On Demand Business

第6章 随需应变电子商务应用技术

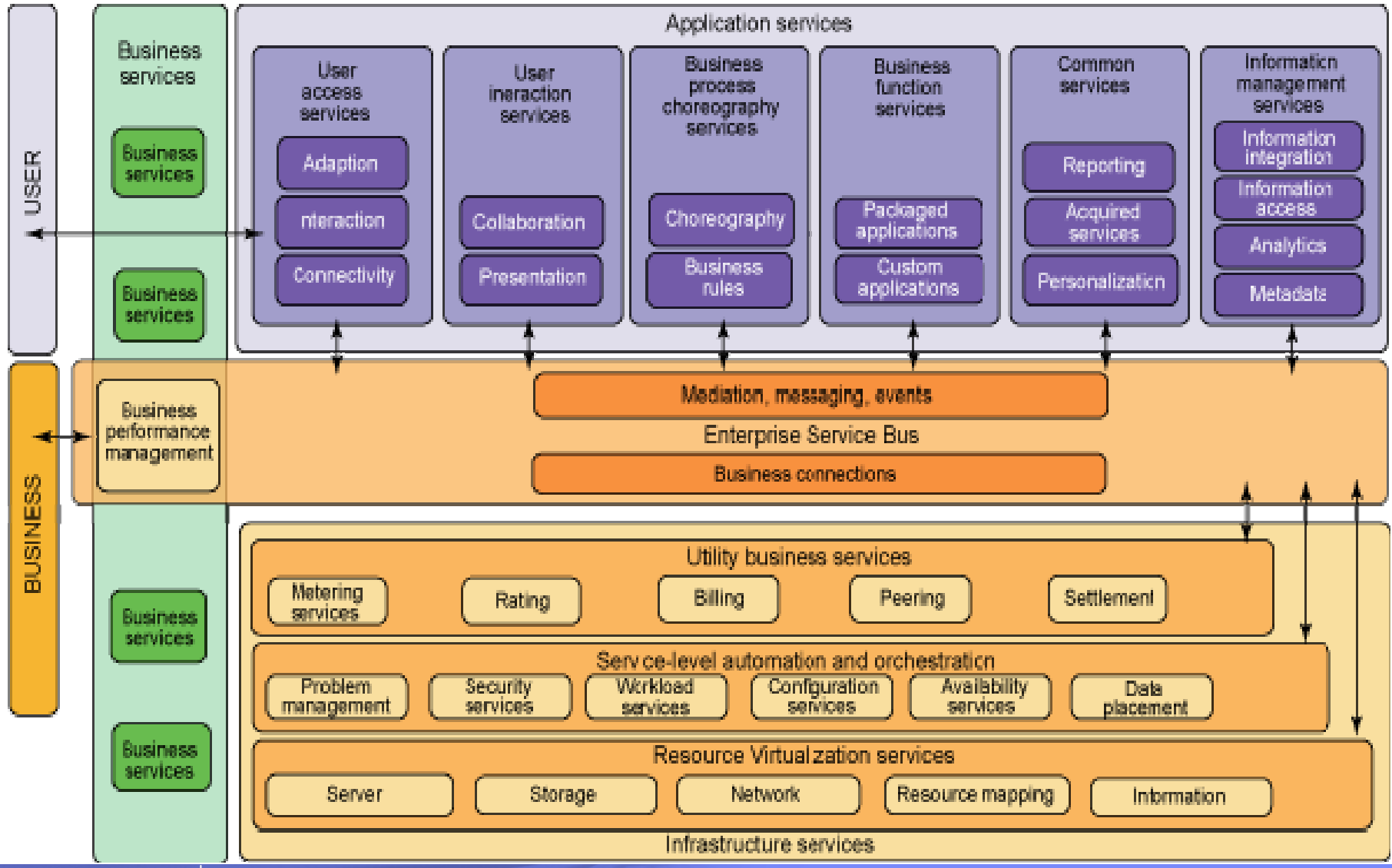
IBM大学合作部

第 6章 —随需应变电子商务应用技术

- 章节目标
  - 1. 了解随需应变运行架构的组成；
  - 2.掌握SOA的特点，了解SOA的体系架构，了解SOA与Web服务的关系；
  - 3.理解ESB的概念以及它与SOA的关系，掌握ESB实现SOA的多种模式，能根据实际情景选择合适的ESB模式；
  - 4了解工作流的概念,了解 BPR、 DesignFlow方法和BPML的功用，理解业务建模的功能和意义；
  - 5.了解RUP以及MDA的功能；
  - 6.理解设计模式的概念，能根据实际场景，选择合适的设计模式。
  - 7.了解SAN的应用领域和功能。

本章目录

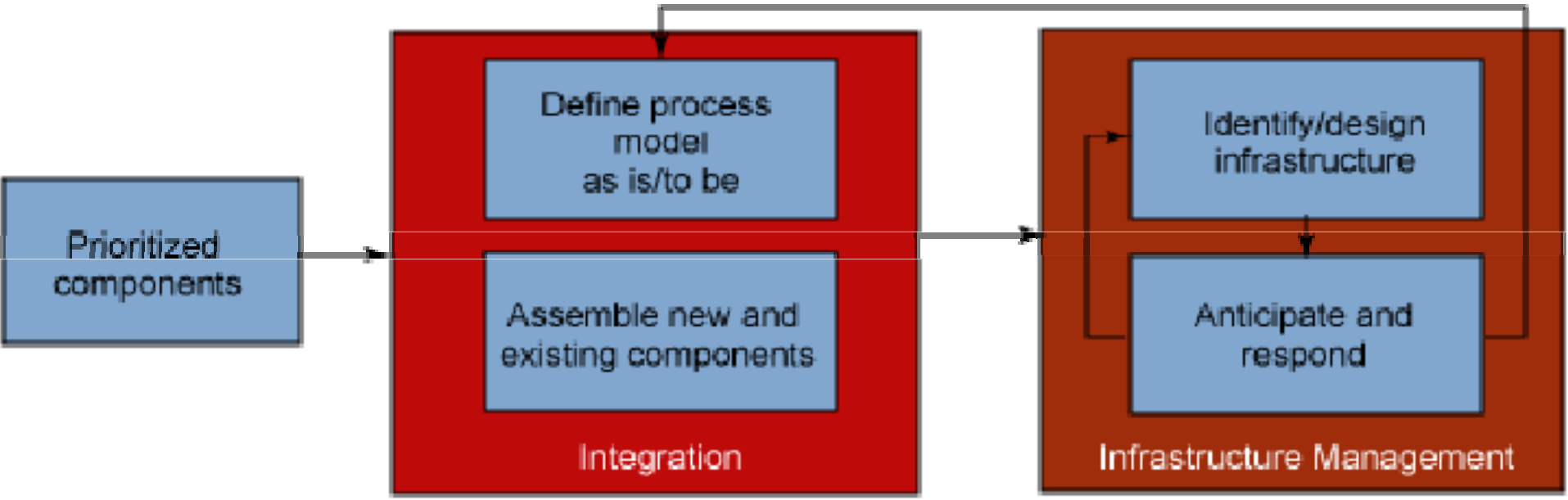
6.1 随需应变运行环境架构



6.1.1随需应变运行环境架构提供的服务

- 集成服务
- 基础设施服务
- 业务性能管理

6.1.2随需应变服务的生存周期



- Deconstruct business model
- Assess priority components

- Model existing and new process and business
- Develop the business case
- Think of software development as a business process
- Create necessary components (new and legacy-based)
- Customize and assemble components

- Underlying infrastructure is modular, built for change and standards-based
- Monitor business and IT status
- Act on autonomic policy
- Choose from on and off premise execution

Open Standards

Service Oriented Architecture

End-to-End Tools Support

6.1.3随需应变运行环境架构的主要元素

- 应用程序服务杂货店
- 企业服务总线金融服务
- 基础设施服务
- 用户

6.2 SOA

6.2.1企业为什么选择SOA

- 选择 SOA 的时机
- 将业务与 IT 结合起来
- 一个重要的机制：服务
- 面向服务的架构SOA、Web 服务与优势保持
- 模型驱动的开发和虚拟企业
- 面向服务的架构SOA与控制问题
- 考虑SOA的理由
- 面向服务的架构SOA与业务
- 面向服务的架构SOA的发展前景
- SOA与信息数据
- SOA适应的场合

6.2.2SOA体系架构

- 6.2.2.1 什么是SOA
- 6.2.2.2 什么是服务？
- 6.2.2.3 松耦合
- 6.2.2.4 明确定义的接口
- 6.2.2.5 无状态的服务设计
- 6.2.2.6 服务粒度
- 6.2.2.7 服务质量需要考虑的问题

6.2.2.1 什么是SOA

- 面向服务的体系结构（service-oriented architecture，SOA）是一个组件模型
- 它将应用程序的不同功能单元（称为服务）通过这些服务之间定义良好的接口和契约联系起来。
- 接口独立于实现服务的硬件平台、操作系统和编程语言。
- 异构系统可以以一种统一和通用的方式进行交互。

6.2.2.2 什么是服务？

- 服务（service）是封装成用于业务流程的可重用组件的应用程序函数。
- 通过定义的通信协议，可以调用服务来强调互操作性和位置透明性。
- Web 服务是比较常用的一种服务方式。

6.2.2.3 松耦合

- 服务请求者到服务提供者的绑定与服务之间应该是松耦合的。
- 服务请求者不知道提供者实现的技术细节。
- 松耦合使会话一端的软件可以在不影响另一端的情况下发生改变

6.2.2.4 明确定义的接口

- 服务交互必须是明确定义的。
- Web 服务描述语言（Web services Description Language，WSDL）是受到广泛支持。
- WSDL用于描述服务请求者所要求的绑定到服务提供者的细节。
- WSDL 的通用定义允许开发工具创建各种各样类型的交互的通过接口，同时隐藏细节 。

6.2.2.5 无状态的服务设计

- 服务应该是独立的、自包含的请求
- 服务不应该依赖于其他服务的上下文和状态。

6.2.2.6 服务粒度

- 粗粒度的接口用于外部集成的最佳实践
- 细粒度的接口多用于企业内部

6.2.3SOA与Web服务

- 服务是SOA的立足点。
- Web 服务就是其中一种服务方法。
- Web 服务使异构的计算机系统能够有效地互操作
- 许多公司正在使用 Web 服务。

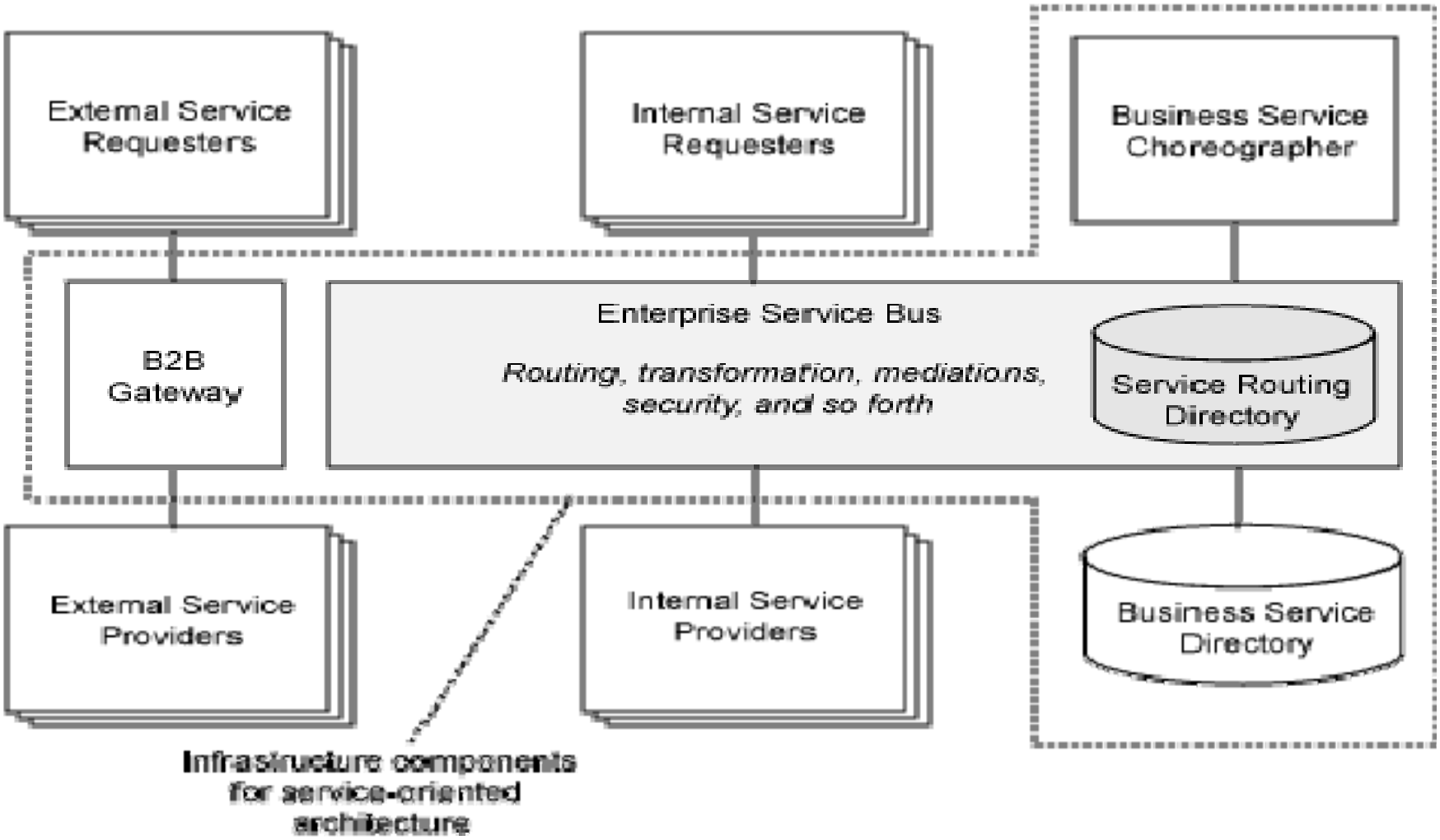
6.3 ESB

- 6.3.1 企业服务总线ESB
- 6.3.2 ESB 在 SOA 内的工作角色
- 6.3.3 企业内部SOA实现的模式
- 6.3.4 扩展企业SOA模式
- 6.3.5 一个特定的场景和对应的ESB解决方案
- 6.3.6 ESB案例分析

6.3.1企业服务总线ESB

- ESB是一种中介。
- ESB是由中间件技术实现并支持 SOA 的一组基础架构功能。
- ESB 支持异构环境中的服务、消息，以及基于事件的交互，
- ESB具有适当的服务级别和可管理性。

6.3.2ESB 在 SOA 内的工作角色



6.3.3企业内部SOA实现的模式

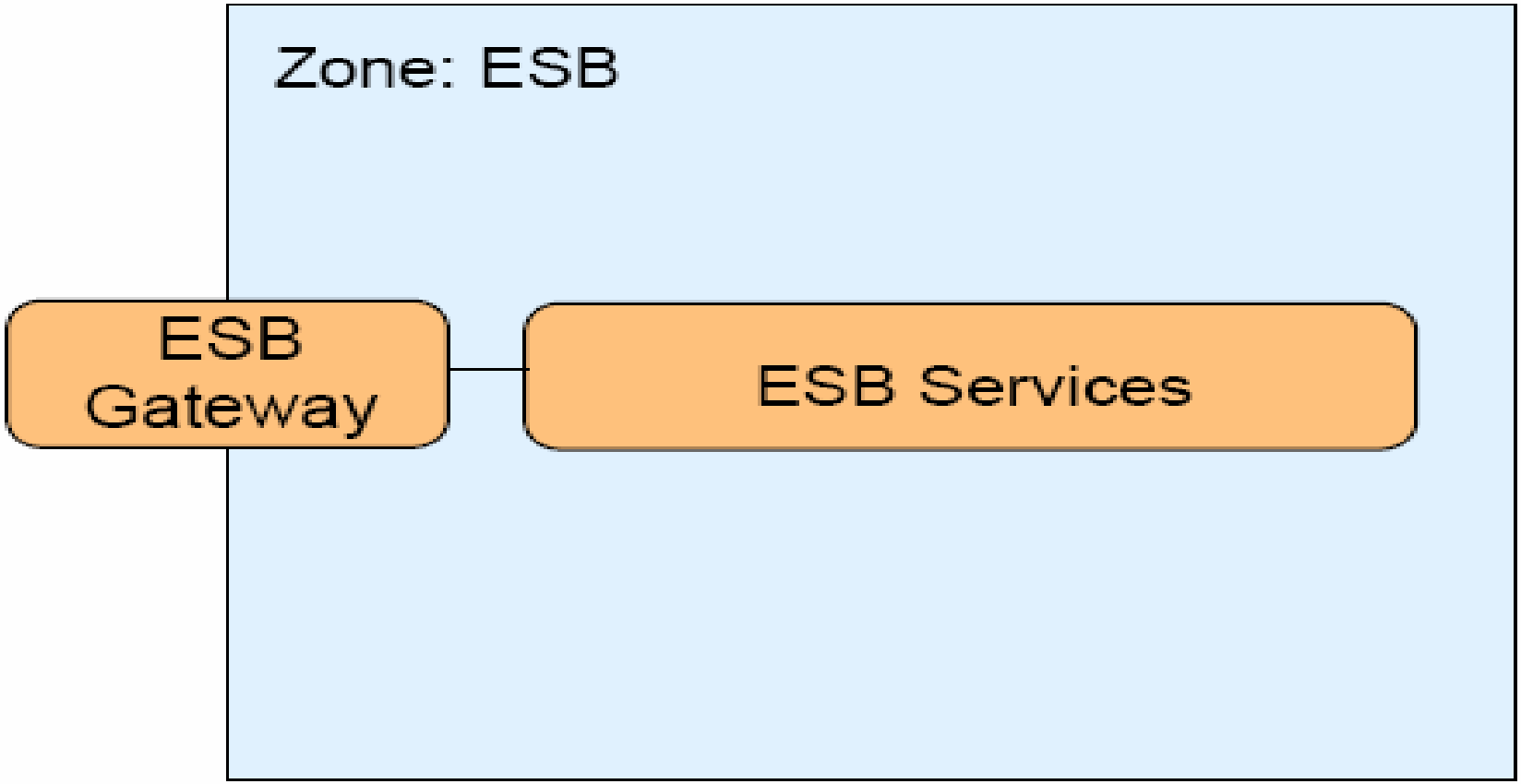
- 6.3.3.1 企业服务总线模式
- 6.3.3.2 企业服务总线ESB网关模式

6.3.3.1 企业服务总线模式

- Enterprise Service Bus pattern
- 提供一个遵循SOA法则的，健壮的，可管理的，分布式的整合框架。
- 使得各独立接口间能够通过服务的方式进行交互。
- 支持SOA，提供整合的框架。
- 支持服务路由（service routing）和替换，协议转换以及别的消息处理功能。
- 支持Web服务和传统的EAI通讯标准和技术

6.3.3.2 企业服务总线ESB网关模式

- ESB gateway pattern

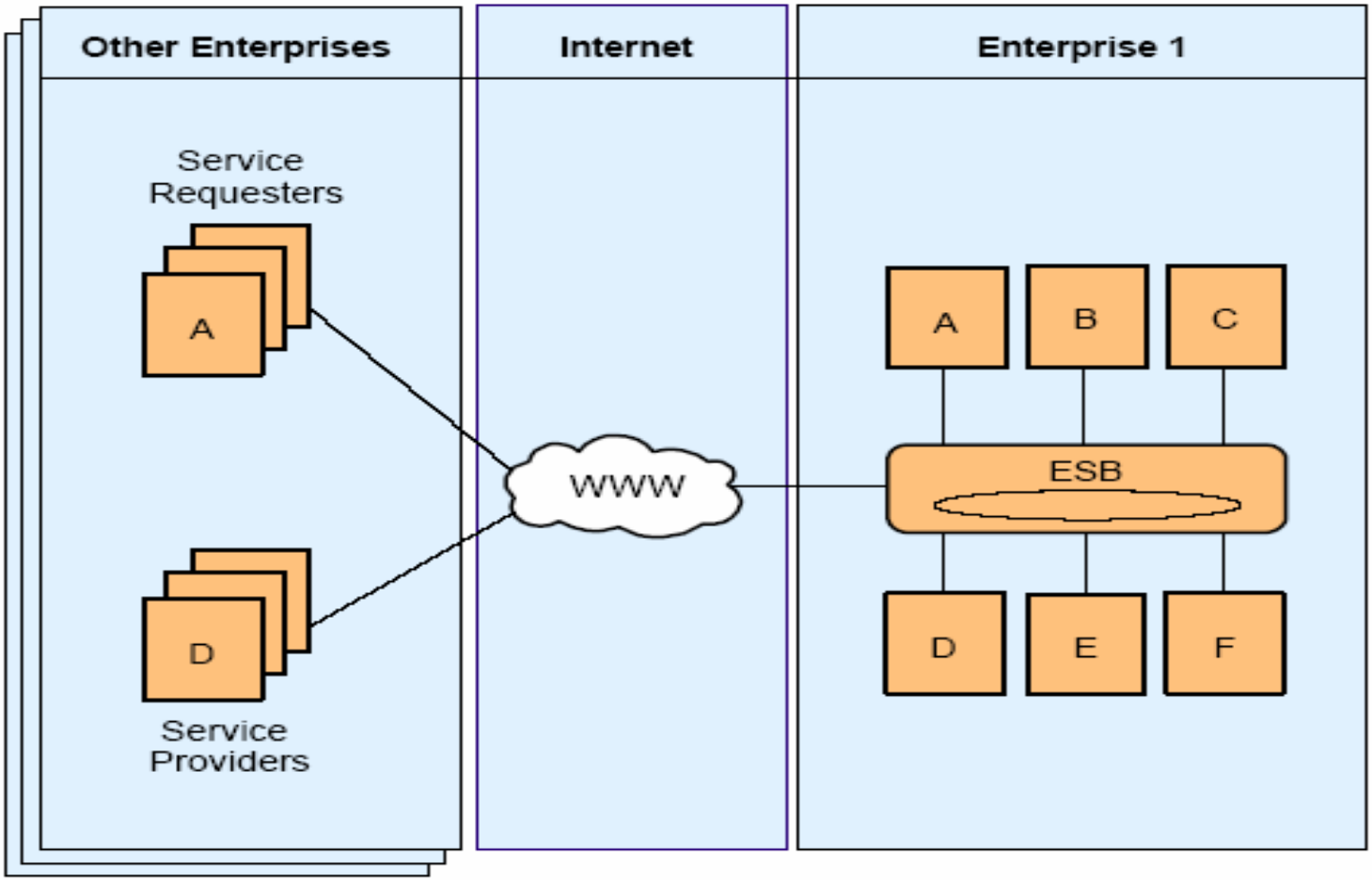


6.3.4扩展企业SOA模式

- 6.3.4.1 ESB外联模式
- 6.3.4.2 ESB网关外联复合模式

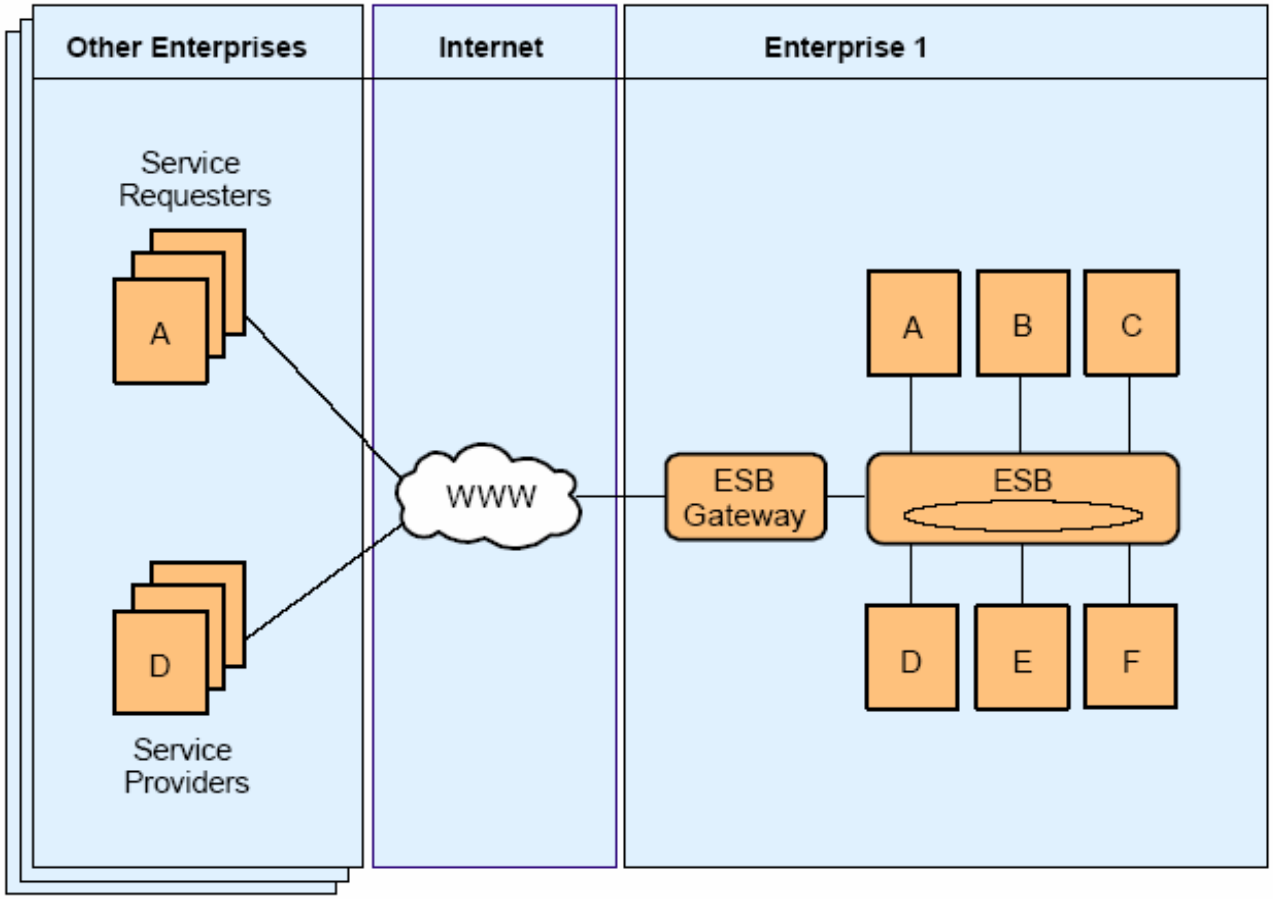
6.3.4.1 ESB外联模式

- Exposed ESB pattern



6.3.4.2 ESB网关外联复合模式

- Exposed ESB Gateway composite pattern



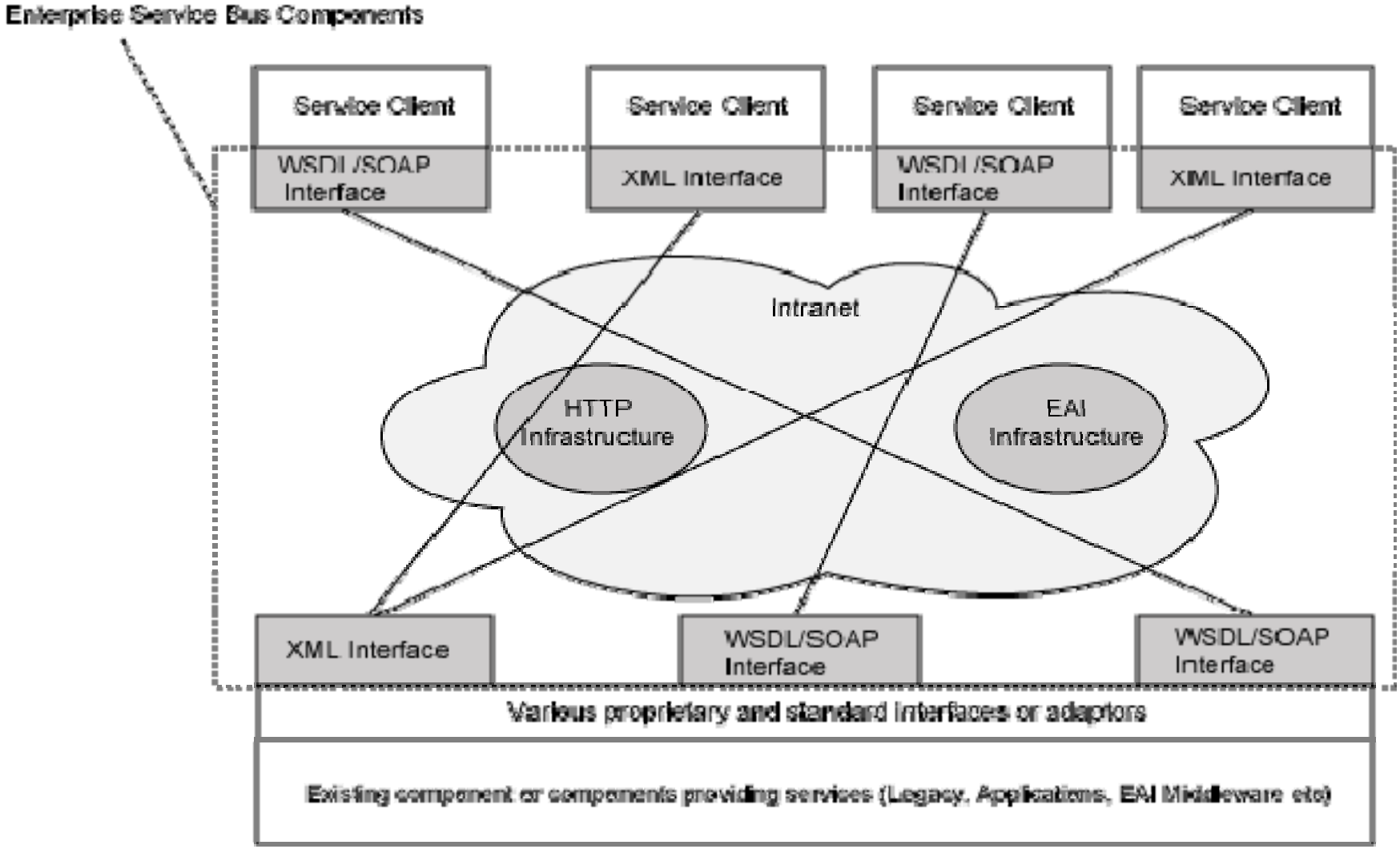
6.3.5一个特定的场景和对应的ESB解决方案

- 6.3.5.1 驱动体系结构的 ESB 场景和问题
- 6.3.5.2 基本适配器解决方案模式
- 6.3.5.3 服务网关解决方案模式

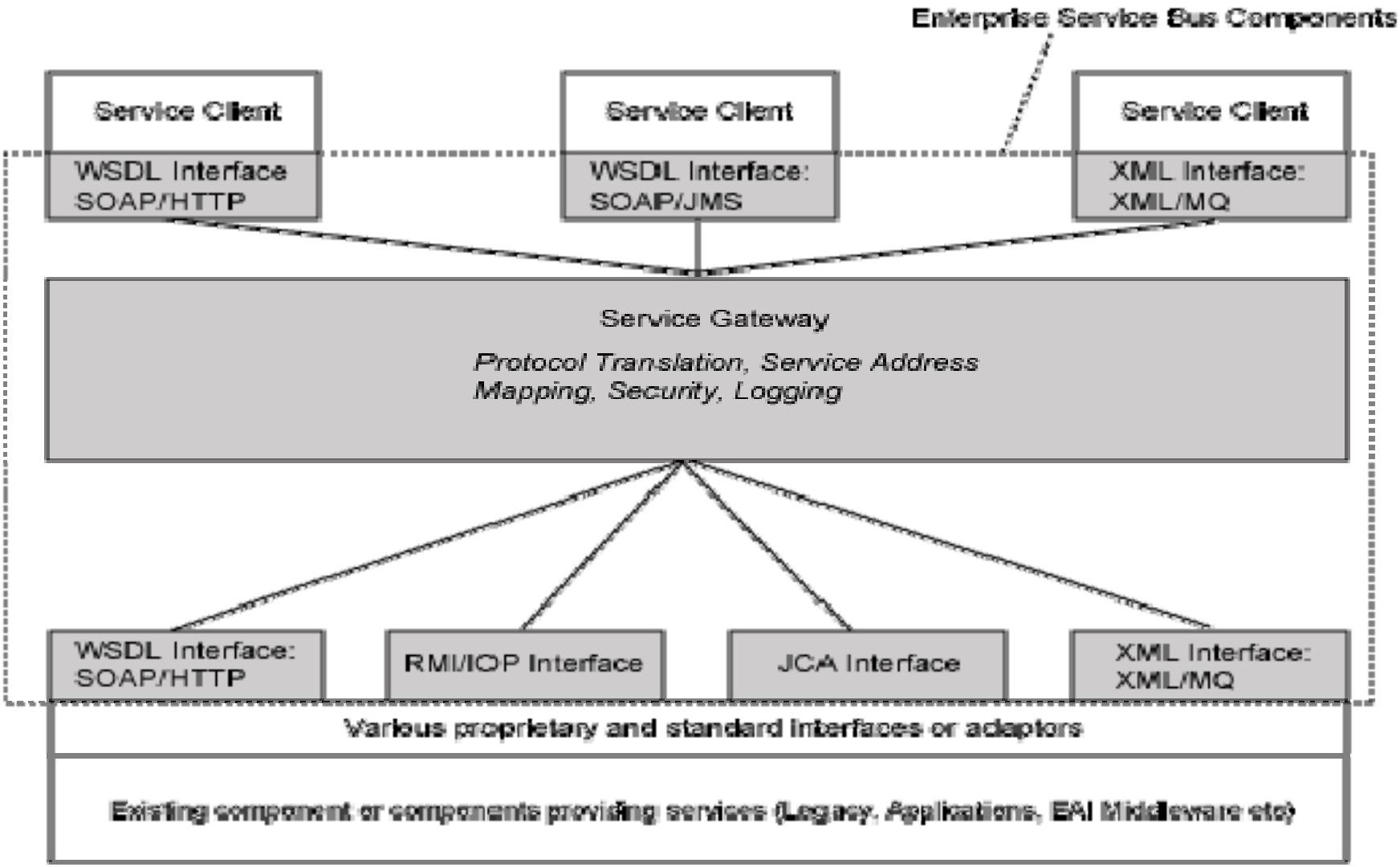
6.3.5.1 驱动体系结构的 ESB 场景和问题

- 两个系统的基本集成的场景
- 企业需要提供用不同的技术（如 J2EE、.NET、CICS 等等）实现的两个系统之间的集成。
- Web 服务 SOAP 标准或消息传递中间件可能是候选的集成技术。
- 是否一开始就选择可扩展解决方案？

6.3.5.2 基本适配器解决方案模式



6.3.5.3 服务网关解决方案模式



6.3.6ESB案例分析（一）

- 案例陈述——施乐公司，用于新的或正在更新的业务应用的定制编码不仅降低了生产效率，还增加了成本
- 为什么选择IBM？——IBM，SOA
- 优势——在24个月内实现100%的投资回报，每年节省720,000美元的部署成本，开发和执行新应用的时间仅为过去的1/4。

6.3.6ESB案例分析（二）

主要软件组件

- IBM WebSphere Message Broker（原名：IBM WebSphere Business Integration Message Broker）
- IBM WebSphere MQ
- IBM WebSphere Application Server Network Deployment
- IBM WebSphere Studio Application Developer Integration Edition

合作伙伴：Software Spectrum

6.4 业务流程设计优化与业务建模

- 6.4.1 工作流及工作流管理
- 6.4.2 业务流程重组BPR
- 6.4.3 DesignFlow方法
- 6.4.4 业务流程建模语言BPML
- 6.4.5 业务建模
- 6.4.6 RUP
- 6.4.7 模型驱动架构MDA

6.4.1工作流及工作流管理

- WfMC 的义，工作流（Work Flow）是自动运作的业务过程部分或整体，表现为参与者对文件、信息或任务按照规程采取行动，并令其在参与者之间传递。
- 工作流就是一系列相互衔接、自动进行的业务活动或任务。
- MQSeries Workflow的流程管理用于管理人、数据、应用和商务过程。
- 流程：公司内部和客户，供应商，商业合作伙伴

6.4.2业务流程重组BPR

- BPR（Business Process Re-engineering）
- BPR创始人，Michael Hammer教授——“追求业务流程变革的根本性和彻底性，希望取得成本、质量、服务和速度方面的显著性改善”。
- Re-position、Re-organization、Re-system、Re-vitalizing
- P(Process)——从管理到销售、采购到财务、生产各层次
- BPR的基本内涵：是以流程运作为中心

6.4.3DesignFlow方法

- DesignFlow——设计优化业务流程
- DesignFlow：表述业务流程，并对业务流程进行改进，以优化流程中不合理环节，实现企业希望的“to-be”业务流程。
- IBM公司——IBM DesignFlow

6.4.4业务流程建模语言BPML

- BPML（Business Process Modeling Language System, 业务流程建模语言）
- 一种开放的XML语言规范
- 定义了一种形式化模型，用于表示抽象、可执行的业务流程。

6.4.5业务建模

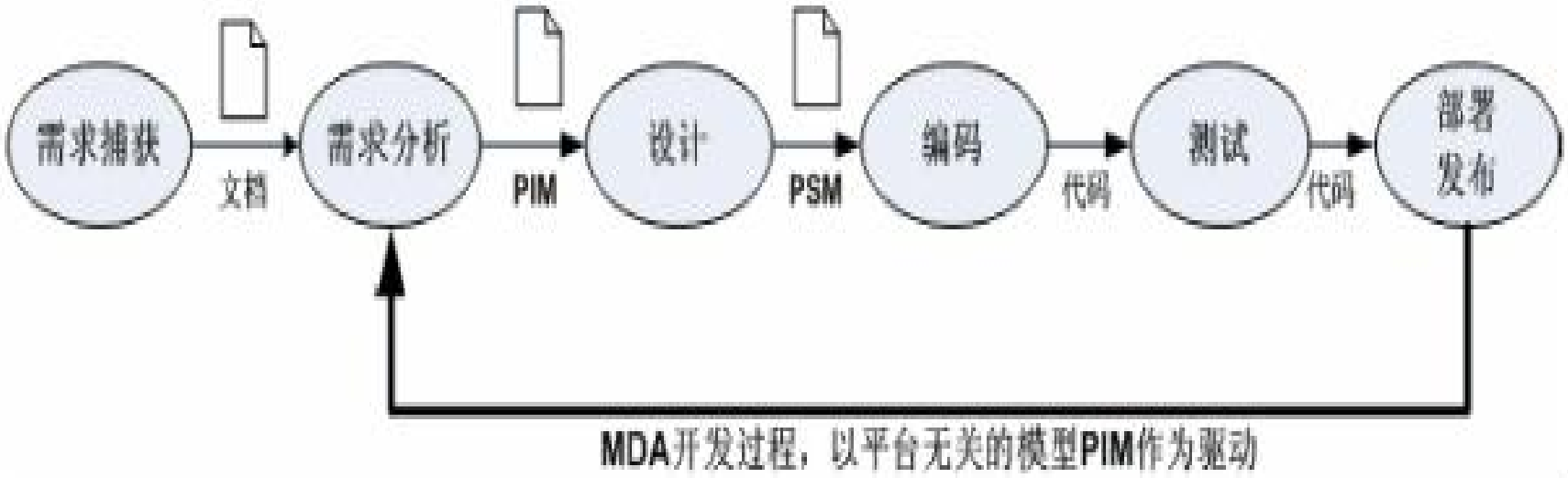
- 包括了对业务流程建模，对业务组织建模，改进业务流程，领域建模等方面。
- 通过业务建模，可以部署、管理、分析现有业务流程，流水线化业务流程，并确认新的商业机会。
- 如何使企业内的多个应用系统共同运作？
- 从业务建模入手

6.4.6RUP

- Rational Unified Process
- 是软件工程的过程。
- 它提供了在开发组织中分派任务和责任的纪律化方法。
- 对开发过程提供自动化工具支持。
- 迭代

6.4.7模型驱动架构MDA

- MDA(Model Driven Architecture) 模型驱动架构
- PIM（Platform Independent Model ）平台无关模型
- PSM（Platform specific Model ）平台相关模型



6.5 设计模式（Design Patten）

- 6.5.1 设计模式（Design Patten）概述
- 6.5.2 MVC设计模式
- 6.5.3 设计模式之Factory
- 6.5.4 设计模式之Singleton(单态)
- 6.5.5 设计模式之Template
- 6.5.6 设计模式之Mediator(中介者)
- 6.5.7 设计模式之State

6.5.1设计模式（Design Patten）概述

- 设计模式是一套被反复使用、多数人知晓的、经过分类编目的、代码设计经验的总结。
- 使用设计模式是为了可重用代码、让代码更容易被他人理解、保证代码可靠性。
- J2EE应用时，常用到设计模式。

6.5.2MVC设计模式

- MVC（Model-View-Controller）模式，是指模型（Model）、视图（View）和控制（Control）相分离的设计方案。
- 模型（Model）只有纯粹的功能性的接口，是一系列的公开方法。
- 视图决定模型以什么样的方式显示给用户
- 控制是和视图联合使用的。用户在与视图发生交互的时候，是通过控制器来操纵模型，从而向模型传递数据、更新模型的状态。

6.5.3设计模式之Factory

- 工厂模式（Factory）定义用于提供创建对象的接口
- 工厂模式就相当于创建实例对象的new

6.5.4设计模式之Singleton(单态)

- Singleton(单态)是一种创建模式。
- 保证在Java应用程序中，一个类Class只有一个实例存在
- 建立目录、数据库连接等

6.5.5设计模式之Template

- Template是一种行为模式。
- 它用于定义一个操作中算法的骨架,将一些步骤的执行延迟到其子类中。

6.5.6设计模式之Mediator(中介者)

- Mediator是一种行为模式。
- 它使用一个中介对象来封装一系列关于对象交互行为。
- MVC是J2EE的一个基本模式, Controller就是一种Mediator,它是Jsp和服务器上应用程序间的Mediator。

6.5.7设计模式之State

- State是一种行为模式。
- 当有不同的状态对应不同的行为，或者每个状态有着相应的行为时，使用State模式。
- 政府OA，网络游戏。

6.6 SAN(Storage Area Network) 技术