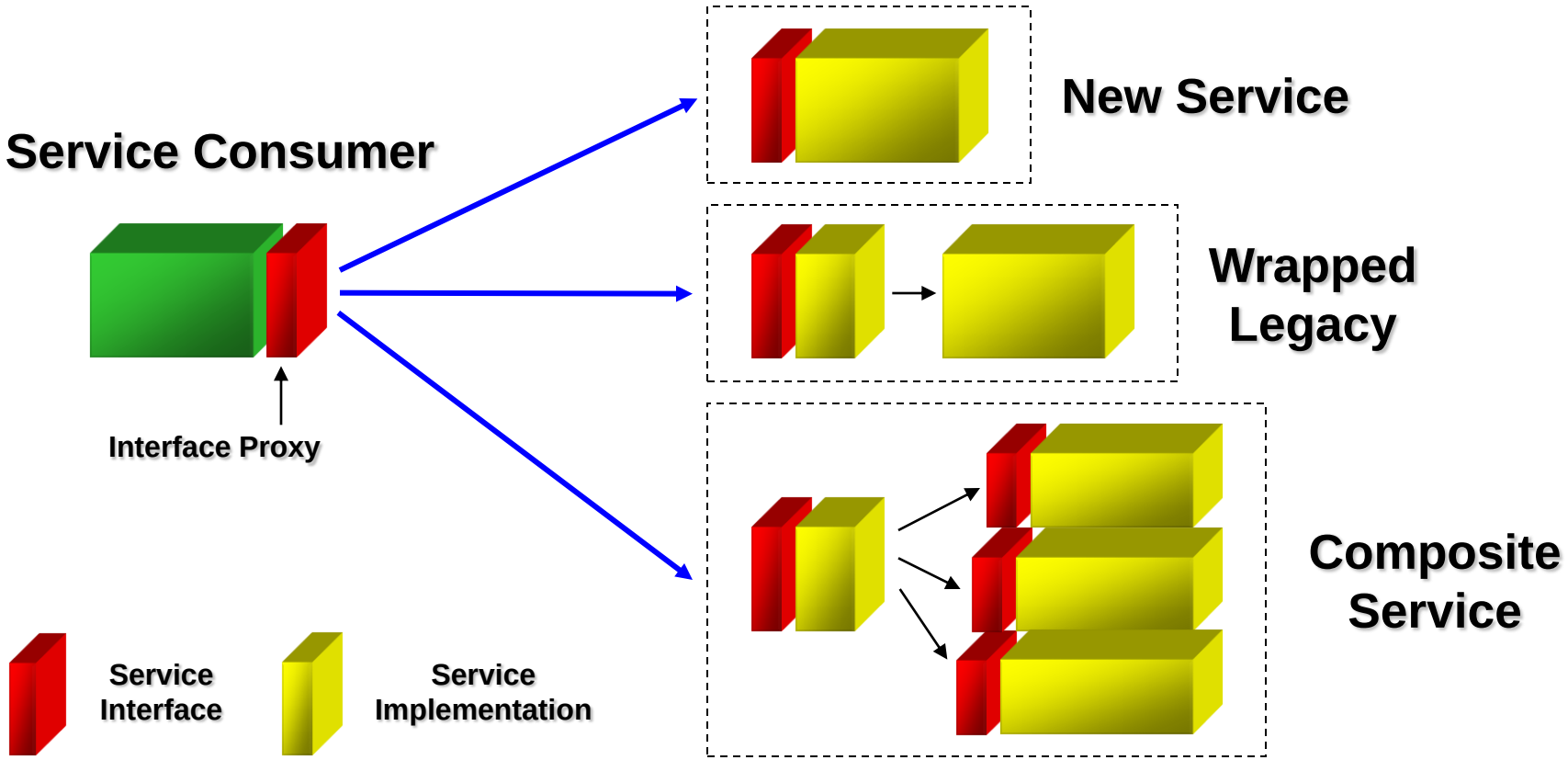


Enterprise Service Bus



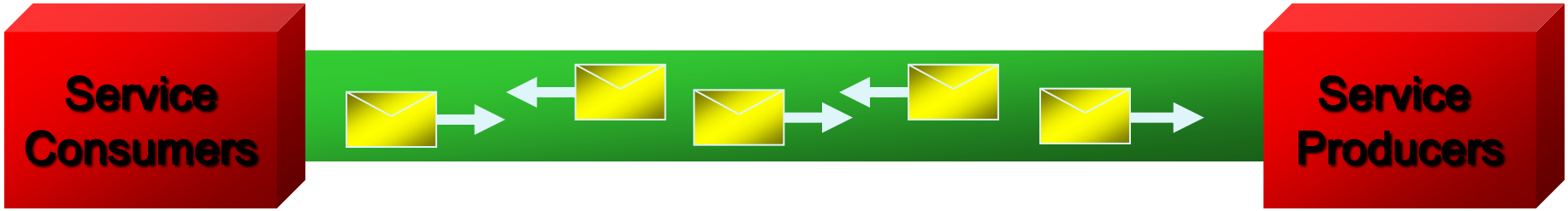
Source: http://www.hinttech.com/websphere/soa.html

Anatomy of a Service

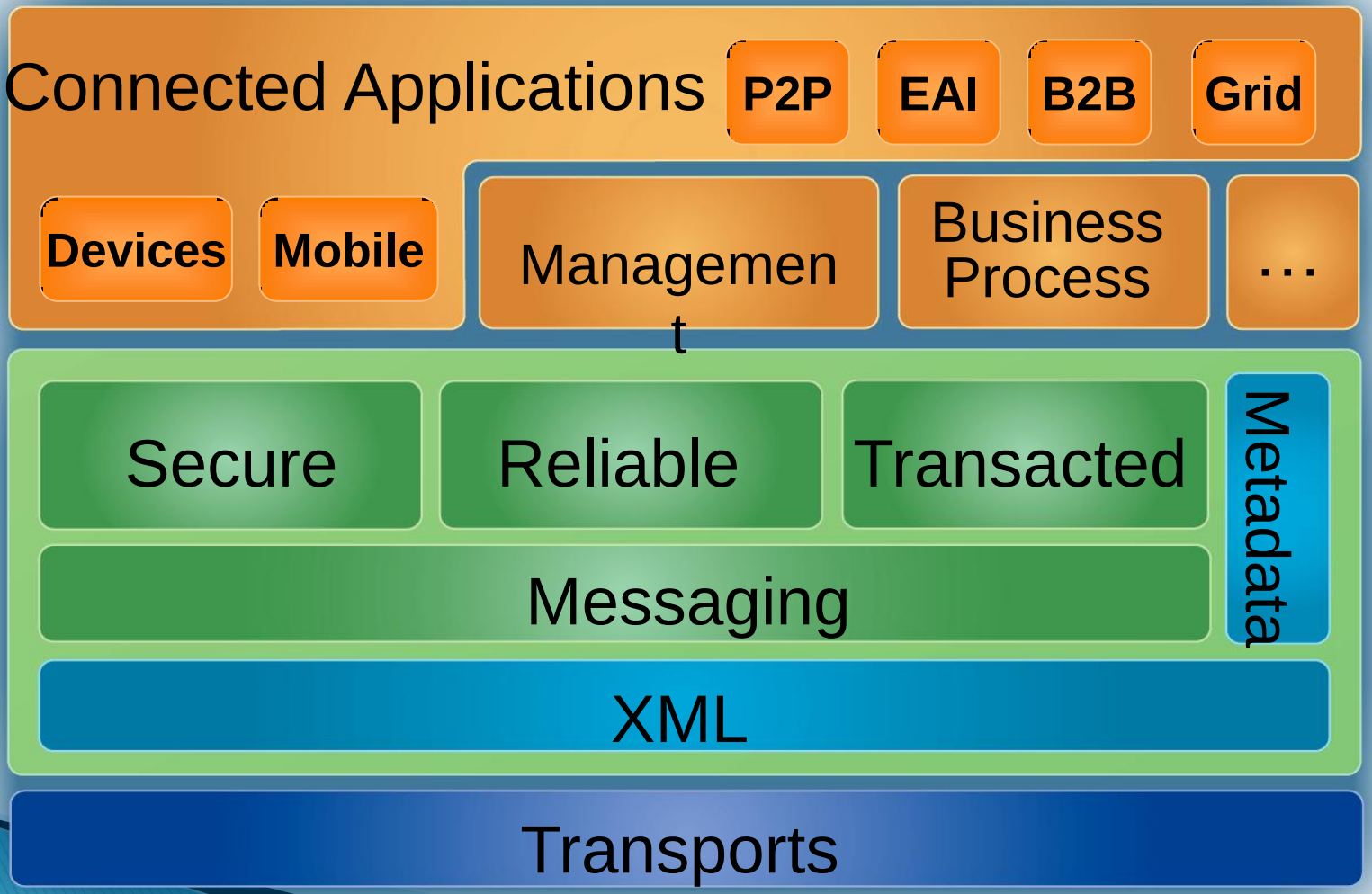


Service Communication

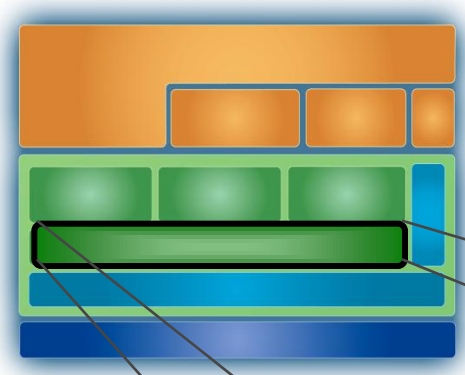
- Communicate with messages
- No knowledge about partner
- Likely heterogeneous



Web Services Stack

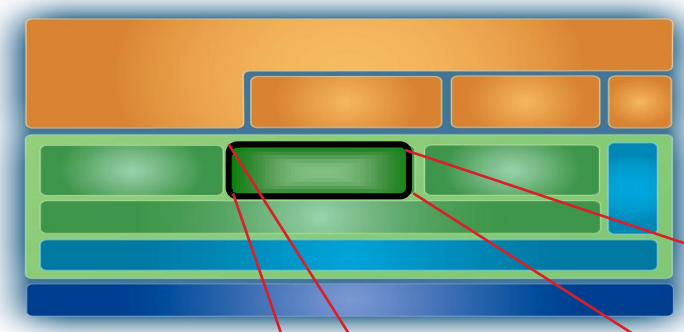


Messaging



- SOAP (We know it)
- URI = URL + URN
- WS-Addressing: construct address of services

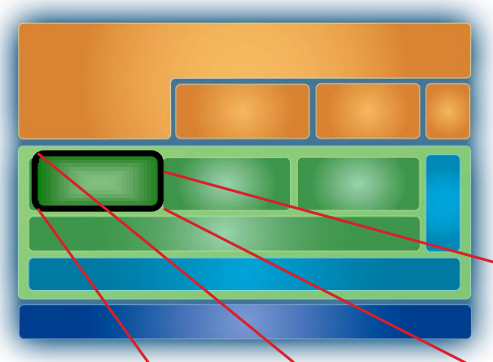
Reliability



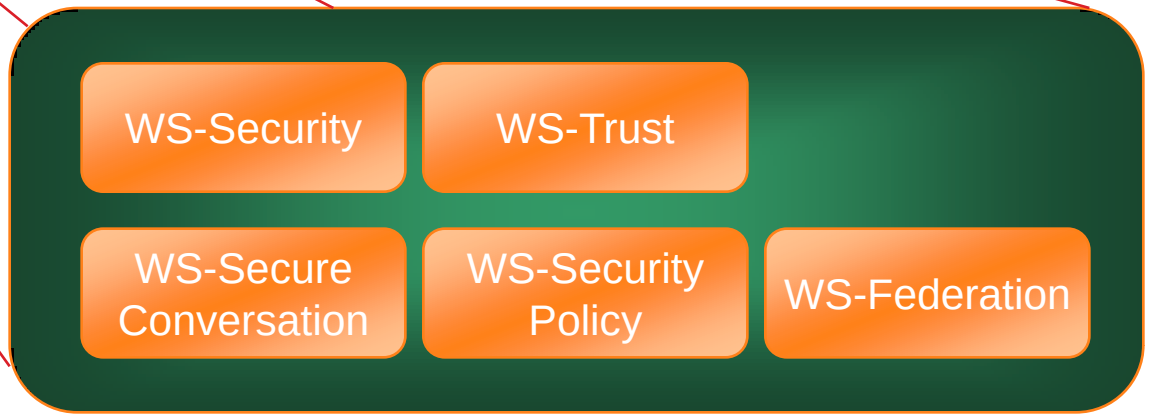
- How to specify QoS of messaging? Such as Exactly once, in-order

WS-ReliableMessaging

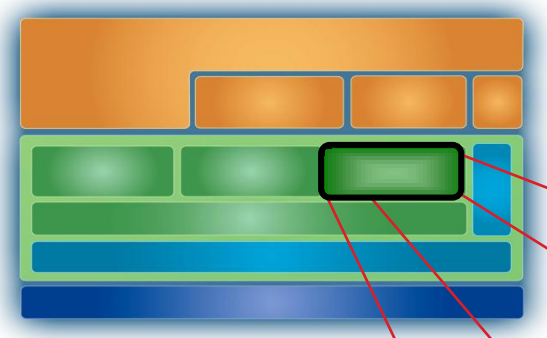
Security



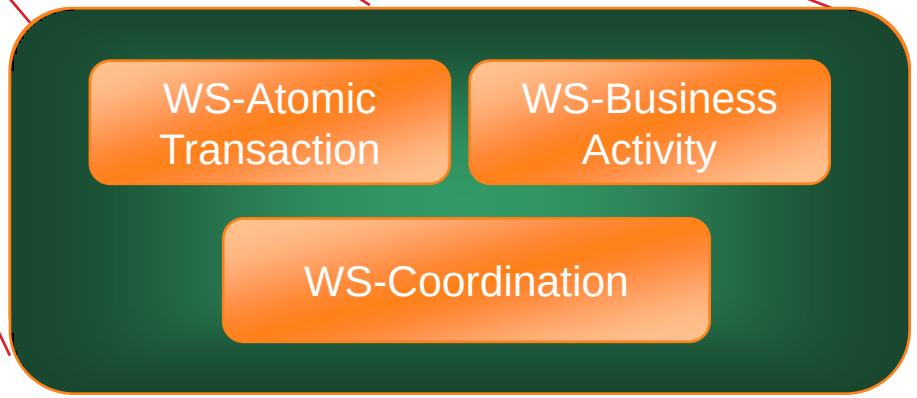
- WS-Security: we know that
- WS-Trust: About security tokens
- WS-Security Policy: how to apply WS-Security for web services
- WS-Security Conversation: how to maintain a secure session/conversation between web services
- WS-Federation: how to enable federation of identifications, authentications, authorisations across multiple services



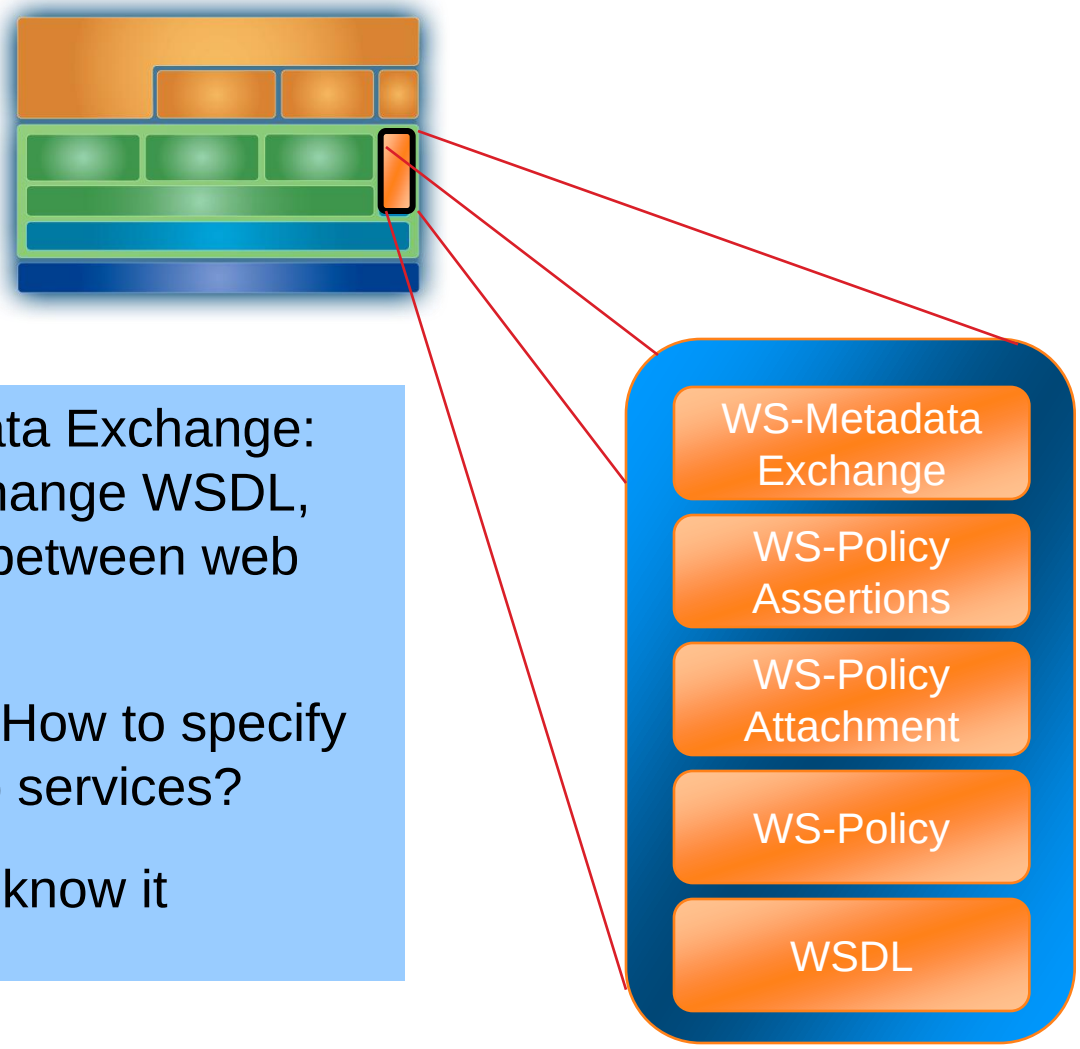
Transacted: Coordination & Consistency



- WS-Coordination: Protocol to set up transaction context
- WS-Atomic Tx: 2PC over SOAP
- WS-Business Activity: Looser consistency model, support compensation ‘rollback’



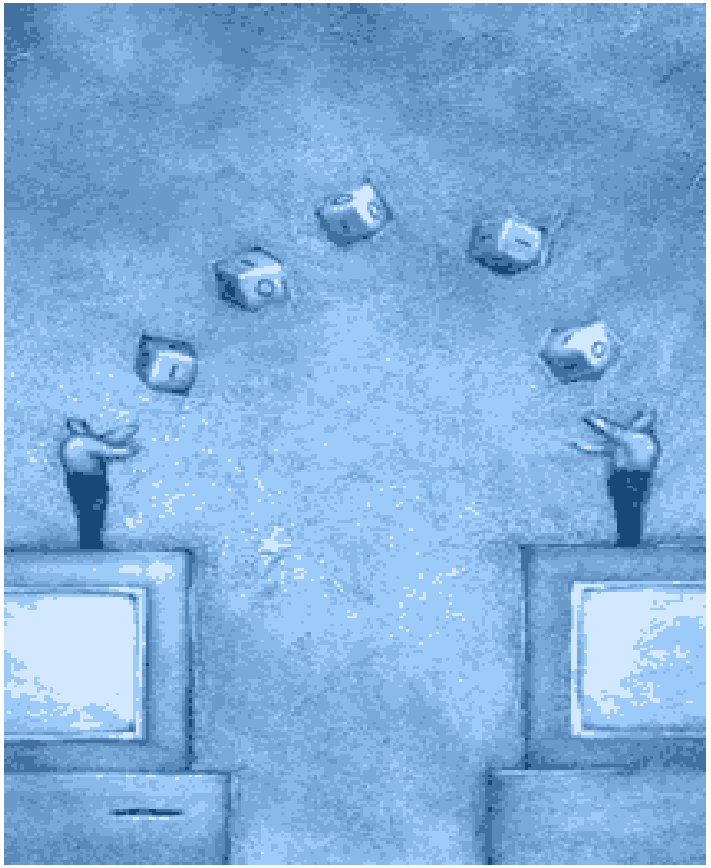
MetaData



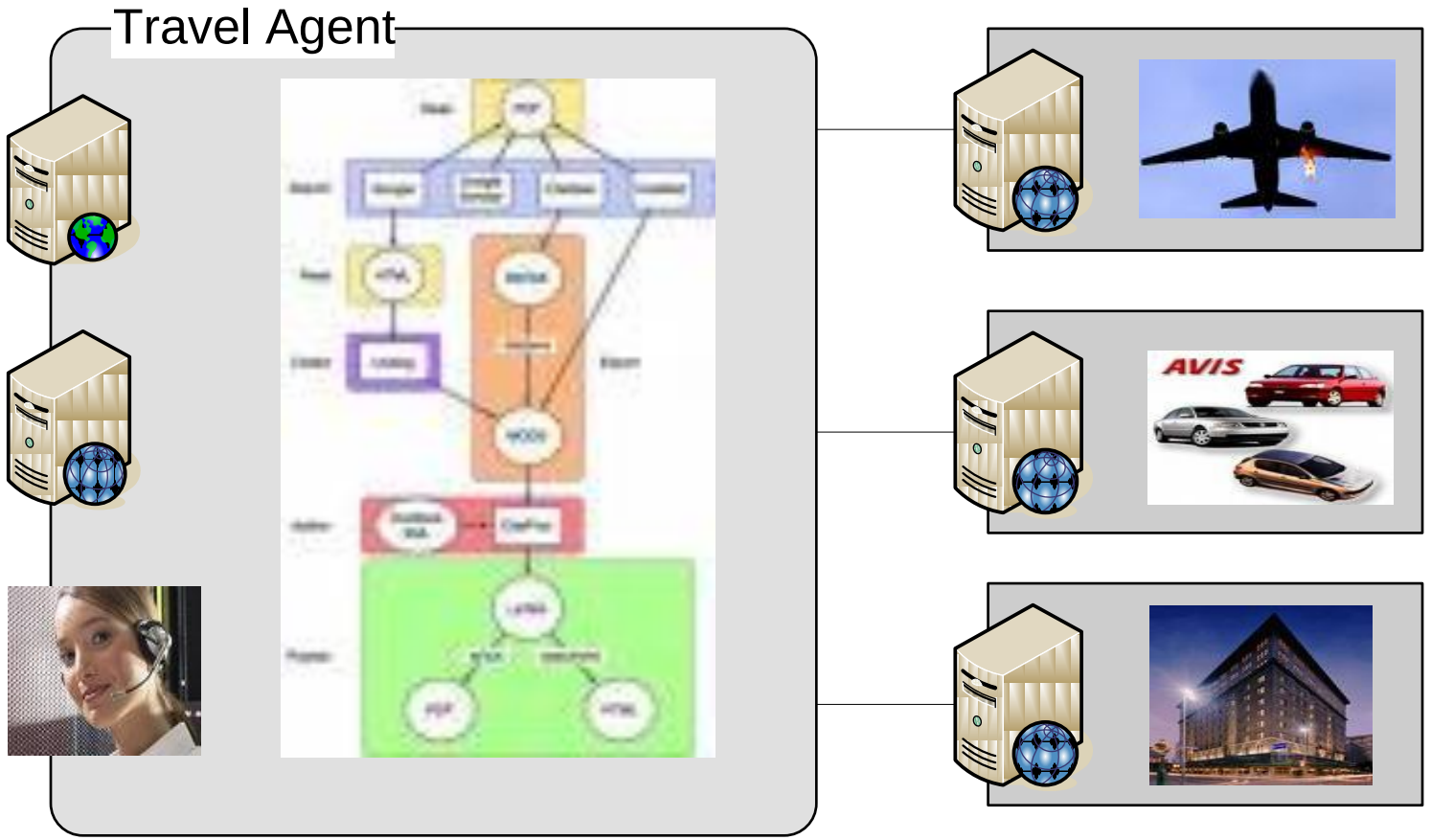
- WS-Metadata Exchange: how to exchange WSDL, WS-Policy between web services?
- WS-Policy: How to specify QoS of web services?
- WSDL: We know it

WS-Interop

- Industry body formed to ensure interoperability of WS*
  - Standard profiles of current standards
  - Compatibility testing to ensure interoperability
  - All major vendors participating
  - But WS is getting to be a big set of standards…
- http://www.ws-i.org/



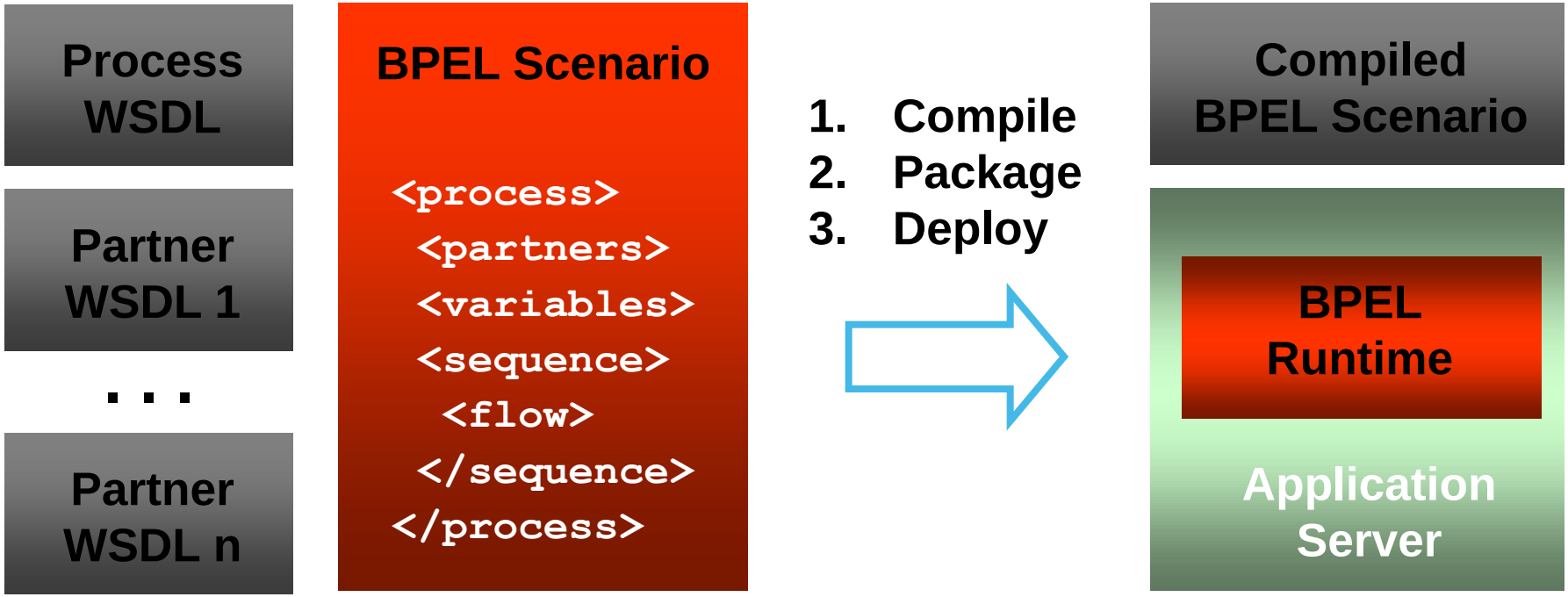
SOA Example. Travel Agent (Application)



BPEL:
Business Process Execution Language

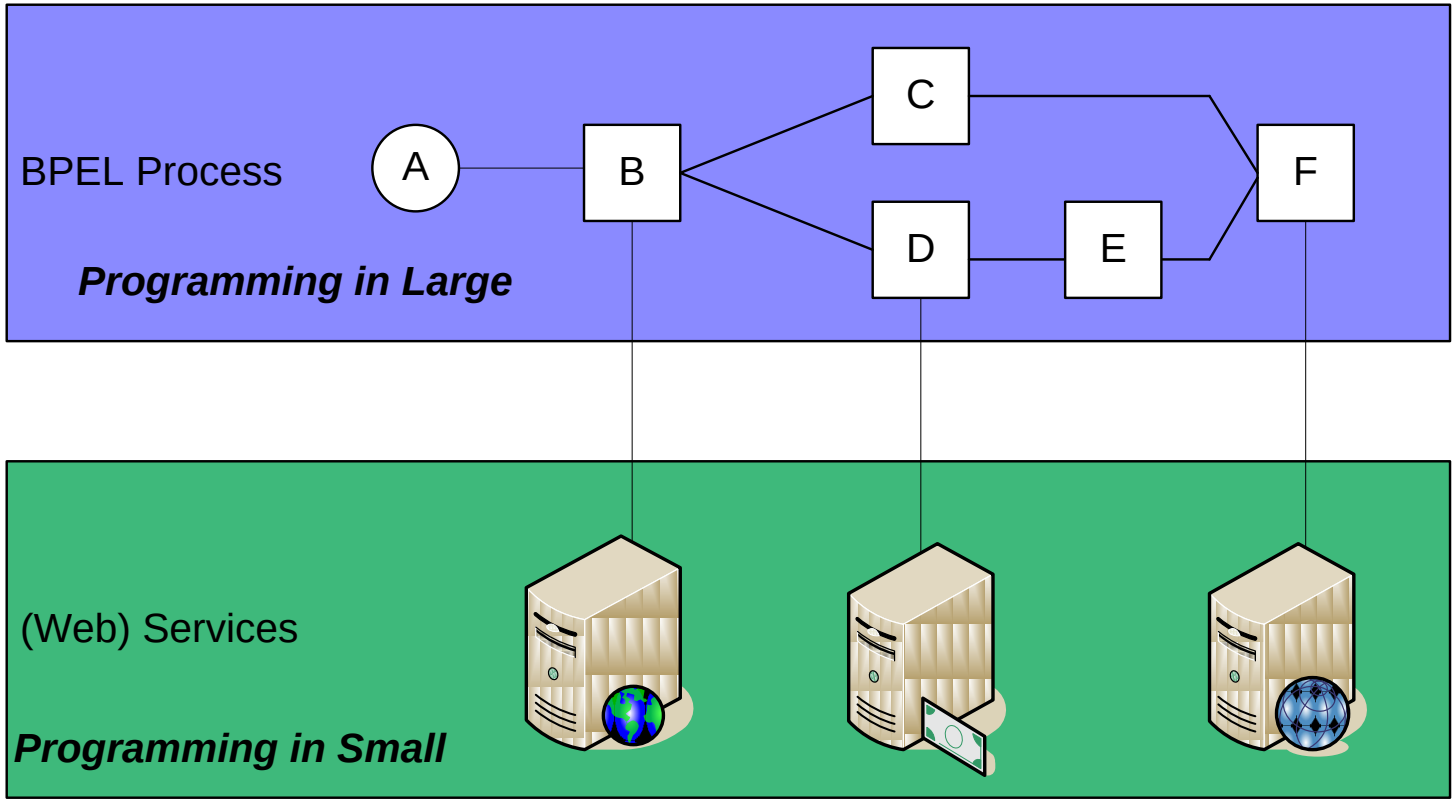
- Also called ‘BPEL4WS’
- Started by IBM and Microsoft, now trend to be standardised
- A XML-based language for the formal specification of business processes and business interaction protocols
- Based on existing WS* spec
- Orchestration Model (instead of choreography)

How Does BPEL Work?



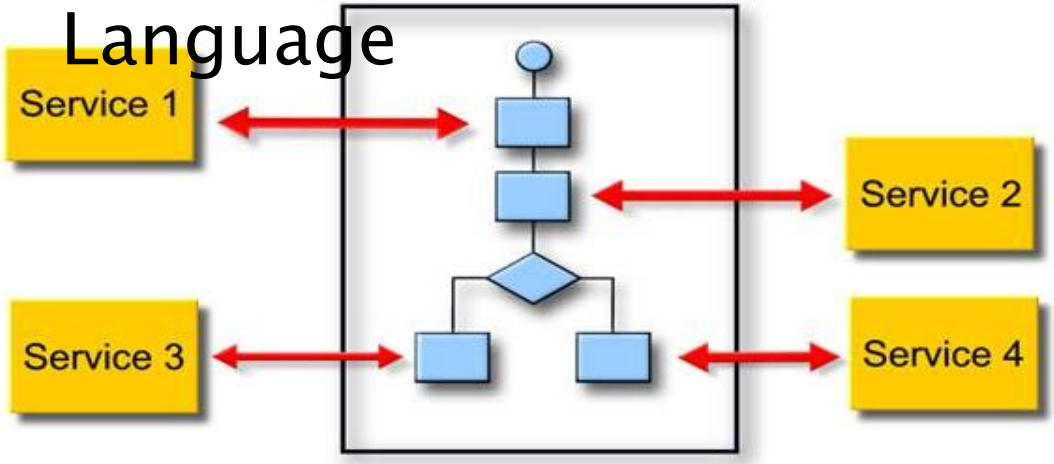
Details refer to http://www-128.ibm.com/developerworks/library/specification/ws-bpel/

Two Level Programming

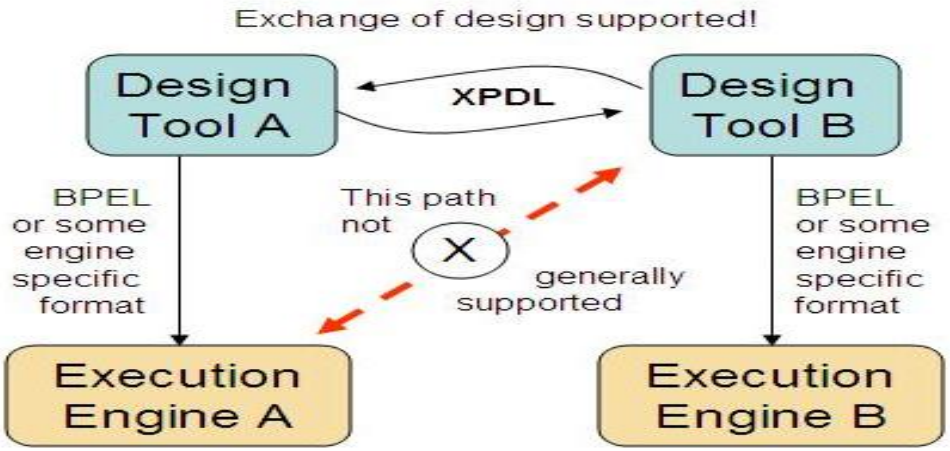


BPEL vs. XDPL

- BPEL – Business Process Execution Language



- **XDPL – XML Definition of Process Language**



Services: Orchestration vs. Choreography

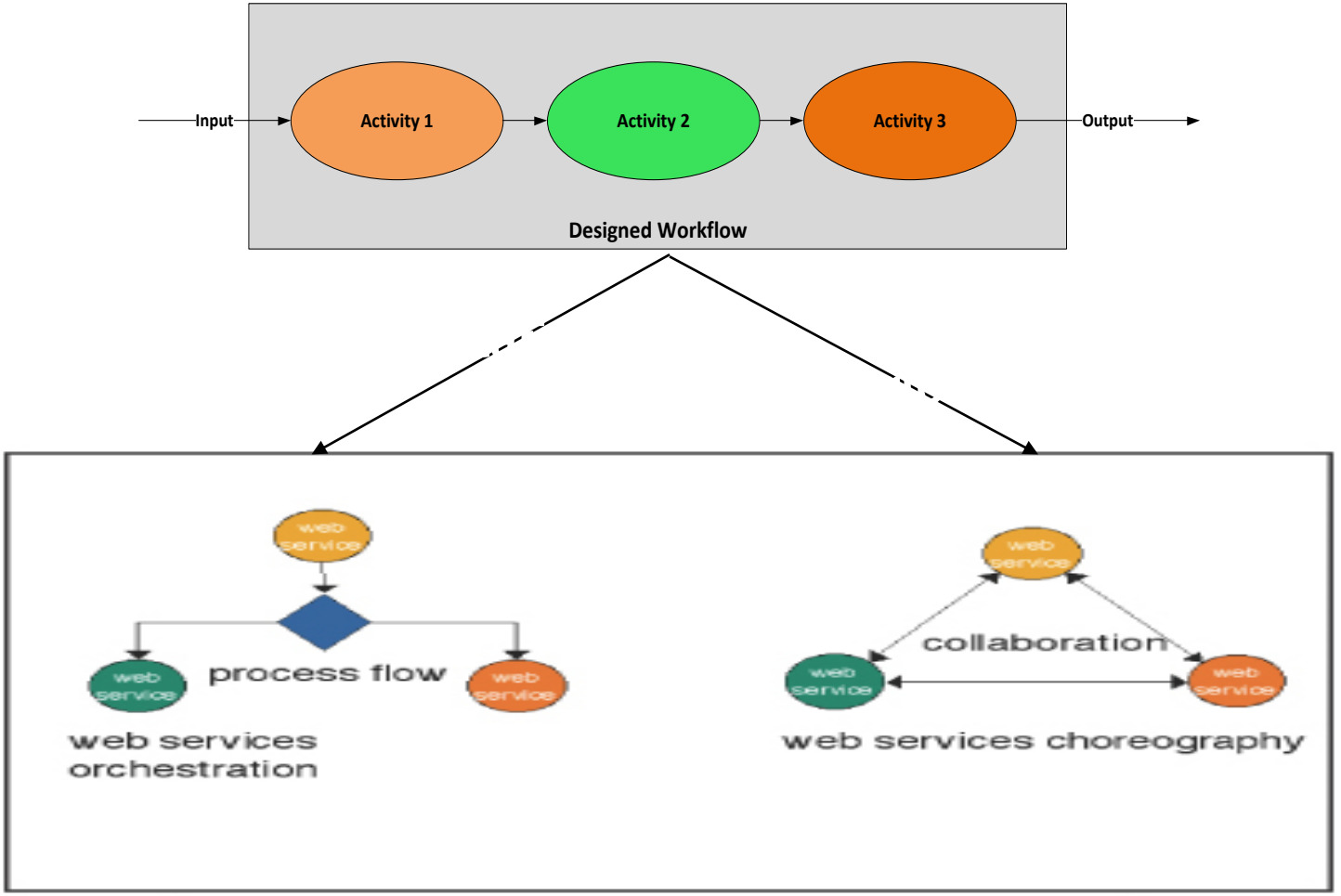


FIGURE 1 | Orchestration and choreography

Industry Support

- BEA provides full support for BPEL-Java in the release of WebLogic Integration following version 8.1
- IBM already offers that support as part of WebSphere Business Integration Server 5.1
- Microsoft BizTalk (from 2004) includes BPEL import and export capabilities
- Oracle acquired Collaxa, a company that focused on implementing the standard since 2002

Open Sources

- Apache ODE
  - http://ode.apache.org/
- Open EBS
  - http://www.open-esb.net/
- Active OVS
  - http://www.activevos.com/
- OW2 Orchestra
  - http://orchestra.ow2.org/xwiki/bin/view/Main/WebHome