أكاديمية الحكومة الإلكترونية الفلسطينية

The Palestinian eGovernment Academy

www.egovacademy.ps

Tutorial III:

Process Integration and Service Oriented Architectures

Session 10
ESB

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	3	Aldasht
Session15: UDDI	2	Aldasht

Session 10: ESB

Session ILOs

After completing this session students will be able to:

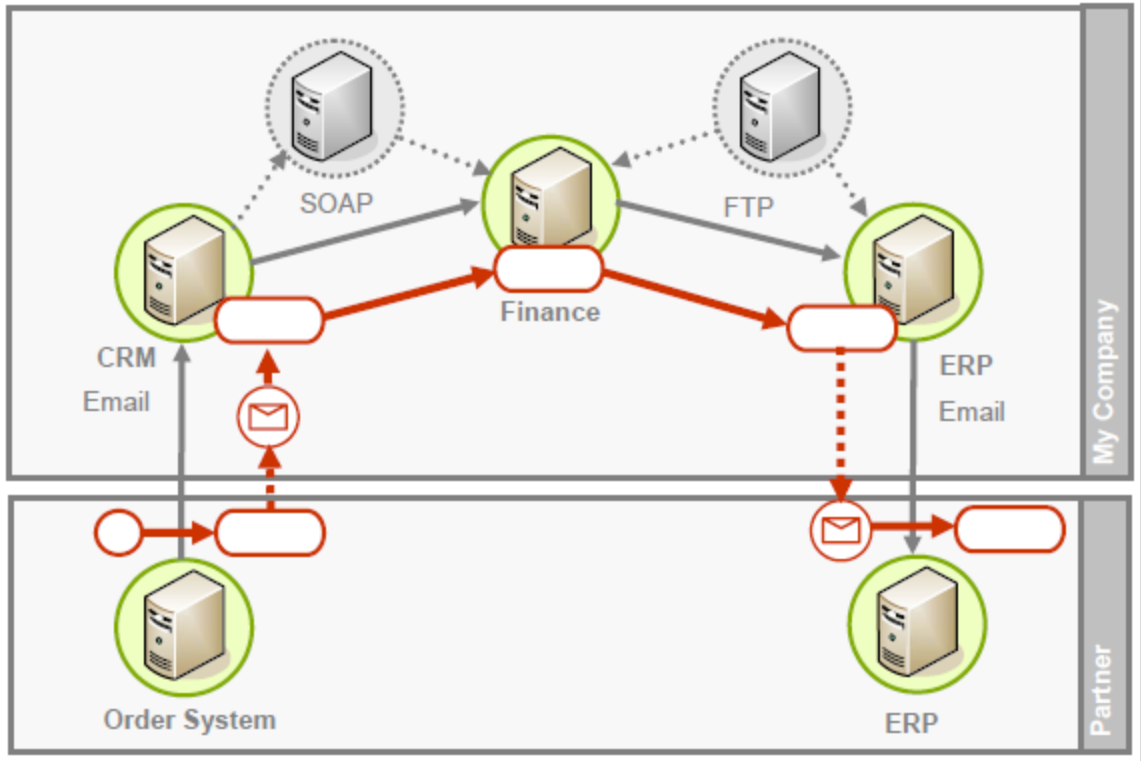
1. Explain the concept of enterprise service bus
2. Setup, invoke and deploy web services.

Session Outlines

- **Introduction**
  - ESB functions
- ESB role
- ESB Components
- Service invocation
- ESB Routing Facilities
- ESB example (Biztalk)

Introduction

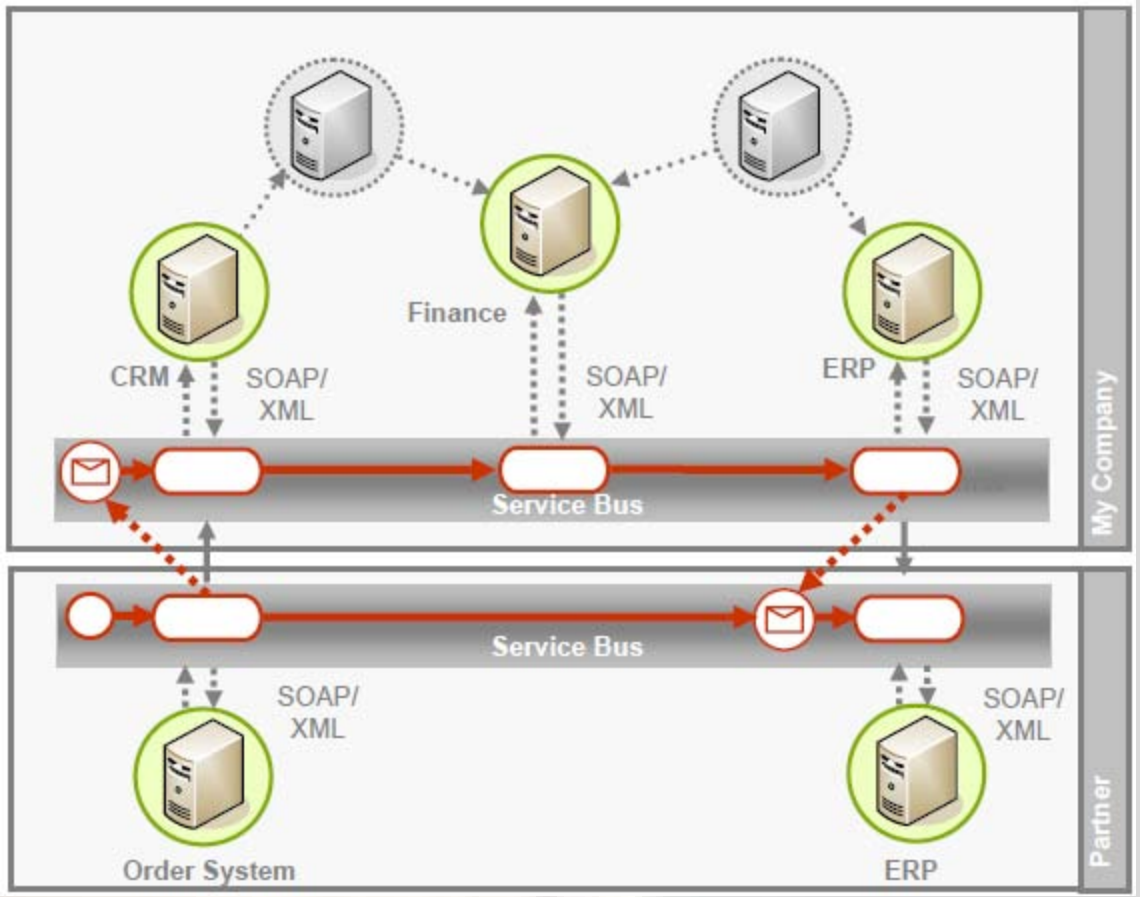
- Using point-to-point integration approach leads to unreliable, insecure, non-monitorable and in general non-manageable communication channels between applications.
- Process logic and data transformation logic is encoded into the applications.



Tightly-coupled applications, Source [6]

Introduction

- In an ESB architecture all kinds of applications are provided as business services and connected via reliable, secure and manageable virtual channels.
- Process orchestration and data transformation logic can be moved to the bus
- Process interactions can be performed in a controlled manner
- The ESB is an innovative approach to enterprise application integration (EAI)



Integrate all kinds of applications using a centralized EAI broker, Source [6]

Introduction: definitions

- The ESB is a new approach to integration that supports a loosely coupled, highly distributed integration network [1].
  - Shortly, a standards-based integration platform
- Also, we can say that an ESB is a flexible connectivity infrastructure for integrating applications and services [4].

ESB 4 functions [4]

- An ESB performs the following between requestor and service:
  - **Routing** messages between services.
  - **Converting** transport protocols between requestor and service.
  - **Transforming** message format between requestor and service.
  - **Handling** business events between different sources.

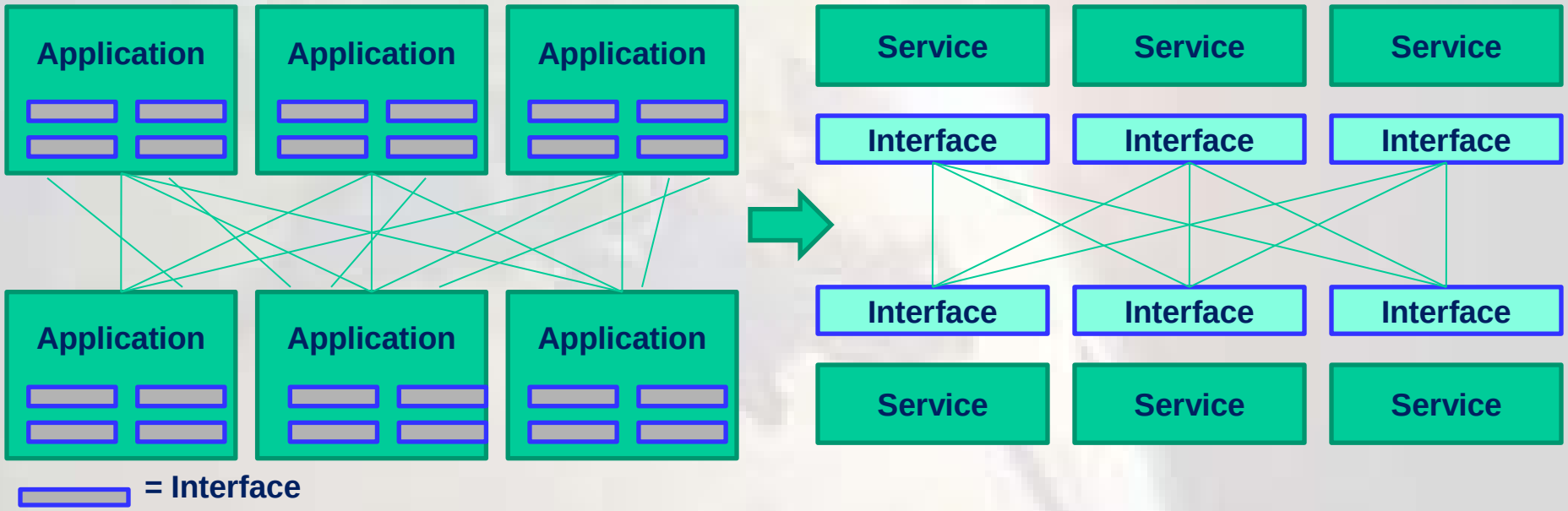
Session Outlines

- Introduction
  - ESB functions
- **ESB role**
- ESB Components
- Service invocation
- ESB Routing Facilities
- ESB example (Biztalk)

SOA Role

- SOA decouples interfaces from applications.
- The number and complexity of interfaces is reduced.
- Business applications and their interfaces become reusable.
- Interfaces tightly coupled with *point-to-point* connections

SOA Role

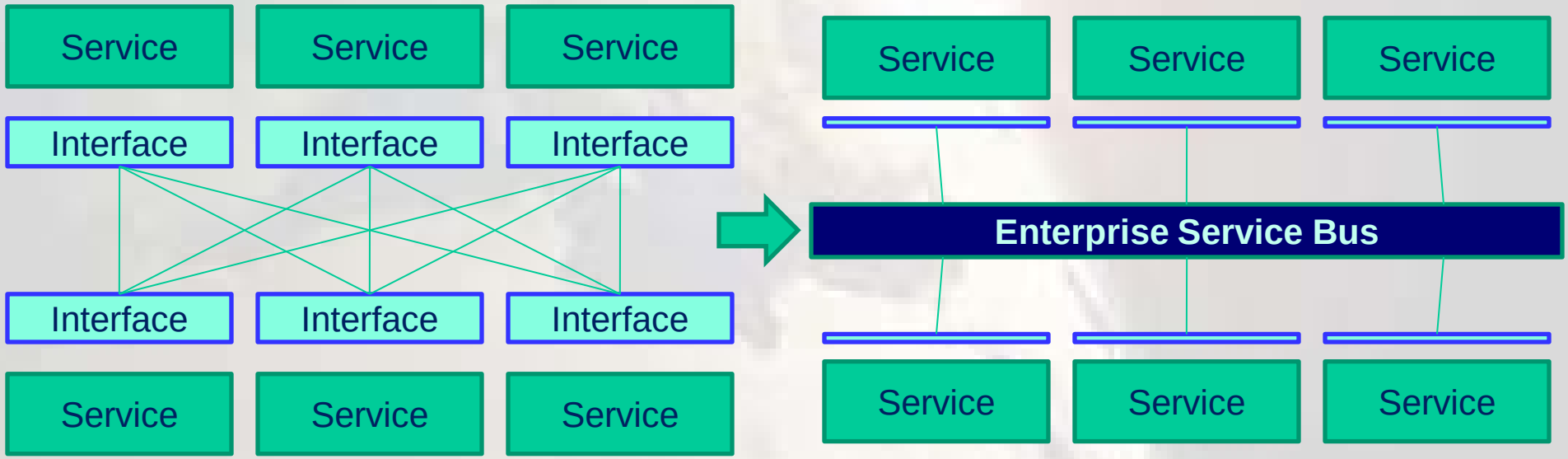


Source, IBM education assistant:
http://publib.boulder.ibm.com/infocenter/ieduasst

ESB Role

- ESB decouples the point to point connections from the interfaces.
- More flexible coupling and decoupling of applications
- Enables for finding applications and interfaces for reuse.
- Add or modify services faster and with less impact on existing services.

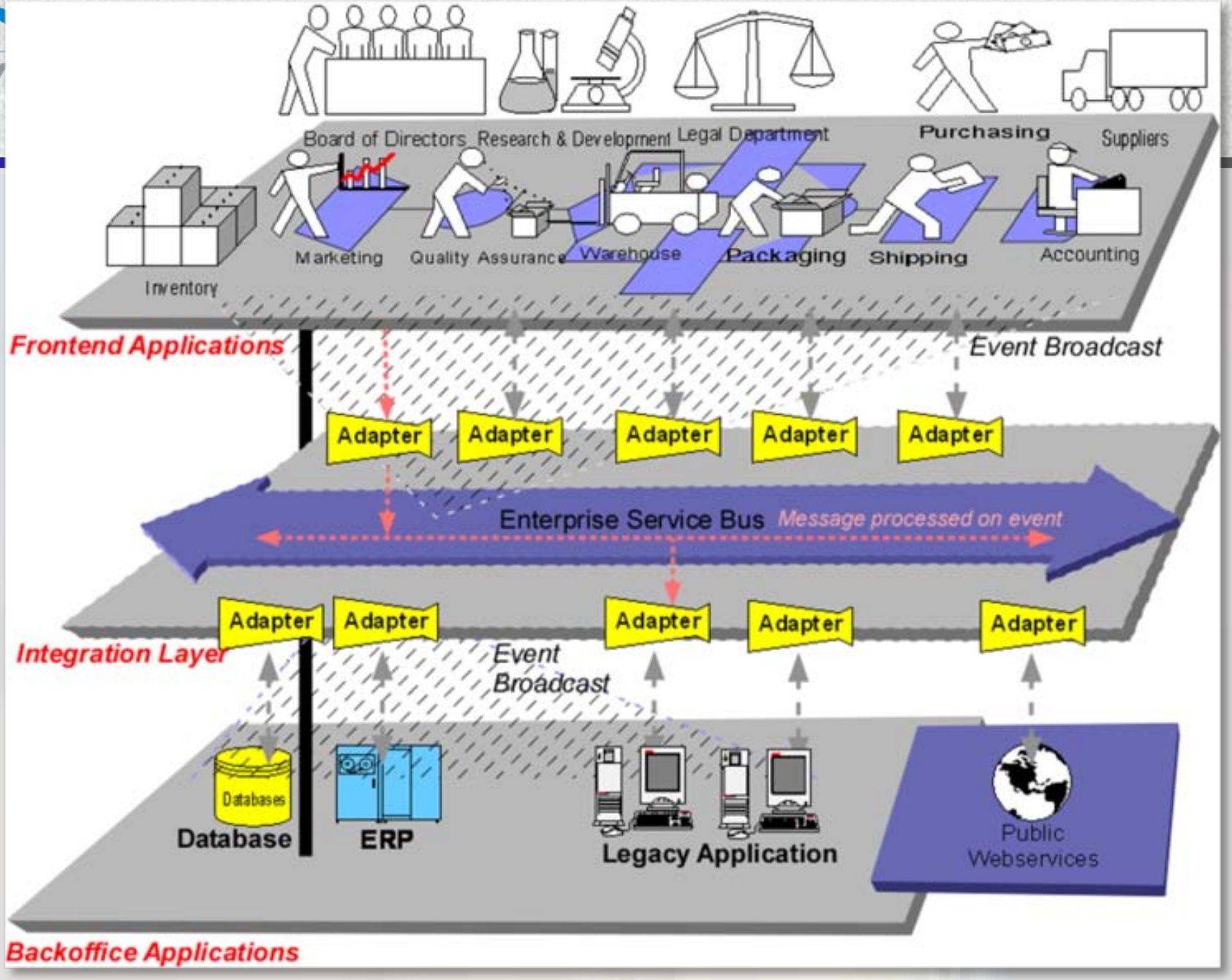
ESB Role



Source, IBM education assistant:
http://publib.boulder.ibm.com/infocenter/ieduasst

ESB role

- In an ESB, applications and event-driven services are tied together in an SOA in a loosely coupled fashion.
- An ESB provides the implementation backbone for an SOA.
- An ESB provides a loosely coupled, event-driven SOA with a highly distributed universe of named routing destinations across a multiprotocol message bus.
- Applications (and integration components) in the ESB are abstractly decoupled from each other, and connect together through the bus as logical endpoints that are exposed as event-driven services.



ESB forms a pervasive grid that can span a global enterprise network, Source [5]

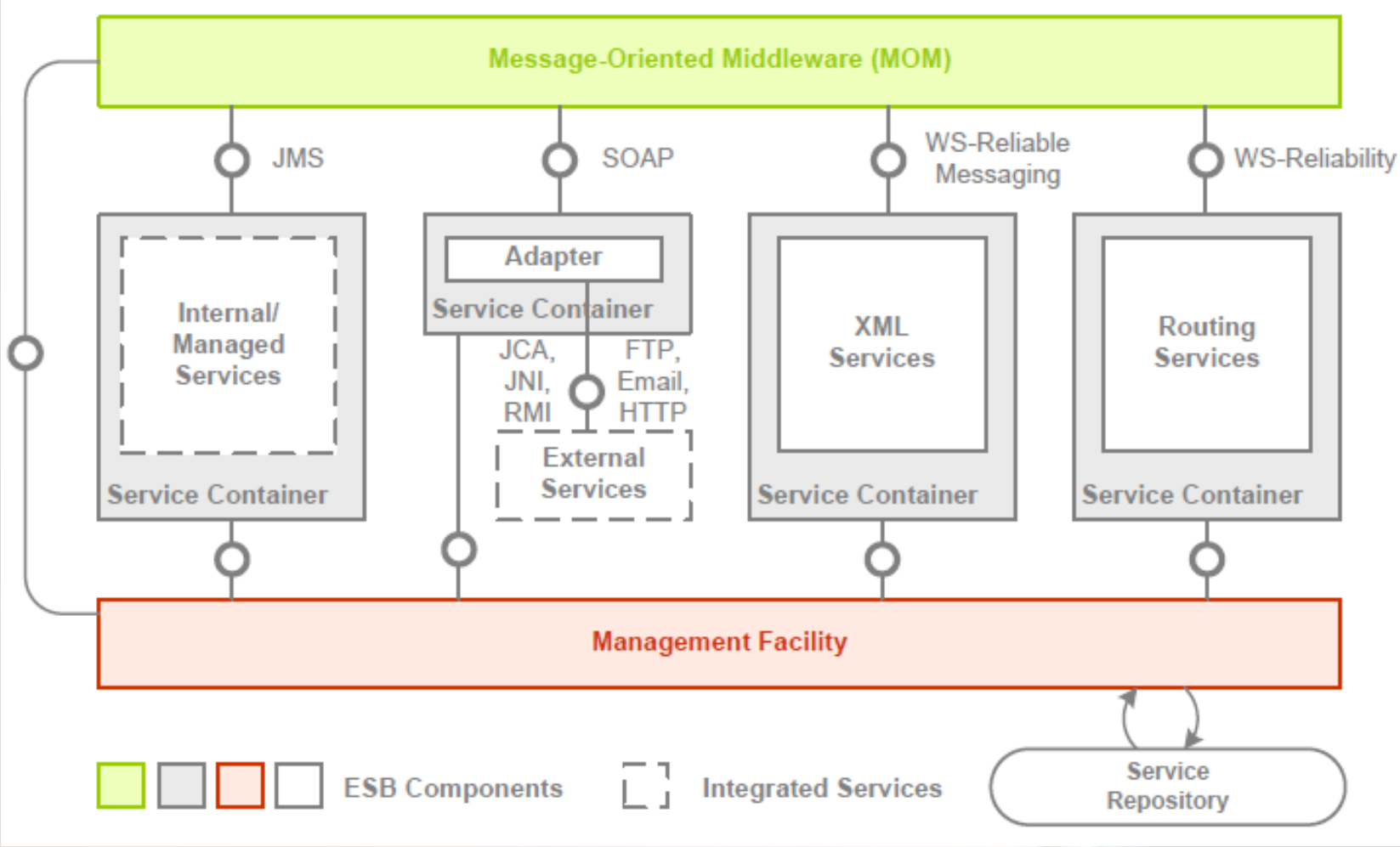
Some ESB Characteristics

- Standards-Based Integration [1].
  - ESB can utilize J2EE components for connectivity.
  - ESB also integrates with applications built with .NET, COM, C#, and C/C++.
  - Can easily integrate with anything that supports SOAP and web-services APIs.
- Distributed Data Transformation.
  - Transformation services can be located anywhere and accessible from anywhere on the bus
- Remote Configuration and Management.
  - Local IT staff can't always be at each remote location

Session Outlines

- Introduction
  - ESB functions
- ESB role
- **ESB Components**
- Service invocation
- ESB Routing Facilities
- ESB example (Biztalk)

ESB Components

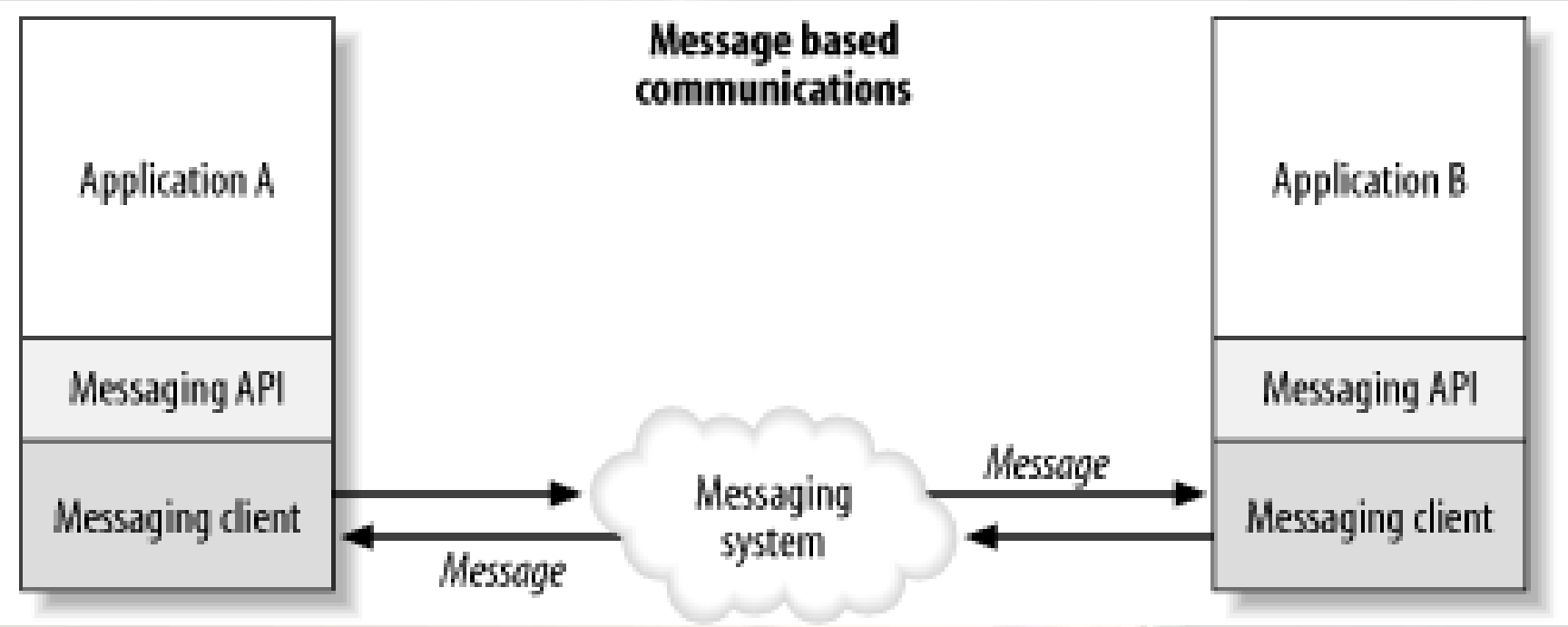


3 key components of an ESB architecture: MOM, Service container, and Management facility, Source [5]

Message Oriented Middleware (MOM)

- A MOM is a key part of the ESB architecture [1].
  - It provides a network of virtual channels that an ESB uses to route messages throughout an enterprise and beyond.
- In a MOM-based communication environment, messages are usually sent and received asynchronously
- MOM in combination with message queues are fundamental components of the ESB

Message Oriented Middleware

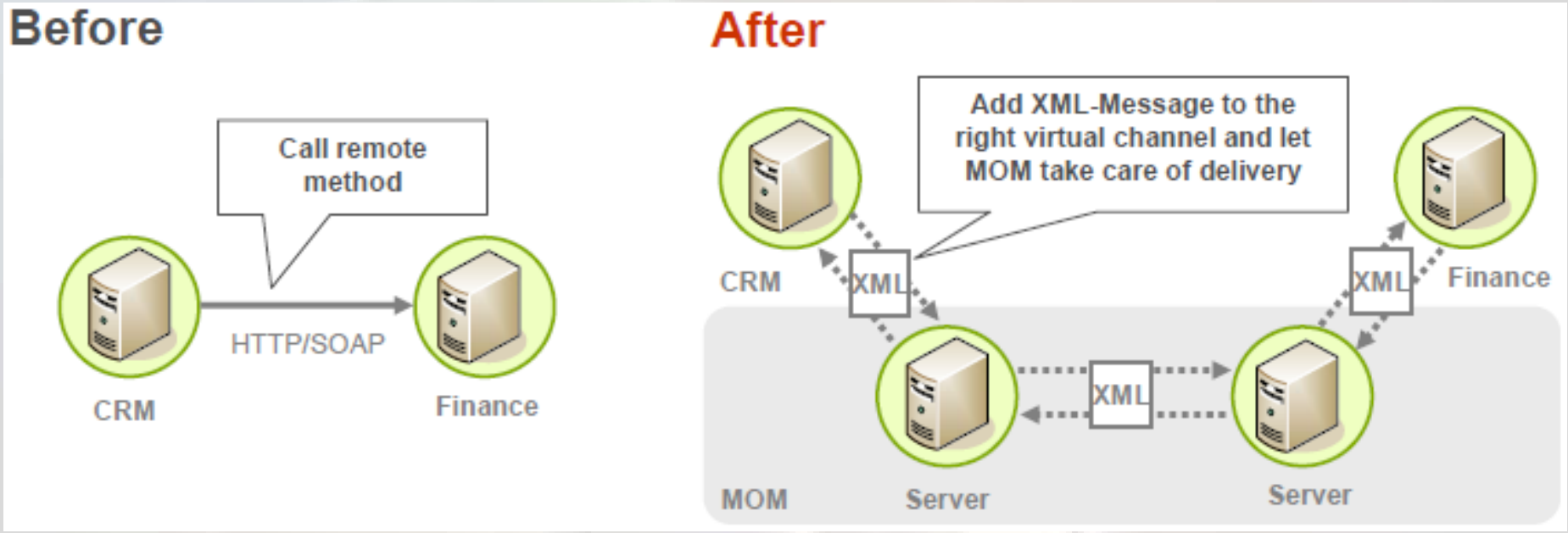


Source, [1]

Tightly Coupled Versus Loosely Coupled Interfaces

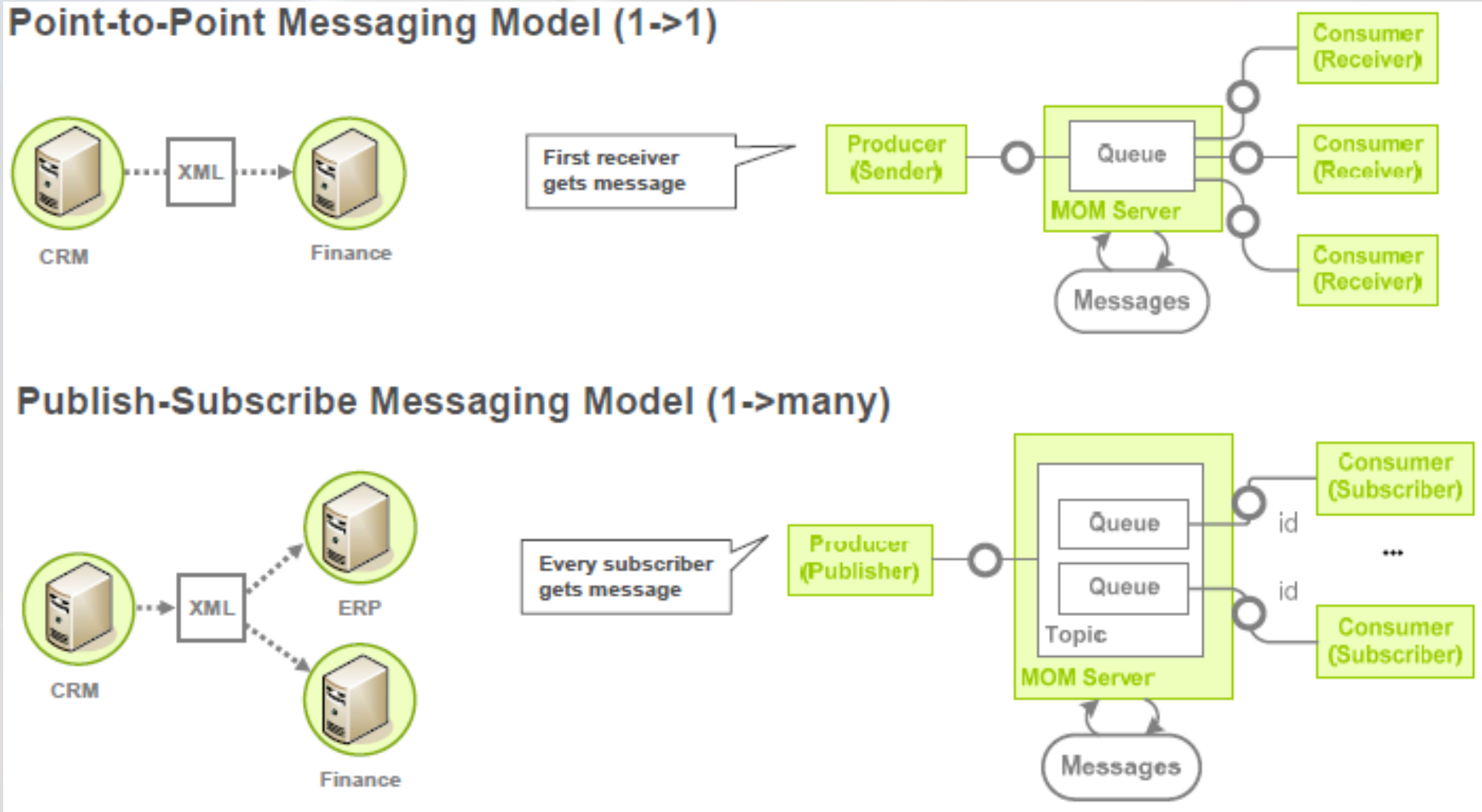
- Synchronous RPC-Style Programming [1]:
  - Technologies that predominantly use RPC-style communication include: CORBA, RMI, DCOM, Sun-RPC, Java API for XML-RPC (JAX-RPC)
  - Synchronous RPC-based interactions consist of multiple point-to-point integrations that depend on each other.
  - Minimum number of application interfaces is n(n-1)/2
- Asynchronous message-based interfaces are much more loosely coupled and manageable

Where MOM fits?



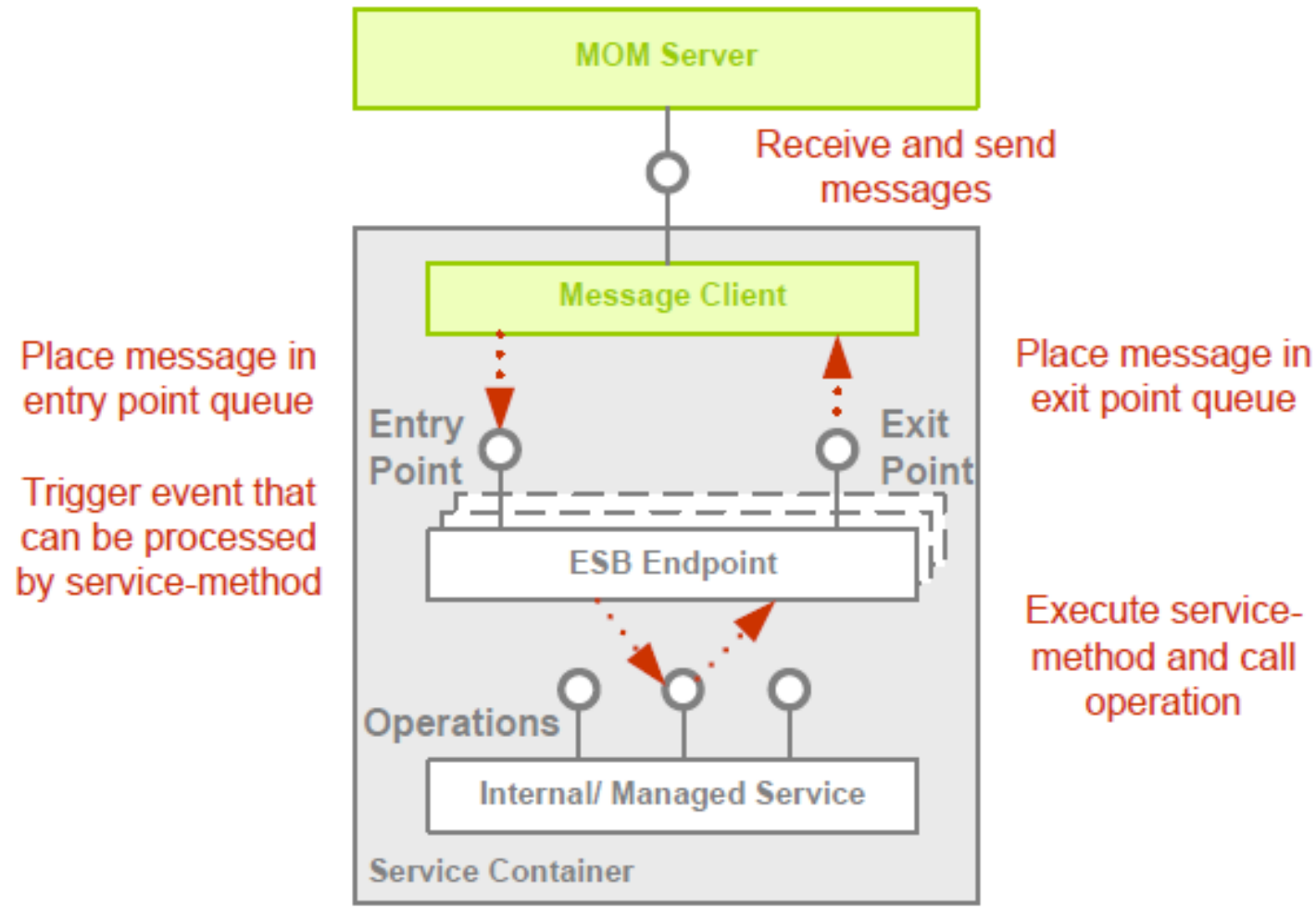
The shift from synchronous remote calls to asynchronous message exchange, Source [5]

Asynchronous Messaging Models



Point-to-point and publish-subscribe messaging models, Source [5].

Service Container



Connecting services to the ESB using ESB endpoints that are managed by a service container, Source [5]

Management Facility

- ESB centrally manages its decentralized infrastructure
- Each ESB, therefore, has a powerful and versatile management facility, which consists of:
  - A central repository
  - A network of management servers
  - Management interfaces at message servers and service containers
  - Different configuration, management and monitoring tools.

Session Outlines

- Introduction
  - ESB functions
- ESB role
- ESB Components
- **Service invocation**
- ESB Routing Facilities
- ESB example (Biztalk)

Service invocation model

- A typical SOA uses the find/bind/invoke model
- This model assumes that there is a registry or a directory that stores the location and metadata about a service implementation
  - In the "*find*" operation, the service client does a **lookup** in the registry for the service, using certain criteria, like names.
  - The "*bind*" operation, for a web services request could simply mean doing an HTTP **connect**.
  - The "*invoke*" operation, means **sending a message** or **invoking a remote procedure**.

ESB Service Invocation

- ESB has the concept of a **registry or directory** service in which information about service **endpoints** is stored
- An inherent find/bind/invoke operation occurs as part of the ESB mechanics, but it is separated out from the business logic.
- In an asynchronous ESB environment, the set of find/bind/invoke operations may actually map to sending an XML message to a queue and processing the reply.
- The means by which the find/bind/invoke operations are defined, is through **configuration and deployment tools**.

Steps of service invocation

1. XML messages are received by the service from an **entry endpoint** that is managed by the **service container**.
2. Upon conclusion of its task, the service implementation places its **output message** in the **exit endpoint**.
3. The output of the service is the reply, which the **ESB routes** to the next step or back to the request.

Session Outlines

- Introduction
  - ESB functions
- ESB role
- ESB Components
- Service invocation
- **ESB Routing Facilities**
- ESB example (Biztalk)

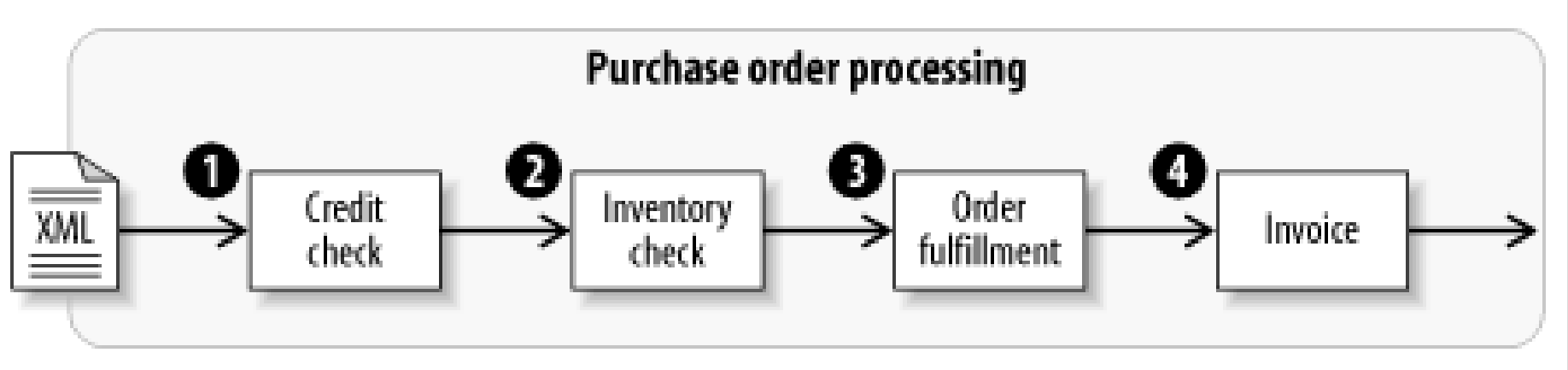
ESB Routing mechanisms

- ESB basically provides 3 routing mechanisms thereby invoking multiple business services:
  - Itinerary-based routing.
  - Service orchestration using BPEL.
  - Content-based routing.

Itinerary-Based Routing

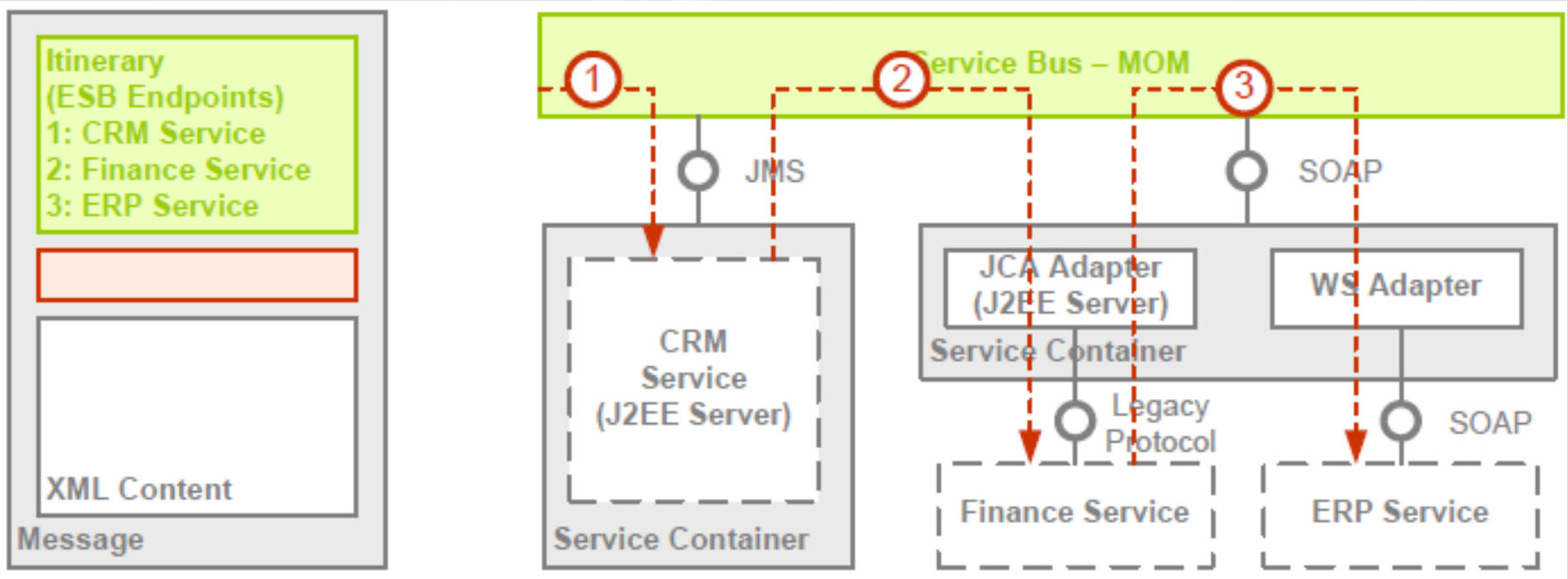
- Message itineraries (*routes*) are key to enabling a highly distributed SOA across an ESB.
- In order to route a message through the bus, each message contains an itinerary.
- The **itinerary** consists of a **list of ESB endpoints** that have to be visited and the information about **already visited ESB endpoints**.
- The **message payload** contains the **current processing state.**

An ESB itinerary represents a distributed business process



Source, [1]

Itinerary-based routing in an ESB

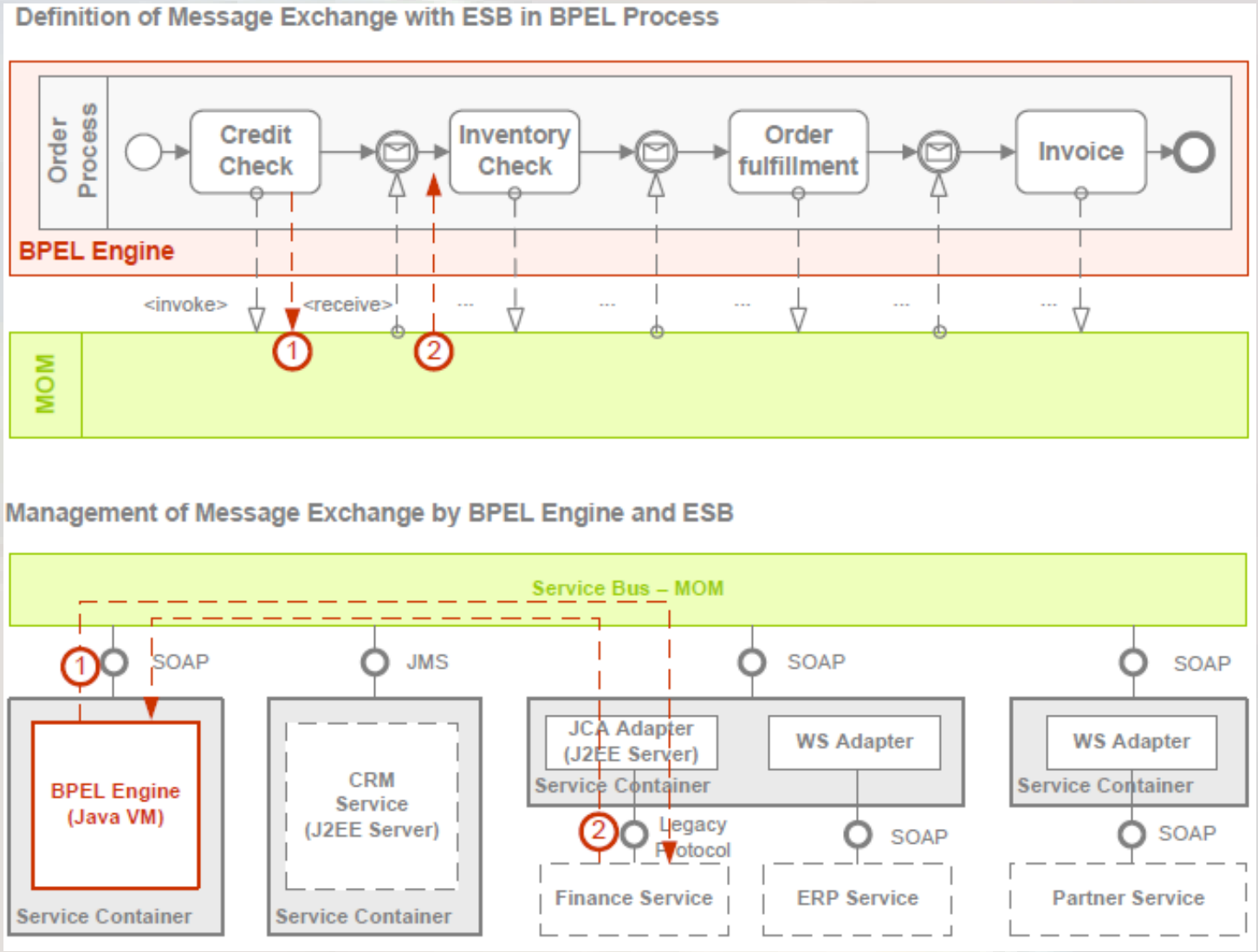


Source, [5]

Service Orchestration using BPEL

- Used to manage **long-running business processes** that might run for **months or years**.
- A BPEL process definition consists of a number of logical steps that are connected to each other by conditional or unconditional links and can be executed in sequence or in parallel.
- A BPEL process definition also allows to define time-based, condition-based and event-based triggers.
- As in the itinerary based routing, each logical step refers to an ESB endpoint

Service orchestration using BPEL

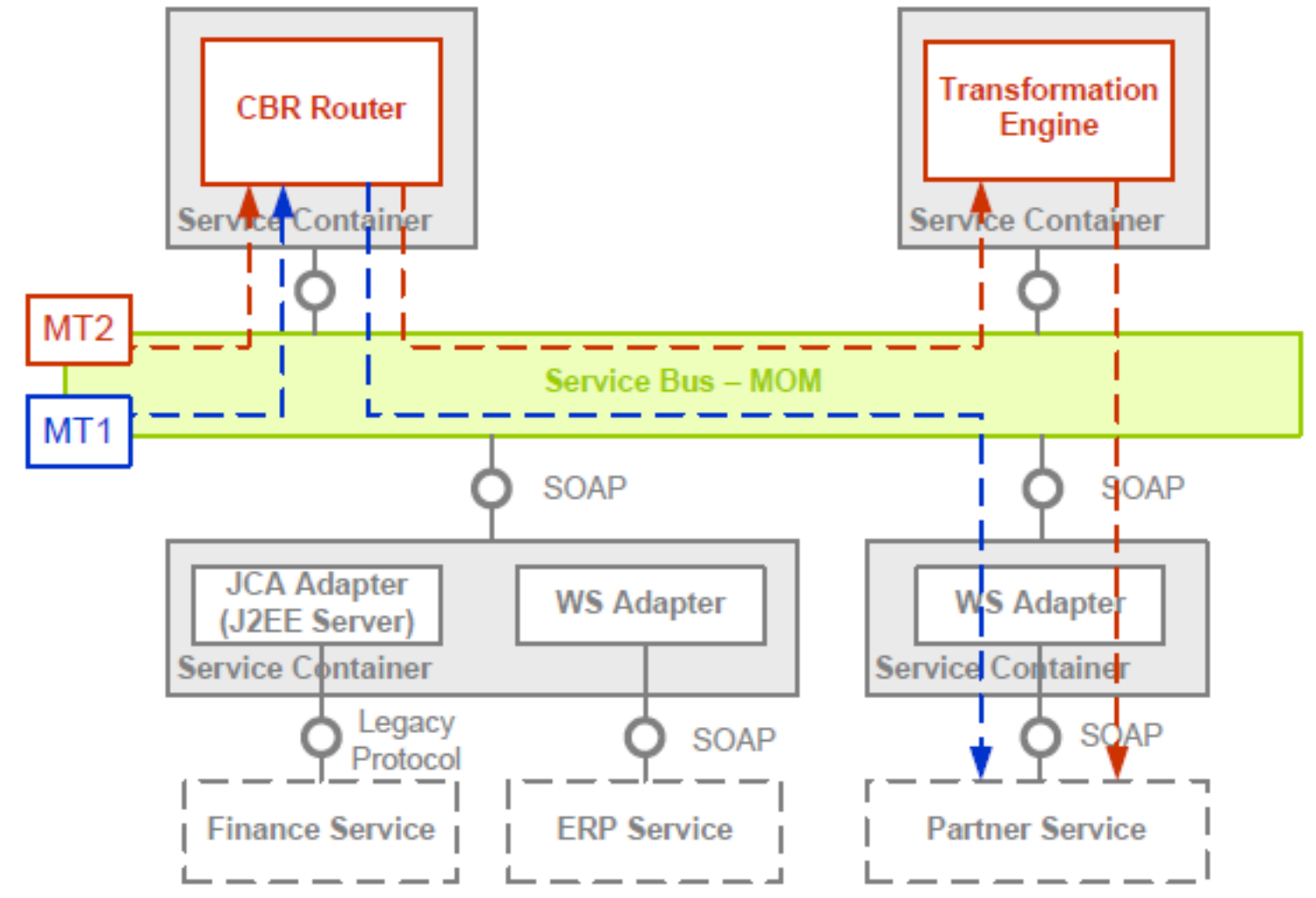


Source, [5]

Content-Based Routing (CBR)

- Based on the fact that XML processing services with different capabilities are plugged into the bus.
- They basically allow to validate, enrich, transform, route and operate XML messages.
- Combinations of these services allow to form lightweight processes with the purpose to process messages.
- Such a lightweight process is plugged as CBR service into the message flow between a message producer and a message consumer

Content-based routing in an ESB



Source, [5]

Session Outlines

- Introduction
  - ESB functions
- ESB role
- ESB Components
- Service invocation
- ESB Routing Facilities
- **ESB example (Biztalk)**

Microsoft BizTalk Server

- Referred to as "BizTalk“
- An Enterprise Service Bus (ESB) created by Microsoft.
- It enables companies to automate business processes
  - Through the use of "*adapters*" which are tailored to communicate with different software systems used in an enterprise.
- Provides the following functions:
  - Enterprise application integration.
  - Business process automation.
  - Business-To-Business communication.
  - Message broker.
  - Business activity monitoring.

Biztalk main competitors

- Commercial:
  - WebSphere by IBM
  - webMethods by Software AG
  - Oracle Enterprise Service Bus (BEA Logic)
- Open source:
  - Apache ServiceMix
  - Apache Synapse
  - JBoss ESB
  - NetKernel

Architecture

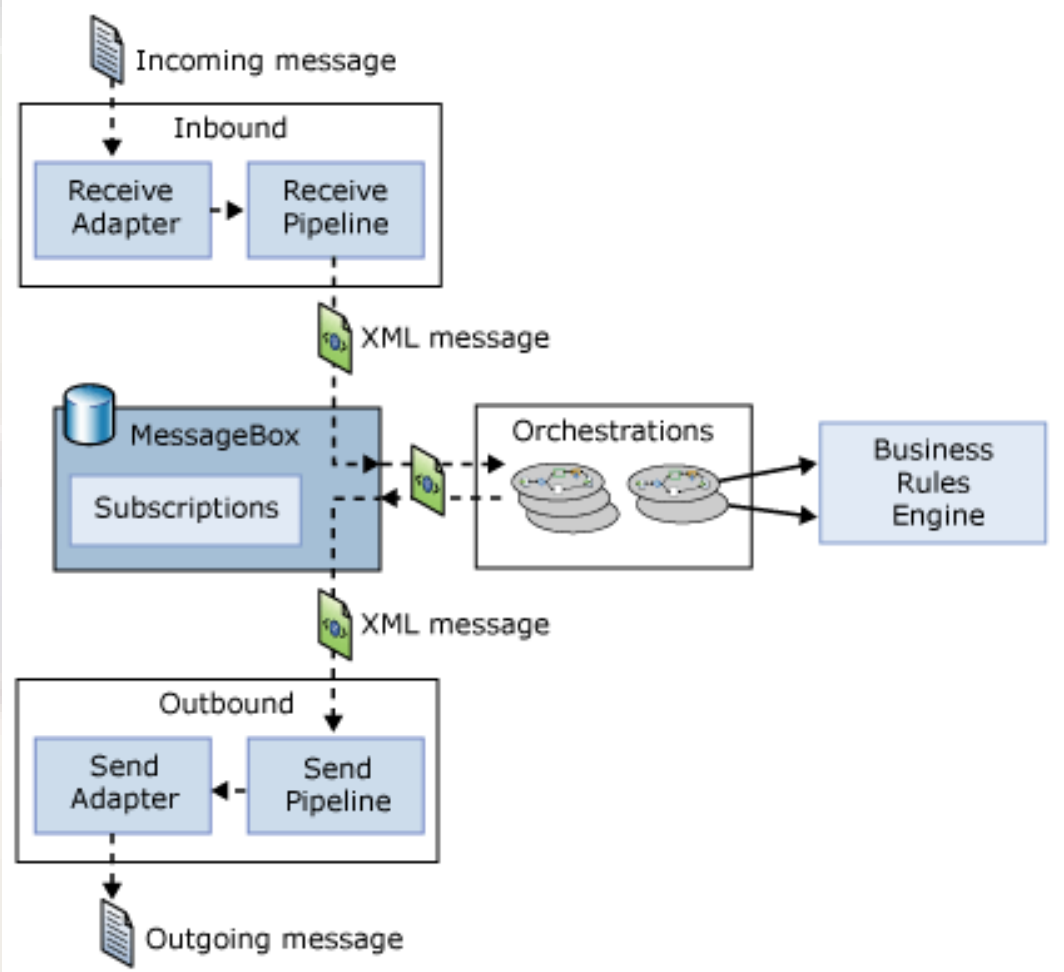
- The BizTalk Server runtime is built on a *publish/subscribe* architecture, sometimes called "content-based publish/subscribe".
  - Messages are published into BizTalk, transformed to the desired format, and then routed to one or more subscribers.
- BizTalk makes processing safe by serialization (called dehydration in Biztalk's terminology)
  - Placing messages into a database while waiting for external events, thus preventing data loss.
- This architecture binds BizTalk with Microsoft SQL Server.
- Processing flow can be tracked by administrators using an Administration Console

BizTalk Server Messaging engine

- Enables users to create business processes that spans multiple applications by providing two primary things:
  - A way to specify and implement the logic driving that business process
  - A mechanism for communicating across the applications that the business process uses

BizTalk Server engine

- **Send** and **receive adapters**, use the appropriate mechanism to communicate with receiver and sender applications respectively.
- **Receive** and **send pipelines**, convert between XML format used by BizTalk Server and the format required the external application, validate/add a digital signature, etc…
- Business process logic is implemented as one or more **orchestrations**, each of which consists of executable code.
- **MessageBox**, a database where the message is delivered into and is implemented using Microsoft SQL Server.



Different people perform different functions using the BizTalk Server engine

- A BizTalk application wraps orchestrations, pipelines, message schemas, etc. into a single logical unit
  - Which is an abstraction for management and deployment
- Three roles are necessary to create and maintain BizTalk Server solutions, this done by:
  - A **business analyst:** to define the rules and behaviors that make up a business process.
  - A **developer** can create a BizTalk application that implements the business process.
  - An **administrator:** setting up communication among the parts, deploying the BizTalk application.

Biztalk Application Example (Practical Session)

- Before start:
  - Install BizTalk server on a single-computer environment
  - Do the basic configuration
- We will create and build the first project in the enterprise application integration (EAI) solution.
  - The project contains message schemas, and a map.
- Then, we will create the business process that routes the messages and evaluates the contents of the request message.
- Then, we will learn administering BizTalk Server artifacts by using the BizTalk Server Administration Console, as follows:
  - Deploy the project
  - Configure and start the application
  - Test the solution

Summary

During this module we have introduced the concept of enterprise service bus; we have covered the following:

1. ESB role
2. ESB components
3. Service invocation
4. ESB Routing Facilities
5. BizTalk

Next module will cover the SOA design and Integration Patterns

References

1. Dave Chappell, “Enterprise Service Bus”, O'Reilly, June 2004.
2. Eric Newcomer, Greg Lomow, “Understanding SOA with Web Services”, Addison Wesley Professional, December 2004.
3. Steve Graham, Doug Davis, Simeon Simeonov, Glen Daniels, Peter Brittenham, Yuichi Nakamura, Paul Fremantle, Dieter König and Claudia Zentner, “Building Web Services with Java, making sense of XML, SOAP, WSDL and UDDI”, Second Edition, Sams Publishing, 2005.
4. Extracted from IBM education assistant: http://publib.boulder.ibm.com/infocenter/ieduasst, 9/2011.
5. Martin Breest, “An Introduction to the Enterprise Service Bus”, Hasso-Plattner-Institute for IT Systems Engineering at the University of Potsdam, Prof.-Dr.-Helmert-Str. 2-3, D-14482 Potsdam, Germany, 2006.

Thanks

Mohammed Aldasht