摘　要

我国物流业目前急需解决的问题，是如何实现物流园区内各个相对独立的业务系统的无缝整合，避免重复建设，消除“信息孤岛”，发挥物流园区产业集约化经营优势。论文依托哈尔滨公路货运主枢纽信息系统建设项目一期工程—“黑龙江龙运物流园区业务系统整合平台”，针对现有业务系统功能落后、应用分割的现状，展开对物流园区业务系统整合平台体系架构的研究。

论文引入 SOA 设计思想，结合 SOA 和 Web Services 技术现状，对业务系统整合模式进行了研究，并选用企业服务总线 ESB 作为平台核心部件，设计了物流园区业务系统整合平台三层体系架构。平台使用了 Web 服务作为逻辑服务基本单元，以 XML 文档作为统一数据格式，保证了异构系统间的松耦合度和转换方式的通用性，解决了数据可靠传输、异构数据转换、Web 服务封装、服务注册、服务传输等难点问题。

论文选用 IBM Message Broker 和 Message Queue 搭建了平台的基础架构，使用 Xfire 开源框架，实现了 Web 服务封装、服务注册，完成了数据在平台中传输以及转换。

论文的研究成果为各地物流园区业务系统整合提供了一个完整的平台体系架构及关键技术的实现方法，对 SOA 的推广应用和促进物流信息化发展具有一定的现实意义。

关键词：物流管理、SOA、业务系统整合、ESB、Web 服务封装

Abstract

The urgent problem need to be resolved in logistic business is how to realize seamless concordance and business collaboration between its' legacy business system, avoid redundant construction and demonstrate the advantage of intensive management in logistic park. The architecture of the logistics park business system integration platform is studied by the papers according to the status of the Laggard function and partitioned application of the existing business systems which is relied on the first of Harbin highway freight main hub's construction project of information system.

The business system integration model is studied by the papers combined with the technology status which is introduced by SOA (service-oriented architecture) design idea. and as the core platform components, ESB (Enterprise Service Bus) is selected by the papers. The three-tier architecture of the logistics park business system integration platform is designed. As the basic unit of logical services of the platform, the Web Services technology is used by the papers using XML documents for a unified data format, the loosely coupling degree of the erogenous system and the conversion of the generic approach are sure. The difficult problems such as data' reliable transmission, heterogeneous data exchange, Web Services' encapsulation, service registry and service transmission were resolved.

Web server encapsulation, service registry, data transfer and conversion in the platform were implemented based on the platform architecture of IBM Message Broker and Message Queue and open-source framework named Xfire in this paper.

Research results of the paper supply an integrated platform architecture and the key technique realizing the integrating logistics park operation system and have real-life significance for popularization and application of SOA and information-based development of logistics.

Key Words: logistics management; SOA; business integration; ESB; Web Services encapsulation

论文独创性声明

本人声明：本人所呈交的学位论文是在导师的指导下,独立进行研究工作所取得的成果。除论文中已经注明引用的内容外，对论文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本论文中不包含任何未加明确注明的其他个人或集体已经公开发表的成果。

本声明的法律责任由本人承担。

论文作者签名：史国栋　　2009 年 6月 2日

论文知识产权权属声明

本人在导师指导下所完成的论文及相关的职务作品，知识产权归属学校。学校享有以任何方式发表、复制、公开阅览、借阅以及申请专利等权利。本人离校后发表或使用学位论文或与该论文直接相关的学术论文或成果时，署名单位仍然为长安大学。

（保密的论文在解密后应遵守此规定）

论文作者签名：史国栋　　2009 年 6月 2日

导 师 签 名：　　2009 年 6 月 2 日

第一章 绪论

1.1 研究背景

物流产业是国民经济发展的动脉和基础产业，物流现代化能降低商品成本、提高劳动生产率，是推动国民经济增长的“第三利润源泉”，其发展程度已成为衡量一国现代化程度和综合国力的重要标志之一。近年来，物流产业已成为国内外产业发展的热点。据统计，2008 年全国社会物流总额达 89.9 万亿元，比 2000 年增长 4.2 倍，年均增长 23%；物流业实现增加值 2.0 万亿元，比 2000 年增长 1.9 倍，年均增长 14%；2008 年物流业增加值占全部服务业增加值的比重为 16.5%，占 GDP 的比重为 6.6%[1]。在全球经济危机的背景下，物流业仍然保持了稳步发展态势。

积极培育第三方物流企业和规划建设现代物流园区是推进我国现代物流发展的两大主题。作为具有综合服务功能的物流节点，近年来我国物流园区建设方兴未艾，自 2001 年以来，我国各地方政府相继推出城市物流规划，其中物流园区建设是规划中的重要内容。深圳、天津、北京、武汉、邯郸、青岛、无锡、成都、西安等地都已建好或在建物流园区。据不完统计全国建成和在建物流园区有 1000 余个[2]。在国务院 2009 年 3 月份审议通过的物流业调整和振兴规划中，更是明确把物流园区工程作为物流产业振兴所需的九大重点工程之一。

随着以信息技术为基础的电子商务的迅速发展以及电子商务向物流领域的大规模渗透，传统的物流模式正在向系统化、信息化、社会化、标准化、一体化的现代物流体系转变，信息化和标准化是推进现代物流发展的两个轮子，其中信息化是现代物流的生命线，直接代表着物流的现代化程度。物流业务系统整合平台既可以灵活地整合现有软件、硬件、数据等相关物流资源，又能为未来应用系统的开发和接入，以及构建一个“大整合、高共享、流程化”的物流信息系统提供基础设施，从而对物流信息化建设起到关键性的作用。

我国的交通、物流信息化近年来发展蓬勃，信息化进程正在加快。但是由于起步较晚，与国际先进水平相比，整体水平较低。由于系统建设往往受全套解决方案的指导思想的制约，以及网络通讯、协同作业业务规范和物流信息化政策法规等多方面缺乏统一的标准与支持，多数物流企业和部门的业务系统自主开发，缺乏统一标准和规划，最终的信息系统较为封闭；同时受垄断及行政体制条块分割制约，物流相关企业、政府部门

的数据资源不能共享，业务系统不能互联互通，形成一个个“信息孤岛”，造成了数据资源的浪费和重复建设等后果，这样的信息系统对于将信息共享视为关键的物流业来说，造成信息流不畅，已成为制约我国物流信息化建设纵深发展的“瓶颈”问题，引起了国家和物流从业人士的极大重视。因此，整合现有资源，解决由于信息系统的分散、异构所导致的信息孤岛问题，建立一个开放、标准、高效的物流业务系统整合平台，对于提高物流信息化水平、提高物流供应链效率和加快现代物流事业的飞速发展具有重要的战略意义。

与此同时，随着不断变化的社会物流需求，各物流企业需要迅速做出相应业务调整，会直接导致业务流程的不断变更和新需求的产生，传统方法开发的业务系统相对静止，难以进行改造或重组，不能适应迅速变化的业务需求。因此，有效地整合异构的物流信息系统和消除“信息孤岛”，实现与物流相关信息系统的信息互通、服务交互，提高物流运作效率和降低物流成本，已成为现代物流技术中急需解决的问题。

近些年来，在社会经济活动中当中频繁出现“服务”一词，服务是一种能满足公共需求的功能。与普通的业务流程不同，服务不针对特定的客户，而是提供一种“公共需求”的功能，即服务强调了“重用”的概念。对于用户来说，它只关心如何使用各种服务，通过服务能得到怎样的整体的实际的效果，而不需要考虑服务的实现流程和实现方法等细节。而对于服务提供者，则需要多种技术和业务保证所提供服务的有效性、真实性和实时性。自 90 年代起，随着互联网技术的兴起和应用，企业可以通过 Internet 获取服务信息甚至直接使用服务功能，大大发展并开拓了自己的业务范围并提高了业务能力，由 Internet 而减少的支出和由提供服务带来的业务收入为企业带来的丰硕的利润，因此各类传统企业纷纷开始从服务中寻找商机，并结合自身特点，开始向服务型企业转型[3]。

基于以上社会经济背景，迫切需要针对物流信息化存在的问题，结合现代物流及企业的变化趋势和发展要求，分析物流产业对信息系统的具体要求，研究适合中国国情的物流信息化关键性技术。

1.2 国内外研究与发展现状

1.2.1 国外物流信息化发展现状

物流概念最早是在美国提出的，同时美国的物流园区信息化水平高度发达，因此最能代表国外物流园区的信息化水平，目前美国物流园区的信息化现状为[4]：

- 服务是物流信息化的核心：物流信息化已成为美国工商企业降低物流成本、改进客户服务、改进客户服务、提高企业竞争力的基本手段，为客户提供的信息服务内容是信息系统建设的重要依据
- 标准是物流信息化的基础：美国行业协会信息交换接口等方面建立了一套适用的标准，使物流企业与客户、分包方、供应商更便于沟通和服务，物流软件也融入了格式、流程等方面的行业标准，为企业物流信息系统的建设创造了良好的环境。
- 应用是物流信息化的关键：将先进的信息技术有效地应用于实际的物流业务之中。公共物流信息平台的发展，为企业间的信息沟通和采用应用服务（ASP）模式降低信息化成本创造了条件。

发达国家物流已经把物流的高度信息化发展作为物流业的发展方向，在其物流信息化过程中，GPS，GIS 等现代高新技术充分得到了利用；对物流信息系统模型、物流服务安全模型和技术、物流商务模型和物流信息服务平台关键技术等方面进行了广泛、深入的理论和应用研究，并建立了相应的物流信息平台。其物流信息平台一般由信息中间商（如美国的 Capstan 公司）搭建经营，整合了采购商、供应商、物流服务商、承运人、海关、金融服务等各类物流活动各类机构，通过平台完成国际物流服务。目前世界上日本，美国和欧洲三大地区的物流信息化最为发达[5]。具有代表性的有美国货运实时信息系统 FIRST[6]（FIRST 是一个基于 Web，可以整合火车、轮船以及火车等交通工具可获取的实时到达信息的一种应用系统）、日本的物流信息服务公用系统 Logilink[7]（基于 Interent 的服务于大型物流企业的信息系统）、以及欧洲多式联运实时信息平台 INTRARTIP[8]（用于多式联运的实时信息平台）。

从美国、欧洲、日本的情况来看，国外物流企业已经形成了以系统技术为核心，以信息技术、运输技术、配送技术、装卸搬运技术、自动化仓储技术、库存控制技术、包装技术等专业技术为支撑的现代化物流装备技术格局。其物流信息系统普遍是构架于网络上的基于 Internet/Intranet 的开放系统，参与企业、部门范围广，业务集成度高，服务丰富，信息资源可以得到充分共享，安全性保密性程度都很高。

1.2.2 国内物流信息化发展现状

国内物流信息化发展大致可概括为三个阶段（或层次）：

第一阶段是基础信息化阶段，即企业不改变作业流程和决策模式，只是把原有流程

中的纸面信息转化为电子信息。这个阶段完成了信息化知识普及和培养人才的过程，但是没有改变人工决策的模式；

第二阶段开始考虑用信息化手段来改造作业流程，提高信息技术对决策过程的作用，逐步改变单一的人工决策模式，如提供自动报警系统协助决策；

第三阶段是物流信息系统走出企业，与上下游合作伙伴结成以信息系统为纽带的供应链。

横向看来，我国约有80%的企业仍然处于第一阶段，进入第二阶段的约占15%，只有少数企业开始了第三阶段的尝试和实施；而来自中国物流信息中心的调查数据也显示，我国物流企业建立信息系统和内部网络的不足50%[9]。

国内物流信息系统比较典型的有2003年建立的深圳现代物流信息公用平台、上海市的亿通网（www.easipass.com）、香港的DTTN（香港数码贸易及运输系统）[10]等。综观目前已建成的物流信息系统，基本上还处于企业级物流信息孤岛阶段，多数仍然停留在相互分割的单项应用层次，缺乏系统性整合，相互连接性不畅，信息资源分散，不能有效交换与共享，功能单一分散，物流业务整合度不高，只在一定程度上解决了物流方面的部分问题。

可见，我国物流信息化现状与国外先进水平相比还有较大差距，有必要研究建立物流业务系统整合平台相关技术，以提高我国物流信息化的整体发展水平，通过信息技术解决与物流活动（如运输、保管、包装、流通、加工等）有关的物流业务资源协同工作，整合社会资源，降低企业的市场风险，提高企业经营管理效率。

1.2.3 物流信息技术应用现状

根据物流的功能以及特点，现代物流信息技术由通信、软件和面向行业的业务管理系统三大部分组成，其中包括集成技术（业务逻辑层集成、数据集成和表现层集成等）、自动跟踪与定位类技术（GIS，GPS等）、自动识别类技术（条形码、扫描技术、射频识别技术RFID等）、企业资源信息技术（如物料需求计划、制造资源计划、企业资源计划、分销资源计划、物流资源计划等）、数据管理技术（如数据库技术、数据仓库技术等）和计算机网络技术等现代高端信息科技。在这些高端计算机技术的支撑下，形成了以移动通信、资源管理、监控调度管理、自动化仓储管理、业务管理、客户服务管理、财务处理等多种信息技术集成的一体化现代物流管理体系[12]。

目前应用十分广泛的现代物流管理系统，有制造资源计划（MRPII）、企业资源计划

（ERP）、供应商管理库存系统（VMI），供应链管理（SCM）等。据调查显示，ERP、SCM 及 VMI 等集成系统软件在我国企业中实施的尚不足 1/10[13]。这些物流系统集成软件相对功能专一，基于传统体系架构设计，在业务交互协同、业务流程管理及动态调整、快速部署等各方面都不能很好的满足现代物流发展的要求。近年来，随着可扩展标记语言技术（eXtensible Markup Language，XML）、Web 服务和业务流程管理（Business Process Management，BPM）等技术的发展和成熟，应用体系结构正在逐渐改变[14]。

面向服务体系结构（Service Oriented Architecture，SOA)是近年来在国内外研究推广迅速的一种 IT 体系结构模式，被誉为下一代 Web 服务的基础框架。SOA 是一种建立、维护、管理 IT 系统和业务流程的方法，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。接口采用中立的方式进行定义，它应该独立于实现服务的硬件平台、操作系统和编程语言。这使得构建在各异构系统中的服务可以以一种统一和通用的方式进行交互[14]。SOA 的中心思想就是实现技术和业务的分离，使得企业应用摆脱面向技术的解决方案的束缚，轻松应对企业业务变化发展需要。

1.2.4 SOA 研究发展现状

1996，世界著名咨询公司 Gartner 最早提出 SOA 的概念。SOA 发展至今，各大主流 IT 公司如微软、IBM、SUN、Oracle 以及万维网联盟(W3C)、结构化信息标准进步组织(OASIS)等 IT 知名组织，都对 SOA 进行了深入研究，并已经完成了 SOA 中基本的协议制定，基于基本协议可以实现 SOA 部署，同时新协议不断提出，以应对部署时出现的问题。总之，目前 SOA 设计模式研究处于繁荣和快速发展阶段。

基于 SOA 的业务系统整合在国外研究推广较好，已经应用于电信、金融、医疗等多个领域，出现了法国电信基于 SOA 的 Boss 系统的经典案例。相比较来说，国内的 SOA 研究发展受各个企业及研究机构规模实力所限，还是相对薄弱。研究方式多以跟随式为主，主要研究如何解决实际业务问题。国内的 SOA 厂商知名的有普元软件、东方通科技、金蝶软件等。国内基于 SOA 进行业务整合的经典案例有山西移动通信公司的基于 SOA 的企业业务流程整合、江苏联通公司的基于 SOA 的统一项目管理系统等。国内 SOA 产品市场主要由 IBM、BEA 等国外知名厂商占据，但是普元软件的普元 EOS、东方通科技的 BOV 系列产品如 TongLink、TongIntegrator 等也已经在国内业务系统整合领域占据一定市场份额。

1.3 研究内容、目的和意义

本文对物流园区业务系统现状、面向服务体系架构 SOA 及相关技术等内容进行了系统分析研究，目的在于提出一种实用的物流园区业务系统整合平台解决方案，可以在业务层面对业务系统进行整合。研究成果一方面可以为政府制定物流信息发展政策、搭建物流信息平台提供决策与技术参考，另一方面也可以为物流园区有效地挖掘与利用已有物流资源，优化园区各企业内部资源及整合整个行业资源提供理论与技术支撑。

SOA 对物流业务系统业务整合有很强的现实指导意义，可以使 IT 系统更加灵活，及时适应新的业务需求，并能使异构系统尽可能进行无缝整合，从而达到减低开发成本和重用现有业务系统的目的。

本文结合 SOA 设计思想，在解决物流业务系统异构整合问题具有以下重要意义：

（1）将 SOA 思想用于业务整合，专注于以业务为中心的体系架构设计，而非底层 IT 基础结构设计，实现业务与技术实现的分离。

（2）使用 SOA 体系结构使得整合平台能充分利用业务系统，尽可能消除了各物流企业遗留业务系统 “信息孤岛” 现状的影响。

（3）消除了不同设计语言、软硬件平台及开发方式带来的业务系统整合问题，实现异构系统松耦合集成。

1.4 课题来源

本课题来源于哈尔滨公路货运主枢纽信息系统建设项目一期工程—“黑龙江省龙运物流园区业务系统整合平台”，得到了交通运输部公路科学研究院公路交通发展研究中心的大力支持。

1.5 论文组织结构

本论文由以下六章组成：

第一章概述我国物流信息化建设面临的机遇与问题，以及引入 SOA 设计模式的初衷。并对国内外物流信息化发展现状进行了剖析；阐述本文研究目的和意义；阐明了本文的研究内容和课题来源。

第二章梳理了面向服务体系结构 SOA 的相关理论及 Web Services、XML、ESB 等相关技术，并有针对性的介绍了这些技术的特点及其相互关系。

第三章介绍了涉及到的相关物流概念，对物流园区业务系统建设现状进行深入分析，对业务系统整合模式进行了研究得出选用 SOA 设计模式的优势。在此基础上详细分析了物流园区业务系统整合平台的体系架构、功能需求。

第四章提出了基于 SOA 的物流园区业务系统整合平台的系统设计方案，并对平台逻辑设计、体系架构、数据处理技术、服务处理技术等进行了深入研究。

第五章对基于 SOA 的物流园区业务系统整合平台实现进行研究，使用 IBM WebSpere Message Broker 和 Message Queue 搭建实现平台，并对业务整合、数据整合的实现进行了研究。

第六章对全文进行总结，并对以后的研究工作进行展望。

第二章 面向服务体系结构SOA及相关技术

2.1 面向服务体系结构SOA

在现代企业应用中，信息系统要依据企业业务逻辑需求去重新组织已存储的数据，从而将现有数据、事务等通过Internet浏览器或者手持设备等新展现方式呈现给用户。现有的主流操作系统大都采用分布式结构，多台工作站同时运行，使得企业业务计算体现出高效、可用及大规模特点。基于以上情况，设计应用企业级解决方案时，必须协调运行群组形式的硬件。应对的方法之一是通过群组服务的方式，将分布式系统组合起来，其中单个服务可以提供一组定义良好的功能集合。此方式的实质就是将系统设计为一组相互关联的服务，其关键在于如何将业务系统功能实现为服务形式。它使得系统中的某些服务能够充分利用其他的服务同时却无需考虑其物理位置。系统可通过添加新的服务来不断的升级，这样的软件组织方式就是面向服务的体系结构（Service Oriented Architecture，SOA）[15]。

2.1.1 SOA概念

1、SOA

1996，世界著名咨询公司Gartner最早提出SOA的概念。SOA发展至今，有不同的组织结构对其分别进行了阐述[16,17,18,19]，以下这种定义是较全面清晰：SOA是一种设计模式，它指导着业务服务在其生命周期（从构思开始，直至停止使用）中包括创建和使用的方方面面。SOA也是一种定义和提供IT基础设施（指机构中与IT有关的各种硬件、软件、服务及数据通信设置等的总合）的方式，它允许不同应用相互交互数据、参与业务流程，无论它们各自背后使用的是何种操作系统或者采用了何种编程语言。SOA可以被看作是一种构建IT系统的方案，它将业务服务（即一个机构向顾客、客户、公民、合作伙伴或其他机构直接或间接提供的服务）作为协调IT系统与业务需求的关键组织原则[20]。

SOA的通用概念模型由三个基本角色和三个基本操作组成，如图2.1所示。

三个基本的角色分别是[21]：

（1）服务提供者（Service Provider）：即服务的拥有者，服务提供者负责向服务注册者发布一个服务的描述，同时作为服务的宿主控制对服务的访问。从企业的角度看，这是服务的所有者；从体系结构角度看，这是托管被访问服务的平台。

（2）服务请求者（Service Requester）：也称为服务使用者。它可以是一个应用程序、软件模块或服务等。它通过服务注册者对服务进行查询、绑定并执行。

（3）服务注册者（Service Broker）：服务注册者是服务发现的支持者，它提供服务存储和发现服务的功能。它包含一个服务提供者所提供服务的存储库，并允许感兴趣的服务请求者查找服务。不过，在 SOA 中也可以没有服务注册者，此时服务请求者需要通过其他方式获得服务的描述，如直接从服务提供者处获取。

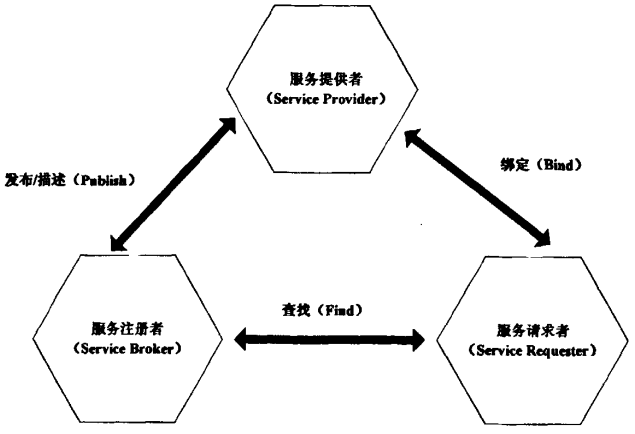

图 2.1　SOA 概念模型

三个基本操作分别是[22]：

（1）发布/描述（Publish）：服务提供者发布一个可访问服务的描述，其中包括服务的网络位置、传输协议和消息格式等调用所需的信息。

（2）查找（Find）：服务请求者使用查找操作定位一个可访问服务。

（3）绑定（Bind）：服务请求者可以在发现一个合适的服务后用服务发布中提供的描述信息调用此服务。

2.1.2 SOA 体系结构

1、典型层次结构

SOA 的一个典型层次结构如图 2.2 所示。此层次结构中自上而下各层次内容为：

（1）服务流程层：用于描述多个基本服务如何通过集成形成更大规模服务

（2）服务发布发现层：存储服务描述，并为应用系统或其它服务提供查询服务描述能力，可采用集中或分布式存储服务描述，并保持服务松散耦合。

（3）服务描述层：提供服务描述的标准格式，包括服务的接口和功能描述，接口描述可以被多个服务实现实例化和引用。

（4）消息传递层：提供传递消息的一致方式。

（5）网络层：定义支持的网络数据传输协议，比如常见的 HTTP、FTP、SMTP 等。

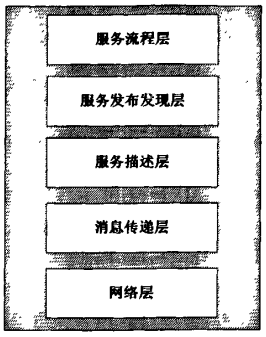

图 2.2　SOA 典型层次结构

2、体系架构特点

SOA 作为一种抽象、松散耦合和粗粒度的软件体系结构，具有以下几个特点[23]：

- 松散耦合性
- 粗粒度服务
- 可重用服务
- 标准化接口
- 可从组织外部访问
- 遗留系统支持
- 响应业务更迅速
- 平台无关性

3、SOA 设计模式

服务是指一个组件的集合，它们向外界提供某个接口，能够完成某种业务功能。服务和组件的不同之处在于服务更多地从业务的角度出发进行设计，向用户提供一个完整的业务的实现，而组件可能只提供完成某个业务的部分功能。

在 SOA 中，服务可以放置在网络中任何位置实现，通过对外发布服务描述，其他的系统（或者服务）就可以发现并且使用此服务。采用不同编程语言、硬件环境、数据库的服务都可在 SOA 中无缝集成。SOA 消除了异构、分布环境对应用系统集成的影响。系统开发者只需要考虑业务逻辑，关注部分业务功能的实现，并包装成合适的服务，而不需要考虑和其他服务之间的互操作问题，因此减少了系统的开发风险和成本。

SOA 的设计模式目前主要有三种形式[24]：

（1）服务注册表（Service Registry）模式：通过服务注册表来实现的松耦合和地址

透明的设计原则。

（2）企业服务总线（Enterprise Service Bus, ESB）模式：通过企业服务总线（ESB）把服务集成到公共的总线上供外部调用并加以控制。

（3）服务编排（Choreography）模式：通过业务流程编排服务形成商业流程服务提供给消费者。

在构建综合应用平台时，可以分别选取上面的一种或多种模式完成架构。

4、SOA 实现方法

目前，SOA 的实现方法有以下几种：

（1）CORBA 组件实现方法：CORBA（Common Object Request Broker Architecture，公共对象请求代理体系结构）遵循了通用的分布式系统解决方案模式，使用 CORBA 接口定义语言（Interface Definition Language，IDL）实现服务描述[25]。

（2）DCOM 组件实现方法：DCOM（分布式组件对象模型）是一系列微软的概念和程序接口，它与 CORBA 十分相似，也采用通用的分布式系统解决模式，其服务描述通过 MS-IDL 实现[26]。

（3）远程方法调用（RMI）实现方法：远程方法调用（RMI）属于 Java API，是 Java 分布式对象技术核心，RMI 允许对象存在于多个指定地址空间，分布在各种 Java 虚拟机上。

（4）Web 服务（Web Services）组件实现方法

Web Services 基于 UDDI（Universal Description, Discovery and Integration）标准协议体系。与 CORBA 组件模型、DCOM 组件模型等异构系统集成技术相比，Web Services 提供了一种分布的、与平台无关的应用程序机制，应用于不同企业所开发的不同平台之上的应用程序的协同工作，实现企业间异构信息的共享、交互与集成，是目前最流行、对 SOA 支持较好的实现方法。

5、SOA 优点

由于 Web Services、XML、ESB 等相关技术标准近年来的发展成熟，使得 SOA 在业务系统整合市场得到应用和大规模地发展，其优点在于：

●编码灵活：低层服务基于模块化，高层服务创建采用不同方式组合实现重用。

●支持多种客户类型：服务接口精确定义，并对 XML、Web 服务标准支持。

●更易维护：服务提供者和服务使用者的松散耦合及对开放标准的采用。

●更好的伸缩性：依靠服务设计、开发和部署所采用的架构模型。

●更高的可用性：服务提供者和使用者松散耦合，使用者无须了解服务实现细节。

2.2 相关技术

2.2.1 Web Services 技术标准

W3C（World Wide Web Consortium，万维网联盟）中 Web 服务体系结构工作组对 Web Services 的定义如下：Web Services 是一种通过 URL 识别的软件应用，并通过 XML 语言进行定义、描述和发现。Web Services 支持通过 HTTP 协议交换基于 XML 的消息或与其他软件代理直接交互[27]。

Web Services 技术标准的一个概念性的协议栈如表 2.1 所示。Web Services 技术标准是基于 UDDI(Universal Description, Discovery and Integration)的标准协议体系。Web Services 技术标准是 SOA 目前最流行的一种实现方法。Web Services 技术标准中主要的关键技术是开放的 Internet 标准：UDDI、XML、SOAP、WSDL 和 WSFL，即扩展标识语言(XML)作为数据交换的格式；超文本传输控制协议（HTTP）；Web 服务描述语言（WSDL，用于服务描述）；统一描述、发现和集成规范（UDDI，用于服务的发现、发布和集成）；简单对象访问协议(SOAP，用于服务调用)和 Web 服务流语言（WSFL，用来定义工作流）等。

表 2.1 Web Services 概念协议栈

协议、语言	层	业务问题		
WSFL	服务工作流	安全	管理	服务质量
Stack ⟶UDDI	服务发现			
Direct ⟶UDDI	服务发布			
WSDL	服务描述			
SOAP	基于 XML 消息			
XML　Scheme	数据模型			
XML	数据表现			
HTTP,FTP,SMTP,MQ	网络传输			

Web Services 技术标准中实现的关键技术有以下几个：

（1）SOAP

SOAP(Simple Object Access Protocol)，简单对象访问协议，是一个基于 XML 的，

用于在分布式环境下交换信息的轻量级协议，可以运行在任何其他传输协议上。W3C SOAP 1.2 规范在服务请求者和服务提供者之间定义使用 XML 格式的消息进行通信[28]。

（2）WSDL

WSDL(Web Services Description Language)，Web 服务描述语言，是一个提供描述服务标准方法的 XML 词汇，它规定了有关 Web Services 描述的标准，业务之间将通过交换 WSDL 文件来理解对方的服务，可以将服务看作是通过 SOAP 访问的对象[28]。

（3）UDDI

UDDI (Universal Description, Discovery and Integration)，统一描述、发现和集成规范，提供了一组公用的 SOAP API，使得服务代理得以实现。UDDI 用于集中存放和查找 WSDL 描述文件，起目录服务器的作用[28]。

（4）WSFL

WSFL (Web Services Flow Language)，叙述网络服务流程语言，由 IBM 提出的基于 Web 服务的工作流，负责组合 web 服务到工作流中[28]。

（5）WS-BPEL

WS-BPEL（Web Services Business Process Execution Language），Web 服务业务流程执行语言，它定义了如何表示业务流程中的活动，以及流控制逻辑、数据、消息相关性和异常处理等[29]。

Web Services 技术作为一种新技术，由于它能提供跨平台、穿越防火墙的访问能力，基于一系列的公开标准，因此 Web Services 的应用成为研究的热点。Web Services 各关键技术关系如图 2.3 所示。

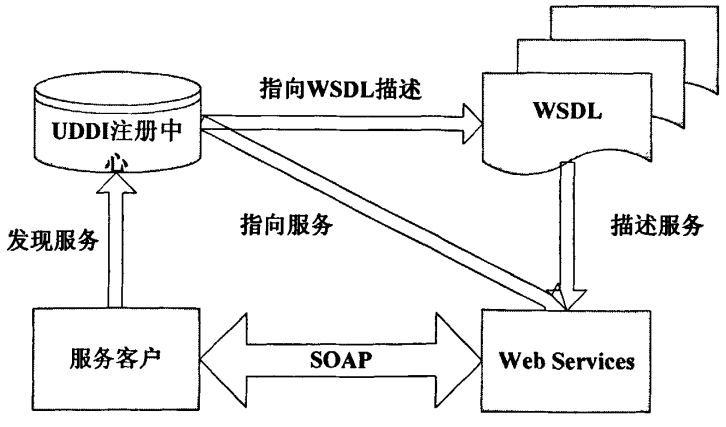

图 2.3 Web Services 各关键技术关系

2.2.2 XML 技术

XML(eXtensible Markup Language)，可扩展标记语言技术，是一种具有数据描述功能、高度结构性及可验证性的置标语言。它是一个基于文本的 W3C 规范的标记语言。与 HTML 使用标签来描述外观和数据不同，XML 严格地定义了可移植的结构化数据。它可以作为定义数据描述语言的语言，如标记语法或词汇、交换格式和通信协议。

为描述 XML 数据结构和约束，至今国内外已提出多种 XML 数据模式语言。最早也是最成熟的是 DTD，而将逐渐取代 DTD 的是 W3C XML Schema。

文档类型定义（Document Type Definition，DTD）就是用 XML 表示特定类型文档的规则集，实际可以看作一个或多个 XML 文件的模板。模式（Schema）表示图解、计划或框架，是描述 XML 数据结构的数据模型。在 XML 中，它指描述 XML 文档的文档。Schema 相对于 DTD 的明显好处是 XML Schema 文档本身也是 XML 文档。异构系统之间要进行数据交换，必须有双方都能理解的共同约定的 XML 消息结构，这个消息结构可以用 DTD 或者 Schema 定义。Schema 是 XML 世界中的标准建模语言，SOAP、WSDL 和 UDDI 的 XML 语法都是采用 Schema 进行描述的。

XML 基于文本的特点，决定了 XML 十分适合作为数据交换统一格式。基于 XML 的数据交换格式具有以下优点[30]：

- 开放性：XML 的规范和标准是开放的，它允许在任何平台上读取和处理数据，允许通过和其他传输协议交换数据。
- 简单性：XML 是基于纯文本的，由于各种应用都提供了对纯文本最好的支持，使得 XML 成为不同应用系统之间进行数据交换的最简单方式。
- 易于扩展：XML 的标记由用户定义的，理论上其类型的数量可以至无限。
- 结构化特性：XML 文档的实现是一种树形的结构，通过标签的嵌套，XML 可以描述任意层次的文档结构。
- 交互性好：用户与应用进行交互时，使用 XML 可以非常方便地在本地排序、过滤和进行其它的数据操作，不需与服务器进行交互，减轻了服务器的负担。
- 语义性强：XML 可以自行设计有意义的标记，便于异构系统之间的数据交换和信息检索，实现机器与机器之间的信息交换。
- 目前所有的主流关系型数据库系统都支持 XML，比如 Oracle8g、DB2 7.0、SQL Server2000 以上的版本中，都直接支持 XML 文档到数据库的双向数据读写。
- 与平台无关：XML 是一种自描述的语言，本身就已经包含了元数据。同时 XML

是基于纯文本的语言，能够被各种平台支持。

综上所述，XML 很适合基于 SOA 的平台中立性的需要，Web Services 中的数据表示和消息描述等规范都是基于 XML 形式描述和表达，XML 的出现对 SOA 是最底层的技术实现手段。

2.2.3 SOA 与 Web Services 关系

SOA 与 Web Services 是两个不同层面的问题。SOA 是面向商业应用的概念模式，而 Web Services 则是面向技术框架的实现模式。面向服务的体系结构所表示的是一个概念上的模型，表述了松耦合的应用如何在网络上被描述、发布和发现。而 Web Services 则是由一组协议构成的协议栈所定义的框架结构，它定义了在异构系统之间通信松耦合的编程框架。Web Services 实际上是 SOA 的一个特定的实现。SOA 将网络、传输协议以及安全等具体的细节都遗留给特定的实现—Web Services。

Web Services 仅仅是实现 SOA 的一种技术，但它建立在开放标准和独立于平台的协议的基础之上，其自身的发展、特点在很多层面上都能满足 SOA 的实现需求：Web Services 中所有的访问都通过 SOAP 访问进行，用 WSDL 定义的接口封装，通过 UDDI 进行目录查找，可以动态改变一个服务的提供方而无需影响客户端的配置，外界客户端根本无需关心访问服务器端的实现，满足了松耦合的要求；基于性能和效率平衡的要求，Web Services 服务提供的是大颗粒度的应用功能，而且跨系统边界的访问频率也不会象程序间函数调用那么频繁，并通过使用 WSDL 和基于文本的 SOAP 请求，可以实现能一次性接收处理大量数据，适合大数据量低频率访问符合服务大颗粒度功能；Web Services 所有的通讯是通过 SOAP 进行的，而 SOAP 是基于 XML 的，XML 是结构化的文本消息。从最早的 EDI（Electronic Data Interchange,电子数据交换）开始，文本消息也许是异构系统间通讯最好的消息格式，适用于 SOA 强调的服务对异构后天宿主系统的透明性，基于标准的文本消息传递为异构系统提供通讯机制。因此 Web Services 成为当前对 SOA 支持较好的、最佳的实现方法。

2.2.4 企业服务总线 ESB

企业服务总线 ESB（Enterprise Service Bus）是从 SOA 发展而来的。SOA 中的各服务之间进行通信时，需要一个智能中介来动态地调度服务，这样的中介角色可以由 ESB 来承担。SOA 实现和解决了服务模块间调用的互操作问题，为了提高企业级服务集成的适应性、灵活性、可扩展性，降低集成门槛，引入了 ESB。事实上，ESB 是 SOA 架构

的主要切入点，它在 SOA 架构中充当实现服务间智能化集成与管理的中介，是逻辑上与 SOA 所遵循的基本原则保持一致的服务集成基础架构，它提供了服务管理的方法和在分布式异构环境中进行服务交互的功能。ESB 是一种新兴的、松散耦合的、基于标准的实现服务和应用无缝集成的中间件技术[31]。

1、ESB 功能模型及特征

ESB 的复杂功能模型如下表 2.2：

表 2.2 ESB 复杂功能表[31]

服务通信	服务交互
集成	服务质量
安全性	服务级别
消息处理	管理和自治
建模	基础架构智能

但是，当前的大多数场景下只需要 ESB 复杂功能模型中的部分功能，表 2.3 是 ESB 支持 SOA 所需的最低功能。

表 2.3 ESB 最低功能表[31]

<table>
<tr><th>通信</th><th>集成</th></tr>
<tr><td>● 提供位置透明性的路由和寻址服务
● 控制服务寻址和命名的管理功能
● 至少一种形式的消息传递范型（例如，请求/响应、发布/订阅等等）
● 支持至少一种可以广泛使用的传输协议</td><td>支持服务提供的多种集成方式，比如 Java2 连接器、Web 服务、异步通信、适配器等等</td></tr>
<tr><td colspan="2">服务交互</td></tr>
<tr><td colspan="2">一个开放且与实现无关的服务消息传递与接口模型，它应该将应用程序代码从路由服务和传输协议中分离出来，并允许替代服务的实现。</td></tr>
</table>

ESB 的功能可以简单概括为这几个方面：在服务与服务之间路由消息；在请求者与服务者之间转换传输协议；在请求者与服务者之间转换消息格式；处理来自于各种异构源的业务事件；保证服务质量（安全、可靠和交互处理）。

ESB 的基本原理是：通过标准的整合技术，将 SOA、Web Services 和 XML 等技术融合到统一的分布式架构中，搭建易于部署、可管理的整合基础设施。它既可集成新的

应用服务，也可通过分解、包装遗留系统，使其提供服务接口，从而集成已有的应用。ESB 还提供了连接企业内部和跨企业间的新的和现有软件应用系统的功能，通过集成松散耦合的、平台独立的服务接口，ESB 充当了服务使用者和服务提供者之间的中介，实现服务组合和业务流程自动化管理[32]。ESB 可以支持多种接口技术，图 2.4 是 ESB 接口技术示意图，其所能支持的接口技术包括但是不局限于图中所示内容。

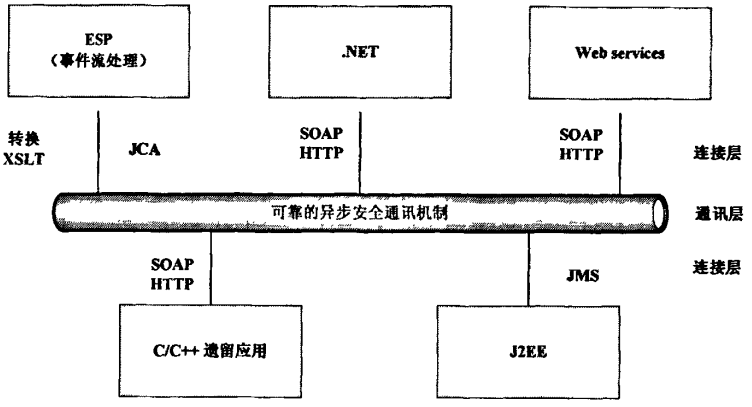

图 2.4　ESB 接口技术示意图

作为 SOA 架构中服务集成功能的重要提供者，ESB 有以下重要特征[33]：

（1）ESB 是在 SOA 架构中实现服务间智能化集成与管理的中介

作为 SOA 的核心和基础架构，ESB 通过与它连接的各种应用的服务级接口实现各种应用之间的连接，控制它们之间的通信和交互。

（2）ESB 明确强调消息处理在集成过程中的作用

这里的消息指的是应用环境中被集成对象之间的沟通。ESB 具有可靠和高可用的异步消息传递能力，同时它也有容错能力和一定的安全性。ESB 不仅仅是提供消息交互的通道，更重要的是提供服务的智能化集成的基础架构。

（3）事件驱动成为 ESB 的重要特征

事件驱动体系结构（Event Driven Architecture，EDA）是一种非常适合 SOA 的体系结构，它作为 SOA 的一种最佳实践备受关注，而 ESB 正是将二者联系起来的关键部分。连接到 ESB 上的松散耦合的各系统之间进行通信和交互时不需要相互知晓通信协议和要求。

2、ESB 内部要素

ESB 是一个预先组装的 SOA 实现，它包含了实现 SOA 分层目标所必需的基础功能部件。

如图 2.5 所示，ESB 内部要素有数据传输协议、软件总线控制器与软件总线控制策略、软件总线管理器与软件总线管理策略、软件总线与构件库的接口策略和元数据五种[34]：

（1）数据传输协议：表明在软件总线上的数据传输的协议标准，具体的数据如某个数据结构、可序列化对象等都需要封装在该协议中。

（2）软件总线控制器则通过元数据数据库中配置的控制策略来控制软件总线上的数据传输以及其他的服务（如安全、事务等）。

（3）软件总线管理器则管理通过元数据数据库中配置的管理策略来管理软件总线，如监控总线状态、异常处理、日志等。

（4）服务构件库则通过接口策略来发布服务、以及通过接口策略来进行数据交换。

（5）元数据则是存储软件总线的策略（控制策略、管理策略、接口策略）以及整个软件总线的状态信息。

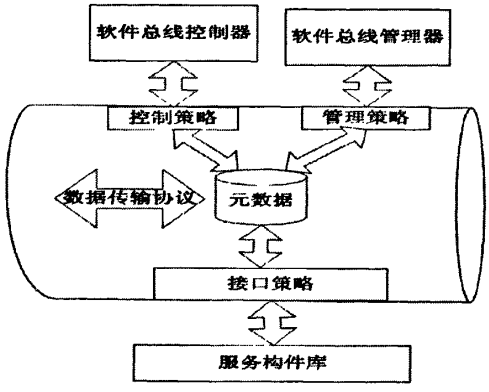

图 2.5　ESB 内部要素

3、ESB 对 SOA 的改进

ESB 是逻辑上与 SOA 所遵循的基本原则保持一致的服务集成基础架构，它提供了服务管理的方法和在分布式异构环境中进行服务交互的功能。ESB 对 SOA 的改进如下图 2.6 所示：

在复杂企业计算机环境中，传统的服务连接方式是在服务请求者和服务提供者之间采用端到端的交互。随着应用程序增加及复杂度提高，最终会导致应用程序之间关联会异常复杂，形成一个网状结构，从而使得系统维护费用非常昂贵，同时使得 IT 基础设施的重用变的困难重重。ESB 作为一种基础设施，消除了服务请求者与服务提供者之间的直接连接，可以使得两者之间耦合度进一步降低。

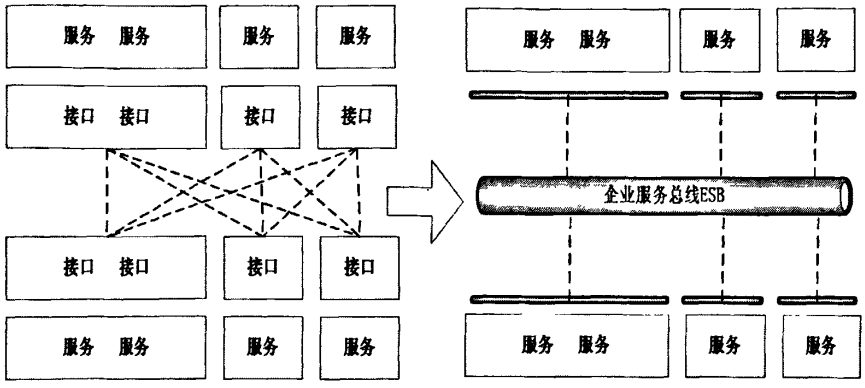

图 2.6 ESB 对 SOA 的改进

ESB 位于 SOA 的中心，并通过减少接口的数量、大小和复杂度使得 SOA 更为强大。ESB 一般应用在基于 SOA 的企业应用环境中，作为其集成服务组件的核心总线。也可不作为 SOA 的组成部分，而作为服务独立运用。

4、现有 ESB 产品

现有的 ESB 产品很多，可以分为两大类—商业产品和开源产品。

（1）商业产品

微软本来没有 ESB 产品，近年来为了争夺 SOA 领域的地位，目前在.Net Framework2.0 的基础上，借助 Visual Studio 2005，推出了 Indigo 产品，目前有最新的版本产生，但也是在此基础上的完善和包装。

IBM 则推出了基于 J2EE 的 WebSphere 系列产品为 SOA 的应用提供解决方案，目前使用最多的是 WebSphere Message Queue（消息队列，简称 MQ）和 WebSphere Message Broker（消息代理，简称 MB）。此后 IBM 又推出了 WebSphere Message Broker Toolkit，它是 Websphere Message Broker 的一个可视化的开发界面，底层是 Eclipse 平台。

BEA 公司则推出了基于 J2EE 的 WebLogic 平台。近来 BEA 公司又开发了 ALSB (AquaLogic Service Bus)，ALSB 用来异构系统集成和互操作。使用 ALSB 作为服务总线不会排斥使用其他技术，ALSB 可以和.Net 应用，可以和 Tibco、Sap、Oracle、JBoss、WebSphere、Siebel 等系统集成。

（2）开源产品

Apache 基金（http://www.apache.org）提供了一系列开源产品，基本涵盖 SOA 的所有领域，从服务器到开发平台。此外还有 http://www.springframework.org 提供的 spring 框架以及 http://www.jboss.org 提供的 JBoss 产品。可以说开源产品为 ESB 企业级应用提

供了丰富的解决方案以及开发工具。

2.3 本章小结

本章对研究内容所涉及到的面向服务的体系结构 SOA、Web Services、XML、企业服务总线 ESB 等关键技术进行了介绍，并有针对性的介绍了这些技术的特点及其相互关系。

第三章 物流园区业务系统整合平台总体研究

3.1 相关物流概念

1、物流

物流是物品从供应地向接受地的实地流动过程。根据实际需要，将运输、储存、装卸、搬运、包装、加工、配送、信息处理等基本功能实施有机结合[39]。物流的产生是满足消费者的需求或为了某一目标、任务。物流中移动的主体是货物和与之相关的物流信息。

现代物流涉及物流企业、工商企业、银行、保险、税务、海关、检验检疫、外贸、交通、信息产业和政府等众多单位与部门，基本上由四个行业组成：交通运输业、储运业、通运业和配送业。它是一项管理技术，是对货物流动和仓储过程的计划、执行和控制，其核心活动是运输和仓储。

2、物流园区

物流园区是对物流组织管理节点进行相对集中建设与发展的具有经济开发性质的城市物流功能区域；同时也是依据相关物流服务设施进行的降低物流成本、提高物流运作效率和改善企业服务有关的流通加工、原材料采购和便于与消费直接联系的生产等活动的具有产业发展性质的经济功能区[35]。

现代物流园区具有产业集约化经营优势，它主要具有两大功能，即物流组织管理功能和依托物流服务的经济开发功能。物流园区的物流组织与管理的功能一般包括：货物运输、分拣包装、储存保管、集疏中转、市场信息、货物配载、业务受理等，而且多数情况下是通过不同节点将这些功能进行有机结合和集成而体现的，从而在园区形成了一个社会化的高效物流服务系统。不同类型的物流园区可以有不同的功能组合。

3、物流信息系统

物流信息系统LIS(Logistics information System)是现代物流企业信息化的基础，它利用现代信息技术对物流活动中各种信息进行实时、集中、统一的管理，使物流、资金流、信息流三者同步进行，及时反馈物流市场、客户和物品的动态信息，为客户提供实时的信息服务，为企业提供管理决策依据。

物流信息系统主要有以下特点：物流节点（企业、园区、货站、仓库等）普遍实行信息化管理；整个系统具有无限的开放性；信息流在整个物流过程中起引导和整合作用；

系统具有明显的规模优势。物流信息系统一般可分为物流企业子系统（运输管理系统订、单处理系统、采购订货系统、进货系统、仓储管理系统、出货系统、配送管理系统等）、运输工具子系统（运输工具识别系统和货物数据采集系统等）、现场子系统（分布于道路、仓库以及场站的现场物流设备与管理控制系统）、用户子系统（为物流信息系统用户提供查询以及交互功能的子系统）、行业管理子系统（车队与货运管理系统、营运货运车辆管理系统等，为各类物流企业提供相关的公共信息支撑）5个子系统[36]。其组成框架如图3.1所示：

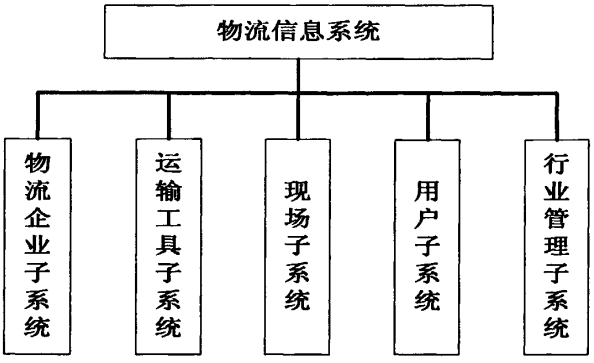

图 3.1 物流信息系统组成框图

3.2 物流园区业务系统建设现状分析

物流园区作为较大规模和综合物流功能的物流集中区，集中了众多的专业化物流企业，包括运输企业、仓储企业、货代企业、配送企业、装卸企业、第三方物流企业等。物流园区为入驻物流企业提供仓储设施、办公场所等物业，是一个物流节点，同时自身又是一个物流企业。它既要从事物流业务活动，对外提供货物集散、货物中转、配送加工等服务，又要负责园区综合管理工作，进行办公管理和物业管理。

近年来，随着我国交通、物流信息化建设进程加快，包括园区入驻物流企业在内，我国很多物流企业为提高效率，降低成本，开始了信息化改造，根据自身核心业务需求分别建设了相应的业务管理和企业内部综合管理系统；同时所有类型的物流园区为了维持基本功能，建立了相应的园区综合管理系统。图 3.2 显示了园区业务系统建设现状。通过这些业务系统，园区内物流企业业务管理能力得到提高，物流园区运营能力得到增强，社会效益和经济效益良好。

随着信息技术的发展，软件设计模式从传统的面向对象、面向组件发展到了现在最先进的面向服务。但是园区业务系统大部分都是基于面向对象设计开发，从企业特定业

务需求出发，独立建立，系统平台、程序开发语言及信息传递标准可能不同，数据标准不统一，接口不一致，单一分散，彼此间很难通信，相互之间无法联动和协同，形成了所谓的“信息孤岛”，这就不可避免的会导致同一业务系统的不同模块间、不同业务系统之间的信息无法共享，流转不畅，业务流程割裂，需要过多的人工介入，引起效率下降和数据精确度降低。

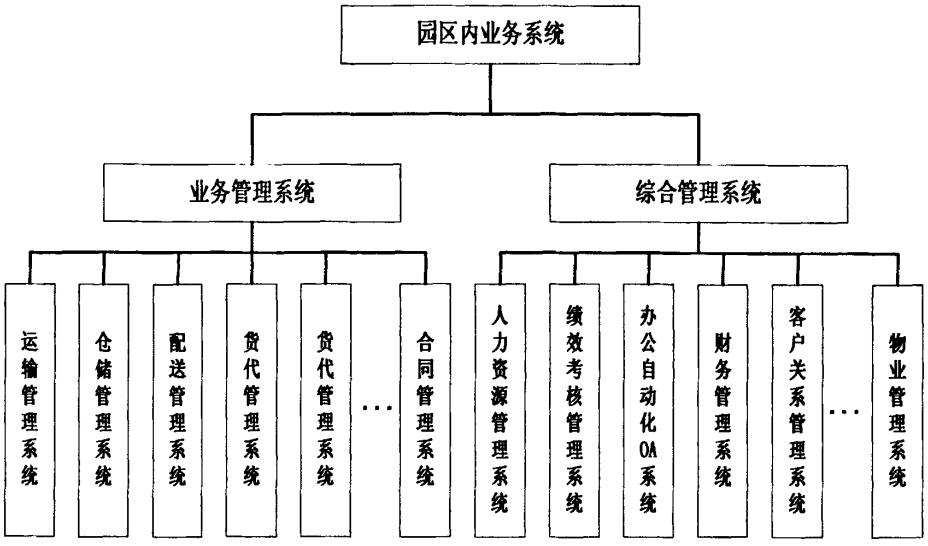

图 3.2 物流园区业务系统框架示意

园区内的业务系统，按其建设年限可以分为近期开发业务系统和遗留业务系统。相比近期开发业务系统，遗留业务系统是物流企业早期开发的，已经运行很久的业务系统，这些业务系统普遍性能落后，技术上过时，不能满足企业现在的业务需求。但是由于这些系统已经融入了企业的业务运行中，不能简单的丢弃重建。这些业务系统，作为企业宝贵的 IT 资源，在这些遗留系统基础上进行信息系统升级改造时，如果使用增量式方法，使遗留系统可以在现代网络化环境下继续发挥作用，可以替各企业节约资金，对于加快其信息化进程将具有重要意义。

园区业务系统按其架构又可以分为 C/S 和 B/S 两种模式，对于 C/S 模式的业务系统，比如合同管理系统，都是单机版本，用于处理相对独立的业务功能，没有过多的考虑与其他系统进行数据和业务交互，基本没有提供外部应用接口。同时建立至今，随着信息技术的飞速发展，以及企业规模、业务模式的变迁，传统的应用开发、升级和变更模式已经不能满足业务需求，成为遗留系统。但是这些业务系统却不能轻易的被替换，这会带来很多的开销，首先是是管理上的开销，需要维护的系统越来越多，很多系统的数据是相互冗余和重复的，数据的不一致性会给管理工作带来很大的压力。次之业务和业务

之间的相关性也越来越大，比如财务管理系统和人事管理系统之间不可避免的有着密切的关系。再次业务办理人员已经熟悉业务操作流程，推倒重做需要考虑投资、人员培训各种因素。

B/S 模式的业务系统，基本上是在最近几年内建设的，技术、架构相对成熟，系统性能相对先进，在开发的过程中会考虑到有些系统资源为外部其他应用所调用，，所以在开发的过程中会预留一些外部调用接口，可以通过直接调用这些接口来与其进行交互。

同时这些业务系统，都有自己的权限管理和用户信息数据，操作人员需要牢记密码等身份认证凭证；业务办理人员在跨系统办理业务时，需要进行多次登录进行身份认证。这都给业务开展带来了极大不便。

目前，我国绝大多数物流企业的特征是“小”（经营规模小）、“少”（市场份额少、服务功能少、高素质人才少）、“弱”（竞争能力弱、融资能力弱）、“散”（货源不稳定且结构单一、网络分散、经营秩序不规范），规模小，底子薄，多数企业专业人才、技术和资金缺乏，无力去建设升级新的业务系统，也会增加企业成本压力，不符合经济原理。

综上所述，物流园区业务系统现状可以概括为：

- 业务系统数量多，提供的业务功能多，用户数量多；
- 业务系统相互异构，业务、数据不易交互，成为“信息孤岛”；
- 技术落后，可扩展性 、灵活性差，不适用于今后业务发展的需求；
- 迫于成本，不宜推倒重做。

物流园区内物流业务系统的现有状况，使得各个业务系统各自为政，相互无法互联互通，不能有效的协同工作，以满足园区和物流企业进一步发展壮大的业务需求，从而物流园区产业集约化、规模化优势无从发挥。

3.3 业务系统整合模式研究

3.3.1 业务系统整合简介

所谓的业务系统整合，是指将各个业务系统的业务流程、应用软件、硬件、各种标准等资源联合起来，实现两个或更多的企业业务系统之间的无缝集成，使其能如同一个整体一样进行业务处理和信息共享。

业务系统整合的目的在于实现企业内人员、组织、管理与技术的高度集中，提升企业内各业务交互部门、组织、个人及业务系统之间的协同工作能力和效果。只有有机整

合各个业务系统，才能实现信息共享，从而提高决策及管理效率，做到“正确的信息在正确的时刻以正确的方式传到正确的地方”。

业务系统整合是按照统一标准，实施数据集中，以此为基础，实施相互交叉业务彼此衔接成，以一体化的方式实现信息系统资源共享和协同工作。其实质上是将分散的要素组合在一起，以松耦合的方式形成一个有效率的整体。

业务系统整合的方式按层次可分为表现层整合、应用层整合、业务逻辑层整合、数据层整合等[37]。

- 表现层整合：面向用户的整合，将原先系统图形界面使用一个标准的界面进行整合，使用企业门户（Portal）技术把不同的系统整合在一起。
- 应用层整合：在应用层直接实现企业内部各业务系统整合并使之协同工作。
- 业务逻辑层整合：实现业务流程的部署和自动化处理，满足企业业务流程管理，这种方式可跨越企业内部不同业务系统、不同业务组织以及不同企业。
- 数据层整合：提供访问和整合分布在不同系统中的数据。

在不同的业务系统整合方案中既可以采用不同层次的整合方式，又可以有机结合几种方式统一运用。

3.3.2 传统业务系统整合模式

传统上，企业进行业务系统整合时，使用的是企业应用集成（Enterprise Application Integration，EAI）技术。EAI 通过对企业中完成不同业务功能的应用系统进行集成，在它们之间建立起可供数据交流和应用沟通的纽带，进而使他们之间的信息交互成为可能。EAI 是对原有信息系统的延伸和扩展，不应该完全推倒原有信息系统重建；同时企业应用也应该是动态、灵活的，能方便的根据业务变化低成本重构[38]。EAI 实际上就是通过中间件来连接企业内外各种业务相关的异构系统、应用以及数据源，从而满足支撑系统等重要系统之间无缝共享和交换数据的需要。

1、传统业务系统整合层次

作为一个涉及广泛主题的企业业务系统整合方案，传统上的 EAI 方式包括 BPM、连接器、事务处理、安全、消息服务等，其按层次可分为数据层集成、应用层集成、业务逻辑层集成三种。各种方式都可实现业务系统整合，但是都会导致一定问题，数据层整合可能会导致数据损坏，数据库的安全缺口打开等问题；应用层整合可能会增加现有业务系统的不稳定性；业务逻辑层整合导致各业务系统紧耦合，降低了系统的灵活性。

2、传统业务系统整合式所存在问题

EAI 对整合的各种方法不做限制，可以方便地针对不同的应用采用不同的整合方法和工具，这就使得整合具有较好的针对性，可以方便的利用任何已有的技术和方法。所以 EAI 可以实现性能较高的集成系统。但是 EAI 技术本身所具有的不确定性也带来了一些问题：实施费用高、产品相关、无法移植、实施强度大、集合系统紧耦合[37]，不利于企业业务流程的调整和重组，缺乏可扩展性和灵活性。

因此，为解决以上问题，急需出现一种面向功能层的企业业务系统整合方式，既能保证原有系统的数据安全性和逻辑安全性，又能实现各系统之间的松耦合，方便系统流程重组和优化。

3.3.3 基于 SOA 的业务系统整合模式

图 3.3 为基于 SOA 的企业业务系统整合参考架构示意图。基于 SOA 的企业业务系统整合模式，各个业务系统只需要以服务的形式出现，选择与该系统交互的其它系统，能够简单发现那些服务，并且在运行的时候或者是设计的时候，与这些服务绑定。面向服务的整合使得 IT 机构能够在已有的应用中提供可重用的服务的功能，可以最大限度的同时满足性能和灵活性的要求[39]。

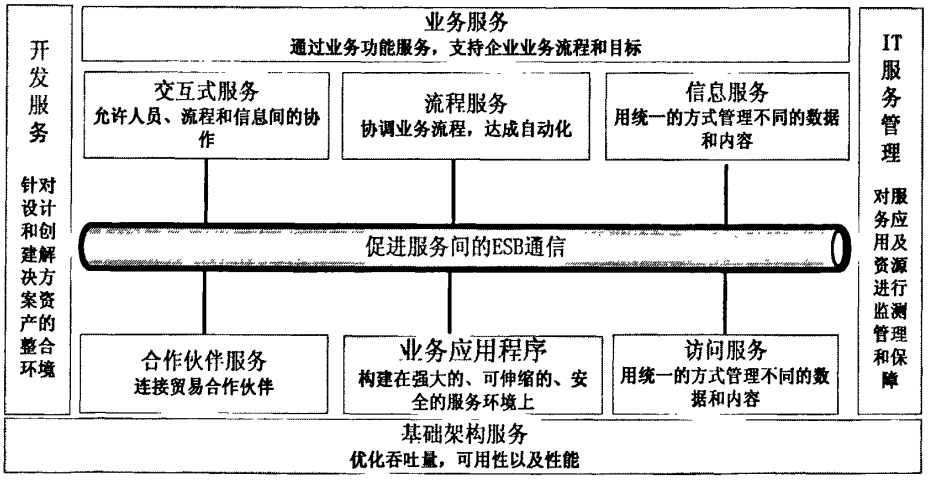

图 3.3 基于 SOA 的企业应用整合参考架构示意

ESB 作为一种中间件技术，实现并支持 SOA。它为 SOA 提供与企业需要保持一致的基础架构，从而提供合适的服务级别和可管理性，以及异构环境中的操作。ESB 的功能主要体现在通信、服务交互、应用集成、服务质量、安全性以及管理和监控等方面，这使得它可以成为 SOA 集成基础架构的关键部分。

我国物流产业处于高速发展时期，各个物流企业的业务模式、业务流程相对不稳定，

这种状况就要求物流业务系统整合平台在建设时，能结合业务模式来驱动应用整合；同时可以体现出物流信息的流通、实时特性。这要求平台应具有对外的流程介入能力，并能提供统一、标准的数据、服务接口。传统整合模式方法并不能很好的满足这些要求。

相比之下，采用基于 SOA 的业务系统整合模式来构建物流业务系统整合平台，具有以下优势：

（1）有利于整合现有业务系统。采用 SOA 的体系结构可以基于现有的系统来发展，并不需要将现有业务系统推倒重建。同时原有功能模块可以通过 Web Services 技术封装接口来进行访问。

（2）便于快捷组合物流业务服务。SOA 体系结构能够方便组合松耦合的服务，从而提供更为优质和快速的响应，应对不断变化的业务需求。

（3）提高系统开发速度。现有的组件、新开发的组件和从厂商购买的组件可以合并在一个定义良好的 SOA 框架内，从而成为了重复使用的架构元素。当需要加入新的物流企业时，可以直接对现有服务和组件进行重用，提高了系统开发部署速度。

（4）降低系统开发成本。使用 Web 服务库来构建和部署服务将显著减少软件开发成本。

（5）便于改进业务流程。SOA 清晰表示物流服务业务流程，这些业务流程通过在特定业务服务中使用组件的顺序来标识，给物流企业提供了监视业务操作的理想环境。

3.4 物流园区业务系统整合平台总体分析

物流园区业务系统现状决定了需要通过整合，建立一个可以提供数据统一、维护统一、用户统一、安全可靠的认证与授权服务，并能整合园区内部业务系统包括软硬件等各种资源在内的物流信息平台，一方面可以通过物流信息平台实现业务协同，满足和适应企业多种业务功能需求，提高物流服务水平，促进园区内物流企业群体间协同经营机制和战略合作关系的建立；另一方面也为各物流企业省去建设升级维护相应物流业务系统的费用，使其可以专注于业务拓展，更好的得到发展。作为物流信息平台的核心部分，物流业务系统整合平台建设成为促进物流园区信息化水平进一步得到发展的关键所在。

物流业务系统整合平台应是一个集成的、开放的、高效的、且具有柔性的 IT 支持平台，在 Web 服务广泛使用的现状下，更应是一个开放的 Web 应用集成平台。它是一项园区内现有业务系统条块结合、联合共建、实现信息网络互联、信息资源共享、业务协同工作的综合性信息系统工程。通过平台建设，可以改进园区内物流企业业务系统不

能互联互通的现状，消灭信息孤岛。

3.4.1 体系架构

体系架构就是一套构建系统的准则。通过这套准则可以把一个复杂的系统划分为一套更简单的子系统的集合，这些子系统之间应该保持相互独立，并与整个系统保持一致。而且每一个子系统还可以继续细分下去，从而构成一个复杂的企业级架构[42]。

体系架构设计是平台建设的核心的问题。作为一个系统的骨架，不同体系架构所能解决问题的类型及实现方法各不相同。平台体系架构设计不能只从功能角度出发，除了要考虑平台体系架构及其应具有的功能行为以外，关注整个架构的可用性，性能问题，容错能力，可重用性，安全性，扩展性，可管理维护性，可靠性等各个相关方面，对任何一个方面的欠缺考虑都有可能为整个平台的构建埋下隐患。

物流园区业务系统整合平台需要这样一种体系架构：先进实用，可以最大程度的重用遗留业务系统，在不改变遗留业务系统原有结构，不影响其正常运行的前提下整合各个异构业务系统；可以提供一种业务交互机制，通过业务交互实现系统动态、松耦合整合，并可以控制和实现交互方式的自动化；提供全面统一的用户管理、可靠的身份认证、有效的分级授权、便捷的信息共享；可以实现业务重用，各物流企业可以根据自身业务需要进行业务组合；有很好的可扩展性和适应性，能够适应未来应用需求的不断变化。

物流园区内物流企业众多，涉及的物流环节多，因此可以提供的物流服务功能也多，要实现业务协同，需要把这些业务服务功能整合起来。

面向服务系统结构 SOA 以服务为中心，服务可以交互重用，实现技术和业务的分离，同时 SOA 在服务数量很大的情况下，其性能优势会更加明显。SOA 这些特点都十分符合物流园区业务系统整合平台的架构需求。

3.4.2 功能需求

物流业务系统整合平台对园区业务系统进行整合，包括业务功能、数据资源及用户三方面的整合。业务功能整合是目的，数据资源整合是支撑，而用户整合是展现方式。

1、业务功能整合

图 3.4 表示在业务整合前，位于一体化物流体系中的园区各物流企业，在开展物流业务时，需要使用多个业务系统，同时如果发生业务交互，则需要将交互产生的业务数据输入到各自的业务系统中，操作不便，同时数据需要通过中间介质进行转发，安全性也得不到保证。

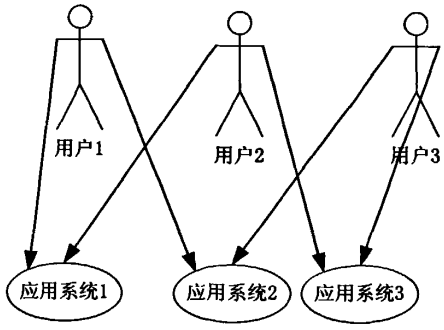

图 3.4 业务整合前状况示意

在传统的实现方式中，使用接口系统或接口模块来满足业务交互时业务系统的访问需求。但是由于接口处数据紧密耦合，而各个遗留业务系统难以修改或升级，企业也不能随着商业环境的变化迅速改变业务流程。因此需要园区各遗留业务系统进行业务整合，以最大限度的减少系统间的耦合，提高可重用性，提高业务敏捷性、客户满意度以及企业之间的无缝连接。

图 3.5 表示，通过建立业务系统整合平台，用户可以通过平台方便快捷的访问多个业务系统，发生业务交互时，数据在平台内部流转，不会暴露于平台外，业务可以方便的进行交互，安全性高；同时遗留业务系统得到重用。

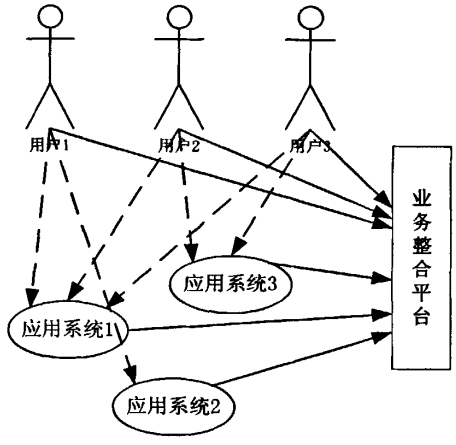

图 3.5 业务整合后状况示意

从技术角度讲，服务是一个可供远程访问的独立的应用程序模块。在其抽象观点中，从服务器应用到打印机到搬运工再到快递公司等一切事物，都可以被视为是一个服务提供者，服务提供者通过接口向外界展现可提供服务的能力。服务模型对外展示出一个全新的、整合的能力。

而遗留业务系统要能够与近期开发系统整合，做到即插即用，则首先必须进行封装。封装实质上是用一个掩藏老系统不需要的复杂性和输出一个现代接口的软件层来包

围遗留系统。园区内每一个新系统的出现都会带来遗留系统封装的问题

在进行业务整合时，可以使用服务模型，结合封装概念，将业务系统的不同功能单元封装为服务，只对外暴露标准服务接口；业务系统通过这些服务之间定义良好的接口和契约联系起来，从而实现接口和底层实现的分离，不同的底层实现也可以共用一个接口。服务调用者只需要懂得接口的含义和使用方法；而服务的底层实现可以随时根据需要而变，并不会影响到用户的使用。

Web 服务是为实现“基于 Web 无缝整合”的目标而提出的概念，它为调用者提供一个通过 Web 进行调用的 API，使调用者能用编程的方式通过 Web 调用来开发应用程序，这就是所谓的 Web 服务封装。

因此，业务系统整合平台设计可采用 Web 服务封装的方法，对业务功能进行整合。无论业务系统是近期开发系统还是遗留系统基于 B/S 模式还是基于 C/S 模式，平台在进行业务功能整合时不会受到业务系统能否提供外部调用接口的困扰，都可以把业务封装为服务单元，整合平台使用这些服务单元，组合后为用户提供业务服务。通过业务功能整合，最大限度的重用遗留业务系统，同时简化了业务组件的开发及平台设计的组装和部署，可提高平台的可移植性、可重用性和灵活性。同时由于业务功能封装为服务，可有效保护企业的业务逻辑资源，安全性高。

2、数据资源整合

业务系统整合平台在完成业务功能整合后，用户需要调用服务进行业务交互，必然要涉及到数据资源整合问题。数据整合是要将处在不同业务系统中的业务数据，通过整合平台实现数据流转，实现数据资源共享。

平台需要解决的数据传输问题是指由于各个业务系统处于不同的企业内，所在地域不同，所处的网络环境不同，所使用的数据传输协议不同，所提供的服务和业务数据同样各不相同等原因所造成的处于不同软硬件和网络环境中的业务系统与平台通信困难。数据不能实现安全准确传输，就不能实现信息孤岛之间的联通，解决数据的异地分布，从而不能保证用户在使用不同的服务时能获得准确的数据。

平台所要解决的数据转换问题一方面是指由于平台所整合的业务系统基于各种技术和数据库平台先后建立，会有数据异构的问题。异构性主要表现在两方面：系统异构和模式异构。各个业务系统本身结构不同，还有所使用的数据库管理系统乃至操作系统不同构成了系统异构；数据在存储模式上的不同造成了模式异构。主流的数据存储模式有多种，但是即使是同一类数据存储模式，其模式结构可能也存在着差异，比如 Oracle

中所采用的数据类型与 SQL Server 所采用的数据类型就不是完全一致的。异构数据会造成服务封装的业务数据与用户所需数据格式不匹配。另一方面，用户在调用所需业务功能时，有时会出现接收到的数据并不符合自身要求，需要进行数据转换。

因此，平台需要建立一个合适的传输机制，并建立一个统一的数据格式以及应用接口，方便业务数据以统一的数据格式从业务系统安全准确传输到平台，同时用户所需业务数据也可以通过平台顺利得到；提供数据转换功能，对数据进行转换、分割或连接处理；同时保持各业务系统的自治性，在进行数据整合同时不能影响已有系统的正常运行。

数据转换问题可以通过建立对每种不同的业务数据格式进行解析的机制来解决，但是这种方法只适合于服务总量较少的状况。对于物流园区这种集中了大量涉及物流各个环节、规模大小不一的物流企业的物流节点，用户调用服务的请求数量是巨大的，这种解决方法开销巨大，不符合业务整合、重用的初衷。XML 的简单、可扩展、基于文本、平台无关性等特点，使得其可以作为整合平台的统一的数据格式，平台中所有数据都可以转换为 XML 格式。同时异构系统进行信息交互时，必须有双方都能理解的共同约定的 XML 消息结构，这个消息结构可以用 XML DTD 或者 XM Schema 定义。XML 文件需要进行转换时可以使用 XML 转化语言 XSLT 进行解析转换。

3、用户整合

建立整合平台的一个目的是使得园区物流企业、园区运行综合管理人员以及社会用户可以方便快捷的使用业务信息系统，提高物流效率。

物流企业业务系统，比如运输管理系统、仓储管理系统、办公自动化系统、财务管理系统、档案管理系统等，服务于物流企业的信息化建设，为企业带来了很好的经济效益。但是，业务办理人员在使用这些业务系统时经常很不方便。园区内物流业务环节多，业务量大，操作繁琐，业务办理人员在开展业务时，需要在多个系统间跳转，进入每个系统时，都必须输入用户名称和用户密码，进行身份验证；而且不同应用系统，所需用户账号不同，业务办理人员必须同时牢记多套用户名称和用户密码。对于业务系统数目较多，用户数目也很多的物流企业，这个问题尤为突出。产生这个问题的原因并不只是是系统开发失误，更多的应该是系统在早期建设中缺少整体规划，缺乏统一的用户登录平台。

同时出于园区内不同物流企业规模、物流业务类型各不相同，其业务办理人员所要使用的业务系统也各不相同，不同用户有其个性需求。这就需要为用户建立一个统一的平台入口，方便用户使用；平台还需要为用户提供个性化业务系统定制功能，满足其个

性化需求。

单点登录（Single Sign On，SSO），目前比较流行的用户整合的解决方案之一。单点登录是指访问同一服务器不同应用中的受保护资源的同一用户，只需要登录一次，即通过一个应用中的安全验证后，再访问其他应用中的受保护资源时，不再需要重新登录验证。使用 SSO 的好处主要有：

- 方便用户。
- 方便管理员。
- 简化应用系统开发

单点登录的出现，为解决用户整合问题提供了一种解决办法。业务平台可通过建立单点登录（Single Sign On，SSO）功能，解决用户整合问题，将多个业务系统的用户统一进行管理，在多个业务系统共存时，用户只需要登录一次就可以访问所有相互信任的应用系统。

单点登录用例如图 3.6 所示：

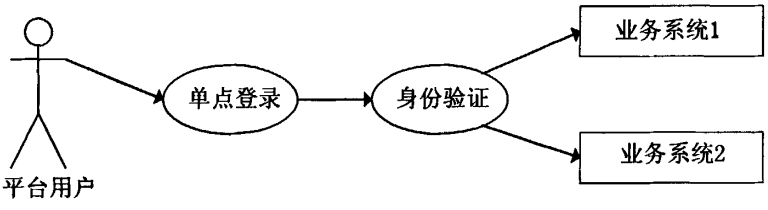

图 3.6 单点登录用例图

而门户（Portal）技术是 IT 领域的新技术，是信息化发展的方向之一。Portal 提供了一个单一的访问各种信息资源的入口，可以将应用、人员、信息与流程有机的结合起来，可以为业务系统整合的开展提供可行的思路和解决方案。

Portal 技术强调以用户为中心，提供身份验证、工作管理与监控、信息集成、安全管理、文档管理等功能，并可以提供单点登录界面，为用户提供统一的个性化桌面应用。Portal 也是业务系统整合平台解决用户个性需求的一种良好的方法。

3.5 本章小结

本章介绍了物流、物流园区等物流相关概念，并在此基础上对物流园区业务系统现状以及业务系统整合模式进行研究分析，得出了基于 SOA 的业务系统整合模式应用于物流业的优势所在。最后对业务系统整合平台体系架构以及功能需求进行了深入分析。

第四章 基于SOA的物流园区业务系统整合平台设计

4.1 设计目标及要求

1、设计目标

根据第三章分析内容，我们可以得出基于SOA的物流园区业务系统整合平台的设计的主要目标是：基于SOA架构思想和相关实现技术，整合物流园区内资源，实现应用和数据的完全集中，建立一个可靠、实时、稳定的物流业务系统整合平台。解决各类异构信息系统的集成和协同；解决各类异构数据源的共享和交互，保证数据的准确性、一致性和完整性；为平台用户提供清晰、友好、易用、安全的操作平台；为其它外部系统提供安全、有效的系统接口；为系统管理员提供方便的、完整的监控管理；为实施人员提供快速的、便捷的进行业务定制。

2、设计要求

构建物流园区业务系统整合平台的技术要求应该为无缝整合、应用提升、物理分散和逻辑集中，即无缝接入园区内各种已有的或新建的业务系统、业务数据等资源，并通过应用整合、数据整合、内容整合、流程整合，支持到统一的信息平台，通过安全可控的方式提供给平台各类用户。其设计应满足以下要求：

- 先进性：平台设计科学合理，采用业界领先的软件架构，体系架构完整，便于以后的升级和维护，具有实用价值。
- 成熟性：平台中所用产品要经过市场考验，尽量避免采用较小厂商开发的产品。
- 松耦合：平台设计可以将应用程序定义为不同组件（或称为服务），通过这些服务之间定义良好的接口和契约联系起来。
- 可靠性：平台运行稳定，易于维护，安全机制健全，能保障所集成数据和服务的安全性。
- 可扩展性：平台集成的数据和应用服务在保证目前需要的前提下，能满足未来发展的需要，易于扩展，灵活升级。
- 可伸缩性：在业务和吞吐量突然增长时，平台能保证平稳运行，能在不进行重构情况下进行升级。
- 标准性：在使用新技术的同时充分考虑技术的国际标准化，严格按照国际国内相关标准设计实施，充分利用现有资源，统筹考虑，长远规划。

- 开放性：数据和业务应用面向所有用户开放，并为其他应用系统或未来需要整合的应用系统提供接口式开放。

4.2 总体设计研究

4.2.1 设计思路

通常，完善的系统或者平台是难以基于单一技术而实现的，基于 SOA 的应用整合问题也是如此，往往需要很多种理论和方法来共同解决。SOA 的要点不只在于关注服务设计细节，更重要的是整体体系结构设计，良好的体系结构可以更多的节省成本，还可以提高平台的可维护性和扩展性，这也是决定园区物流业务系统整合平台建设成败的关键因素。

SOA 目前的最佳实现方法是 Web Services。传统的基于 SOA 应用整合中，主要选择 Web Services、XML 和 WSDL 技术，依赖于提供 HTTP 和 SOAP 协议的基础上的跨平台的数据传输处理。随着 UDDI、WS-BPEL 等技术的出现和发展，解决了传统 SOA 整合模式中出现的服务注册、客户端不稳定、私有服务暴露等问题。尤其是 BPEL 标准的实现和产品化进一步加速了 Web Services 以 SOA 构建基础的身份在全球应用，甚至出现了跨企业、跨领域的服务组合。伴随大规模的 Web Services 集群出现了服务集成关联复杂度问题，并需要一种混合的消息传递模型将 Web Services 的优点与传统的异步消息传递结合在一起以满足更大规模任务需求。ESB 的出现解决了这个问题，所以 ESB 是解决目前 SOA 应用整合中面临各种问题的一种最直接有效的方法。

考虑到物流相关企业遗留业务系统应尽量做到重用，而且所有的服务、功能、数据需要通过一种有效方式连接起来，因此引入 SOA 构架及 Web Services，使用相关技术、工具将已有应用中需要开放的功能封装为 Web 服务。这样处理后，各业务系统可以将自己提供的功能以 Web 服务的方式在网络上提供，用户就可以方便地从平台上调用这些服务而无需了解服务实施细节，又能保证原有的应用不受影响。

同时使用 ESB 技术平台来构造 SOA 的架构，把封装后的服务注册到 ESB，完成服务的注册，并由 ESB 负责服务的发布、管理、请求和调用，这样既充分的发挥了 ESB 的作用，又完全符合 SOA 的设计模式，可以解决应用接口潜在的兼容性和维护冲突的问题。

因此，物流园区业务系统整合平台应选用基于 SOA 的以 ESB 为中心的体系架构，使用 Web Services 封装服务，以 XML 作为数据交换标准，采用应用层整合和数据层整

合两种整合方式有机结合实现业务整合目标，以提高业务整合的完整性和功能完备性。

4.2.2 整合平台逻辑体系设计

结合物流园区业务系统整合平台的业务需求，可以得出基于 SOA 的物流园区业务系统整合平台总体逻辑体系，其结构如图 4.1 所示：

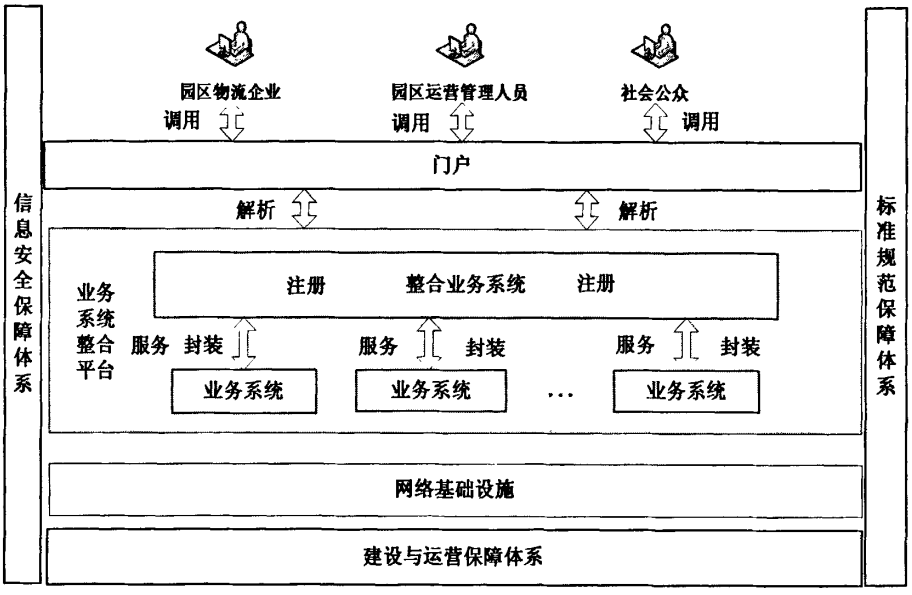

图 4.1 业务系统整合平台总体逻辑体系图

总体逻辑体系图展示了园区物流信息系统的逻辑组成以及业务系统整合平台在其中的逻辑地位。

物流园区物流信息系统由网络基础设施、业务系统整合平台、门户以及保障体系组成。

网络基础设施包括园区内的基础通信网络、网络设备、主机及存储系统等，是保证业务系统整合平台运行平稳的 IT 基础设施。信息安全保障体系、标准规范体系、建设与运营保障体系属于业务系统整合平台保障体系。信息安全保障体系采用相应技术、权限管理手段和管理体制等，充分保证系统信息等数据的安全；标准规范体系规范工程相关数据的设计、存储、交换；建设与运营保障体系包括基于建设模式及运营模式上的保障体系。落实机构、人员和资金，制定一整套科学合理的建设管理体系以及长效运营机制，规范系统的建设管理。门户是园区业务系统整合平台对外服务主要形式。

图中灰色部分显示的是业务系统整合平台。业务系统整合平台通过整合园区业务系统，可以为园区开展一体化物流业务，提高物流效率，降低运营费用，提供更强大的 IT 支持，是园区物流信息系统的核心部分。其体系架构设计，是本文重点研究内容。

整合平台建立在园区的网络基础设施之上，负责对园区内部包括遗留系统在内的各个业务系统，比如运输管理系统、仓储管理系统、配送管理系统、OA 系统、财务管理系统等，进行业务功能整合、数据资源整合、访问安全控制、事务日志管理等，所有业务系统将功能连接到整合平台上来，由平台为用户提供统一的应用接口和数据格式，从而避免了传统点对点整合方式中业务功能之间交叉引用频繁的弱点，可以实现整个平台统一规则，统一调度。

园区运营管理人员、园区内物流企业以及有物流需求的社会公众在业务系统整合平台建立后，通过平台统一入口进入相应业务系统开展业务，同时原有遗留业务系统通过重用的方式继续发挥作用。

4.2.3 整合平台体系架构设计

结合物流园区业务系统整合平台的业务需求，结合园区业务系统整合平台的逻辑架构，我们给出基于 SOA 的物流园区业务系统整合平台的体系架构图，其结构如图 4.2 所示。

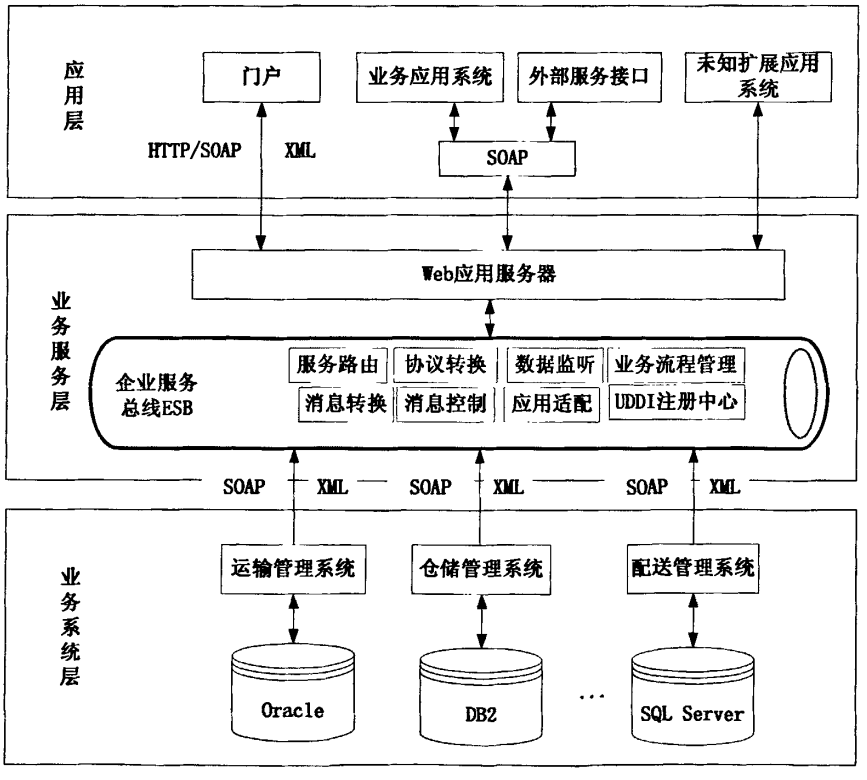

图 4.2 业务系统整合平台体系架构图

物流园区业务系统整合平台体系架构采用分层结构设计，清晰明了。平台共分为三

层，从下到上依次为业务系统层、业务服务层和应用层。

业务系统层是平台所需要整合的资源层。包括物流园区内各物流企业业务系统、园区综合管理系统等在内的各种信息系统以及软硬件资源。业务系统层是业务系统整合平台的基础层，用于支撑物流园区物流业务顺利开展。

业务服务层是平台的核心层。它由企业服务总线 ESB 以及 Web 应用服务器组成，负责对业务系统层各类资源提供的功能按照规定协议和标准封装成不同粒度大小的服务，注册，将服务发布，供应用层查找和绑定，以及对外提供定义明确合理的接口。其既可以提供与业务系统一致的服务，也可以通过服务组合，提供扩展服务，体现了平台架构的灵活性和扩展性，以应对迅速变化的业务需求变动，做到“随需应变”。服务层还可以提供服务安全、交换数据标准等功能。

企业服务总线 ESB 作为平台核心和基础架构，位于业务系统整合平台的中心位置。其功能是将园区内业务应用系统包括软件、硬件都作为服务提供者，通过整合平台对外提供服务。ESB 对园区内业务系统进行服务封装，每个服务都是一个不可再分的单一业务逻辑单元。并将这些逻辑单元注册到 ESB，服务调用者从 ESB 中获得所需服务。使用 ESB 可以减少接口的数量、大小和复杂度，主要完成的功能有：服务之间的消息路由；请求者和服务之间的传输协议转换；请求者和服务之间的消息格式转换；保证服务质量（安全、可靠和交互处理）。业务系统整合平台建立统一的标准和接口，为平台内部业务应用系统以及外部服务调用提供交互信息，标准化操作。

应用层包括门户、整合后的业务应用系统、外部服务接口以及未知扩展应用系统等企业级应用。供平台用户进行所需业务操作，对业务逻辑进行调用和触发，对数据进行显示和处理。应用层一方面为园区内企业提供各种整合后的业务服务支持，另一方面也可以把园区的业务信息传递给社会有物流需求的公众，有助于园区业务开展，从而使得园区内业务信息可以互通交互。

业务系统整合平台中以 XML 作为数据交换标准格式，层与层之间、服务之间通讯也是采用 XML 格式的消息，独立于业务系统。业务应用系统、外部服务接口采用 SOAP 协议与业务服务层通讯。

基于 SOA 的物流园区业务系统整合平台的设计建设后，园区内的物流企业在进行综合业务处理时，可以通过业务系统整合平台随时获取相关业务系统的数据，免除了以往手工数据导入导出的麻烦，而且可以保证数据的准确性、安全性和有效性，对于提升物流园区的物流效率具有积极意义。

4.3 整合平台数据处理技术研究

前述章节中我们已经对物流园区业务系统整合平台的数据整合需求进行了详尽分析。归纳分析结论，可以看出整合平台进行数据资源整合，对数据进行处理时需要解决以下几个问题：统一数据格式、数据传输、数据转换。

1、统一数据格式

业务系统整合平台需要整合多个异构系统，数据格式各不相同，因此平台的统一数据格式必须是采用一种公开的、被广泛接受的、稳定的并且具有预期长久生命力的规范或标准。

第二章中我们梳理了 Web Services 相关技术标准，其中 XML 的简单性、易于扩展、结构化特性、交互性好、语义性强、主流关系型数据库系统支持、与平台无关等特性，使得 XML 十分适合作为对象或标准的描述语言。园区业务系统整合平台选用 XML 作为统一数据格式，通过将所有异构业务系统业务功能或业务数据封装为 Web 服务单元，以 XML 文档的格式存在，从而以统一的数据格式得到整合。

2、数据传输

对于整合平台中的以 XML 文档统一数据格式存在的包括逻辑服务单元、调用消息、服务间通信信息等在内的各种数据，要实现数据安全可靠传输，必须使用标准开放的传输协议来负责通信。

在 Web Services 技术标准中，SOAP（简单对象访问协议）是其标准通信协议，具有平台无关性和厂商无关性。SOAP 基于 XML，是用于在分布式环境下交换信息的轻量级协议，可以运行在任何其他传输协议上，包括常见的 HTTP（超文本传输协议），SMTP（简单邮件传输协议）等。SOAP 协议运行机制可简单理解为：SOAP=RPC+HTTP+XML，其中 RPC（远程过程调用协议）是调用途径， HTTP 是底层通信协议，XML 则是数据传输的格式。

图 4.3 显示了 SOAP 协议的消息结构，它包括 Envelope（封装）、Header（信息头）和 Body（信息体），封装和信息体是必须有的，而信息头则是可选的。HTTP 协议是广泛使用的广域网传输协议，它是个无状态协议，效率不高，但是很多网络安全设备比如防火墙对其都没有很多限制规则，十分适合于异构松耦合系统信息传输。

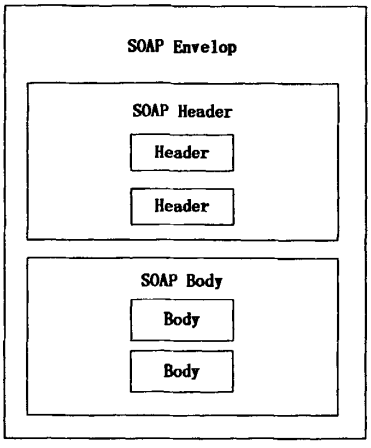

图 4.3 SOAP 消息结构

由此可见，SOAP 的底层通信协议无关性及基于 XML 的数据传输格式，使得 SOAP 十分适合作为平台的数据传输标准协议。外网的社会用户访问平台时，需要把反馈的 SOAP 消息封装为 HTTP 消息格式，只需要给 SOAP 消息上加上 HTTP 消息头即可完成，而平台内部使用 SOAP 协议即可。

由于 HTTP 效率不高，同时 XML 文档转换成数据需要解析，需要额外耗时，所以 SOAP 协议响应速度不快，因此对于效率要求不是很苛刻的情况下，园区整合平台使用 SOAP 协议作为数据传输协议。

3、数据转换

传统的业务整合模式对于异构业务系统数据转换，采用的方法一般是采用建立中间数据库的方法，如图 4.4 所示，即建立所有数据的抽象模型，建立统一的中间数据库，将所有待整合业务系统的数据集中到中间数据库，再通过中间数据库与不同应用系统相连，完成数据集中、转换。连接业务系统与中间数据库通常由数据库中间件来完成。

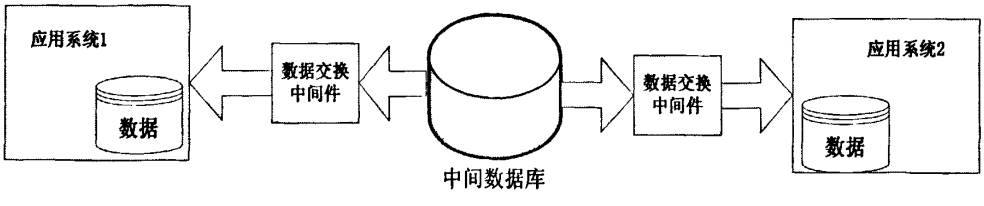

图 4.4 传统数据转换方法

业务整合平台中所有数据以 XML 文档格式存在，解决了数据异构问题。但是对于 XML 文档，因其具有简单性、开放性、易于扩展等特点，其格式灵活，不同人设计以及同一个人不同时间设计的 XML 文档都会存在数据层次关系不同等异构问题，因此需要进行转换，否则 XML 就不能很好的作为平台统一数据格式解决方法。对于异构 XML

文档，进行转换时，可以通过 XML 转化语言 XSLT（Extensible Stylesheet Language Transformations，扩展样式表转换语言）来解决。XSLT 是一项实现 XML 广泛应用的重要技术，它可以将 XML 文档转换成其他结构（包括 XM L 结构）的文档，比如 HTML 文档、纯文本文档等。XSLT 在转换过程中，一般会对数据进行过滤、排序、计算等操作。基于 XSLT 上述特性，在 XML 技术已在网络中普遍应用的背景下，使用 XSLT 转换异构 XML 文档是非常有效的转换方法，实用范围也很大。

因此，基于 SOA 的物流业务系统整合平台解决数据转换的方法是建立交换数据的元数据 XML Schema，通过 XSLT 实现 XML 文档的转换解析，其转换过程如图 4.4 所示。

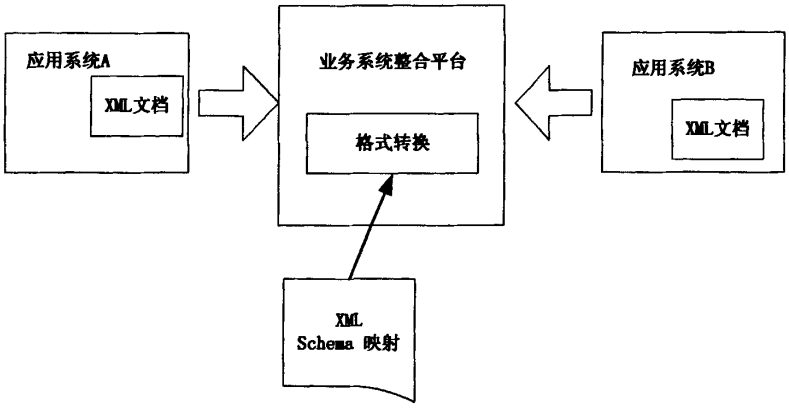

图 4.4 基于 XSLT 的数据转换

XSLT 对 XML 文档进行转换的方法是把 XML 文档看作为树结构，有根节点和子节点，转换的过程就是从源文件树结构进行遍历获得各个节点，解析文档后，将各个节点按照转换模版重新生成目标文件树结构。其转换原理图如图 4.5 所示。

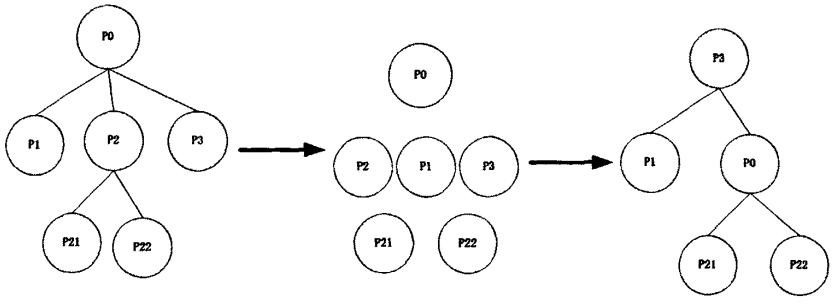

图 4.5　XSLT 转换原理

除了 XML 文档格式的转换解析，用户在进行服务调用时，还会遇到业务数据聚合和分割问题。用户所需业务数据来自不止一个服务时，需要把各个服务业务数据进行聚合后传递给用户；用户所需业务数据只是服务中业务数据的一部分时，需要把业务数据进行分割，因此需要使用 XSLT 结合聚合和分割算法分别进行处理。

如图 4.6，来自应用 1、应用 2、应用 3 的业务数据，经过聚合算法进行处理，成为一个统一的数据传递给应用 4 中。

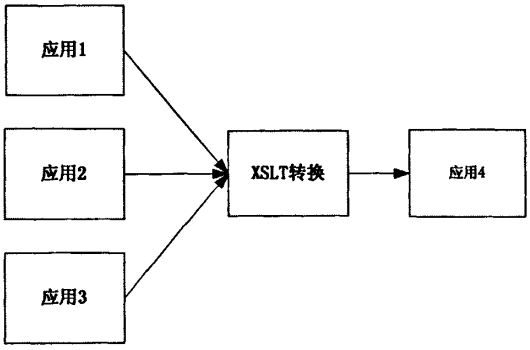

图 4.6 基于 XSLT 的数据聚合

聚合中需要处理三个问题：

（1）数据分组。决定有哪些需要聚合的数据。

（2）开始及结束标志。确定聚合起始点。

（3）转换。各个应用的服务需要进行转换才能聚合成应用 4。

分割是聚合的逆运算，其处理过程正好相反，所需处理的三个问题一样，其处理过程如图 4.7 所示。

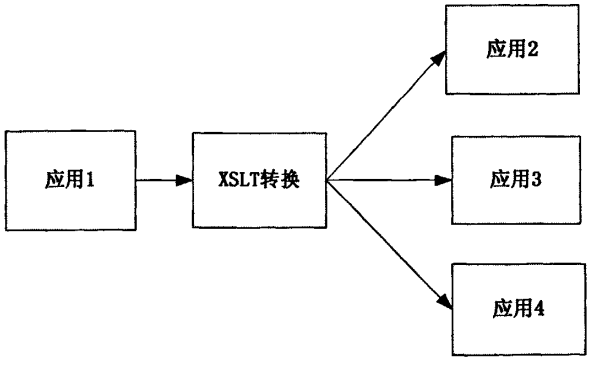

图 4.7 基于 XSLT 的数据分割

4.4 整合平台服务处理技术研究

4.4.1 服务处理技术

平台进行异构业务系统整合是基于服务进行的。服务要经过建模、封装、传输、注册和管理。

1、服务建模

面向服务对象 SOA 的一个特性就是粗粒度服务，进行业务系统整合，首先就是要对业务系统进行服务建模，确定服务。服务建模可以采用多种方式，比如采用自顶向下的方式（分解业务流程抽象成服务并快速组合成新的服务）、自底向上方式（分析业务系统来抽取服务以实现系统重用）等，也可以直接对具体业务目标进行分析生成服务。

服务建模要分析到位，要对业务模式、业务系统包括其子系统、遗留业务系统都进行分析，得出的服务还要进行筛选，根据具体业务需求筛选出粒度大小合适的服务。

2、服务封装

Web Services 技术标准是 SOA 目前最流行的一种实现方法，平台使用 Web Services 技术对服务进行封装。

对于一些近期建立的业务系统，技术上相对先进，可以提供外部接口，在进行业务封装时，只需要把其接口封装为服务即可，此过程类似于开发适配器，建立数据导入、导出。接口封装为服务以后，注册到整合平台上，由平台提供统一的服务接口，以保证业务系统数据访问的安全性。

对于另一些业务系统，包括遗留业务系统在内，只能提供一些业务数据，或者是处于安全考虑，不能提供外部接口，对于这些业务系统进行服务封装时，只需要对其提供的业务数据进行 Web 封装，然后将封装好的数据作为 Web 服务注册到平台，作为逻辑服务单元提供给用户调用。用户调用服务时，查找到所需逻辑服务单元，对其进行解析获得所需业务数据。

Web Services 协议体系中，WSDL 作为 Web 服务描述语言，规定了有关 Web Services 描述的标准，业务之间将通过交换 WSDL 文件来理解对方的服务，所以封装后的 Web 服务以 WSDL 文件格式存在。

3、服务注册

服务封装后，传输进入整合平台，进行服务注册。Web Services 协议体系中，UDDI 用于集中存放和查找 WSDL 描述文件，起目录服务器的作用。它给服务提供者提供注册服务函数，同时为服务发现者提供服务调用函数。所以服务注册可由建立 UDDI 注册中心方式来解决。在本章的物流园区业务系统整合平台中，ESB 已经集成了此项功能。ESB 中自建有 UDDI 注册中心，服务封装为 WSDL 文件后，传输进入 ESB，注册到 UDDI 库中，外部应用访问时需要搜索 UDDI 服务库，查找后即可绑定服务，调出 WSDL 文件解析后，得到所需功能服务。

4、服务传输

园区业务系统整合平台是建立在大量服务基础上的，在服务注册后传输进入 ESB 时，如何管理这些服务使其不会造成互相等待、阻塞等异常情况，需要设立一种良好的服务传输模式。

常见的程序通信方式有以下三种：

第一种是通信公用编程接口（Common Programming Interface for Communications，CPI-C）。如图 4.6 所示，CPI-C 方式是同步通信模型。通信发起方负责控制信息流动，发送数据给接收方，数据流动是双向的。CPI-C 是由 IBM 发展的一个独立平台应用编程接口，其通常用为基于 SNA 网络的高级程序对程序通信应用中提供便利。CPI-C 中参与通信的程序必须要跟踪对话的状态以及对错误的处理和恢复，所以其编程接口相当复杂。

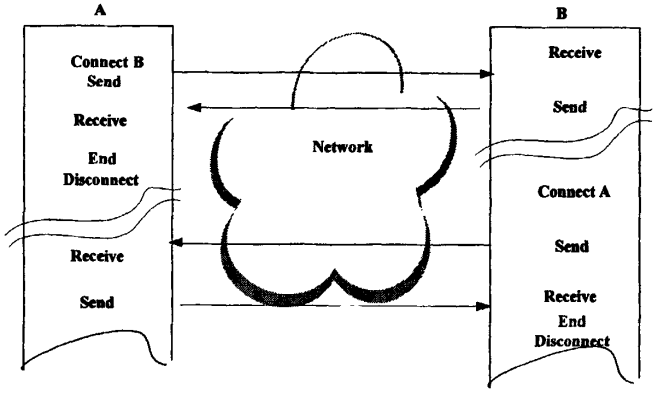

图 4.6 CPI-C 通信模式

第二种是远程过程调用 RPC。RPC 也是一种同步通信模型，如图 4.7，发起方和接收方关系固定，难以实现对等通信。使用 RPC 的程序不必了解支持通信的网络协议的情况，提高了程序的互操作性。和 CPI-C 一样，由应用程序处理错误，并且在申请的服务得到响应之前，服务申请者被阻塞。

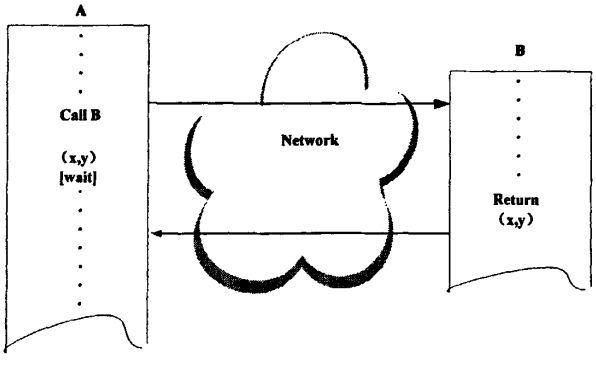

图 4.7 使用 RPC，基于连接的同步通信

第三种则是消息队列接口（Message Queue Interface，MQI）。消息队列接口是一种异步通信方式，如图 4.8 所示。MQI 是一种应用程序对应用程序的通信方法。应用程序通过写和检索出入列队的针对应用程序的数据（消息）来通信，而无需专用连接来链接它们。消息传递指的是程序之间通过在消息中发送数据进行通信，而不是通过直接调用彼此来通信，直接调用通常是用于诸如远程过程调用的技术。排队指的是应用程序通过队列来通信。队列的使用除去了接收和发送应用程序同时执行的要求。通信的方式和使用的传送协议无关，可使用 TCP/IP、SNA 或者其他局域网协议等标准协议。

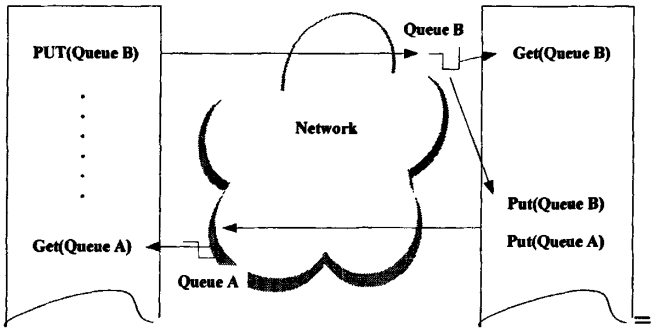

图 4.8 使用 MQI、基于队列的异步通信

通过以上三种通信方式的比较，可以比较得出 MQI 是比较好的通信方式。业务系统整合平台采用 MQI 方式进行传输通信，可以利用消息中间件这种目前较流行的消息传递方式。消息队列是为了消息可靠传输而采用的传统消息传递技术，虽然 SOA 架构中已经定义了基于 Web 服务的相关可靠性标准。业务系统整合平台区别于单一的软件实体，系统与系统之间，服务与服务之间消息的传递大多不在同一机器上，都需要通过网络来进行数据传输，这种离散性以及网络传输的不可靠性，就需要通过系统整合来提供消息传递的可用性和稳定性，通过消息中间件实现服务与服务之间的消息传输。

5、服务管理

大量的服务进行平台后，需要对服务进行管理。WS-BPEL 是 Web Services 技术标准中的业务流程执行语言，其作用是把一组 Web 服务重新组合起来，定义为一个新的 Web 服务。整合后的服务其接口同样由 WSDL 文件描述，成为业务流程。因此，平台使用 WS-BPEL 定义业务流程，将不同的服务组合成为业务流程，使得各个服务之间联系起来，从而实现服务管理。

4.4.2 服务处理流程

基于 SOA 的物流园区业务系统整合平台的服务处理流程如图 4.9 所示，其流程具体包括：

（1）作为服务提供者，园区各业务系统所能提供的功能由平台进行 Web Services 封装。业务功能被封装为 WSDL 文件传输进入 ESB 中。

（2）服务在 UDDI 注册中心中进行注册，成为注册服务。

（3）作为服务请求者的平台用户，在业务操作时，发出服务调用请求，获得反馈后在 UDDI 中查询已注册服务列表。

（4）发现有合适的服务后，平台对服务描述 WSDL 文件进行解析，如果解析的为应用调用接口，则直接输出给用户，由用户进行服务调用。

（5）如果服务内封装的是业务数据，则要与用户所需业务数据格式进行比较，符合要求，直接输出给用户；不符合，则要进行数据转换。转换后的业务数据符合用户要求后，再输出给用户。

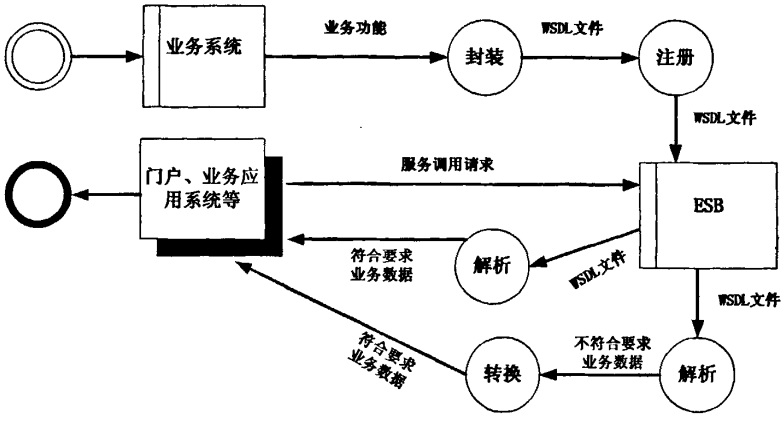

图 4.9 平台服务处理流程图

4.5 整合平台网络拓扑结构

网络拓扑结构如图 4.10 所示。整合平台网络拓扑结构采用数据库服务器、应用服务器、Web 集群三级架构，集中式部署。业务整合平台部署有网路安全管理设施设备区、应用服务器区以及数据区。其中应用服务器区部署有 ESB 部署服务器、Web 应用服务器、身份认证服务器、门户服务器、邮件服务器，以及负载均衡设备、WEB 应用防御设备和防垃圾邮件网关系统，以保障服务器的安全与性能。园区内各个业务系统统一接入应用服务器区，由 ESB 负责服务传输、消息路由等功能，业务系统整合平台通过各

种网络设备接入 Internet，供用户访问。

网络设备包括交换设备、路由设备、安全设备。网络出外而内分别为 DDoS 网关，核心路由器，防火墙，防病毒网关，核心交换机，以及网络安全设施设备区的入侵检测系统、漏洞扫描系统、安全审计系统和网络管理系统。各种网络设备均采取双份冗余方式设计，避免单点故障，满足系统可靠连续运行需求。基于网络健壮性和可靠性考虑，在整个网络设计时所有的骨干链路和服务器都采取备份设计，保证全网不会出现任何单点。对于服务器集群，设计 Web 集群实现负载均衡功能，同时也克服负载均衡设备风险集中、扩展性差的缺点。

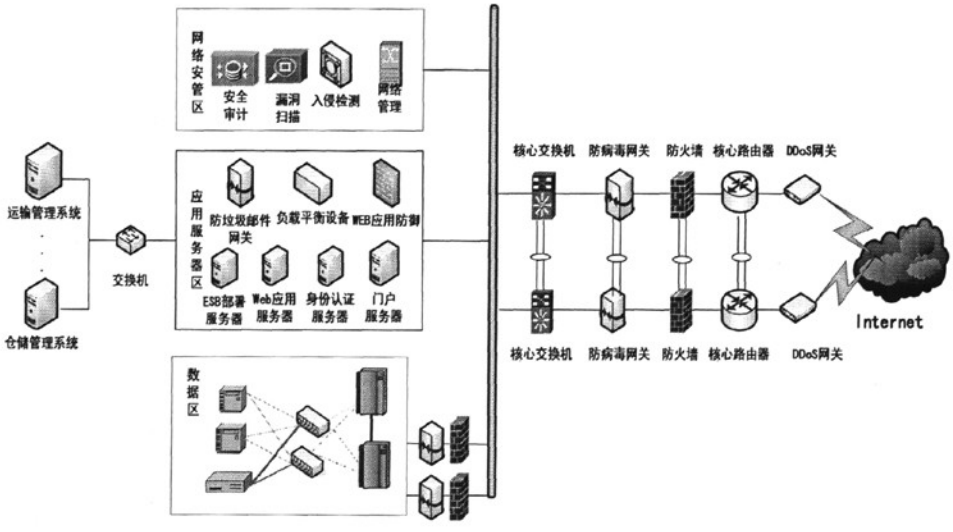

图 4.10 网络拓扑结构图

4.6 本章小结

本章对基于 SOA 的物流园区业务系统整合平台的设计目标及要求进行了阐述，并给出设计思路，对平台逻辑体系、体系架构、数据处理技术、服务处理技术、服务处理流程进行了研究。

第五章 基于SOA的物流园区业务系统整合平台实现研究

5.1 概述

根据物流园区业务系统现状，我们提出了基于SOA的物流园区业务系统整合平台的解决方案，采用基于SOA的体系架构，并使用企业服务总线ESB作为平台核心组件。平台通过对业务系统提供的业务功能封装为Web服务，注册并发布服务，供用户查找、调用，从而实现园区业务系统松耦合整合，因此，根据平台的架构及功能，平台的实现具体可以分为以下步骤：

（1）服务建模

服务建模是指对园区遗留业务系统进行分析，抽取或者生成业务服务。

（2）平台的基础架构搭建

平台的核心组成部件是企业服务总线ESB，ESB是一种中间件技术，创建满足功能需要的ESB作为平台的核心部件，进行平台基础架构搭建。搭建好以后，把封装好的服务部署到应用服务器上，作为ESB集成的外部服务。

（3）业务功能整合

业务整合包括服务封装（对第一步分析生成的服务进行Web服务封装）和服务注册（将部署到应用服务器的服务传输到ESB中进行注册，注册后并生成统一的服务入口WSDL文件供外部调用）。

（4）数据整合

服务注册完成后，要对业务数据进行解析、传输，满足用户调用服务时的数据需求。

（5）服务调用

在应用平台上调用服务并实现业务流程，即开发应用平台，并在应用平台上进行服务调用，实现业务流程。开发应用平台可采用平台基础架构支持的各种成熟的开发技术。

（6）用户整合

即建立单点登录功能以及门户，以实现多系统多用户统一入口统一登录。

本章节主要研究的实现内容为（2）—（4），即从平台基础架构搭建到数据整合。

5.2 平台架构搭建研究

5.2.1 ESB选型

平台架构搭建，主要在于创建能满足平台功能需求的企业服务总线ESB。ESB本身就是一种中间件技术，中间件是介于应用系统和系统软件之间的一类软件，它使用系统软件所提供的基础服务（功能），衔接网络上应用系统的各个部分或不同的应用，能够达到资源共享、功能共享的目的。

前述章节已经提到了现在基于ESB的中间件产品有很多。平台基础架构选用IBM的基于J2EE的WebSphere系列产品：WebSphere Message Queue（消息队列，简称MQ）和WebSphere Message Broker（消息代理，简称MB）来进行平台基础架构搭建。

选型原因在于：

- 作为SOA支持的业务流程管理领域的行业领袖，IBM在各个行业拥有数量庞大的、客户群体广泛的部署消息通讯解决方案，全球超过62000加客户使用WebSphere产品，包括财富100强中93%的客户，其在整合产品创新领域具有10年以上的投资和经验。
- WebSphere系列产品是基于J2EE的产品，而J2EE作为Sun公司推出的一个标准体系结构，是一个成熟的、成功的企业级应用解决方案，拥有大量的客户。它作为企业级的Java平台，安全性高，对Web Services规范支持性好，并可以实现跨平台。
- WebSphere MB是高级的ESB，可以提供智能路由、格式转换、发布/订阅等功能，支持消息中间件，得到所有主流平台支持。WebSphere MQ是IBM的消息中间件，可以提供一个具有工业标准、安全、可靠的消息传输系统，可以实现跨平台操作，为不同操作系统上的应用软件整合提供服务。

WebSphere MB和WebSphere MQ共同搭建组成平台的基础架构，构建基于SOA的物流园区业务系统整合平台。MB和MQ是优秀的用于应用系统间联系的软件，它们基于消息传递机制，通过不断流动的消息将松耦合关系的应用系统联系起来，实现功能叠加。

MB作为ESB连接各个应用系统，其与各个应用的连接方式可以有以下三种：

- MQ方式：利用MQ将MB与应用互联；
- 文件方式：MB监听制定文件目录，有变化时自动读取；

- Web Services 方式：直接利用 Web Services 进行通讯。

我们选用 MQ 方式。MQ 用于保证消息的可靠传输，负责在两个异构系统之间传递消息，通过调用 API 即可实现互相通信，而不必考虑底层系统和网络的复杂性，性能高，支持各种主流平台。MB 建立在 MQ 基础之上，作为 ESB 连接各个应用系统。

MB 支持消息流（对消息的一系列处理的集合），MB 内部业务逻辑可以利用 IBM 的 MB 可视化开发工具 WebSphere Message Broker Toolkit 设定。

MQ 基本由一个消息传输系统和一个应用程序接口组成，其资源是消息（消息就是一个信息单元）和队列（一个安全的存储消息的地方）；通信方式是基于 MQI（Message Queue Interface，消息队列接口）的。MQI（Message Queue Interface，消息队列接口）为程序提供了一种异步通信方式，可以使用各种标准协议。

5.2.2 平台基础架构搭建

构建平台基础架构，首先需要使用 MB 建立多个信息流，然后启动 MB 代理服务器，使得信息流运行其中，代表不同的服务交互，结合 MQ 搭建起平台基础架构。

1、建立消息流

在 MB 中对消息的运算处理、格式转换和路由等功能是通过消息流实现的，每个消息从输入 MB 到从 MB 中输出，都将被一个消息流处理，然后发往目的应用系统。消息流由各种消息处理节点（Message Processing Node）组成，消息处理节点可对消息进行各种处理操作，节点与节点相连，便组成了一个消息流。消息处理节点实际上是被 Broker 运行环境调用的动态链接库（DLL），节点能够对流经自己的消息执行特定的功能。每一个消息流定义一个消息的路由的方式和对数据信息的转换服务，代表着不同的业务流程。这项工作需要在 MB Toolkit 中创建。图 5.1 显示的是 MB Toolkit 界面。

图 5.2 显示了一个信息流。In（MQInput）节点，代表消息流的入口，Out(MQOutput)节点代表消息流出口，消息在进入消息流前以及退出后都进入 MQ 中。Filter 节点，实现 if-else 语句，返回 4 种状态：failure、true、false、unknown。除了这三个消息处理节点以外，MB 中还定义有很多重要节点，比如 Publication 节点，用于消息发布；Mapping，消息到消息和消息到 DB 的映射等。在处理不同业务逻辑时，可以选用不同的消息节点进行处理。

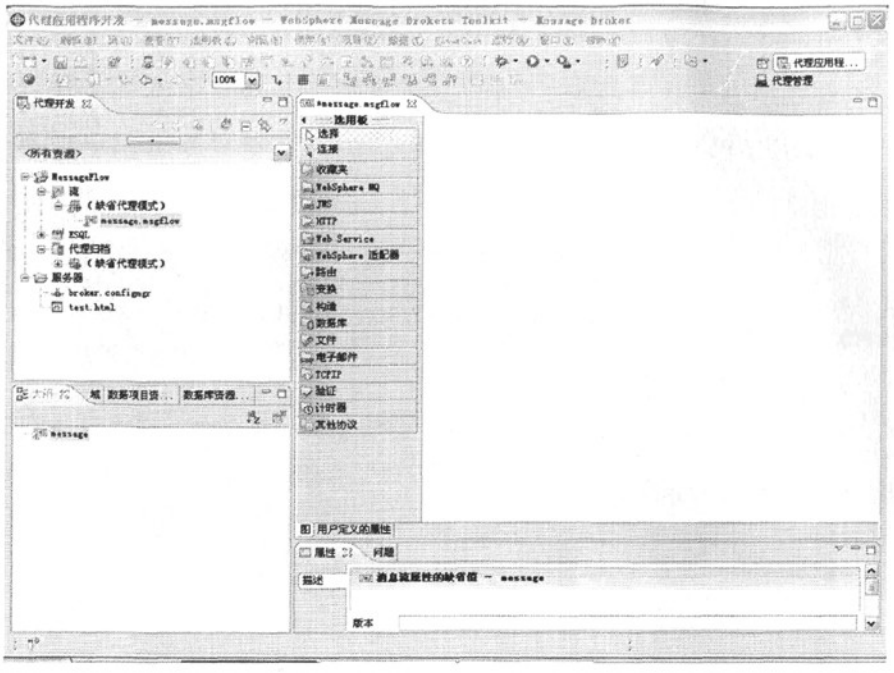

图 5.1　MB Toolkit 界面

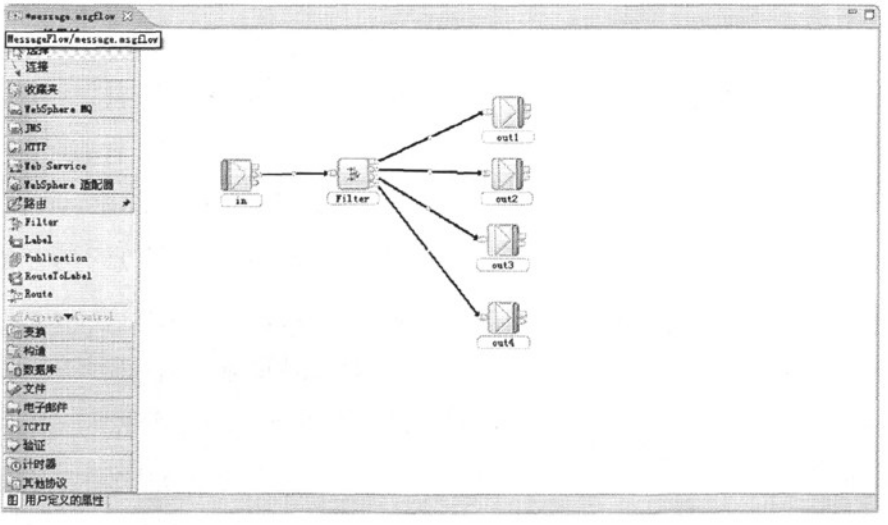

图 5.2　消息流示例

具体到物流园区业务系统整合平台，以第三方物流公司业务开展为例，第三方物流公司专门从事物流工作，为企业提供运输、仓储等一系列物流服务。第三方物流公司进行业务联动时，一方面需要明确所运输货物种类、数量等货物数据，同时要了解仓库中储位信息，以便提前清理储位，准备接受货物。这就同时需要运输管理系统与仓储管理系统的数据，并按照货物储存业务规则进行匹配。货物储位匹配消息流可以创建如图 5.3 所示。

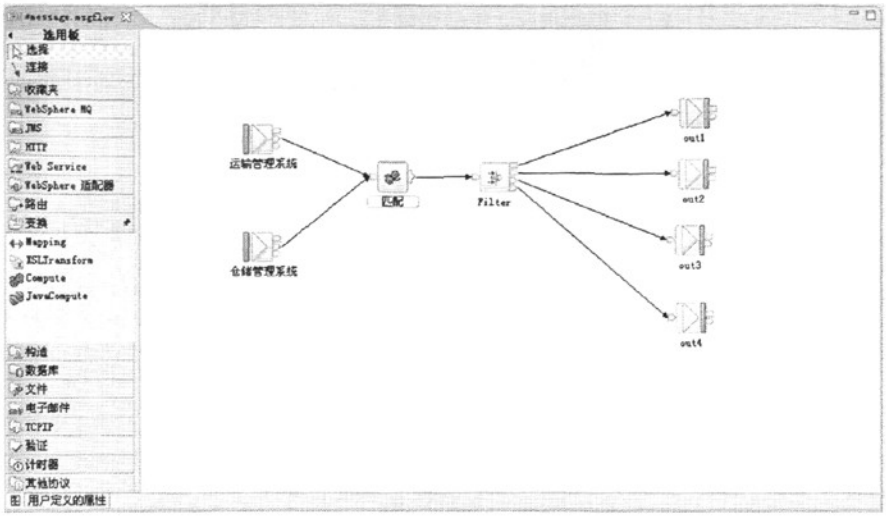

图 5.3 货物储位匹配消息流

WebSphere Message Broker Toolkit 是可视化开发工具，建立消息流可以实现“所见即所得”，而且不需要进行代码编写，十分方便，其实施的关键在于要弄清业务交互规则，才能建立流程清晰准确的信息流。对整合平台中的所有的消息流都可依次进行建立。

2、启动代理服务器

建立完消息流后，需要建立并启动 MB 代理服务器。MB 可以提供代理服务器功能，同样在 MB Toolkit 中进行设置。如图 5.4 所示，MB 提供域连接服务，MB 需要将所建立的消息流打包成 bar 包进行部署，这具有更强的灵活性，更高的性能。前述的第三方物流企业货物储位匹配消息流在建立后，被打包成 bar 包，然后部署到 MB 建立的代理服务器上，所有的消息流部署在代理服务器上后，成为 ESB 集成的外部服务。搭建完成后的平台基础架构如图 5.5 所示。

5.3 业务功能整合实现研究

业务整合包括服务封装和服务注册两个步骤，对其实现分别进行研究。

5.3.1 服务封装

搭建起平台基础架构后，需要对园区业务系统进行服务封装，将业务系统提供系统的不同功能单元封装为 Web 服务逻辑单元，然后发布到整合平台，进行服务注册后，只对外暴露标准服务接口，这种模式既可以保证数据信息的完整性又可以保证数据信息访问的安全性，保护了企业业务逻辑资产安全。

园区业务系统相互异构，各个业务系统情况各异，有些业务系统可以提供应用程序

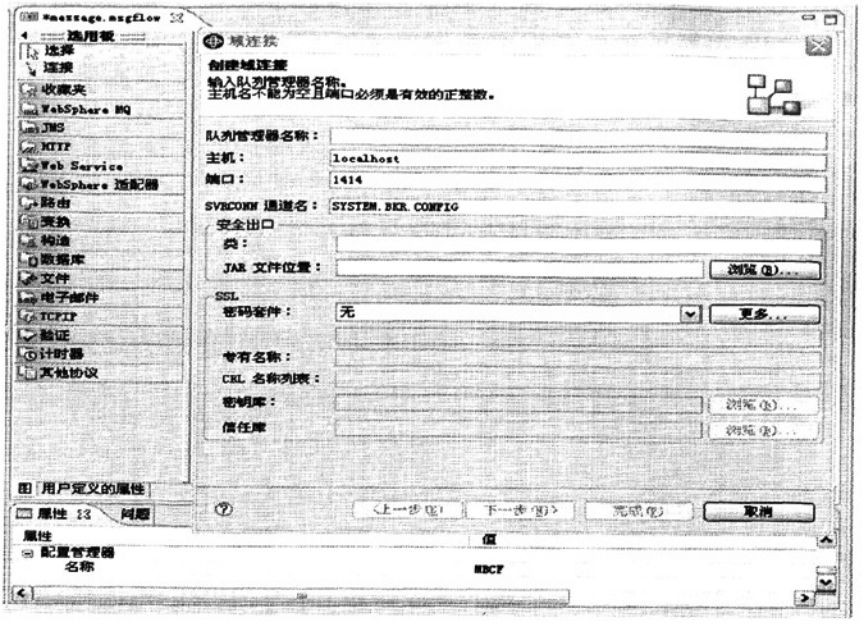

图 5.4 MB 域代理服务

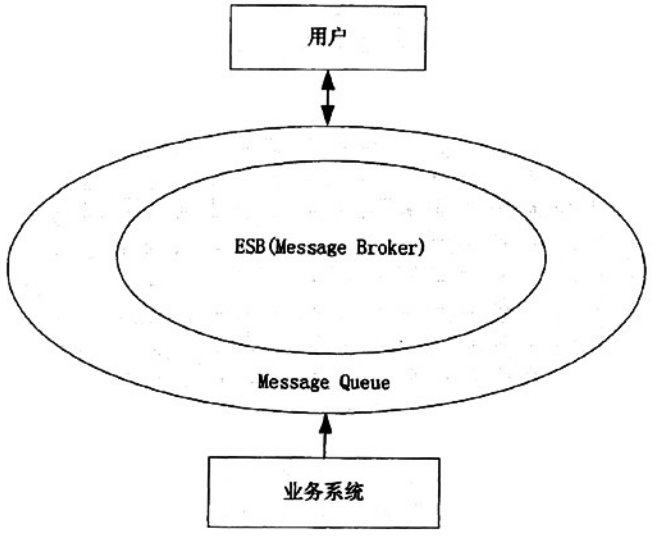

图 5.5 平台基础架构

编程接口函数 API，有些业务系统只能提供业务数据支持，对那些对这两种情况分别进行研究。

（1）可提供应用接口的业务系统服务封装

这种业务系统在建设时预留了应用程序编程接口函数 API，允许用户根据需求进行系统扩展，但是，出于系统安全性考虑，业务数据不能随便被外系统使用或是需要特定接口来实现调用，因此预留的接口有其使用规则和局限性。限于以上情况，对预留接口

的遗留业务系统，服务封装采取把接口封装成服务并传输到业务系统整合平台上进行注册，通过平台为用户提供统一的服务接口，制定调用规则。这种服务封装方式可以解决接口限制，也有利于安全访问数据以及拓展业务。

（2）没有提供应用接口的业务系统服务封装

这种业务系统建成时间较长，建设时没有预留接口，进行业务整合时，只能提供业务数据共享。对于此类业务系统，对其业务共享数据进行服务封装，然后处理方法同第一种情况一致，封装后的服务发布到业务系统整合平台上供用户调用。这种方式需要编写接口程序，然后把接口程序封装成服务，处理过程与第一种情况类似。

进行服务封装需要合适的现在应用比较广泛的新一代 Web Services 平台有 XFire 与 Axis2，两者都是开源产品，都支持一系列 Web Services 的新标准，相比起来 XFire 性能更强，响应时间快，简单易用，可以通过提供简单的 API 支持 Web Services 各项标准协议，快速地开发 Web Services 应用。因此这里使用 Xfire 进行服务封装。

使用 Xfire 进行服务封装时，服务器端的内容需要创建服务接口、服务接口的实现类，最后要配置 Xfire 要求的 Web Services 定义描述文件 services.xml。

以一个简单的仓储管理系统为例，它没有提供外部接口，表 5.1 为其库内货物信息表，具体数据项定义如下：

表 5.1 库内货物信息表（KNXX）

字段名	数据类型（长度）	字段说明
RKDH	int(8)	入库单号
HWMC	varchar(20)	货物名称，主键
HWLB	varchar(20)	货物类别
HWSL	decimal(20,0)	货物数量
HZMC	varchar(20)	货主姓名
CWQH	decimal(10,0)	储位区号
CWJH	decimal(10,0)	储位货架号
HZDH	decimal(15,0)	货主电话
RKSJ	char(8)	入库时间
BCJZR	char(8)	储存截止日期
…	…	…

现需要进入此系统内，根据货物名称查询出相关信息。要完成此接口调用，根据 Xfire 规则，首先创建服务接口类，其内容如下：

```
public interface    KNHWXXCX
{
    /*获取货主名称*/
    public    String getHZMC(int rkdh);
    /*获取货主电话*/
    public    String getHZDH(int rkdh);
    ....
}
```

然后创建服务接口类的实现类，内容如下：

```
public class KNHWXXCXImpl implements KNHWXXCX
{
   public String getHZDH(int rkdh)
   {
       String HZDH = "";
       String sql = "select HZDH from knxx where rkdh = '"+rkdh+"'";
       try
       {
           HZDH = DataUtil.getMessage(sql,"HZDH");
       } catch (Exception e)
       {
           e.printStackTrace();
       }
       return HZDH;
         ....
}
```

此查询服务需要对数据库进行操作，需要分别创建一个数据连接类和数据处理类，数据连接类内容如下：

```
    public static Connection getCon()
    {
      ...
       Connection con = null;
           try
       {
           Class.forName("com.mysql.jdbc.Driver");
           con = DriverManager.getConnection(URL, username, password);
       } catch (Exception e)
       {
           System.out.println("连接数据库");
           e.printStackTrace();
       }
```

```
        return con;
    }
    public static String getString(String tmp)
    {
        return tmp.substring(tmp.indexOf('/')+1);
    }
}
```

数据处理类：

```
public class DataUtil
{
    public static String getMessage(String sql,String type) throws Exception
    {
        String message = "";
        Connection con = null;
        con = DataCon.getCon();
        Statement stmt = con.createStatement();
        ResultSet rs = null;
        rs = stmt.executeQuery(sql);
        while (rs.next())
        {
            message   = rs.getString(type).toString();
        }
        return message;
    }
}
```

最后创建 services.xml 文件，具体内容为：

```
<?xml version="1.0" encoding="UTF-8" ?>
<beans xmlns="http://xfire.codehaus.org/config/1.0">
    <service>
        /*服务名称*/
          <name>KNHWXXCXService</name>
        /*服务对应的接口类*/
          <serviceClass>service.KNHWXXCX</serviceClass>
        /*服务接口类的实现类*/
          <implementationClass>impl.KNHWXXCXImpl</implementationClass>
           <style>wrapped</style>
           <use>literal</use>
           <scope>application</scope>
    </service>
</beans>
```

以上步骤完成后，需要使用 XFire 进行 Web 服务封装，生成 WSDL 文件，具体内容如下所示：

```
<?xml version="1.0" encoding="UTF-8" ?>
<wsdl:definitions targetNamespace="http://service" xmlns:tns=http://service
xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:soap12=http://www.w3.org/2003/05/soap-envelope
xmlns:xsd=http://www.w3.org/2001/XMLSchema
xmlns:soapenc11=http://schemas.xmlsoap.org/soap/encoding/
xmlns:soapenc12=http://www.w3.org/2003/05/soap-encoding
xmlns:soap11=http://schemas.xmlsoap.org/soap/envelope/
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
<wsdl:types>
    <xsd:schema xmlns:xsd=http://www.w3.org/2001/XMLSchema
    attributeFormDefault="qualified" elementFormDefault="qualified"
    targetNamespace="http://service">
    <xsd:element name="getHZMC">
        <xsd:complexType>
            <xsd:sequence>
              <xsd:element      maxOccurs="1"      minOccurs="1"      name="in0"
                  type="xsd:int" />
            </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
        <xsd:element name="getHZMCResponse">
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:element  maxOccurs="1"  minOccurs="1"  name="out"
                      nillable="true" type="xsd:string" />
                </xsd:sequence>
            </xsd:complexType>
          </xsd:element>
          ...
</wsdl:types>
<wsdl:message name="getHZMCRequest">
    <wsdl:part name="parameters" element="tns:getHZMC" />
</wsdl:message>
<wsdl:message name="getHZMCResponse">
    <wsdl:part name="parameters" element="tns:getHZDHResponse"
</wsdl:message>
    ...
<wsdl:portType name="KNHWXXCXServicePortType">
<wsdl:operation name="getHZMC">
<wsdl:input name="getHZMCRequest" message="tns:getHZMCRequest" />
<wsdl:output name="getHZMCResponse" message="tns:getHZMCResponse" />
</wsdl:operation>
    ...
```

```
</wsdl:portType>
<wsdl:binding name="KNHWXXCXServiceHttpBinding"
type="tns:KNHWXXCXServicePortType">
<wsdlsoap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http" />
<wsdl:operation name="getHZMC">
<wsdlsoap:operation soapAction="" />
<wsdl:input name="getHZMCRequest">
<wsdlsoap:body use="literal" />
</wsdl:input>
<wsdl:output name="getHZMCResponse">
<wsdlsoap:body use="literal" />
</wsdl:output>
</wsdl:operation>
    ...
</wsdl:binding>
<wsdl:service name="KNHWXXCXService">
<wsdl:port name="KNHWXXCXServiceHttpPort"
binding="tns:KNHWXXCXServiceHttpBinding">
<wsdlsoap:address
location="http://192.168.117.74:7002/HWXXCX/services/KNHWXXCXService" />
</wsdl:port>
</wsdl:service>
</wsdl:definitions>
```

WSDL 文件生成后，服务封装完成，下一步要做是进行服务注册。

以上处理方法适用于不能提供接口的业务系统服务封装，对于可以提供接口的业务系统，创建其服务接口实现类和 services.xml 文件，最后生成 WSDL 文件，具体处理方法类似，不再赘述。

5.3.2 服务注册

在对业务系统完成服务封装后，把封装后的逻辑服务单元送入 MQ 中，MQ 负责把服务传输进入 MB，因为 MQ 本身提供 API 接口支持很多主流平台，所以平台服务注册功能可以通过把服务单元送入 MQ 中从而完成服务注册，即实现消息与 MQ 绑定。

MQ 中信息是以消息的形式进行传输的。消息本质上就是一段数据，是应用程序之间传递的信息载体。

将逻辑服务单元送入 MQ 中，需要以下步骤：设置连接环境属性；连接到队列管理器；打开发送方消息队列；放置消息到发送方队列上；创建消息缓冲区；放置消息到接收方消息队列；关闭队列和队列管理器对象。

下面给出代码实现：

```
/*设置 MQ 环境属性*/
MQEnvironment.hostname = hostName;
MQEnvironment.channel = channel;
MQEnvironment.CCSID = 1381;
MQEnvironment.properties.put(MQC.TRANSPORT_PROPERTY,
MQC.TRANSPORT_MQSERIES);
MQEnvironment.disableTracing();
MQException.log = null;
/*连接到队列管理器*/
qMgr = new MQQueueManager(qManager);
Int openOptions = MQC.MQOO_OUTPUT | MQC.MQOO_FAIL_IF_QUIESCING;
/*打开队列*/
mqQueue = qMgr.AccessQueue(qName, openOptions);
/*设置放置消息选项*/
MQPutMessageOptions mpmo = new MQPutMessageOptions();
MQMessage mqMessage = new MQMessage();
/*写入消息*/
mqMessage.writeString(new Java.util.Date().toString());
mqQueue.put(mqMessage, mpmo);
}
finally {
  try {
/*关闭队列和队列管理器对象*/
if (mqQueue != null) {
    mqQueue.close();
}
if (qMgr != null) {
    qMgr.close();
    qMgr.disconnect();
    }
```

5.4 数据解析传输实现研究

服务封装后，以 WSDL 文件形式传输到 MB 上，完成服务注册，MB 对 WSDL 文件进行解析，获得服务封装的应用调用接口，调用用户所需业务数据，并根据 MB 中已设定好的信息流，通过 MQ 输出到有服务调用要求的用户端。

对 WSDL 文档进行解析仍可以使用 Xfire 进行，因为 XFire 支持从 Java 文件转换到 WSDL 和从 WSDL 转换到 Java 文件。

对 getHZMC 方法解析完后的调用接口为：

```
//请求
@XmlAccessorType(XmlAccessType.FIELD)
    @XmlType(name = "",  propOrder = {"in0"})
@XmlRootElement(name = "getHZMC")
public class GetHZMC
{
        @XmlElement(required = true,  nillable = true)
        protected String in0;
    public String getIn0() {return in0;}
    public void setIn0(String value) {this.in0 = value;}
}
//响应
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "",  propOrder = { "out"})
@XmlRootElement(name = "getHZMCResponse")
public class GetHZMCResponse
{
        @XmlElement(required = true,  nillable = true)
        protected String out;
    public String getOut() {return out;}
    public void setOut(String value) {this.out = value;}
}
```

解析后的数据需要通过 MQ 传输给用户。从 MQ 上获得消息，需要以下步骤：设置环境属性；连接到队列管理器；打开接收方消息队列；创建消息缓冲区；放置消息到缓冲区上；放置消息到接收方消息队列；关闭队列和队列管理器对象。

具体实现代码如下：

```
/*设置 MQ 环境属性*/
MQEnvironment.properties.put(MQC.TRANSPORT_PROPERTY,
MQC.TRANSPORT_MQSERIES);
MQEnvironment.CCSID = 1381;
MQEnvironment.hostname = hostName;
```

```
MQEnvironment.channel = channel;
/*连接到队列管理器*/
MQQueueManager qMgr = new MQQueueManager(qManager);
/*设置打开选项*/
Int openOptions = MQC.MQOO_OUTPUT | MQC.MQOO_FAIL_IF_QUIESCING;
/*打开队列*/
mqQueue = qMgr.accessQueue(qName, openOptions);
MQGetMessageOptions gmo = new MQGetMessageOptions();
gmo.options = gmo.options + MQC.MQGMO_SYNCPOINT;
gmo.options = gmo.options + MQC.MQGMO_WAIT;
gmo.options = gmo.options + MQC.MQGMO_FAIL_IF_QUIESCING;
gmo.waitInterval = 3000;
/*打开队列*/
MQMessage inMsg = new MQMessage();
mqQueue.get(inMsg, gmo);
String msg = inMsg.readString(inMsg.getMessageLength());
System.out.println("this message    is " + msg);
qMgr.commit();
}
finally {
  try {
/*关闭队列和队列管理器对象*
 if (mqQueue != null) {
   mqQueue.close();
}
 if (qMgr != null) {
   qMgr.close();
   qMgr.disconnect();
  }
```

逻辑服务单元送入 MQ 以后，用户可以根据 MB 中制定的业务流程，在进行具体业务操作时，调用逻辑服务单元。数据在整合平台中起到支撑业务的关键作用。

5.5 本章小结

本章主要是对基于SOA的物流园区业务系统整合平台的实现进行了研究。使用IBM WebSphere MQ 和 MB 搭建了平台基础架构，并在其基础上进行了业务整合和数据整合功能实现研究。

总结与展望

总结

物流业务系统整合平台是物流园区信息化建设的关键性工程，而面向服务体系架构SOA是一种设计模式，其目的是实现异构业务系统松耦合度整合，并最大限度重用业务服务以提高适应性和效率，为企业级应用系统的开发提供了良好的结构框架和崭新的应用模式。

本文结合我国物流信息化现状，引入SOA设计思想，主要完成了以下工作：

（1）梳理了SOA设计模式及其相关实现技术Web Services、XML、ESB等技术规范，对业务系统整合模式进行了研究，并选用企业服务总线 ESB 作为平台核心部件，设计了物流园区业务系统整合平台三层体系架构。

（2）平台使用了Web服务作为逻辑服务基本单元，以XML文档为统一数据格式，保证了异构系统间的松耦合度和转换方式的通用性，并对平台中数据可靠传输、异构数据交换、Web服务封装、服务注册和服务传输等平台运行所需关键技术进行了研究，给出了解决方法。

（3）选用IBM Message Broker和Message Queue搭建了平台实现基础架构，使用Xfire开源框架，实现了Web服务封装、服务注册，完成了数据在平台中传输以及转换。

基于SOA的物流园区业务系统整合平台设计具有以下特点：

（1）结构简单，易于实现：平台功能按照层次划分，清晰明了，整体规划完善。在不改变现有业务系统的前提下，平台可以将业务系统及应用封装为服务，并由平台提供服务给用户，因此不需要改变物流业务系统体系结构，重用能力强。

（2）松散耦合：平台基于SOA设计，将服务使用者和服务提供者在服务实现和客户如何使用服务完全隔离开来。这就意味着，服务请求者不知道提供者实现的技术细节，也可以不知对方所在的位置，同时也可以在完全不影响服务使用者的情况下进行修改。

（3）信息高度共享并与底层数据格式无关：各个业务系统的数据以服务的方式暴露出来并在平台中注册，用户可以通过平台搜索查询服务和使用服务，而且数据交换格式用的是XML格式，数据交换简单方便。

（4）接口统一，标准化：整合平台有良好而又基于标准的接口，服务的描述易于

理解，而且标准一致。

（5）渐进式整合方式：业务系统的相关功能封装为服务，服务可以重用，未来新的整合需求可以通过组合已有服务来完成，整个整合方式呈现渐进特点。

基于 SOA 的物流园区业务系统整合平台体系架构设计已经在哈尔滨公路货运主枢纽信息系统建设项目一期工程—“黑龙江龙运物流园区业务系统整合平台”中得到验证和应用。

展望

由于物流业务涉及社会生活的方方面面，业务功能复杂，而基于 SOA 应用经验的不足，以及课题来源项目经费、时间限制，本文的研究成果不可避免的会有不足之处，在本文工作总结的基础上，对下一步研究工作归纳如下：

（1）可以将物流活动涉及到的政府主管部门业务系统整合进来，以实现物流业务一体化运作，增大业务整合力度。

（2）除了选用 ESB 以外，可以对基于 SOA 的业务系统整合模式的多种解决方案进行研究，做到随业务情况不同解决方案不同。

（3）研究完善平台业务服务层功能，规划补充完善日志管理、事务管理等。

（4）平台实现研究基于 IBM MQ 和 MB 进行，MQ、MB 都是商业中间件，考虑到各地各种类型物流园区条件限制，可以研究基于 Jboss 等开源产品的实现方法。

参考文献

[1] 国务院.国务院关于印发物流业调整和振兴规划的通知[EB/OL], http://www.gov.cn/zwgk/2009-03/13/content_1259194.htm,2009.3.10/2009.3.15

[2] 重庆市交通委员会,重庆市交通运输业现代物流发展规划（2006-2020 年）[Z],2005.12：96

[3] 宋健.基于 SOA 的第四方物流管理系统的关键技术研究与实现[D].天津：天津大学,2007

[4] 卢云帆.我国物流园区信息化现状分析[J].商品储运与养护,2007,29(6)：13-14

[5] 钟艳玲.物流信息平台的建设要点[J].广东科技,2001,3(2)：12-15

[6] Freight Information Real-Time System for Transport (FIRST)[EB/OL], http://ops.fhwa.dot.gov/freight/documents/first.pdf 2002.3/2008.9.20

[7] 陈金川.日本物流企业中的信息系统应用[EB/OL], http://www.siod.cn/siod_article/58/4832.html，2007.12.1/2008.11.24

[8] 徐红梅. 吉林省共用物流信息平台系统设计及关键技术研究[D]，长春：吉林大学,2008

[9] 王亚.物流业：有待突破信息采集瓶颈[J].中国自动识别技术,2007,6：85

[10] 王东.城市现代物流信息平台规划与和谐制造研究[D].上海：上海交通大学，2003

[11] 张教赟.浅谈现代物流信息技术的发展与应用[J].综合管理,2009,1(2)：169-170

[12] 何振.物流信息技术的集成应用研究[J].内蒙古科技与经济，2008,162(8)：74-75

[13] 秦天保.现代物流信息系统之技术架构[J].上海海事大学学报，2005,26(4)：57-58

[14] Sujatha Kumar， Vijay Dakshinamoorthy. An Empirical Analysis of the Impact of SOA Adoption on Electronic Supply Chain Performance[J]. IEEE Communication Magazine, 2002:102-114

[15] Eric Newcomer，Greg Lomow 著，徐涵译.Understanding SOA With Web Service 中文版[M].北京：电子工业出版社，2006：221-270

[15] JieWu 著，高传善译.Distributed System Design[M].北京：北京机械工业出版社，2001：10-15.

[16] OASI. SOA Reference Model TC[EB/OL],

http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=soa-rm, 2006.8.2/2008.10.8.

[17] The Open Group. Service-Oriented Architecture [EB/OL], http://opengroup.org/projects/soa/doc.tpl?gdid=10632, 2006.6.8/2009.2.9

[18] Object Management Group. Service Oriented Architecture SIG[EB/OL], http://soa.omg.org/, 2006.3.24/2008.12.20

[19] XML.com. What Is Service-Oriented Architecture[EB/OL], http://www.xml.com/pub/a/ws/2003/09/30/soa.html, 2003.9.30/2009.2.15

[20] Bill StArnaud, Andrew Bjerring, Omar Cherkaoui, etal. Web Services Architecture for User Control and Management of Optical Internet Networks[J]. Proceeding of IEEE, 2004, 92(9):1490-1500

[21] 张巧霞, 范黎林. 基于SOA实现企业应用集成[J]. 微计算机信息, 2007, 23(10-3): 5-13

[22] 柴晓路, 梁宇奇. Web Services技术、架构和应用[M]. 北京: 电子工业出版社, 2003

[23] 吴家菊, 刘刚, 席传裕. 基于Web服务的面向服务（SOA）架构研究[J]. 现代电子技术, 2005, 28（14）: 71-72

[24] Carey M J. SOA What[J]. Computer, 2008, 41(3):92-93

[25] Paul Patrick. Impact of SOA on enterprise information architectures [J]. Proceedings of the 2005 ACM SIG-MOD international conference on Management of tata. 2005:844-848

[26] 徐罡, 黄涛. 分布应用集成核心技术研究综述[J]. 计算机学报, 2005, 28(4):434-445

[27] Tsalgationdou A, Pilioura T. An overview of standards and related technology in web services[J]. Distributed and Parallel Database, 2002, 12(2-3):135-162

[28] Vinoski S. Integration with Web Services[J]. IEEE Internet Computing, 2003, 7(6):73-80

[29] 李德生, 王海洋. 一种将业务规则与BPEL有效集成的方法[J]. 计算机应用, 2005, 25(11): 2705

[30] 黄强. 基于SOA的电子政务应用集成研究[D]. 武汉: 武汉理工大学, 2008

[31] IBM. 理解面向服务的体系结构中企业服务总线场景和解决方案[EB/OL]. http://www.ibm.com/developerworks/cn/webservices/ws-esbscen/,2006.7.1/2009.2.3

[32] 曾文英,赵跃龙,齐德昱. ESB 原理、构架、实现及应用[J]. 计算机工程与应用,2008,44(25): 225-228

[33] IBM. 企业服务总线解决方案剖析[EB/OL]. http://www.ibm.com/developerworks/cn/webservices/ws-esbl/,2006.4.10/2009.2.4

[34] 童鑫, 基于SOA的企业服务总线的研究与设计[D]. 长沙：湖南大学,2008

[35] 孙承芳. 物流园区信息平台规划研究[D].西安：长安大学,2006

[36] 欧阳文霞. 物流信息系统[M].北京机械工业出版社，2004

[37] 卢致杰，SOA体系设计方法研究[J].工业工程，2004，7（6）：83-87

[38] 谢小轩，张浩，夏敬华，王坚，李琦.企业应用集成综述[J].计算机工程与应用，2002,(22)：2-4

[39] 周竞涛，王明微．Web Services Vs EAI：比较与分析[EB/OL] http://www.ibm.com/developerworks/cn/webservices/ws-wseai/index.html,2002.10.1/2009.3.17

[40] 丁昭华.基于ESB的企业应用集成技术研究与应用[D] 长沙：中南大学,2007

[41] GB/T18354—2001, 物流术语[s]. 北京：中国标准出版社，2006

[42] 杨志明.几种常见软件体系结构模型的分析[J].计算机工程与设计，2004，25(8)：1326-1328

致 谢

首先感谢我的导师赵祥模教授，本论文的撰写，得到了赵老师的精心指导和悉心教诲。赵老师严谨求实、忘我工作以及一丝不苟的治学作风激励着我，时刻提醒自己要珍惜时间，努力学习。他渊博的知识、正直的品格和谦虚宽厚的生活态度将使我受益终身。

感谢杜瑾老师对我论文的无私指导。感谢袁绍欣老师、徐志刚老师平时对我在学习和生活上的指导和帮助，他们踏实认真的工作态度、灵活敏锐的学术思维深深地熏陶着我。感谢信息工程学院各位老师的谆谆教诲，他们教给了我丰富的专业知识，更让我懂得了许多人生哲理，他们的教诲将伴随我走过一生。

感谢论文评审和答辩小组的老师们，感谢他们在百忙之中对论文的审阅，并提出了宝贵意见。

特别感谢北京交通运输部公路科学研究院公路交通发展研究中心的顾敬岩主任、杨英俊博士、吴金忠高工以及项目组的杨复峰、唐小淳等同事，本文的撰写离不开他们的大力帮助和支持。

感谢在论文写作过程中给过我支持和帮助的李卓文、李伟兵等同学。感谢在三年读研期间给予我支持与帮助的同学和朋友们。

最后，再次向关怀、帮助我完成学业的老师、同学们表示深深的感谢！