分类号 ..... 密级 .....

U D C ..... 编号 .....

CENTRAL SOUTH UNIVERSITY

硕士学位论文

论文题目 基于ESB的企业应用集成技术研究与应用

学科、专业 计算机软件与理论

研究生姓名 丁昭华

导师姓名及专业技术职务 李建华 教授

2007年5月

摘 要

一个典型的电子政务系统是涉及到众多职能部门在网上联合开展行政服务的业务系统，如何将这些业务系统有机地集成在一起发挥最大的功效是一个关键问题。随着企业应用集成（EAI）技术的发展，出现了面向服务的体系结构（SOA）。在 SOA 中，服务之间通过统一的业务服务接口进行通信，它可以达到服务的可重用性。为了处理复杂集成的挑战，通过选择企业服务总线（ESB）作为 SOA 系统中消息传输与服务交互的主干道，可以为未来业务的变化奠定良好的基础架构。

本文首先介绍了 EAI 的相关概念和研究内容，然后在 EAI 的多个层次上重点对 SOA 和 Web Services 技术进行了剖析和研究，提出了一种基于 ESB 的 EAI 集成框架－CreatorSOIF，它可以作为面向服务应用集成的一种实现模式。然后通过研究目前主流的 ESB 模型提出了 CreatorSOIF 中的一种基于服务执行引擎的 ESB 参考模型－SEE-ESB，为 SOA 的基础设施－ESB 的理解、设计和实现提供一个参考。接着对服务执行引擎中的服务适配网关模型以及消息路由器的体系结构进行了设计，并且在分析 ESB 内消息路由中介者和过滤机制的基础上，提出了一种基于优先级和一种基于规则的过滤机制，从而使得 ESB 中的消息路由机制更丰富。

最后，基于 CreatorSOIF 开发了电子政务并联审批原型系统，通过分析与扩展开源 Mule ESB 来集成政府各部门的审批业务系统。实践效果表明，通过本文提出的集成模式可以实现更为灵活多变的电子政务集成系统，有效地提高了政府服务效率。

关键词 企业服务总线，面向服务体系结构，企业应用集成，消息路由，电子政务

ABSTRACT

A typical E-government system includes various business systems of departments which can carry out administrative services together on the network. How to integrate them together to get the highest efficiency is a key problem. With the development of the enterprise application integration (EAI) technology, the Service-oriented architecture (SOA) emerged. In SOA, Services communicate with each other through standard business services' interfaces, and SOA's goal is to make services reusable. In order to handle complicate integration, by choosing ESB in the SOA system as the major trunk road for information transmission and service interaction, we can establish a good foundation structure for the future business change.

In the beginning, this paper introduces the relative concept of EAI and the research contents. Then, in multiple levels of EAI, it focuses on the analysis and research of SOA and Web Services technology, and proposes an integration framework-CreatorSOIF which is based on ESB. Next, through the study of the current mainstream ESB models, it brings forward an ESB reference model in CreatorSOIF, that is SEE-ESB, which is based on service execution engine, and that can provide a reference for understanding, design, and implementation of SOA infrastructure-ESB. After that, this paper gives out the working model of service fitting gateway in SEE as well as the architecture of message router. And also on analyzing the message routing mediators of ESB and the traditional filtration mechanisms, it puts forward two new filtration mechanisms, that are priority-based and rule-based filtration mechanisms, which can enrich the message routing mechanisms in ESB.

Finally, according to CreatorSOIF, we have developed a parallel examination and approval prototype system of E-government. Through analyzing and improving open source of Mule ESB, We have integrated various examination and approval business systems of governmental departments. The practice shows that, by using the method presented in this paper, we can establish a new set of dynamic E-government systems, and effectively improve the efficiency of government services.

KEY WORDS enterprise service bus, service-oriented architecture, enterprise application integration, message routing, E-government

目 录

第一章 绪 论

1.1 引言

1.1.1 研究的背景

信息化的发展在给企业带来难得机遇的同时，也给企业带来了新的挑战。巨大的投资为企业建立了众多的信息系统，以帮助企业进行内外部业务的处理和管理工作。但是这些信息系统可能由不同的品牌导入实施，只关注于各自领域内的数据与业务处理，由于缺少相应的接口标准和规范，它们各自为政，相互之间无法进行信息共享与业务集成，从而形成“信息孤岛”[1]。

同样的情况在电子政务系统中也存在。传统的电子政务系统由于缺少不同业务系统相互集成的技术，导致很多关键的信息被封闭在相互独立的系统中，部门间重复着冗余的工作，这直接导致了政务工作效率的降低和运营成本的上升。如何将这些政务服务系统有机地集成在一起，使用最少的资源，发挥最大的效力是一个关键问题。

随着企业规模的不断扩大，应用系统不断增加，企业迫切需要一种集成方法，将各种旧的应用系统和新的应用系统集成起来，这使得企业应用集成（Enterprise Application Integration，EAI）技术产生与发展起来。但是传统的EAI不能很好的满足企业集成的需求，传统的EAI往往使用如CORBA和COM等组件化技术进行分布式、跨平台的程序交互，系统整体的拓扑结构是较复杂的，组件的连接协议私有、非标准[2]。面向服务体系架构（Service-oriented Architecture，SOA）带来了一种新的集成思想，根据它可以构造出灵活的以服务为中心的架构。

作为一个很有前景的应用系统架构，SOA尚在不断的发展中，肯定存在许多有待改进的地方。第一：服务需要支持多种传输协议，但如何解决多种传输协议的转换；第二：服务是分散的，但如何对服务进行有效的流程控制；第三：服务是独立的，这样势必造成服务连接上需要耗费更多的负担，但如何解决服务之间的通讯性能；第四：传统的遗留系统都使用自己的专用的通信方式，但如何把MOM技术和Web分布式技术相结合；第五，服务是公开的，但如何解决服务之间的网络安全问题；第六：如何通过分层结构来实现服务的松耦合；第七：如何在松耦合和敏捷性之间达到更好权衡优化。以上几个问题，都是在SOA架构的实际软件项目中所需要解决的。

Web Services 技术是实现 SOA 的一种方式。Web Services 是完全基于标准的分布式通讯技术，它使用 XML 技术作为基础，使它独立于操作平台和编程语言，是真正意义上的中立技术[3]。使用 Web Services 技术可以使异构环境下的信息系

统能相互操作。但从整体的拓扑结构来看仍然比较复杂，每一个服务被发现和定位之后，就被耦合到另一个服务上。

早在二十世纪八十年代初，一些公司开始意识到企业应用集成的价值和必要性，点到点（Point-Point）集成[4]技术开始出现。在点到点的集成中，各应用系统之间通过各自不同的接口进行点到点的简单连接，从而实现信息和数据的共享。它适用于应用系统不多的情况，具有良好的可集成性，缺点是系统柔性差，耦合性高。

二十世纪八十年代末和九十年代初，在进行复杂的应用系统集成时，通常需要集成多个应用系统，这时人们选择使用集成中间件如 MOM 等来完成绝大部分集成工作。集成中间件是星型集成[4]体系结构的应用集成中心，所有需要集成的系统都和该中心相连，原来点对点集成中 n 个系统之间的 n×(n-1)/2 个点对点连接减少为 n 个连接。星型集成对应用系统的集成能力要求不高，整个系统柔性好、耦合性较低，其缺点是较为复杂，而且当集成的系统的数目较大时，集成中间件的负担会变得很大，容易形成瓶颈。

随着IT技术的发展和企业应用集成需求的增加，企业服务总线（Enterprise Services Bus，ESB）在2002年被正式提出，如图1-1所示。它继承了星型集成体系结构中将各个系统之间点对点的连接转化为多个系统与集成中心连接的理念，但在这种体系结构中，集成中心被扩展成可以分布在多个物理结点上的总线，从而有效解决了星型集成中单点失效和效率问题。同时ESB并不意味着仅仅是简单地将集成中心扩展成总线，它以成熟的消息中间件作为其物理消息传递基础，还提供消息路由、数据转换等各种EAI模式的支持，它是一种在松散耦合的服务和应用之间标准的集成方式。

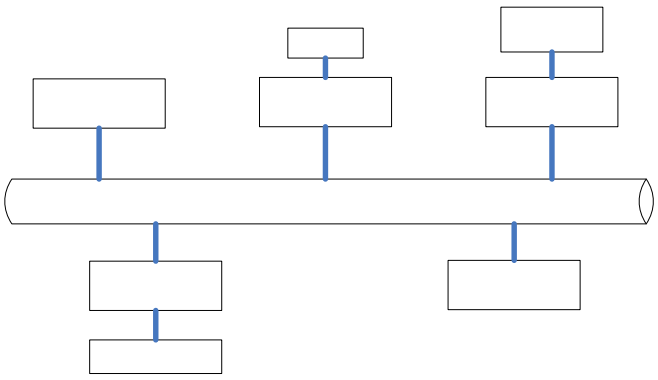


图1-1 企业服务总线

在 ESB 产生后的五年时间里，ESB 被越来越多的人所接受，很多企业和机构已经在生产中部署了 ESB，ESB 的效果得到了一定程度的校验，同时全球范围内对 ESB 的技术研究以及相关产品开发也进入了更为繁荣的阶段，众多的 ESB 供应商正在架构、连接性、易用性以及服务质量的保证等方面进行竞争[5]。

ESB的出现，为EAI和SOA注入了新的活力。ESB在SOA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的先进性表现在：作为SOA的一个最好切入点，ESB的出现改变了传统的软件架构，可以提供比传统中间件产品更为廉价的解决方案，同时它还可以消除不同应用之间的技术差异，让不同的应用服务器协调运作，实现了不同服务之间的通信与整合。

目前，ESB 正处于快速发展的时期，市面上出现了多种 ESB 产品，有专业集成提供商提供的 ESB 产品，也有开源组织提供的 ESB 产品，这些产品可以满足不同场合的集成需要。但是 ESB 是属于新生事物，各 ESB 产品在提供的功能不一样，实现的程度也并不统一，这有待于进一步发展和完善。

1.1.2 研究目的与意义

企业应用集成已经在电子政务、金融、电力、电信、航空等领域内得到了广泛的应用。然而传统的企业应用集成技术已经凸现其较大的局限性：如传统的 EAI 基本上是组件级的重用，基于组件的架构没有统一的标准；各个厂商都有各自不同的 EAI 解决方案，使用各种各样的中间件平台；集成后的系统更加复杂，降低了系统的重用性；连接和维护的费用增多；企业容易长期受到特定 EAI 提供商的限制等。

基于 SOA 进行企业应用集成是当前集成的主流[6]，ESB 是广义企业实现面向服务整合的关键。ESB 是 SOA 架构的核心，这是一个基于标准的、面向消息的、高度分布式的、具有智能性路由的系统整合平台[7]。ESB 的使用，正在使企业应用集成领域内发生着一场新的革命。所以，希望通过对 ESB 的研究来克服传统 EAI 的不足，从而最终实现面向服务计算环境下动态、灵活的分布式企业应用集成。

1.2 国内外研究现状

目前，国内外 ESB 市场上的所有厂商都一致认为，ESB 是一种实现 EAI 的新技术，可以帮助企业快速简便地实现 SOA。但是，对于如何实现 ESB，这些厂商存在非常大的歧见。不同的组织对 ESB 有着不同的理解，所以它们对 ESB 的研究重心也不一样，每个厂商都在根据自己的现状和目标来表明自己的观点。

1．ESB 在 SOA 中功能覆盖面的研究

专门以 ESB 为主要产品线的中间件厂商的 ESB 产品覆盖面较广，他们一般认为自己的 ESB 产品可以帮助用户实现所有与 SOA 有关的工作，其中也包括了

业务流程管理（BPM），其中有些厂商还把门户功能也加入其中。开源 ESB 中的 ServiceMix[8]已经提供了 BPEL 引擎－Apache PXE[9]，Mule ESB[10]目前是通过配置 XML 文档来编写业务流程，但是它将在新发布版本中通过提供 BPEL 引擎来支持 BPM。而原来提供 SOA 平台产品的软件厂商则认为 ESB 只是 SOA 的一部分，他们的 ESB 产品是其整个 SOA 平台软件中的一块，通常会把 BPM 作为另一个单独的产品，例如 BEA、甲骨文、IBM、中国的东方通等公司都推出了单独的 ESB 产品和 BPM 产品。

2．ESB 集成标准的研究

ESB 的总线方式可以是多样的，例如，总线可以是一个网络，而不是一个中心 Hub，甚至还可以直接通过点对点的方式。多样的方式是为了减少总线的压力，具体的形式可以很灵活。鉴于不同厂商对于实现 ESB 的体系架构不统一，因此无法通过统一的行业标准来进行互相操作。

到目前为止，还没有一个统一的得到大多数厂家认可 ESB 标准。JBI（Java 业务集成）[11]是一个由 SUN 公司提出的 ESB 集成标准，对它的一个实现即为基于 JBI 规范的 ESB。ServiceMix ESB 和 JBoss ESB 就是典型的基于 JBI 规范的 ESB 产品。一些原来不支持的 JBI 的 ESB，主要是开源 ESB，如 Mule ESB，都开始在自己的发布版本中逐渐增加对 JBI 的支持力度。但是这样的例子不多，JBI 并没有得到其他集成巨头如 IBM、BEA 等公司的支持，因此，目前业界基本上不看好 JBI 的发展前景，SUN 公司也有意减少对 JBI 标准支持的力度[12]。

另外，ESB 对服务的发布、发现、调用等功能的支持程度不统一。不同的软件提供商采用不同的机制来实现其服务的发布、发现与调用，因此开发人员开发的服务很难得到组织外开发人员的共享。IBM 因为以前就有成熟的 SOA 集成平台，所以它的 ESB 中提供对类似于 UDDI 的服务注册库的支持，但是开源 ESB 中的 Mule、ServiceMix 没有提供类似 UDDI 的功能。此外，ESB 中的消息路由种类多，实现各异，大多数 ESB 产品都只支持其中的一种或几种。智能路由是许多 ESB 产品都宣传其拥有的功能，但是它们都没有对智能路由进行定义，没有统一的标准。WS-*标准，如 WS-Routing，WS-Addressing 等路由协议以及 WS-Security 等安全协议本身处于不断的变化和完善中，ESB 产品对它的支持程度也各不一样。

3．ESB 产品是否与特定 Web 服务容器或平台紧密结合

ASB[13]是 BEA 公司的 ESB 产品，它基于 Web 服务技术、SOA 技术，是工业界第一个问世的架构于 SOA 和 Web 服务技术之上的 ESB 产品。ASB 与 BEA 的 J2EE 应用服务器 WebLogic Server 紧密结合，它为采用 WebLogic Server 的组织进行应用、服务集成提供了很大的方便。IBM 公司把 ESB 技术、Web 服务技

术、SOA 的思想融入到了其优秀的 J2EE 应用服务器 WebSphere 中，WebSphere 是一个产品组，包括应用服务器、业务整合、开发包及其无线技术等。而大多数开源 ESB 则没有规定与特定 Web 服务容器或平台的紧密结合，比如 IONA Celtix[14]、Mule 等，它们提供了多种部署模式，可以在任何一种 J2EE 或 JBI 容器中部署，包括使用基于 Spring 和/或 Apache Tomcat 项目的轻量级容器。

1.3 本文的研究内容

1．面向服务集成框架的研究

企业应用集成技术发展迅速，在当前面向服务的计算环境下，企业应用集成的主要对象将主要是粗粒度、松耦合和标准化的服务。本文通过对企业应用集成技术、面向服务的集成，以及企业服务总线的研究，使得能够基于这些技术或产品来提供一个灵活松耦合的可扩展的基础集成平台。

2．ESB 模型的研究

对现有主流的 ESB 模型进行研究，分析它们的优势与不足，并且提出一个基于服务执行引擎的企业服务总线参考模型，同时对该参考模型的核心－服务执行引擎进行研究。

3．SEE-ESB 参考模型中服务执行引擎的关键技术的研究

服务适配功能和消息路由功能是服务执行引擎的核心功能。这包括服务适配网关模型的设计，消息路由路径模型的分析、中介者的分类、消息路由器总体结构的设计，以及新的过滤机制的提出等。目前，ESB 中的消息路由研究是一个热点，这也是企业业务集成场景日益复杂的需要。

4．ESB 在电子政务并联审批应用场景中的研究与应用

推行电子政务是提高审批效率、增强审批的公开性、公正性和透明度，降低审批运行成本的必由之路。构建完善的网上审批系统是电子政务体系的有机组成部分，ESB 能将审批系统中的各子业务系统无缝集成起来，这对提高政务审批的工作效率，降低集成的复杂性和费用等都有明显的好处。

1.4 本文的组织结构

第一章，绪论：介绍了当前进行企业应用集成存在的问题，并且介绍了企业应用集成研究的背景，分析了国内外企业应用集成技术的研究成果，在此基础上提出了本文的研究内容。

第二章，面向服务的应用集成：首先介绍了企业应用集成的概念、研究成果、分类等，然后介绍 SOA 的概念、参考模型及关键实现技术，如 XML、SOAP、

UDDI、WSDL、XPath 和 XSLT 等。接着对 ESB 的定义、功能模型等进行了介绍。

第三章，基于 ESB 的企业应用集成框架 CreatorSOIF：本章首先提出了面向服务的集成框架的设计要求，然后提出了一种面向服务的集成框架 CreatorSOIF，并对它的总体结构、层次、关键技术以及实施流程给出了相应的分析，最后对 CreatorSOIF 应用于电子政务解决方案的优势进行阐述。

第四章，企业服务总线参考模型：在对现有主流 ESB 模型分析和研究的基础上提出了一种基于服务执行引擎的企业服务总线参考模型－SEE-ESB。接着对它的总体结构进行了分析，给出了服务执行引擎的结构模型，并对 SEE-ESB 的部署模型进行了分析。

第五章，服务执行引擎关键技术的研究：首先给出了服务执行引擎中服务适配网关的模型，接着设计了一个服务适配网关的例子。然后对消息路由进行概述，分析 ESB 中消息路由器工作模式，提出了一种消息路由器的总体结构，以及在对主流过滤机制分析的基础上提出了一种基于优先级和一种基于规则的过滤机制。

第六章，基于 Mule 的电子政务并联审批原型系统：分析了 Mule 的工作机制，以及针对电子政务系统中的并联审批流程，我们基于 CreatoSOIF 框架，并应用扩展的 Mule ESB 将各部门的子业务系统集成起来，从而实现一个完整的政务并联审批流程。

第七章，总结与展望：对本文进行总结，说明不足之处，并进行前景展望。

第二章 面向服务的应用集成

2.1 企业应用集成概述

2.1.1 企业应用集成的概念

最初企业应用集成的概念可以说是一个狭义上的EAI，仅指企业内部不同应用系统之间的互连，以期通过应用整合实现数据在多个系统间的同步和共享[15]。

伴随着EAI技术的不断发展，它所被赋予的内涵变得越来越丰富。现在EAI的概念已经扩展到业务整合的范畴，不仅要提供底层应用支撑系统间的互连，同时还要实现存在于企业内部应用与应用之间、本企业和其他合作伙伴间[16]端到端的业务流程的管理。它包括用户互动、应用整合、B2B整合、自动化业务流程管理、人工流程管理、企业门户以及对所有应用系统和流程的管理监控等方方面面。

EAI的目标就是集成和跨越不同应用系统的过程，同时使企业的员工、决策者和商业合作伙伴能够很容易地访问企业和客户的数据，而不必管这些数据在什么地方和哪个系统中[17]。此外，EAI更进一步的目标是集成跨企业和跨组织的信息和过程，其中包括实现传输流、数据流、信息流、过程流和B2B流。此处的流是以一种平滑、连续、实时或准实时的方式把数据发送给用户或系统的。

2.1.2 企业应用集成的研究内容

企业应用集成所涉及的技术与方法比较广泛，本文认为目前企业应用集成的研究主要集中在分布式系统技术、组件方法、中间件技术、软件体系结构方法和企业服务总线技术五个方面。

1．分布式系统技术的研究主要通过对分布式计算模型和中间层负载均衡的研究[18]来满足人们对系统集成的可扩展性、可靠性、适应性和性能方面的需求，同时，还要解决并发控制、事务和异常等问题。

2．组件方法的研究主要集中在组件的接口层次，包括接口命名、输入/输出参数类型、交互协议方面[4]，以及基于领域分析的软件组件选取、组件库的组织和检索、组件的组装[19]。Web 服务是目前组件研究的新方向。

3．中间件技术主要研究组件运行和交互的基础架构，为分布式应用集成提供通信中间件、组件运行容器及相关的服务。

4．软件体系结构方法的研究范畴包括软件体系结构语言及形式化、体系结构模型、特殊应用领域体系结构框架和基于体系结构的软件开发环境和工具。目前，面向服务体系架构是软件体系结构研究的新领域[20]。

5．企业服务总线为 SOA 中不同服务之间的集成提供了有力支持。目前，企业服务总线技术的研究主要集中在：服务发布和发现、服务的组合、语义 Web 服务、消息路由模式、路由安全性、服务 QoS、ESB 的事务性以及服务集成标准、业务流程编排等方面。随着企业服务总线技术的不断成熟，ESB 必将承担起更多更重要的集成任务，从而为我们带来了新的研究内容。

2.1.3 企业应用集成的分类

文献[4]依据分布式应用集成中集成点的不同，将集成层次从低层到高层分为传输机制、数据集成、接口集成和过程集成。传输机制层是分布式应用集成的基本层次，它提供两个或多个集成点间连接和移动数据的传输渠道，前提是在传输层上连接多个系统，其方法包括 IP、FTP 及特定的通信中间件。数据集成能够从应用系统和数据存储中抽取、插入数据和元数据，同时还必须解决应用句法和应用语义等问题。接口集成主要针对业务逻辑层，它允许应用系统间的业务逻辑共享。接口集成的核心是使用分布式组件封装应用系统的业务逻辑，通过远程方法调用业务逻辑。过程集成实现的是面向过程的集成。过程集成的对象不是物理实体（如数据或组件），而是过程实体，通常表示为逻辑实体。过程集成的逻辑表示需要映射到物理实体。

文献[21]按集成的范围或广度划分，企业应用集成可分为松集成和全面集成、横向集成与纵向集成、企业内集成和企业间集成。松集成是指两个系统之间仅仅交换信息，而不管对方是否能够解释这个信息，或者说他们的集成仅仅是语法层的集成，而不是语义层的集成。全面集成的含义是：（1）每个系统的定义仅仅由本系统知道，另外一个系统不知道其他系统的含义；（2）两个系统共同为完成一个任务作贡献；（3）两个系统对于它们之间交换的信息有相同的定义。横向集成是指从产品需求到产品发运业务过程的集成，包括物理和逻辑的集成。纵向集成是指企业内部不同管理层之间的集成，分为企业级集成、业务单元级集成和操作级集成。企业内集成是指一个企业内部业务过程的集成，而企业间集成就意味着一个企业的业务过程与另外一个企业的业务过程集成，或者不同的企业共享一些业务过程[22]。

文献[23]从集成深度的意义上将市场上主流的企业应用集成模式划分为面向信息的集成，面向过程的集成和面向服务的集成。面向信息的集成模式聚焦于接口层次的应用和系统间的数据转化和传输，它给了大多数组织一种风险较低的切入企业应用集成的方式，其主要优势是较低的成本。它又可以划分为三种类别：数据复制、数据聚合以及接口集成。面向过程的集成按照一定的顺序实现过程间的协调并实现数据在过程间的传输，其目标是通过实现企业相关业务过程的协调和协作实现业务活动的价值最大化，还可以减少错误。总的来说，它是一种过程

流集成的思想。面向服务的集成可以实现动态的应用集成和大范围的业务逻辑共享，这种目标是通过整合业务层服务来实现的，其中的关键是要将面向服务、接口以及过程的集成模型加以整合利用，同时利用标准的 Web 服务接口对现存应用进行封装。

2.2 面向服务体系结构概述

目前，企业大都关心并正在处理两个问题：迅速改变的能力和降低成本的要求。为了保持竞争力，企业必须快速地适应内部因素（如兼并和重组）或外部因素（如竞争能力和顾客要求），这需要经济而灵活的 IT 基础设施来支持企业[24]，相应地应用程序彼此进行通信的方式也发生了典型变化，为迎合这种需求，SOA 应运而生。从技术上讲，SOA 并不是一个新概念。早在 1996 年，Gartner 就提出了面向服务架构的概念。Gartner 于 2002 年预测，到 2008 年，SOA 将成为占有绝对优势的软件工程实践方法，它将结束传统的整体软件体系架构长达 40 年的统治地位，届时，将有 70%的企业在进行企业 IT 建设时会转向 SOA[25]。

近年来，随着 Web Services 等相关标准的出现和日趋成熟，SOA 开始从概念走向实用。Web 服务的发展使 SOA 成为主流，反过来，SOA 的实践架构又促使 Web 服务的应用更为广泛。

2.2.1 SOA 的概念

SOA 是英文 Service-Oriented Architecture，即面向服务架构的缩写。SOA 是一种新型的软件体系架构模式，它是在计算环境下设计、开发、应用、管理分散的服务单元的一种规范，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[26]，它可以根据需求通过网络对松散耦合的粗粒度服务进行分布式部署、组合和使用。这些服务是自包含的，具有定义良好的接口，允许服务的使用者了解如何与其进行交互。SOA 的目标在于让 IT 变得更有弹性，以便更灵活、更快地响应不断改变的企业业务需求[27]。

文献[6]将 SOA 定义为：“它本质上是服务的集合。服务间彼此通信，这种通信可能是简单的数据传送，也可能是两个或更多的服务协调进行某些活动。服务间需要某些方法进行连接。所谓服务就是精确定义、封装完善、独立于其他服务所处环境和状态的函数。”

文献[28]将 SOA 定义为：“按需连接资源的系统。在 SOA 中，资源被作为可通过标准方式访问的独立服务，提供给网络中的其他成员。与传统的系统结构相比，SOA 规定了资源间更为灵活的松散耦合的关系。”

文献[29]对 SOA 的定义如下：“面向服务的体系结构是一种用于创建企业 IT

体系结构的体系结构样式，利用了面向服务的原则来实现业务和支持业务的信息系统之间更为紧密的关系。”

目前 SOA 并没有一个统一的、标准的参考模型来给出其定义。但从上述的定义中可以看出 SOA 的几个关键特性：一种粗粒度、松散耦合服务架构，服务之间通过简单、精确定义接口进行通讯，不涉及底层编程接口和通讯模型。

粗粒度：SOA 总体上采用粗粒度服务接口，也就是说暴露在整个系统外部的服务尽量使用粗粒度的接口。而在企业系统架构内部通常采用相对较细粒度的服务接口。采用粗粒度的服务，可以减少服务间互调、查询、通讯等延迟，但与细粒度的服务相比，其重用性、灵活性比较差。

松散耦合：SOA 不像传统的 DCOM，CORBA 等计算技术依赖于具体的协议、实现等。它独立于各个平台，与开发环境、协议、厂家等都没有直接关联。SOA 架构中的不同服务之间保持一种相对独立无依赖的关系。

SOA 最主要的应用场合在于解决在 Internet 环境下的不同商业应用之间的业务集成问题。

2.2.2 SOA 架构的分层模型

文献[30]中提出的 SOA 架构分层模型如图 2-1 所示。在 SOA 系统中不同的功能模块可以被分为 7 层:第 1 层就是系统已经存在的程序资源，例如 ERP 或者 CRM 系统等。第 2 层就是组件层，在这一层中我们用不同的组件把底层系统的功能封装起来。第 3 层就是 SOA 系统中最重要的服务层，在这层中我们要用底层功能组件来构建我们所需要的不同功能的服务。总的来说，SOA 中的服务可以被映射成具体系统中的任何功能模块，但是从功能性方面可以大致划分为以下三种类型:（1）商业服务（business service）或者是商业过程（business process），这一类的服务是一个企业可以暴露给外部用户或者合作伙伴使用的服务。比如说提交贷款申请，用户信用检查，贷款信用查询。(2)商业功能服务(business function service)，这类服务会完成一些具体的商业操作，也会被更上层的商业服务调用，不过大多数情况下这类服务不会暴露给外部用户直接调用，比如说检索用户帐户信息，存储用户信息等。（3）技术功能服务（technical function service），这类服务主要完成一些底层的技术功能，比如说日志服务以及安全服务等。在服务层之上的第 4 层就是商业流程层，在这一层中我们利用已经封装好的各种服务来构建商业系统中的商业流程。在商业流程层之上的就是第 5 层表示层了，我们利用表示层来向用户提供用户接口服务，这一层可以用基于 portal 的系统来构建。

以上这 5 层都需要有一个集成的环境来支持它们的运行，企业服务总线提供了这个功能，它能够通过多种通信协议连接并集成不同平台上的组件将其映射成为服务层的服务。第 7 层主要为整个 SOA 系统提供一些辅助的功能，例如服务

质量管理，安全管理这一类的辅助功能。

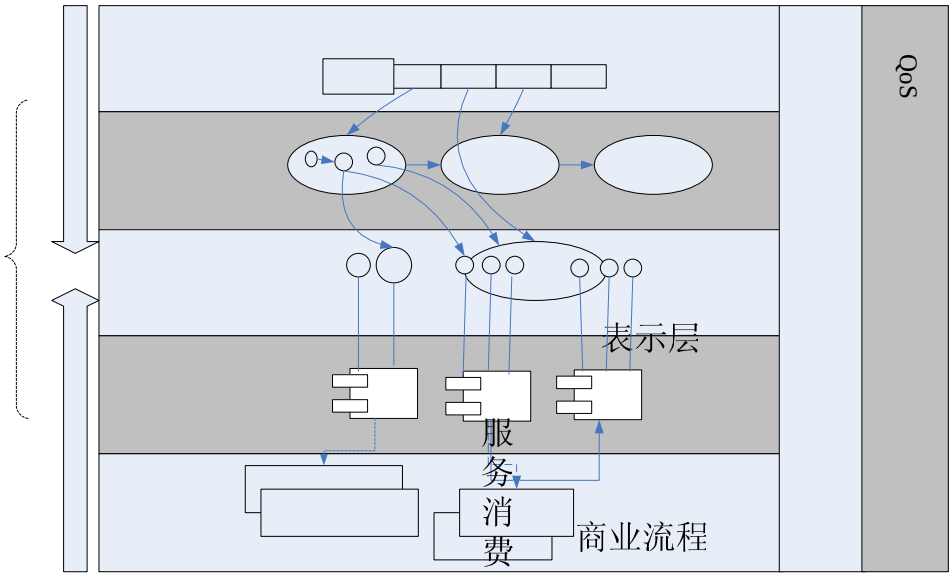


图 2-1 SOA 分层模型

2.2.3 SOA 的关键实现技术

在 SOA 架构下，服务成为应用系统的基本单元，使得 IT 与业务有机地结合在一起。Web 服务技术是对 SOA 的实现。这节详细讨论 SOA 中关键技术：Web Services 技术，下一节讨论 ESB 技术。

1．Web 服务定义

Web 服务可以从多个角度来定义。从技术角度来说，一个 Web 服务是可以被 URI 识别的应用软件，其接口和绑定由 XML 描述和发现，并可与其他基于 XML 消息的应用程序交互。从功能角度看，Web 服务是一种新型的 Web 应用程序，具有自包含、自描述以及模块化的特点，可以通过 Web 发布、查找和调用，使得 EAI 变得更加容易。

2．Web 服务体系结构

Web 服务采用了 SOA 的体系结构，通过服务提供者、请求者和注册中心等实体之间的交互完成服务调用。服务级别协议（Service Level Agreement，SLA）保证的 Web 服务第一代体系结构[31] 如图 2-2 所示。

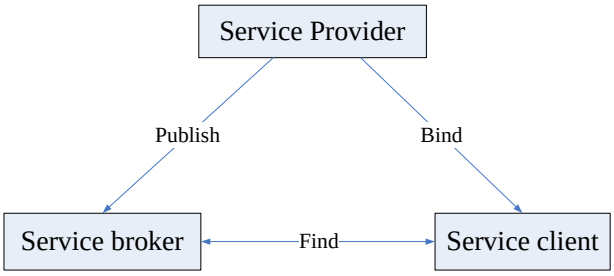


图 2-2 Web 服务体系结构

文献[31]提出了一些原因，认为图 2-2 中的体系结构已不适合如今日益复杂的 Web 服务应用程序，在 SOA 中已显得过时。IBM 提出了基于 SLA 保证的第二代 Web 服务体系结构[31]。

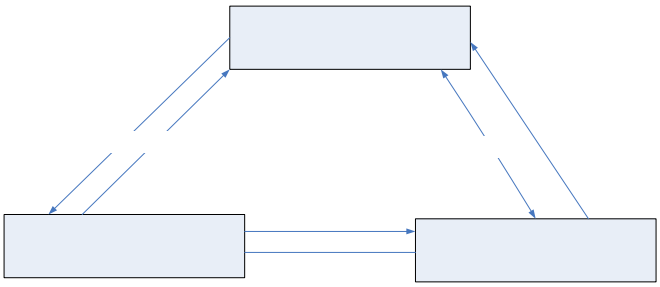


图 2-3 SLA 保证的第二代 Web 服务体系结构

第二代体系结构如图 2-3 所示，这里将提供者通知（alert）和客户通知分为三类：确认、验证和故障信息。每种类型的内容对每个事件来言都不一样。

例如，应用程序提供者给代理发送发布服务的请求。如果代理成功地发布了资源库中的服务，它将发送发布成功的通知信息给提供者。否则，发送发布失败的通知信息。应用程序客户发现 Web 服务应用程序过程类似。客户一旦接收到成功发现的通知，客户就将其与提供者绑定。彼此间通信，以便进行下一步工作。

3．Web 服务的技术支持

Web 服务平台需要一套协议来完成分布式应用程序的创建。任何平台都有它的数据表示方法和类型系统。要实现互操作性，Web 服务平台必须提供一套标准的类型系统，用于沟通不同的平台、编程语言和组件模型中的不同的类型系统。目前相关协议有：

（1）XML 和 XSD：

可扩展的标记语言 XML（Extensible Markup Language）是 Web 服务平台中表示数据的基本格式。XML 不仅易于建立和易于分析，而且是跨平台的，在传输信息时软件和硬件是相互独立的工具，并且与厂商无关。Web 服务利用 XML 来传送数据。Web 服务平台用 XSD 作为数据类型系统[32]。

（2）SOAP

XML 使数据易于理解和共享，但应用实体之间要发送和接收 XML 文档，还需对网络协议、访问点等细节达成共识[33]。Web 服务实体间使用简单对象访问协议 SOAP（Simple Object Access Protocol）交换 XML 编码信息。SOAP 是交换 XML 编码信息的轻量级协议，可以运行在任何其他传输协议上。SOAP 消息包括三个主要部分：封装结构、编码规则和 RPC 机制。

（3）WSDL

在应用程序调用一个 Web 服务之前，必须知道其调用接口。Web 服务采用

Web 服务描述语言 WSDL（Web Services Description Language）来描述其服务接口。WSDL 基于 XML 语言，具有语言无关性，用于描述 Web 服务及其函数、参数和返回值。WSDL 将 Web 服务定义为网络端点的集合，使用类型、消息、端口等元素来描述服务接口。请求者根据这些可以知道服务要求的数据类型、消息结构、传输协议等，实现对 Web 服务的调用。

（4）UDDI

为了使服务请求者能够查找到所需要的服务，业界制订了注册和查找 Web 服务的 UDDI 技术规范。UDDI（Universal Description, Discovery, and Integration）是一套基于 Web 的、分布式的、为 Web 服务提供信息注册中心的实现标准规范，同时也包含一组使企业能将自身提供的 Web 服务，以使别的企业能够发现的访问协议的实现标准。

（5）XSLT

XSLT 的英文标准名称为 eXtensible Stylesheet Language Transformation。根据 W3C 的规范说明书[34]，最早设计 XSLT 的用意是帮助 XML 文档转换为其它文档。但是随着发展，XSLT 已不仅仅用于将 XML 转换为 HTML 或其它文本格式。其更全面的定义为：XSLT 是一种用于转换 XML 文档结构的语言。XML 从根本上解决了应用系统间的信息交换，XSLT 就是我们用来实现这种转换功能的语言。

（6）Xpath

Xpath 是 XSLT 的重要组成部分。Xpath 是一种专门用来在 XML 文档中查找信息的语言。在转换 XML 文档时可能需要处理其中一部分（结点）数据，在查找和定位 XML 文档中信息时，就要用到 Xpath。

2.3 企业服务总线概述

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。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[35]。

2.3.1 ESB 的定义

ESB 为 SOA 中服务提供了交互、组合和治理的基础架构。有了它，才能释放 SOA 的最大价值。ESB 是一种基于标准的、保护投资的软件平台产品。ESB 将属于不同所有者的应用系统所提供的功能抽象到服务级别，使系统的互联不再

纠缠于接口细节的描述[36]。企业服务总线的定义通常如下：

文献[37]对 ESB 的定义为：它是企业服务运行的基础平台，负责对分散在整个企业以及企业外延的服务进行中央配置、部署和管理。它是传统中间件技术与 XML、Web 服务等技术相结合的产物，采用总线模式来管理和简化应用之间的集成拓扑结构，以广为接受的开放标准为基础支持异构环境中的服务、消息以及基于事件的交互，并且具有适当的服务级别和可管理性。

文献[38]对 ESB 的定义为：ESB 是从中间件产品中的存储转发机制发展而来。现在，ESB 是用来实现、部署和管理基于 SOA 解决方案的一种基于开放标准的消息骨干，以及所必需的安全、策略、可靠性等。

2.3.2 ESB 的功能模型

IBM 公司对 ESB 的功能提出了一个功能模型[35]，如表 4-1 所示。但是，当前的大多数场景只需要 ESB 功能模型中的部分功能，4-2 表是 ESB 为支持 SOA 所需的最低功能[35]。

表 4-1 ESB 的功能模型

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

表 4-2 最低的 ESB 功能

通信	集成	服务交互
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 支持至少一种可以广泛使用的传输协议	支持服务提供的多种集成方式，比如连接器、Web 服务、异步通信、适配器等	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

上面的许多功能既可以使用专有技术实现，也可以通过利用开放标准实现。然而，使用不同的技术来实现 ESB 可能会使它们的性能、可伸缩性和可靠性这些特性显著不同，同时 ESB 功能和所支持的开放标准也会有所不同。由于这些原因，再加上最近制订和正在兴起的一些相关标准，当今实现 ESB 的许多关键决策都涉及到成熟的专有技术和不成熟的开放标准之间的权衡[35]。

2.3.3 ESB 的特征

作为 SOA 架构中服务集成功能的重要提供者，我们可以总结出 ESB 的一些重要特征[39]如下：

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。在 ESB 系统中，被集成的对象被明确定义为服务，而不是传统 EAI 中各种各样的中间件平台，这样就极大简化了在集成异构性上的考虑，因为不管有怎样的应用底层实现，只要是 SOA 架构中的服务，它就一定是基于标准的。

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。成熟的 MOM 通常成为 ESB 中消息通信总线的选择。

以往传统的 EAI 实施过程中碰到的最大问题就是被集成者都有自己的方言，即各自的消息格式。因此其中的消息处理大多是被动的，消息的处理需要各自中间件的私有方式的支持，例如 API 方式。

ESB 由于集成对象统一到服务，消息在应用服务之间传递时格式是标准的。如果 ESB 能够在底层支持现有的各种通信协议，那么对消息的处理就完全不考虑底层的传输细节，而直接通过消息的标准格式定义来进行。这也是 ESB 总线功能的体现。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。EDA 使用的时机是业务需要单向消息收发，或者涉及长时间运行的异步流程，同时事件源不需要知道事件接收者是谁。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，这利用异步消息技术比同步更容易实现，而 ESB 可以为这种松散耦合系统的异步交互机制提供一个基础性的骨干通路。事件驱动机制的引入，能更加准确地反映现实世界中各种商业处理模式的异构性和通信交互的异步性。

2.4 ESB 对面向服务集成的支持

基于 SOA 的企业应用集成就是“面向服务的集成（Service-Oriented Integration，SOI）”。SOI 通过强制分开每个服务请求者和服务提供者，从而超越

了脆弱的、紧耦合的传统企业应用集成方法。SOI 提供了一个抽象的接口，通过这些接口，系统之间可以进行交互，而不是使用低层的协议和自定义的编程接口来规定系统如何与其它系统进行通讯。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。面向服务的集成使得 IT 机构能够在已有的应用中提供可重用的服务的功能。

IBM 认为，为实现 SOI，应用程序和基础架构都必须支持 SOA 原则。启用 SOA 应用程序要创建服务接口，而服务接口则可直接或间接地通过使用适配器用于现有的或新的功能。从最基本的级别来看，启用具有规划功能的基础架构将服务请求消息经路由并传递给相应的服务提供者。然而，在不影响服务的客户端的情况下，与之交互的服务的另一个实现也是至关重要的[35]。

ESB 最初被描述为分布式的基础架构，其中具有服务路由和服务替代等功能，并通过提供集成的通信、消息传递以及事件基础架构来支持异构环境中的服务、消息以及基于事件的交互功能[40]。

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。

2.5 本章小结

ESB 是特定环境下（SOA 架构中）实施 EAI 的方式，它是为了支持 SOA 设计的集成基础平台。本章的内容是后面章节分析和研究的基础。在本章中，我们先给出了企业应用集成的概念、研究内容以及分类，在此基础上引入了对 SOA 的定义、原则、研究现状、分层模型的介绍，接着对企业服务总线的定义、功能、体系结构以及优势进行了论述和分析。

第三章 基于 ESB 的企业应用集成框架

本章首先提出了面向服务集成框架的设计要求，然后提出了一个面向服务的集成框架－CreatorSOIF，并且对 CreatorSOIF 中的各层给予了分析；最后对基于 CreatorSOIF 进行电子政务系统集成的优势进行了分析。

3.1 SOI 框架体系结构的设计要求

对于 SOI 的理解不能狭隘地只停留在应用系统之间互连的层面上，SOI 的体系结构应该体现出 SOA 的动态特性。依照 SOA 体系结构来建设、改造、封装各类企业服务，使得这些服务可以被简单地发现、调用、管理。同时，基于 SOA 的企业应用集成系统可以随着企业业务的变化而逐渐变化，能够实现“柔性化”的软件系统。服务在整个 SOI 系统中是一个相对概念，因为一个服务可以由若干更小的服务按照一定的互动规则组成，因此整个 SOI 体系和内部的各个基础服务概念都是广义和可嵌套的。

SOI 通过提供一个构建、部署和管理集成的体系框架，减少了不同类型的 IT 系统的依赖性，降低了集成费用和 IT 操作的复杂性，可以提高已部署系统的灵活性。这个新的方式超出了传统集成的范围，能够合理化地将有用的技术进行合并，同时排除了抑制业务创新的障碍。根据上面的论述，构建一个 SOI 框架的体系结构应满足特定的要求，这些要求包括：

1．可以重用现有的资产。现有的应用系统很少可以被抛弃，它们通常都包含对于企业很有价值的东西。SOI 框架必须有能力集成现有系统，以便随着时间的推移，可以在管理、渐进式项目中分化或替代它们，以期获得更大的投资回报率。

2．具有良好的兼容性。任何应用的传输协议，不管是 J2EE RMI、.NET、CORBA、DCOM 还是 Web Services 等，它都能够提供支持并进行相应的协议转换，不管是同步服务还是异步服务事件都能处理。

3．基于开放标准而构建。基于开放标准而不是使用专有的和厂商自定义的技术来构建 SOI 框架，使得集成更容易，解决方案的选择面更广；同时，如果以后选择新技术，移植将更为便利。

4．服务的地址和传输协议的透明化。也就是说客户端调用 SOI 框架中的服务时不需要知道服务的具体实现。当客户端通过 SOI 框架调用这些服务时，SOI 框架将会代理调用这些服务。

5．安全性高。安全性越来越受到人们的重视，SOI 框架必须能够集成企业

现有的安全系统，同时支持服务请求者与服务提供者之间的安全交换。

6．对事务的支持。常见的 Web 服务事务的规范有 WS-C/WS-T、WS-CAF 等，他们都支持原子事务或者是 ACID 事务，同时都能很好的处理长事务，尤其是 WS-C/WS-T 规范，它能处理传统的 ACID 事务，使得 Web 服务和传统的系统之间能实现互操作。为了能够使企业客户接受，SOI 框架中的事务处理必须基于 WS-C/WS-T 事务规范，支持事务语义，以保证一致可靠的结果。

7．允许实现新的计算模型。特别是提供对新的基于 Portal 的客户端模型、网格计算和按需计算的支持功能。

3.2 一种基于 ESB 的企业应用集成框架

3.2.1 总体结构

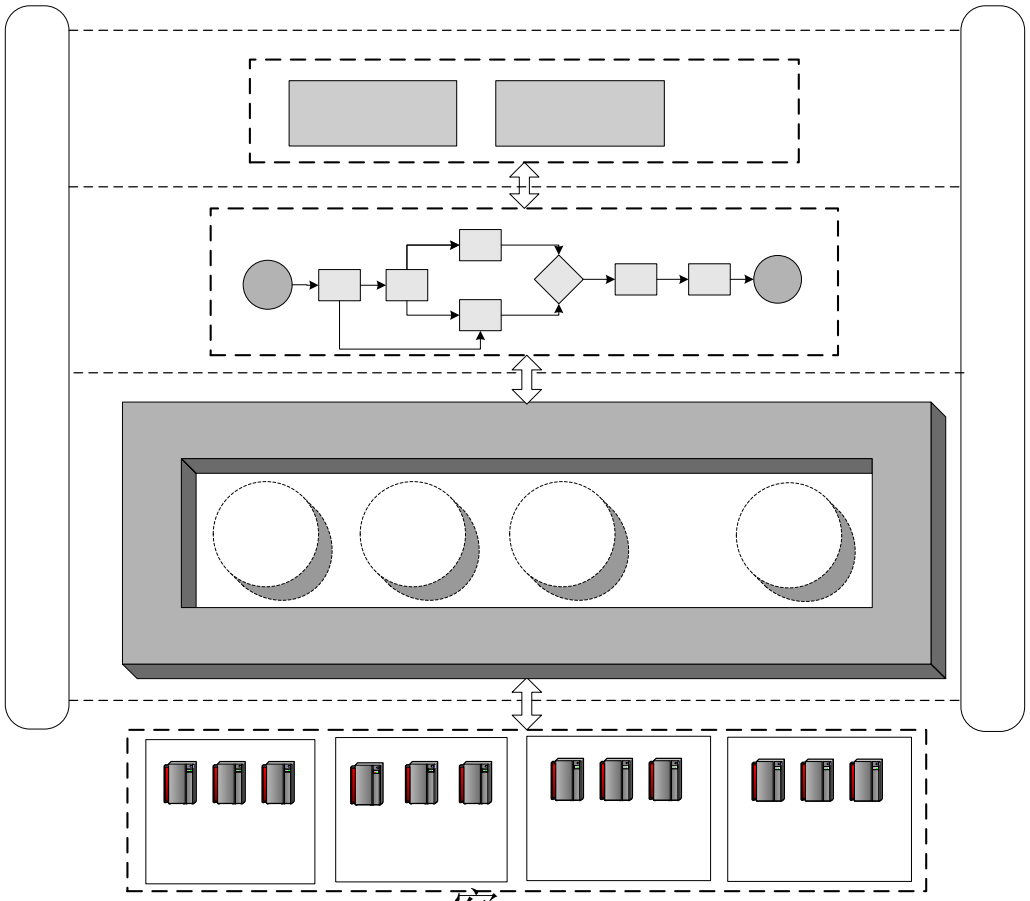


图 3-1 CreatorSOIF 结构图

伴随着 XML 技术的广泛应用和 Web Services 的发展，基于面向服务体系架构，并且结合企业服务总线技术来实现新一代的企业应用集成可以使得应用集成的技术不再依赖于单个集成厂商，为企业应用集成的技术采用提供了多种选择，同时还可以保护企业现有的 IT 资源，快速高效的对企业的资源进行集成，有效降低企业的集成成本，最终实现信息技术和企业业务目标的紧密协调。

本文给出了一个完整的基于 ESB 的 EAI 框架－CreatorSOIF（Creator Service-oriented Integration Framework）。CreatorSOIF 是一个灵活的用于集成各种异构环境中应用和服务的连接基础架构，能够有效地实现各应用系统和服务间的信息沟通和数据共享。它使用 ESB 作为服务集成基础，这种集成基础关注更为低层的网络交互协议，而开发者关注于各种业务流的组织。如图 3-1 所示，该框架实现了分层整合架构。

3.2.2 层次分析

1．服务资源层

重用是 SOA 思想的一个重要原则，因此必须充分重用现有的应用系统。这一层是可用于业务服务流程的可重用软件对象集合，它既包括企业原有的各种信息系统和遗留数据库、EJB、POJO 等，又包括为建立新的企业应用而重新开发的功能构件，ESB 将所有这些都视为服务。

以 Web 服务的集成为例：为了使系统中的遗留系统能以一种松散耦合的方式集成，可以将现有的遗留系统封装为 Web 服务，封装可以采用 Web 服务封装器。封装的意思就是从服务请求者的视角去看，只能看到其与一个 Web 服务进行交互，而 Web 服务背后是使用什么样的技术细节是无需知道的。这些 Web 服务以统一的方式暴露接口，即将它们原来以各种 API 形式暴露的接口用 WSDL 重新描述，然后使用服务接入层提供的 HTTP 或 JMS 等作为其与外界交互的桥梁。这样，通过 Web 服务的封装使得外部客户程序可以以统一的松散耦合的方式使用该服务，当业务的实现逻辑需要更改时，只要 Web 服务的 WSDL 接口不变，无论系统的业务逻辑、实现技术甚至是更换全新的应用系统，客户程序都不需要作任何改动。

2．ESB 服务层

ESB 服务层是整个 CreatorSOIF 的关键部分。它通过适配功能将分散在整个企业以及企业外延的服务资源封装为能被 ESB 部署和管理的业务服务，这些业务服务可供服务请求者进行调用。ESB 能提供与面向服务集成需要保持一致的集成基础架构，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、服务质量等方面。

当连接到 ESB 上的服务请求者发出请求消息时，在其中也可能要请求其他的服务。被请求的服务可以是基本服务、由静态组合而产生的服务或通过动态组合而产生的服务。ESB 服务层分离了与业务逻辑相关的应用层和与实现平台相关的服务资源，遵循该体系结构所构造出来的业务应用能适应业务和技术变化。

3．业务服务流层

为了达到某个业务目的而去按照特定的顺序去调用一组业务相关的活动称

之为业务服务流程。企业服务总线层之上是业务服务流层，通过通用和标准的对象和服务模型，我们可以在这一层上定义可重用和基于业界标准的业务服务流程，它是支持业务过程快速重构的关键。当企业的业务需求发生变化时，只需调整服务间的组装方式就能快速响应业务的变化，使得企业能以最快的速度满足市场的需求，这也是企业 SOA 服务集成的目标。

4．客户应用层

客户应用层为权限满足要求的用户提供了良好的应用服务访问接口，从而使得用户可以调用企业的应用服务。也可以根据不同的用户需要、根据不同时期的应用实现重点，非常方便、快捷地推出基于不同侧重点的门户，从而能为他们提供可用、高效、一致的访问界面。

5．系统管理、监控和安全

系统管理、监控和安全功能贯穿整个 CreatorSOIF。系统管理提供对基于 CreatorSOIF 实现的集成系统的管理。CreatorSOIF 提供对服务的执行情况进行监测的机制，它主要履行两个主要功能：分析服务的运行状况和监控服务异常。系统安全模块则通过用户身份认证、授权、消息加密以及数字签名的安全技术来保证在 ESB 上传输的消息是安全和可靠的，从而保证正确、安全的服务交互。

3.2.3 关键技术

ESB 服务层是服务请求者和服务提供者之间的中介，它提供对消息的传输、路由选择以及进行数据转换等重要功能。本文提出的 CreatorSOIF，可以为基于 ESB 进行企业应用集成提供一个良好的指导模型。它通常涉及到下列关键技术：

服务适配网关：通过该模型可以实现服务请求者和服务提供者之间的松散耦合、位置透明、协议独立等特征。对服务适配网关的研究将在第五章介绍。

消息路由技术：它配合服务适配网关技术可以完成消息的正确路由，这是通过在服务请求者和服务提供者之间一系列的中介者（Mediator）之间进行过滤和转发来完成的，这个过程涉及到消息路由器的设计与过滤机制的研究等问题，本文也将在第五章介绍。

安全技术：通过有效的安全技术可以保证传输的消息是安全和可靠的。消息从服务请求者逐级发送到服务提供者，继而服务结果返回给服务请求者，在这个过程中可以根据需要来设置是否需要数字签名、身份验证和消息加密，这满足了消息的机密性、完整性和不可否认性的安全需求，实现了消息级的安全。

服务的匹配、选择：服务请求者向服务注册中心发出查询请求，服务注册中心将服务请求与中心存储的服务描述进行匹配，服务发现的问题就转化为请求描述与服务描述之间的匹配问题。良好的服务匹配算法应该保证较高的查准率、查全率、良好的灵活性，同时也应该具有较好的效率。UDDI 是现有的典型的 Web

服务发现解决方案，但是它存在一些不足，如服务匹配是基于关键字的；不支持对服务的自动发现和组合等。目前，Web 服务和语义 Web 结合产生的语义 Web 服务技术是一种可行的解决方案。

此外，还有用于完成转换功能的消息转换技术，可以适配不同协议的连接器，以及服务适配器、组件管理器等关键组件。

3.2.4 框架的工作机制

CreatorSOIF 采用的是松散耦合的方式，服务部署灵活、简单快速。在该集成框架中，ESB 负责统一管理服务的执行与交互过程，确保集成应用系统的业务逻辑的正确性。在 ESB 的管理支持下，CreatorSOIF 中的服务执行过程如下：

Step1：客户应用层发出服务调用请求。

Step2：ESB 中的服务适配网关将该请求消息转换成 ESB 能够理解的格式。

Step3：在 ESB 中流动的消息可能需要验证服务请求者的身份以及所请求服务的访问权限，如果请求是合法的，则转 4，否则转 8。

Step4：这些可靠的请求消息通过 ESB 来实现服务注册中心或服务目录中元数据的搜索，从而找到请求者所需要的服务，如未发现，则转 8；如发现，则转 5。

Step5：查询到的匹配服务的有关信息被返回到请求用户。

Step6：通过 ESB 中的消息转换、消息路由等功能，用户将请求消息发送给服务提供者。

Step7：所请求的服务提供者由于自身性能，或者当时的负荷状态，不一定在收到请求消息后，马上执行，如果它能够执行，则转 9；如不能立即执行，则转 8。

Step8：在服务请求者请求服务和调用服务的过程中，如果出现异常，则将进入 ESB 提供的异常处理模块进行处理，转 10。

Step9：服务提供者接收到请求消息后，执行服务，同时将处理的结果返回给服务请求者，转 10。

Step10：结束。

3.3 基于 CreatorSOIF 进行电子政务系统集成

实施电子政务工程，有利于全面提升政府职能部门的办事效率和管理能力[41]。由于政府内部“信息孤岛”的存在，使得政府部门无法实现对整体政务运作和流程管理的全面掌控[42]。CreatorSOIF 集成框架中的 ESB 是一种新的集成方法，它类似于 PC 总线。PC 总线能够实现计算机各个功能部件之间的通信，与

其原理类似，ESB 能够实现政务系统中不同业务系统之间的通信，从而改变政府原有点对点的工作模式，有效的提高政府服务效率，将“一站式”、“一网式”、“虚拟政府”等理念的实现变为可能。与传统的集成方式相比，基于 CreatorSOIF 进行电子政务系统集成的特征主要体现在下面几个方面：

1．基于标准和开放的整合技术

ESB 是 CreatorSOIF 的核心，它支持 Web Services 系列开放协议和规范，并且通过支持 SOAP、JMS、JCA 等连通标准使得 ESB 能够整合企业和合作伙伴的应用，极大简化了电子政务集成系统在异构性上的考虑。连接到 ESB 上的不同松散耦合的业务系统之间通过 ESB 来进行通信，这些开放标准的采用避免了传统集成中间件为一家或几家集成商所私有的局面，屏蔽了版本和 API 的变化，使得在业务系统版本或者服务发生变化时无需改动另一端服务的代码，易于实现服务的重用和替换。

2．通过 ESB 将分散的服务进行集成，隐藏了服务的细节

外部服务提供者的地址和传输协议都已经被 ESB 实现了封装和隐藏。CreatorSOIF 对框架内的业务系统进行了封装，再将所有服务统一对外发布。框架中所有服务的具体实现、位置和传输协议对服务请求者来说都是透明的，也就是说客户端调用服务时不需要知道服务的具体实现，CreatorSOIF 中的 ESB 将会代理调用服务。此外，ESB 还具有修改消息格式和修改消息内容等功能。

3．基于事件驱动的服务流程编排

CreatorSOIF 通过 ESB 的消息通信服务在松散耦合的各个业务系统间传递事件或消息，进行正确的消息转换和路由，并且调用各业务系统的接口实现数据的存取，从而实现定义的业务流程。通过基于事件驱动的服务流程编排机制，企业和公众可以获得全面、准确的政务信息；实现政府内部的协同作业，提高办事效率，同时获取实时公众申报进度数据和部门业务处理统计数据，推动政府部门工作绩效的考核。

4．对安全的支持

基于 CreatorSOIF 进行电子政务集成时，由于系统中管理着大量安全级别相对较高的业务和数据，这就要求尽可能的保证系统的安全。按照其保护的内容可分为网络安全、主机系统安全、信息安全和应用系统安全四大部分，并且可以通过集成 PKI 架构提供全局统一的信任服务体系，建立信息安全、统一认证的网络信任域、以及协调完整的网络安全管理系统。

3.4 本章小结

本章的工作主要围绕 CreatorSOIF 来展开。首先提出了 SOI 集成框架体系结

构的设计要求，然后基于此设计了一个基于 ESB 的企业应用集成框架 CreatorSOIF。ESB 是 CreatorSOIF 的关键部分，它作为一种中间件技术，实现并支持 SOA。接着对该框架的关键技术与工作机制进行了分析，最后对 CreatorSOIF 应用于电子政务集成系统的优势进行了说明，充分显示了它的应用潜力。

第四章 企业服务总线参考模型

4.1 现有 ESB 模型研究

目前，已有不少厂商推出了自己的 ESB 产品，但是这些产品的体系结构各异，所具有的功能和实现的程度都不一样，这种情况给企业应用集成带来了新的问题。这一节将分析四种典型的 ESB 模型，比较其优势与不足。

1．BEA 公司的 ASB ESB 模型

ASB 是 BEA 公司的 ESB 产品，它基于 Web 服务技术、SOA 技术，是工业界第一个问世的架构于 SOA 技术和 Web 服务技术上的 ESB 产品。它支持多种消息格式和传输协议，消除了消息之间的差距，支持消息路由，服务编排，发送方和接收方可以在不替换现有基础架构的情况下，实现服务之间的快速集成和部署，可以跟踪消息和事件，支持有效的 SOA 运行，有在线建模能力。

粗粒度的 XML 通信协议，以及实际交付消息的面向消息中间件（Message Oriented Middleware，MOM）内核是 ESB 的主要元素[43]。ASB 模型中的 MOM 简化了连接到 ASB 上的应用之间数据的传输，屏蔽了底层异构操作系统和网络平台，提供一致的通讯标准和应用开发，确保分布式计算网络环境下可靠的、跨平台的信息传输和数据交换。但是 ASB 中使用的 MOM 系统是 BEA 的 MessageQ，如果它要和 IBM 的 MQSeries，Microsoft 的 MSMQ 进行交互，那么这些 MOM 系统很难实现互操作和无缝连接，这是因为没有一个通用的标准。此外，ASB 与 BEA 公司的 J2EE 应用服务器 WebLogic Server 紧密结合，它与平台的紧耦合性也给集成带来了不便。

2．中和威 ESB

在国内，北京中和威软件有限公司于 2005 年 9 月率先发布了基于 SOA 的应用集成中间件产品 InterESB[44]。InterESB 是对当前中和威公司拥有的基础中间件及领域中间件的各类技术和平台实施有效整合，形成的一个面向企业应用需求的完善的中间件整体解决方案。它由基础平台层、服务总线层和 EAI 应用集成层共三层组成。它将多种通信模式融为一体，其中包括目标通信模式、点对点通信模式、发布/订阅通信模式、扩展的发布/订阅集群模式。InterESB 将上述多种通信方式有机封装成一个整体，并通过多种标准接口方式对外进行发布，从而使得基于 InterESB 构建的企业应用能够以透明、一致、高效的方式应用不同的底层通信机制。

但是 InterESB 不具备面向服务的业务过程集成功能，也无法实现基于面向服务集成的其它高层需求。而且它主要应用在企业内部，如一些查询、浏览、数

据调用，而涉及安全性、可靠性要求高的如企业级事务方面的应用还很不成熟，此外，涉及新的商业机会，新的商业模式所牵动的各种产业环境也尚未丰满。

3．Mule ESB

Mule 是一个基于 ESB 架构理念的消息平台，Mule ESB 的部署模型是一种由部署在服务容器中的协作服务结点－UMO 组件构成的集成网络。服务注册在总线上，Mule ESB 驱动系统中的所有服务。连接到 Mule 中的各种平台上的应用和服务可以通过 UMO 来进行交互，UMO 组件的功能类似于 mediator。每个 UMO 组件都能发送和接收事件，其中的消息传输和事件处理可以使用不同的技术如 JMS、HTTP、SMTP 和基于 XML 的 RPC 等，并且能对路由到它们的数据进行转换。

灵活的配置 UMO 组件可以较容易的实现各种异构系统和应用之间的数据转换、路由和服务流程编排等功能，此外，Mule 中的事务处理机制和可自定义异常处理的安全管理则保证该集成框架有效地实施和正确运行。但是 Mule 没有提供类似 UDDI 的功能，还不能支持对服务的注册、查找，此外，它不支持可视化的编排业务服务流程，也还没有提供对 BPEL 的支持。

4．ServiceMix ESB 模型和 JBI 集成标准

JBI 是第一个 ESB 产品的集成标准。JBI 提供了一种正规消息路由器（Normalized Message Router，NMR），通过它，所有基于消息的数据片段－SOAP 片段、MOM 消息、HTTP 数据或其它信息－被聚合、集中、应用到业务逻辑、传输，如果有必要则被转换成其它格式并随后被分派到最终目的地。JBI 通过一种总线型架构的基于消息的手段到达了适应大范围的消费者和提供者的目的。ServiceMix 是基于 JBI 的 ESB，支持 JBI 规范的所有功能要求，是轻量的 ESB 实现。

在 ServiceMix ESB 提供的业务流程解决方案中，强调使用 BPEL 进行业务流程描述，然后集成一个第三方提供的 BPEL 引擎来执行业务流程。BPEL 描述文档中包含了它要使用的服务的引用（如服务名称、使用服务的地址和 WSDL 文档的存放位置），并且标明了每个服务的使用顺序。BPEL 引擎能够理解 BPEL 过程描述文档，该引擎负责按顺序或必要的逻辑来使用服务。

但是 BPEL 并不是理想的业务过程描述语言。BPEL 流程的核心概念是 Web 服务，它所调用的业务逻辑对应的实现只有 Web 服务，这样的设计对于能集成异构服务的 ESB 而言并不合理。众所周知，基于 XML 的 Web 服务相对于本地调用而言效率要低得多，在一个实际的业务流程中，需要 Internet 上的业务伙伴协作的活动数量毕竟是有限的，大多数的应用调用仍在本地。同时，将一个企业的所有业务逻辑均暴露在 Internet 上，也增加了安全隐患。此外，尽管 ServiceMix

的这种基于 JBI 的 ESB 模型很适合企业级的应用，但是 Java/J2EE 领域的其他主要推动者，如 IBM 和 BEA 等，都不支持 JBI，所以它的发展前景很不明朗。

4.2 一种基于服务执行引擎的 ESB 参考模型

4.2.1 SEE-ESB 的总体结构

根据对现有 ESB 模型的分析，并且基于国内外现有 ESB 的研究成果，本文认为 ESB 是服务运行时的基础平台，它能提供对以服务为中心的基础设施的有效管理。同时，服务的集成方式不应该仍然采取僵硬的点到点集成方式，它定义了总线上服务之间的连接和这些服务通过异构 MOM 进行的交互[45]，应该支持服务的动态发现、动态集成。因此，本文提出了一个以服务执行引擎（Service Enactment Engine）为核心的 ESB 参考模型即 SEE-ESB，为解决上述问题提供一个参考，如图 4-1 所示。

SEE-ESB 是一种可以提供可靠的、有保证的消息的新型中间件，它是网络中最基本的连接中枢。SEE-ESB 提供的服务可用于承载各种不同质量（可靠、不可靠）、不同类型（如同步的请求/响应、异步的发布/订阅）、不同领域（数据、事件、指令、文件等）消息的传输需求。它的总体结构中各部分如下。

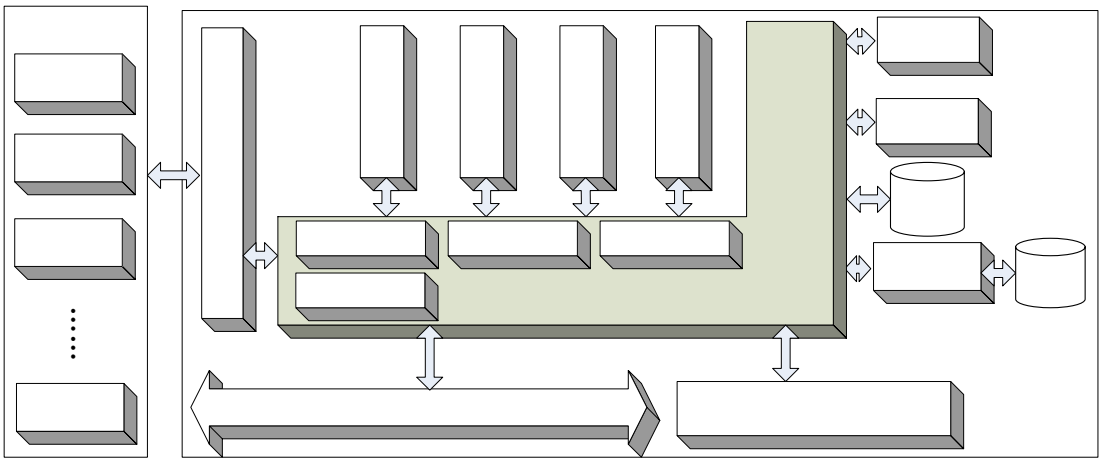


图 4-1 基于服务执行引擎的 ESB 参考模型（SEE-ESB）

1．服务执行引擎

服务执行引擎是 SEE-ESB 的核心，它处理对服务的申请并控制服务间的交互和消息转发，并通过调用 ESB 中的其他组件可以提供交互所必需的支持服务和管理服务，这些组件包括：服务编排器，安全管理器，事务管理器，服务监控器。

2．服务编排器

通过服务编排器，我们可以定义可重用和基于业界标准的业务服务流程。这

里编排的业务服务流程是对粗粒度服务的组装和排序，服务的不同组合方式代表了企业不同的业务流程，从而实现动态业务模型。当企业的业务需求发生变化时，只需调整服务间的组装方式就能快速响应业务的变化，使得企业能以最快的速度满足市场的需求，这同时也是企业 SOA 服务集成的目标。

3．安全管理器

安全管理器用来实现对 SEE-ESB 安全方面的管理，对授权用户和非授权用户进行认证机制，保证 ESB 始终处于一个安全的环境之中。这样可以保证传输到 ESB 中的消息是安全和可靠的，从而保证正确的消息转发。

4．事务管理器

它是基于二阶段提交协议，任务是确定事务协调者和参与者，由协调者控制整个事务的提交和失败后的回滚，它使得系统管理员可以良好地控制连接到 ESB 上的参与服务之间的交互。

5．服务监控器

服务监控器可以监控和管理服务的状态，具体来说其作用为：有效实现负载均衡；实现优化网络，增加对边缘资源的利用率；监控服务之间消息的转发，服务的调用以及业务流程的执行；除此之外，它还为 ESB 提供了良好的日志功能。

6．服务适配器

服务适配器能够提供对多种主流接入协议的支持，如 HTTP、JMS、JCA、.NET 和 IBM/CICS 等，这样可以使各种异构服务能够通过 ESB 进行交互操作。它将服务资源层中的各种异构服务资源传递过来的消息转换成 ESB 能够识别的信息表示方式，交给服务执行引擎执行，并将执行的结果返回给异构服务。由于 ESB 支持标准接口，因此这些适配器具有通用目的，不再为单个的企业集成商所私有。

7．消息总线

消息总线提供 SEE-ESB 内的消息通信服务，SEE-ESB 内的消息总线推荐采采用成熟的面向消息的中间件，如 JMS，这时消息总线就是 JMS 总线。

JMS 是一种面向消息的中间件，它是 SUN 提出的旨在统一各种 MOM 系统接口的规范[46]。基于 JMS 来提供 ESB 内消息通信服务，它包含点对点（Point to Point，PTP）和发布/订阅（Publish/Subscribe，Pub/Sub）两种消息模型，可以为连接到 ESB 上的服务提供可靠消息传输、事务和消息过滤等机制。

JMS 消息发送者将消息发送给消息服务器，消息服务器将消息存放在若干队列中，在合适的时候再将消息转发给接收者。这种模式下，发送和接收是异步的，二者的生命周期未必相同：发送消息的时候接收者不一定运行，接收消息的时候发送者也不一定运行；一对多通信：对于一个消息可以有多个接收者。由于 SEE-ESB 对 JMS 标准的支持，这使得 JMS 接口具有通用性。

8．互操作接口

当所需集成的服务个数很多时，如果把它们都连接到一个 ESB 上，通常会引起一系列的问题，如效率下降，负载过重，维护难度增大等。这时我们可以把一个 ESB 和其他 ESB 连在一起联合工作从而构成 ESB 群，这样可以提供比单个 ESB 更强大的集成功能。SEE-ESB 上有互操作接口，通过它可以和其他 ESB 进行交互。

9．服务发布、发现工具、服务选择器

SEE-ESB 提供了一个服务注册中心，连接到 SEE-ESB 上的各种异构服务可以通过服务发布工具注册到该中心上，注册中心存储着服务的描述文件，以便为用户和其他应用系统提供服务发现和服务查找。服务发现工具则根据指定的服务地址如 URI 来调用具体的服务，这是一种静态地调用服务。服务选择器基于一系列 QoS 规则集，它可以依据规则用来在服务注册中心查找具体服务，查找的结果是返回服务的绑定地址，这是一种动态地调用服务。

4.2.2 服务执行引擎的结构

在这一小节里，本文将设计一个服务执行引擎的结构模型。服务执行引擎是SEE-ESB的核心，它用来驱动整个业务服务流程的执行。它主要由引擎管理器、服务适配网关、消息路由器、组件管理器等部分组成，如图4-2所示。

服务适配网关可以实现服务请求者和服务提供者之间的松散耦合、位置透明、协议独立等特征。它提供各种协议的适配功能来调用各种异构的服务，可以屏蔽各种通信协议的差异性，为 ESB 提供统一的服务调用接口。

消息路由器可以用来进行消息的转发和过滤。当一个服务请求消息进入SEE-ESB后，它通常要经过一系列中介者，形成一条请求消息的服务（中介）处理序列，请求消息必须经过这条服务处理序列来完成一次完整处理，最后请求消息才被送到正确的目标服务提供者。在消息路由过程中，可以根据不同的过滤机制，如基于内容或基于消息属性等机制进行转发，中介者起到了消息路由器的作用。SEE-ESB中的消息路由服务非常适合于高度分布式的网络，这种消息路由过程，不需要服务请求者和服务提供者的直接介入，从而使得基于不同平台的异构应用通过SEE-ESB集成在一起。

组件管理器包括组件池工厂、组件注册器、生命周期管理器。它对业务流程定义文件中定义的各种组件进行管理，它会将各种组件注册到组件池中，这些组件是可插拔的，由生命周期管理器对其管理。

引擎管理器根据业务流程定义文件来执行，并通过组件管理器调用相应的组件来进行数据的处理，这里的组件可以是连接到ESB上的业务服务对象。这些业务服务对象可以起到参与ESB通信与交互的中间处理（协调）作用。

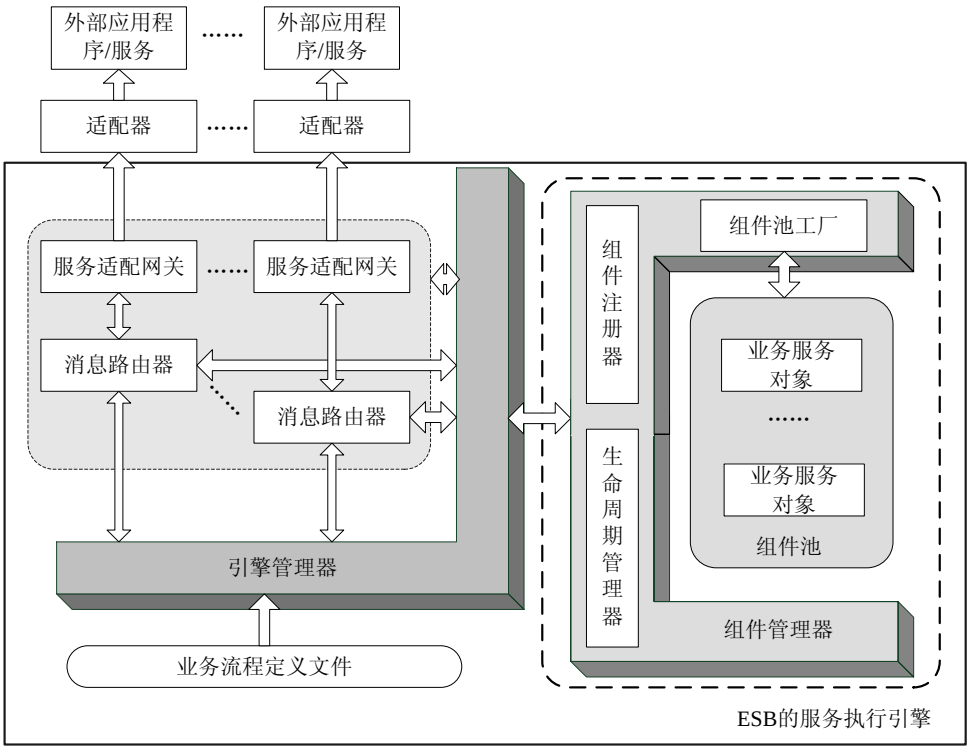


图4-2 服务执行引擎体系结构

下面是服务执行引擎运行时的工作机制：

Step1：服务执行引擎启动。

Step2：服务执行引擎中的引擎管理器解析业务流程定义文件，并根据业务流程定义文件对ESB进行相应的配置。

Step3：引擎管理器激活组件管理器，组件管理器对业务流程定义文件中的组件进行注册。

Step4：服务适配器通过适配器接收到外部应用程序的一个请求消息后，发送给服务适配网关。

Step5：服务适配网关接收该消息，并根据需要使用转换器对该消息进行数据格式的转换。

Step6：根据一定的路由机制转发到业务服务对象，由其准备处理。

Step7：分布式业务服务对象对该请求消息进行业务处理。

Step8：处理后的消息由组件管理器接收，根据流程定义文件，由引擎管理器将消息送至路由器，由其进行转发和过滤，传递到下一个访问点，如果下一个访问点是最后的访问结点，则转9；否则，转6。

Step9：整个流程执行完毕，得到返回消息，通过服务适配网关发送给服务请求者。

4.2.3 SEE-ESB 的部署模型

图 4-3 是一种 SEE-ESB 的部署模型。应用系统通过 ESB 提供的接口将数据提交给 SEE-ESB。SEE-ESB 在接收到数据后，通过内部的处理，再把数据提交给服务提供者进行业务处理。处理完成后，通过 ESB 将结果返回给应用系统（服务请求者）。SEE-ESB 为服务请求者提供一个基础的支撑环境，为服务请求者提供诸如消息路由、数据转换、日志、事务、协调等服务，使得服务请求者关心其如何利用数据而不是考虑服务如何查找、调用，服务提供者则关心其如何实现业务功能。

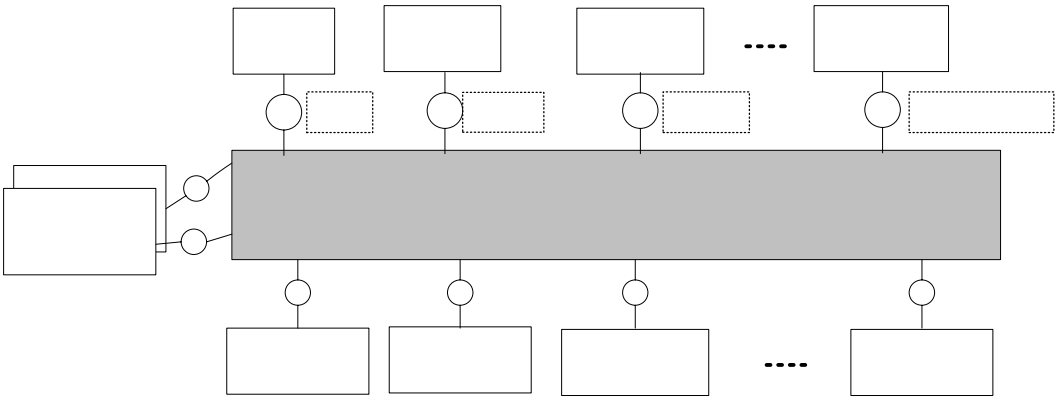


图 4-3 SEE-ESB 的部署模型

使用 SEE-ESB 可以将选择恰当服务提供者的负担从服务请求者转到 SEE-ESB 上，从而为服务请求者提供一个简单得多的体系结构。SEE-ESB 是调用服务的请求者和这些服务的提供者之间的中介，即服务请求者不再直接调用服务提供者，而由 SEE-ESB 对服务提供者进行调用。图 4-4 显示了一个在调用服务的请求者和该服务的提供者之间进行协调的 ESB。

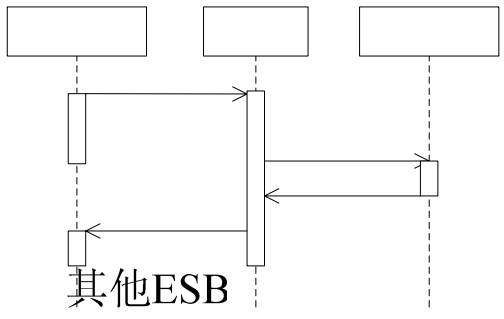


图 4-4 企业服务总线序列简图

在企业应用集成场景中，SEE-ESB 还可支持其他不同的部署模式。ESB 本身可以是单个引擎，甚至还可以是由许多同级和下级 ESB 组成的分布式系统，这些 ESB 一起工作，形成不同的拓扑结构，以保持 SOA 系统的运行。这可以称

为“企业服务网络”[47]。图 4-5 是 IBM 提出的一些常见 ESB 互联模式。

1．全局 ESB：所有服务共享一个名称空间，每个服务提供者对环境中所有服务请求者均可见，所有服务都可能在整个组织中应用。

2．直接连接的 ESB：公共服务注册中心使几个独立的 ESB 安装空间中的所有服务均可见。

3．代理 ESB：桥接服务有选择地将请求者或提供者公开给其他域中的合作伙伴，ESB 间的服务交互通过实现桥接服务的公共代理进行。

4．联合 ESB：将多个依赖 ESB 联合到其中的主 ESB，服务请求者和提供者连接到主 ESB 以访问整个网络中的服务。

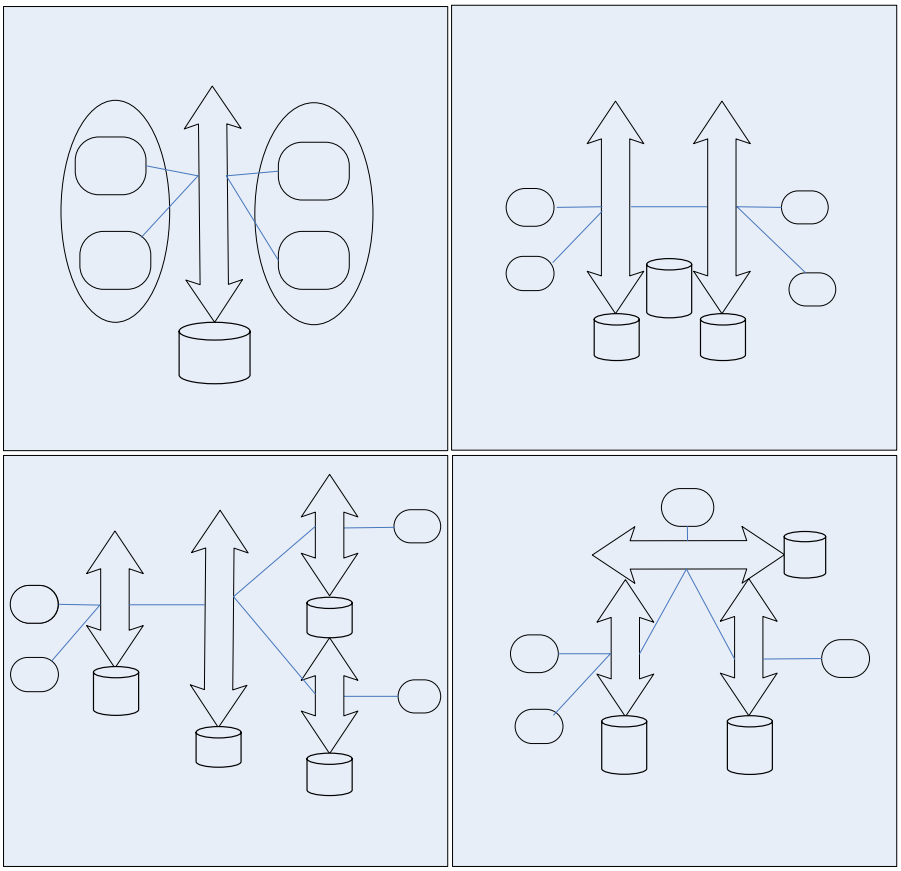


图4-5 ESB的互联模式

4.3 本章小结

本章在分析现有主流 ESB 模型的优势与不足后，提出了一种基于服务执行引擎的企业服务总线参考模型 SEE-ESB，从而为 ESB 的理解、设计和实现提供了一个参考，并提出了它的核心－服务执行引擎的体系结构，以及其部署模型和 ESB 的互联模式。

第五章 服务执行引擎的关键技术

服务适配网关和消息路由器是服务执行引擎的两个关键组件，它们体系结构的设计以及具有的功能将直接影响到 SEE-ESB 参考模型设计的好坏。本章第一节将研究服务适配网关，第二节将研究 ESB 中的消息路由机制。

5.1 服务适配网关的研究

5.1.1 服务适配网关概述

服务适配网关是 SEE-ESB 中重要的一个功能模块，它主要通过建立各种协议的适配功能来调用各种异构服务，可以屏蔽各种通信协议的差异性，为 ESB 提供统一的服务调用接口。它可以实现服务请求者和服务提供者之间的松散耦合、位置透明、协议独立等特征。它具有如下优势：

1．通过服务适配网关将服务访问逻辑与应用逻辑分隔开来，可以轻松更改应用程序所访问的服务，并达到模块间的松耦合。外部应用程序不直接访问与之交互的应用程序或服务，不受任何实现细节和服务位置的影响。

2．隐藏外部应用程序或服务之间交互的复杂性。通过配置服务适配网关中的连接器以及消息转换器等组件，对外部应用程序或服务之间的交互可以屏蔽各种通信协议的差异性，为 ESB 提供统一的服务调用接口。

5.1.2 服务适配网关的工作模型

在这一小节里，我们将设计一个服务适配网关的工作模型。该服务适配网关模型提供了一个开放的、基于标准的基础结构，主要由以下四个模块组成：消息接收器、消息分派器、消息转换器、连接器等组成。服务适配网关模型如图 5-1 所示。

消息接收器可以用来接收外部应用程序或服务发出的请求消息，请求消息通过它进入服务适配网关。

消息分派器可以将经过处理的请求消息发送出去。这里发出的请求消息已经是接收者能识别并处理的消息格式了，这是因为消息转换器的作用。

消息转换器可以提供转换服务，它把请求消息的格式转换成另外一种消息格式。消息接收和消息分派的目的地址都是由消息路由器上配置的访问点 URI 去决定的。

连接到 SEE-ESB 上的异构服务之间相互通信所使用的消息格式各异，为了实现它们之间的通信和交互，消息转换器必须将服务请求者发出的消息转换为目

标服务提供者所能理解的格式。

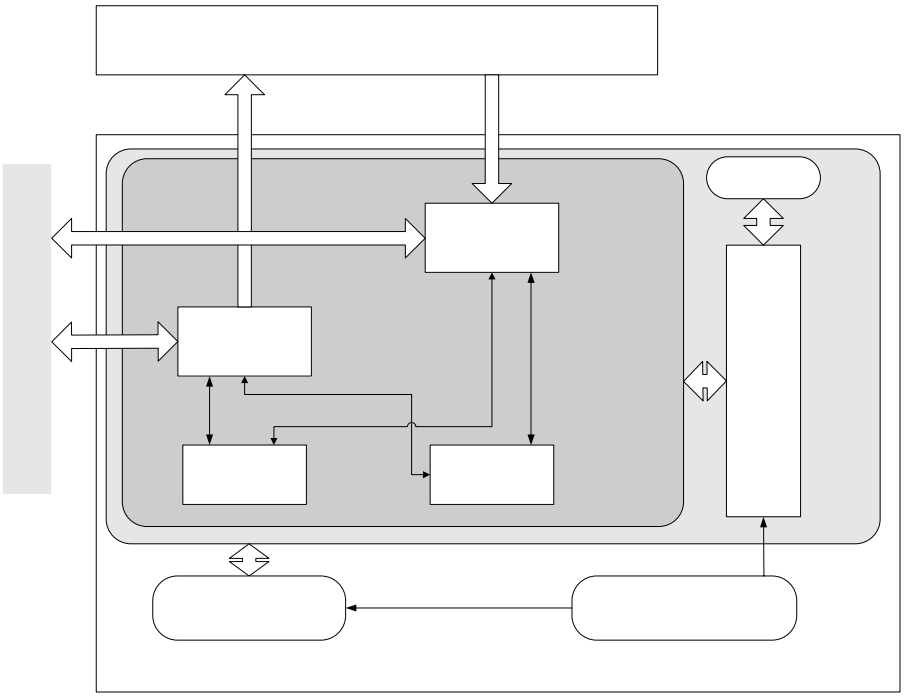


图 5-1 服务适配网关模型

XML 独立于操作平台和编程语言，它提供了强大的信息表示和转换能力，一方面可以将其他格式转换为 XML 格式进行处理和向前转发，另一方面可以根据需要将 XML 转换为其他格式[43]。因此，SEE-ESB 内的请求消息通常是基于 XML 的，消息转换器就可以方便地采用 XSLT 和 XQuery 等技术来实现消息的转换。消息转换器提供的服务是分布式部署的，这样避免了传统的企业应用集成在中央 Hub 上进行集中转换所引起的性能瓶颈。

连接器：连接器是服务适配网关的核心。在连接器上可以定义消息接收器和消息分派器的端点 URI，用于指定消息从何处接收或者发送到何处。此外连接器使用消息转换器来转换消息。

访问点 URI：它是访问点 URI 指向每个端点（endpoint）的地址。

连接器描述符：它是定义在业务流程定义文件中的 descriptor，它描述了是怎样的一个连接器，以它对 JMS 连接器的描述为例：包括了对连接工厂（ConnectionFactory）、JNDI 初始化工厂（jndiInitialFactory）、规范（specification）等的详细说明。

下面是服务适配网关的工作机制：

Step1：外部应用程序/服务发出请求消息后，消息接收器上的监听进程监听到该消息，消息接收器根据业务流程定义文件中定义的配置信息，获取访问点 URI。

Step2：消息接收器通过解析访问点 URI，获取所使用的连接器描述符。

Step3：消息接收器根据连接器描述符的具体配置快速定位到相应的连接器。

Step4：根据业务流程定义文件，如果访问点 URI 上配置有消息转换器，则转 5，否则，转 6。

Step5：连接器调用消息转换器进行消息转换。

Step6：消息分派器获得转换后的消息，将该消息发送给外部应用程序/服务接收者或消息路由器，转 7。

Step7：流程结束。

它的时序图如图 5-2 所示。

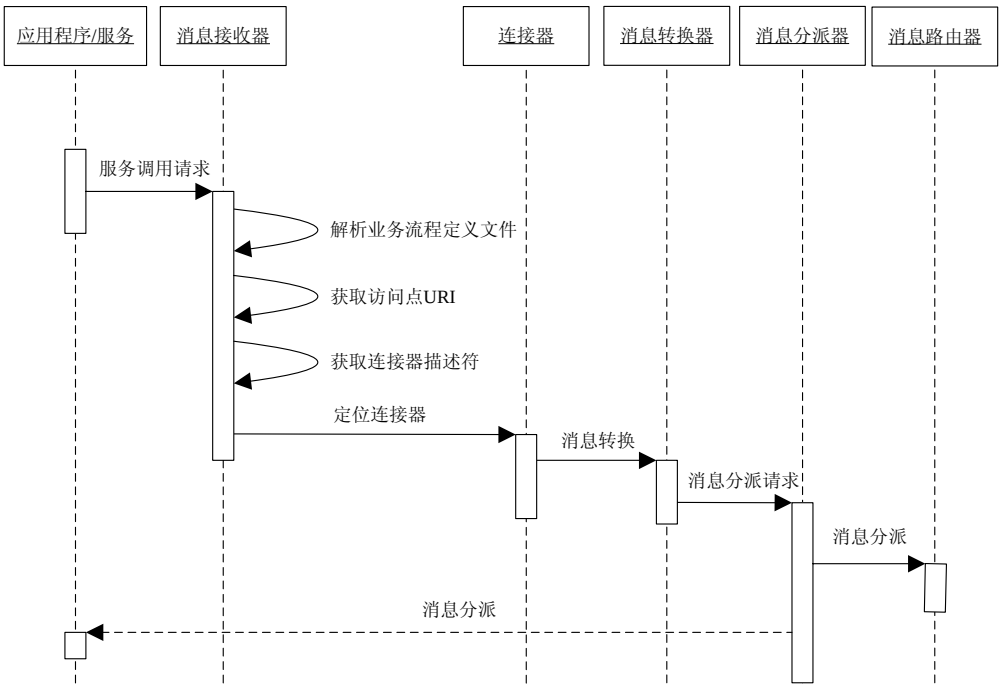


图 5-2 服务适配网关的工作机制时序图

5.1.3 服务适配网关的设计

当 ESB 需要调用各种服务时，它并没有直接调用这些服务，而是通过使用相应协议的服务适配网关去调用服务。对于不同的通信协议提供相应的服务适配网关，如 HTTP 适配网关，JMS 适配网关，CORBA 适配网关、EJB 适配网关和 COM/DCOM 适配网关。SEE-ESB 在启动时都会构造出相应的适配网关，去实现消息的发送和接收。

每种服务适配网关的接口方法都是一样，所以 ESB 可以根据服务的类型来选择对应的网关，其调用方式是统一的，只是网关的实现类不同。下面具体介绍 EJB 服务的适配网关。

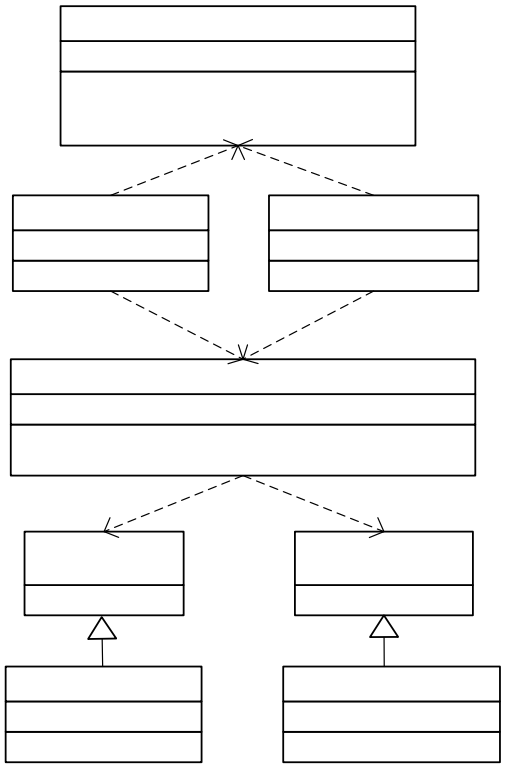


图 5-3 EJB 服务适配网关的类图

如图 5-3，Transformer 类是进行消息格式转换，ESBDispatchAction 类是 SEE-ESB 中进行消息发送的，ESBReceiveAction 是进行消息接收的。具体要调用哪种发送器和接收器根据访问点 URI 来决定的，这个功能由 AbstractConnector 来完成。MessageReceiver 和 MessageDispatcher 是 SEE-ESB 中消息接收和发送的两个接口，EJBMessageReceiver 和 EJBMessageDispatcher 则是消息接收器和发送器的具体实现，这两个类根据访问点 URI 和消息的具体格式进行发送或接收。

5.2 ESB 内消息路由机制的研究

5.2.1 消息路由机制概述

目前，在 ESB 领域内消息路由的研究是一个热点。ESB 中消息路由是对传统消息路由技术的进一步发展与完善，它通过路由器种类的选择，以及路由过滤机制的引入等多项技术和机制的有机组合与协同工作，大幅提升了 ESB 的可用性、可靠性和可控性，为服务参与方构建了一个简单、透明、可靠的消息路由机制。这一小节先介绍与 ESB 内消息路由一脉相承的消息路由工作机制，然后分析 ESB 内的消息路由工作机制，包括其路由流程和特点等。

1．消息路由的工作机制

基于 XML 的 Web Services 技术和面向服务的体系架构的研究近年来蓬勃发展，它们被认为是实施企业应用集成的一种有效解决方案。Web Services 解决了

在不同平台，不同开发语言，不同的协议之间如何进行安全的互操作的问题。Web Services 通过 SOAP 消息进行交互，SOAP 是一种使用 XML 来定义的规范，是一个在分布式环境下传递消息的轻量级协议，通过它可以在不同系统之间实现消息的传输和路由。

基于消息目的地址的路由、基于内容的路由、基于主题路由等是目前消息路由的主流模式，具体分析如下：

（1）基于目的地址的路由

它通过指定消息的目的地址来将消息转发到目标接收者，如相互通信的应用之间 IP 地址紧密绑定。当使用 SOAP 消息进行通信时，SOAP 只是规定了消息的接收者，并没有规定消息传递过程中所使用的路径。目前，学术界、产业界提出了一些 Web Services 核心规范和协议，致力于研究系统集成中消息的路由机制，主要有：

1）使用 WS-Routing[48]进行路由

WS-Routing 是一个无状态协议，它扩展了 SOAP 协议，使用 SOAP 扩展，在 SOAP 消息结构内部提供了描述消息路由路径的方法，该规范允许 Web 应用以传输中立的方式指定消息路由和调度信息。这个路径将从消息源，经过若干中介，最后到达消息的最终接收者。

但是这种消息路由存在严重的安全问题。在消息路由过程中，中间结点必须修改 WS-Routing 标头之后才能把 SOAP 消息发送到下一个结点。所以消息的最初发送者无法对包括 WS-Routing 标头在内的消息进行加密、签名，这导致了作为 Web Services 基础消息设施的 SOAP 的安全性无法保障。

WS-Routing 没有提供中间结点动态路由的构造，这个功能由 WS-Referral 来提供。WS-Referral 是一个基于 SOAP 的无状态协议，它是与 WS-Routing 相正交的协议，它提供了如何配置中间结点以建立一个消息路径的方法。

2）WS-Addressing[48]对 WS-Routing 的改进

作为 WS-Routing 的改进，WS-Addressing 规范不再使用完整的消息转发路径，而是为每个结点指定下一个发送目标，即 Next-Hop[48]方式，这样就避免了修改标头。它意味着消息发送者不需要知道整个消息的路由路径，它只需要知道执行路由指令的一个端点就可以开始转发。WS-Addressing 规范定义了在 Web Services 消息和服务描述中表示消息寻址信息以及在服务地址中包含特定属性的标准，用标准的方法来指定消息的目的地、如何返回响应或者在哪里报告错误。

（2）基于内容的路由

实现基于内容路由的主要问题是如何将到达消息基于内容而不是基于目的地址映射到下一跳地址[49]。

与基于目的地址的路由相比，它的主要特点有：

1）消息发送者不需要知道消息接收者的任何先验知识，允许消息接收者表达希望接收消息的愿望而不必了解消息发送者的信息。

2）基于内容的路由具有更大的灵活性，实现了应用之间的松散耦合。在发布/订阅模型中，消息按照内容进行分类，由于可以针对事件的内容进行订阅，使得订阅者能够更有针对性地订阅它们感兴趣的事件，降低了无用事件的干扰；对于消息过滤条件的更改不会影响其他用户的订阅。

3）基于内容路由能够以一致的方式实现消息的单播、多播和广播，而这在基于目的地址路由中是很难实现的。

发布/订阅模型是分布式网络中的一种有效通讯机制，其异步、多点通讯的特点很好地满足了 Internet 上松散耦合的大型应用系统的需要[50]。WS-Eventing[51] 规范是对发布/订阅模型的实现。语义上说，WS-Eventing 规范不是路由规范，但是实际上，它可以用来实现基于内容的路由。WS-Eventing 提供了根据消息内容的某些规则进行触发消息的机制，它是基于内容在 Web 服务之间转发消息的强大手段。采用 WS-Eventing，当事件源判定有事件发生时，它就会将此信息提供给订阅管理器，订阅管理器然后可以将该事件传送给所有匹配的订阅。

但是由于 WS-Eventing 原本不是路由规范，它有一些局限性，比如对每一个中转结点来说，它上面的路由器都要检查消息，然后基于消息的内容来进行处理，这使得在中转结点上加入安全性特征是非常有限的。

（3）其他路由机制

除了基于内容的发布/订阅模式外，对消息或事件的发布/订阅还有基于主题和基于类型的订阅模式等。

基于主题的发布/订阅模式[52]：主题是一个字符串类型的关键字，它作为区分事件类型的标识，每个发布的事件都携带有主题信息；订阅者的订阅信息则包含某个主题，说明订阅者所感兴趣的主题。事件通知服务依据主题信息在事件和订阅信息中进行匹配，从而将事件转发到对它感兴趣的订阅者。

基于类型的发布/订阅模式：它根据消息的类型来路由消息。

另外，由于应用集成场景的复杂性和企业实际业务的需要，还可以有其他路由模式。

2．ESB 中消息路由的工作机制

承担服务请求者和服务提供者之间通信协调任务的各种中介者，通过它们可以实现连接到 ESB 上的不同服务间的交互和集成[53]。ESB 中的消息路由不是工作在网络级，它工作在应用级[54]。中介者可能包含零个或者多个，是否部署中介者依照应用的体系结构决定。中介者可能需要对接收到请求消息进行修改、必

要的处理以及增值服务等，如在实际的应用环境下可能对服务请求消息进行格式验证、安全认证、计费功能等。中介者具有消息路由的功能，在路由消息时可根据服务请求消息的内容、类型等进行转发。

ESB 中的消息通常是基于 XML 格式的，它可以方便有效地表示结构化数据，这就使得 XML 可以作为表示和存储数据的手段。当一个 XML 服务请求消息进入 ESB 后，它先经过一系列中介者（如果需要的话），即中介处理结点，这些中介处理结点要侦听消息，以对消息进行部分处理或完整处理，从而形成一条 XML 消息的服务（中介）处理序列，如图 5-4 所示。XML 消息必须经过这条服务（中介）处理序列来完成处理，最后 XML 消息才被送到正确的目标服务提供者。

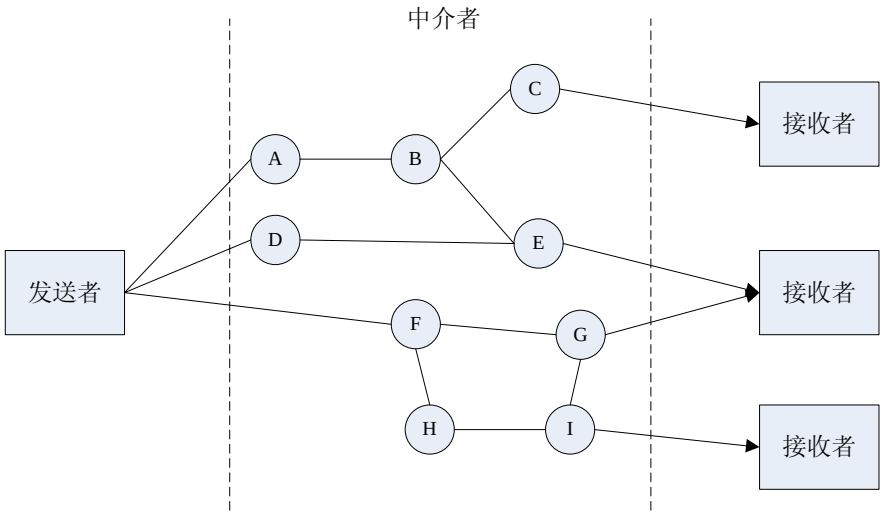


图 5-4 中介者对消息进行处理

基于 ESB 的消息路由具体流程：

（1）当有消息到达消息接收结点时，接收结点就用消息接收器接收消息。

（2）消息在路由过程中可能有中介处理结点，它们负责对消息进行必要处理，比如，将消息的格式转换成为下一个中介处理结点所能理解的消息格式，或者将来自不同源地址的消息进行处理再进行汇聚然后再送到下一个中介处理结点等。在消息转发过程中出现错误时，还能将消息转发到异常处理结点进行异常处理。

（3）消息的发送者和接收者都是相对的，消息的最终接收者在接收送达的消息并且进行处理完成后，根据需要还可以向消息的原始发送者返回一个应答消息，而且应答消息的路由路径可能不会和消息的原始路由路径保持一致。

（4）消息的传输和路由支持可靠的端到端传输，另外，还可以利用独立的审核跟踪服务来记录消息的转发过程。

5.2.2 ESB 内消息路由器体系结构的设计

随着 Internet 技术的迅猛发展，大型跨区域企业应用集成系统的各个部门之间、不同企业之间在通过基于 Internet 进行业务往来的过程中，逐渐发现多个异构 Web 应用系统之间的语义差异、平台差异、协议差异、数据差异是低成本的集成这些异构系统的主要困难。另外，目前异构 B2B 系统集成度越来越高、越来越复杂，特别是多级代理的出现，使得消息不再是从服务消费者直接发送到服务提供者，而是在转发过程中可能要经过多个中介的处理[55]。ESB 处理异构服务之间交互的方法就是在路由路径中配置多个中介，这些中介有不同的种类，可以对消息完成不同的处理，从而实现服务请求者和提供者之间的协调[56]。这一小节先介绍各种中介者，接着设计了 ESB 内的通用中介者的体系结构，由此设计出了 ESB 内的消息路由器的体系结构。

1．中介者

中介者通常独立于具体平台和技术，中介者通过构成 mediator 链来形成逻辑总线拓扑结构，可以修改服务请求者和服务提供者之间传递的消息，从而实现连接到 ESB 上的不同服务间的交互和集成。中介者可以起到不同的作用，本文总结了一些常见的中介者种类，如下所示：

过滤中介：它能够对消息进行检查，根据一定的过滤准则，让不同的消息通过不同的路径进行转发，用于有选择的过滤消息。

消息格式转换中介：用于对消息格式进行转换，将消息的有效负载（内容）从请求者的模式转换为提供者的模式。

协议转换中介：允许服务请求者使用各种交互协议或 API（如 SOAP/HTTP、JMS 或 MQ 等）发送其消息，由该中介负责服务请求者和提供者之间协议的转换。

充实中介：通过添加来自外部服务或数据源的信息来扩充消息的有效负载（内容）。

日志中介：记录消息的过滤，并记入日志，以供日后审核之用。

监控中介：监控服务之间消息的过滤，它可以和日志中介结合起来一起使用。

数据库查询中介：用来执行对数据库的查询功能。

Cache 中介：存储已经执行的请求的结果，这可以节省时间。

可定制中介：允许用户使用外部组件作为中介者的实现。目前，IBM WID 定义的 SCA[57]就是一种这样的外部组件。

逻辑操作中介：消息的并、交、聚合、分离等操作。

采用 mediator 的好处：首先，它能够很容易的用来集成遗留系统，企业可能不想改变这些遗留系统的代码，这是因为代价太大；其次，因为独立，与平台无

关的 mediator 的出现，可以实现 mediator 的重用，避免了根据服务请求者和服务提供者使用的技术使得它们只有一种交互方式的可能，可以实现它们之间的多种交互模式。

中介者相互之间可以进行组合，以实现更为复杂的中介者处理模式。例如，可以将转换、日志、过滤、监控、逻辑操作模式结合在一起，以提供加密、日志记录、审核及过滤等功能。

2．ESB 中通用中介者的设计

如果 ESB 所做的工作仅仅是机械地把消息从服务请求者通过 ESB 传送到另一端的服务提供者，那么这对企业级应用来说还完全不够。为此就需要在传输的同时对接收到的消息做进一步地加工，这就是引入中介者的目的。它可以动态地处理总线上的消息（事件），由服务请求者发出的消息会转换为不兼容的服务提供者能够理解的消息，以帮助不匹配的服务请求者和提供者进行交互。

通过在中介者之间进行转发可以协调服务请求者和提供者之间的通信和交互，因此，中介者实际上也具有路由器的功能。

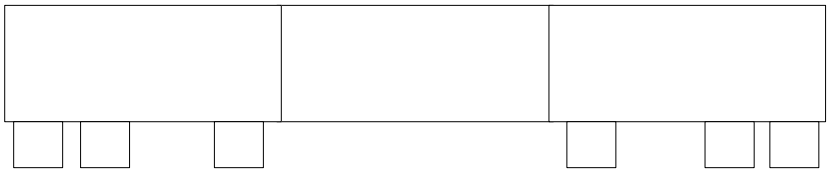


图 5-5 ESB 中通用中介者的架构

一个通用中介者由消息接收器、消息发送器以及中介处理组件组成，如图 5-5 所示。消息接收器上有一个或多个输入终端用于接收输入的消息，消息发送器上有一个或者多个输出终端用于转发消息，其中，输出终端可用于输出两种类型的消息流：失败流和成功流。一般情况下它最多可以有一个失败终端，而成功终端数目不限。中介处理逻辑用于对通过消息接收器进入的消息进行中介处理。

3．基于中介者的消息路由器的设计

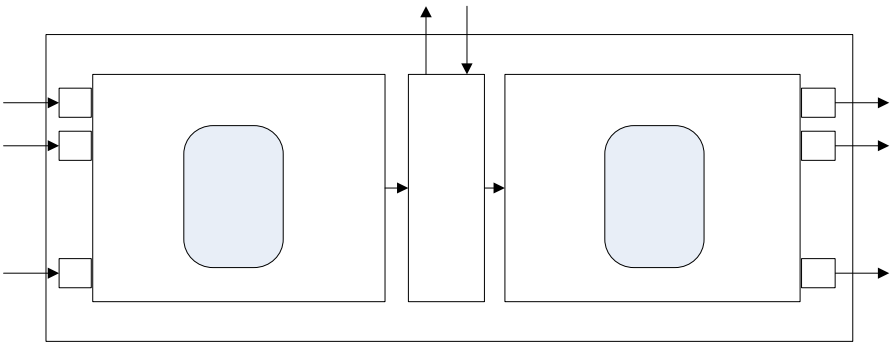


图 5-6 消息路由器的体系结构

消息接收器 中介处

根据上面的通用中介者结构，我们设计了一种 ESB 中消息路由器的结构，它由入站路由器和出站路由器以及中介处理组间组成，如图 5-6 所示。

入站路由器：其上有一个或多个入口，消息通过它们进入到中介处理组件进行必要的处理。

出站路由器：其上有一个或多个出口，用来将经过中介处理的消息按照一定的过滤准则转发到相应的消息接收者；另外，它还具有一个端口用于将不匹配过滤条件的消息转发到相应的容错组件进行处理。

入站路由器和出站路由器上都设置有过滤器（Filter）。过滤器根据其上设置的不同过滤准则将消息过滤到不同的接收者。实际应用中，可以根据需要在入站和出站路由器上设置过滤器。

中介处理组件提供调用接口，它通过调用第三方提供的服务如 Web Services 等来对消息进行处理。

在有些情况下，中介处理组件不对经过它的消息作任何处理，而只是将消息从入站路由器上进入的消息直接转发给出站路由器，它包含了消息的入站和出站地址。入站路由器和出站路由器的中间是转发器，转发器的作用就是直接转发消息。

另外，本文通过参考 Mule ESB 的体系结构和文献[58]，提出了一些不同种类的入站路由器和出站路由器，归纳如下，它们可以满足 ESB 中不同应用场合下消息路由的需要。

（1）入站路由器

1）选择型路由器：符合所设置过滤准则的消息将被转发到相应的中介处理组件上。

2）聚集路由器：它可以接收多条消息，并将这些消息聚合成一条消息，然后将它进行入站处理。聚集路由器是以选择型路由器为基础的，在完成聚合功能的同时，它上面也可以设置不同的过滤准则来过滤消息。

3）重排路由器：它将对收到的一组消息进行重新排序，然后将它们按照排好的顺序转发给中介处理组件进行处理。它也是以选择型路由器为基础的，可以在其上设置不同的过滤准则。

4）去重路由器：它能确保中介处理组件不会收到重复的消息。判断消息是否重复，是根据消息的某个属性，比如 ID 是否重复来判断的。该路由器同时也指明了存储该属性的文件地址，供消息入站时使用。

（2）出站路由器

1）过滤路由器：和入站路由器的选择型路由器类似，它上面可以设置不同的过滤器，用来将匹配过滤准则的消息转发到一个或多个消息接收者。

2）链式路由器：它将消息发送到多个接收结点，但是它采用上一个结点的输出作为下一个结点的输入来将这些结点串连起来。它尤其适用于将同步的请求一响应结果发送到下一个处理结点的场景。

3）广播路由器：将中介处理组件处理完成后的消息转发到多个不同的消息接收者。在路由过程中，通常需要根据不同消息接收者的要求对消息进行必要的格式转换，使得对方能够正确接收。

4）分片路由器：将中介处理组件处理完成后的消息分成多个不同的部分或片断，并将这些部分转发到不同的消息接收者。分成不同片断的消息通常都携带有一些顺序信息，比如 ID，这样方便消息接收者使用入站路由器，如聚集路由器或重排路由器对它们进行处理。

5）容错路由器：在它上面通常设置有多个端点，消息通过它出站时，总是选择第一个能够完成消息出站的端点，即如果第一个端点不行，再试第二个，不行再试第三个，直到找到合适的端点。

这些不同的路由器，可以被组合和配置，从而形成复合路由器，可以适用于更复杂的应用场景。

5.2.3 消息路由器的过滤机制

前面提到了 ESB 在转发消息时，它是在中介者之间进行消息的转发，从而最终将消息转发给消息接收者。在中介者之间进行转发，有不同的转发策略，本文称之为过滤机制。

1．主流的过滤机制

目前，根据对 ESB 中过滤机制的分析，有如下三种主流过滤机制：

（1）基于消息内容的过滤

基于内容的过滤，指的是可以根据实时消息或事件的内容来将消息转发到下一个消息接收者。请求消息通常是基于 XML 的，XPath 是一种专门用来在 XML 文档中查找信息的语言。当一个 XML 服务请求消息进入 ESB 后，它通常要经过一系列中介者即中介处理结点，形成一条 XML 消息的服务（中介）处理序列，XML 消息必须经过这条服务处理序列来完成一次完整处理，最后 XML 消息才被送到正确的目标服务提供者。

ESB 中传输的消息通常基于 XML，我们可以使用 JXPath 来查找和定位信息。基于 JXPath 来实现基于内容的过滤器，该过滤器将阻止那些不符合过滤条件的消息，如图 5-7 所示。

```
<filter expression="(msg/header/resultcode) = 'success'"
//过滤器的过滤条件设置为"(msg/header/resultcode) ='success'"
className="org.apache.JXPath.xml.JXPathFilter"/>
//表示使用基于 JXPath 来实现的消息内容过滤器
```

图 5-7 基于内容过滤的配置文件

（2）基于消息属性的过滤

消息属性包含了消息的重要信息，它也可以用来作为路由过滤准则。图 5-8 是一个使用 HTTP 的 SOAP 请求实例。

```
   POST/StockQuote HTTP/1.1
   Host: www.stockquoteserver.com
   Content-Type:text/xml
   charset = "utf-8"
   Content-Length: nnnn
   SOAPAction: "Some-URI"
<SOAP-ENV: Envelope
   xmlns:SOAP-ENV = "http://schemas.xmlsoap.org/soap/envelope/"
   SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <SOAP-ENV:Body>
     ……
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

图 5-8 使用 HTTP 的 SOAP 请求实例

例如，在路由配置文件中，我们可以添加如下代码，如图 5-9 所示。

```
<filter expression="Content-Type:text/xml"
//用来表明过滤准则为消息属性是 Content-Type，值是 text/xml
className="org.apache.routing.filters.MessagePropertyFilter"/>
//表示使用 MessagePropertyFilter 消息属性过滤器
```

图 5-9 基于消息属性过滤的配置文件

（3）基于消息类型的过滤

消息都是属于某一类型，如 String，或者也可以是自定义的类型如 org.apache.broker.esb.routers.BankQuote 等，我们可以使用基于消息类型的过滤器根据消息的类型来进行过滤，如图 5-10 所示。

```
<filter  expectedType="java.lang.String"
//指定过滤准则－消息类型为 String 类型
className="org.apache.routing.filters.PayloadTypeFilter"/>
//表示使用 PayloadTypeFilter 消息类型过滤器
```

图 5-10 基于消息类型过滤的配置文件

这些过滤模式的出现较好地解决了 ESB 中消息路由问题，但是随着 IT 的发展和企业业务的日趋复杂，企业应用集成系统需要更多类型的过滤模式。本文根据企业实际业务的需要，提出了一种基于优先级和一种基于规则的过滤机制，它们和其他过滤机制的结合能够使得 ESB 中的消息路由机制更丰富。

2．一种基于优先级的过滤机制

ESB 通常应该具备消息的优先级控制功能，以保证实时业务服务质量和适应业务的多样性。在 ESB 对消息进行过滤的场景中，优先级控制功能是指可以按各类请求消息的优先级高低来控制服务质量，例如当不同优先级的消息在消息队列中等待处理时，应该保证优先级高的消息先得到处理，而优先级低的消息在缓冲器或队列中等待。但是目前在各主流 ESB 中，如 Mule，ServicesMix 等，都没有提供对基于优先级过滤的支持，针对这种情况，本文引入了一种基于优先级的过滤机制。它首先确定接收到的消息的初始优先级，然后使用当前优先级计算算法来选择一个当前优先级最高的消息，接着将它转发到消息接收的下一站地址。

首先引入几个符号，来描述我们的算法：

Q：消息队列；

M_i：表示队列 Q 中的消息，即 $M_i \in Q$；

$M_i.T_s$：消息 M_i 进入 ESB 的时间；

$M_i.T_e$：计算消息 M_i 当前优先级的当前时间；

$M_i.P$：消息 M_i 的初始优先级；

P_{max}：被转发消息的当前优先级；

M_{max}：被转发的消息；

当外部应用程序/服务发出的请求消息进入 ESB 后，ESB 中的某个特定中介者会对接收到消息进行初始优先级的确定，这时通常是根据某种分级标准，如消息的某个属性等进行分级的。这时候确定的初始优先级是静态的，在整个请求过程中它都不会再发生改变。但是消息的过滤除了参考初始优先级外，还需参考动态因素，这里是消息从进入 ESB 到调度该消息进行当前优先级计算的时间差，这两者的乘积为当前优先级，即紧急因子 K，它表示了 ESB 中请求消息过滤的紧急程度。如下面的公式所示：

$$K = M_i.P \times (M_i.T_e - M_i.T_s)$$

根据以上的论述，我们可以提出一种基于优先级的过滤算法。该算法中，拥有最大的紧急因子的消息将被赋给 M_{max}，而该消息也即将被过滤出去。

基于优先级的过滤算法如图 5-11 所示。

```
输入：消息队列 Q
Pmax←0；
for（Mi∈Q）
      {
       K = Mi.P×(Mi.Te－Mi.Ts)；
       if（K > Pmax）
          {
              Pmax←K；
              Mmax←Mi；
          }
       }
发送 Mmax；
```

图 5-11 基于优先级的过滤算法

3．一种基于规则的过滤机制

随着大量业务的变化，导致企业应用集成系统不停的变更，这时我们希望能找到一种解决商业逻辑的架构，来解决当商务和业务不停的变化时，可以保证我们的应用系统具有较好的柔韧性，可以适应特定的商务和业务的变化。目前，ESB 基本上都还没有实现基于规则将消息在连接到 ESB 上的应用之间进行转发的机制。针对企业应用集成系统的环境和特点，在确定 ESB 中消息路由的下一个结点时，本文提出了一种基于规则的过滤机制。消息通过规则集以确定基于哪种规则进行过滤，这样经过过滤后的消息就确定了服务请求消息转发的下一站点地址。

规则使得开发人员能够轻松地定义、更新和管理关键的决策以及监管业务流程和应用程序的策略，例如业务流程内部变化的业务策略可以使用规则来收集。利用它就可以在应用系统中分离商业决策者的商业决策逻辑和应用开发者的技术决策，让它们能在运行时可以动态地管理和修改，从而为企业保持灵活性和竞争力提供有效的技术支持。

一条规则可以看作是“IF…THEN…”语句块，或者一个简单的 IPO（即输入、处理和输出），描述了一组输入，一组判断和一组输出。

规则集是规则容器，所有使用到的 Rule 都加载到规则集中，由规则集来管理。规则集的基本设置如图 5-12 所示。

```
<ruleset>
  <rule>
    <condition>
        Condition     //具体的条件
    </condition>
    <action>
        Action        //后续的动作
    </action>
  </rule>
</ruleset>
```

图 5-12 规则集的基本设置

所有的规则都以 ruleset 元素内的 rule 元素的形式出现。Contidion 内可以将条件与逻辑运算符链接，Action 表明通过规则匹配的后续动作。

基于规则的过滤包含了一个规则编辑工具、一个规则引擎和 SDK。该编辑工具可供编程人员和业务分析人员共同使用，声明规则。规则引擎是一个用 Java 编写的快速、有效地与 JSR-94 兼容且基于 RETE 的引擎。SDK 则通过规则编辑应用程序来支持规则生成。如图 5-13 所示，基于规则过滤的工作机制为：

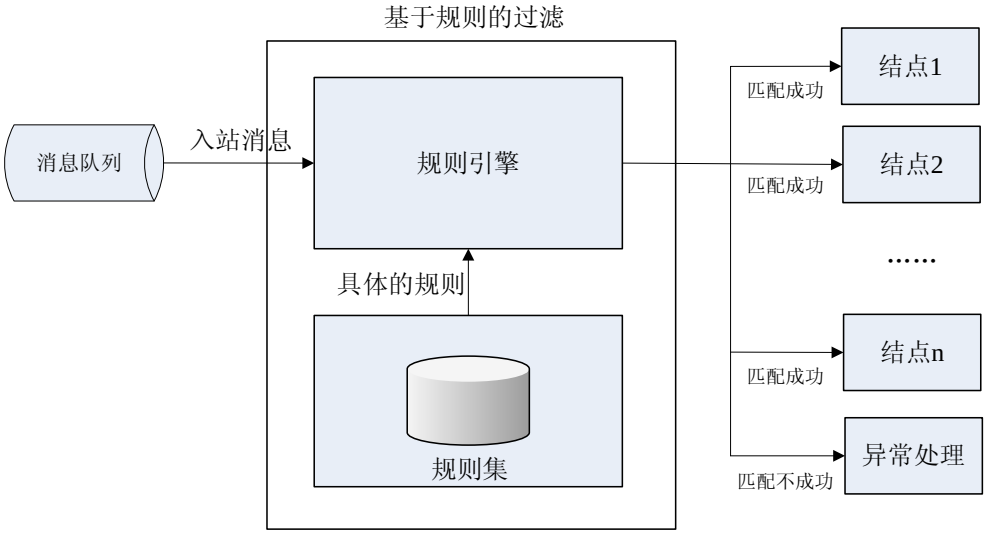


图 5-13 基于规则过滤的工作机制

Step1：路由规则编辑器根据相应的路由规范生成规则集，它里面是具体的规则；

Step2：来自服务请求者的请求消息中的路由关键部分被识别出来，并与规则集里面的具体规则进行匹配，以此来决定该消息的下一步处理。如果匹配结果为真，即成立，转 3；如果匹配结果不为真，则转 4；

Step3：按照定义顺序执行与此规则相关的后续动作，如将结果送到正确的下一个服务接收者进行处理，转 6；

Step4：如果规则集里面还有下一条规则，则取出该条规则，转 2；如果规则集里没有规则了，则转 5；

Step5：没有规则与该消息匹配成功，将该消息转发到异常处理模块进行处理，转 6；

Step6：基于规则的消息过滤过程结束。

在 SEE-ESB 中，可以通过集成合适的规则引擎来实现基于规则的过滤。开源 Drools[59]就是这样的一个规则引擎，它被设计为可插入式的语言实现。Drools 使用域描述语言（Domain Specific Languages，DSL）－专为问题域定义了某种模式的 XML，DSL 包含的 XML 元素（Element）和属性（Attribute）代表了问题域中的各种要素。

SEE-ESB 通过调用规则引擎 Drools 完成对规则配置文件的解析，然后对于每一个进入 SEE-ESB 中的消息，它都会检查以确定消息是否匹配定义在规则配置文件中的具体规则。任何匹配的消息均会转发至在规则配置文件中指定的目的地。

5.3 本章小结

本章的主要工作围绕服务适配网关和 ESB 中消息路由机制的研究而展开。本章先提出了服务适配网关的工作模型，分析了其工作机制，并且设计了一个 EJB 服务适配网关的实例。在消息路由机制研究方面，本章提出了通用中介者和消息路由器的体系架构，并且在分析主流过滤机制的基础上另外提出了一种基于优先级和一种基于规则的过滤机制，从而使得 SEE-ESB 中的消息路由机制更丰富。

第六章 基于CreatorSOIF的电子政务并联审批原型系统

6.1 应用场景

电子政务并联审批系统能迅速的搭建起面向具体业务的办公环境，以及方便的增减和修改原定的业务内容和业务流程，快速有效地处理政府的审批、审核、登记、核准等业务，并且能够实现多项业务、多个部门的联合办公，真正的实现政府网上审批的互联互通、信息共享和交换。特别是重大项目或服务对象的申请需要两个以上部门审批的事项的联办件，需要实行内部流转审批或并联审批制。如市规划局的建设工程规划许可行政审批项目，需要建设局、消防大队、环保局的联合办理。

发起单位规划局窗口接收申办单位的基本材料，经初步审核材料决定受理以后，建设局、消防大队和环保局进行联合审批，其中，建设局进行建设工程初步审计审查，消防大队进行建设工程消防设计审核，环保局进行建设工程环境影响报告书审批。经三局联合如有一局审查不通过则流程扭转到规划局窗口告知申办单位审查结果；若全部审查通过，规划局业务科组织现场勘查，主管领导审查，以及进行算费，最后办结告知。在此过程中可能联办单位有本局业务系统，而这些部门的业务系统又是相对独立建设的，不存在一个统一的标准。因而，对这样涉及多个部门业务系统的流程采用 ESB 作为架设系统互连的基础平台将能充分发挥原有系统的能力，同时提高效率。

电子政务并联审批系统通常涉及到多个具备自治性、分布性和异构性的部门应用系统[60]。我们基于 CreatorSOIF 来构建一个电子政务并联审批系统，该系统的核心是 ESB。这里采用开源 Mule ESB 作为 CreatorSOIF 的集成基础设施，并且针对开源项目 Mule ESB 存在的未提供服务注册及发现机制、以及路由过滤机制种类少等不足，我们对 Mule ESB 进行了扩展，并在电子政务并联审批原型系统中得到了应用。

6.2 开源 Mule ESB 的研究

6.2.1 Mule 的体系结构

Mule 是由 SymphonySoft 公司在 2003 年发起的开源 ESB 项目。它是一个轻量级的消息框架，同时也是一个分布式的对象代理，能够无缝的用不同的技术、传输和协议来处理与其他应用之间的交互。它提供了一个可以部署业务组件（business component）的高度分布的环境，能够透明地管理处于不同 VM（虚拟

机）上或者 Internet 上组件之间的交互，而不用考虑底层的传输细节[10]。

Mule 围绕着 ESB 架构来设计，这个 ESB 架构确定了不同的组件或者应用通过一个公共的消息总线来进行通信，这通常是用 JMS 或者是其他的消息服务器来实现，Mule 将 JMS 和其他的传输技术从在总线上接收消息的业务对象（business object）中抽取出来。

Mule 的最终目标是提供与不同源地址的数据进行交互的统一方法，而不妨碍开发人员了解数据的发送、接收方式以及协议等细节[10]。这样使得 Mule 是一个具有高度可伸缩性，轻量级，并且使用快速简单的 ESB。它的首要目标则是简化和加速开发分布式服务的网络的过程。

6.2.2 Mule 的核心功能模型

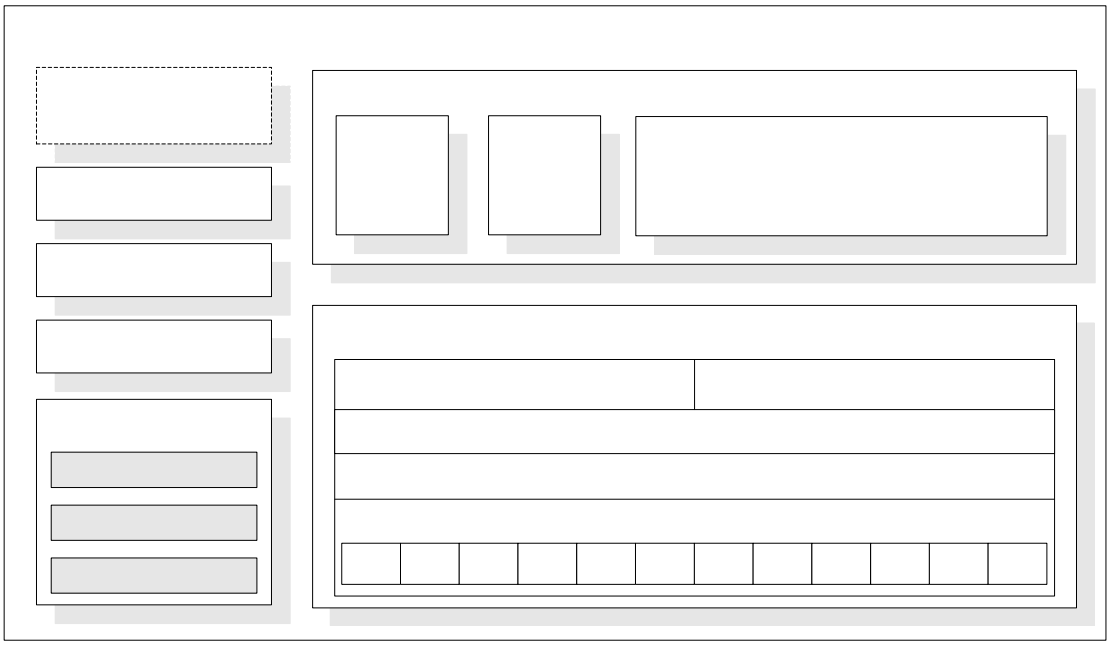


图 6-1 Mule 的核心功能模型

Mule 的核心是 Mule 管理器（Mule Manager）。它管理着这个模型的所有核心服务的配置以及它管理的组件。主要由 Mule 模型（Mule Model）、传输管理（Mule Transport）和代理（Agents）等组成，文献[10]提出了 Mule 的核心功能模型，如图 6-1 所示。

其中，Model 封装和管理一个 Mule 容器实例的运行时行为。它负责维护 UMOs 实例和它们的配置。Mule 配置机制来决定 Mule 怎么样与它的 UMO 组件进行交互。进入点分解器（Enter Point Resolver）用来在事件被接收时，决定 UMO 组件上调用的方法。生存周期适配器（Lifecycle Adapter）将 Mule 组件生存周期映射到基础的组件上。默认的生命周期适配器（Default Lifecycle Adapter）直接将生存周期事件指派给组件，而这些组件实现了 0 个或者多个 UMO

生存周期接口。组件池工厂（Component Pool Factory）模型定义了为一个给定的UMO组件创造一个组件池时所采用的工厂类。传输提供者（Transport Providers）是一个Mule插件，能够使得Mule组件在一个特殊协议，储存库（repository）消息或者其他技术上发送和接收信息。转换器（Tranformer）用来转换来自不同类型的消息和事件负载，路由器（Router）结合过滤器（filter）来完成消息路由功能。

6.2.3 Mule的不足

Mule因其易扩展等特性，得到使用者偏好，我们的原型系统也架构于Mule ESB之上。本文通过对Mule实现机制及其源码等的分析，得出Mule的不足之处如下：

1．Mule中没有提供统一的服务注册、发现机制

目前最新的 Mule1.4 版本中，仍未提供统一的服务注册和发现机制，对于ESB的客户端而言，它必须明确地知道Mule所提供的各类异构服务的地址，这个不足，严重的影响了Mule ESB的广泛应用。

2．Mule服务器与客户端的紧密耦合

基于Mule的应用系统中，对于每个Mule客户端来言，在初始化MuleClient时，必须各自启动一个Mule服务器，通过多个Mule服务器间相互通信提供分布式服务，这种模式，造成了客户端与服务端代码的紧密耦合，从而降低了应用系统的分布性。

3．Mule的分布式执行模型增加了客户端系统的负载

为了使用Mule管理的异构服务，每个客户端均需启动各自的Mule服务器，必然将增加客户端负载，易形成负载过重，影响客户端速度等各方面性能。

4．消息过滤机制有待加强

随着业务的发展，需要更多更强大的消息过滤机制。需要丰富Mule ESB现有的消息过滤机制来使其更好的应用于面向服务的集成。

5．服务流程编排不方便

基于Mule的业务流程里的服务种类不只包括Web服务，还有其他服务，如EJB，POJO等。这克服了BPEL的不足，但是基于Mule的服务编排没有可视化的编排工具,如果开发人员没有对Mule的工作机制和各组件的功能很清楚的话，那么就会带来不便。

6.2.4 Mule的路由过滤机制的扩展

在我们的电子政务并联审批原型系统中，审批请求有轻重缓急之分，需要根据优先级来路由审批服务请求。因此，我们对Mule进行了扩展，加入了基于优

先级的过滤机制，并将其应用到了入站路由器中。

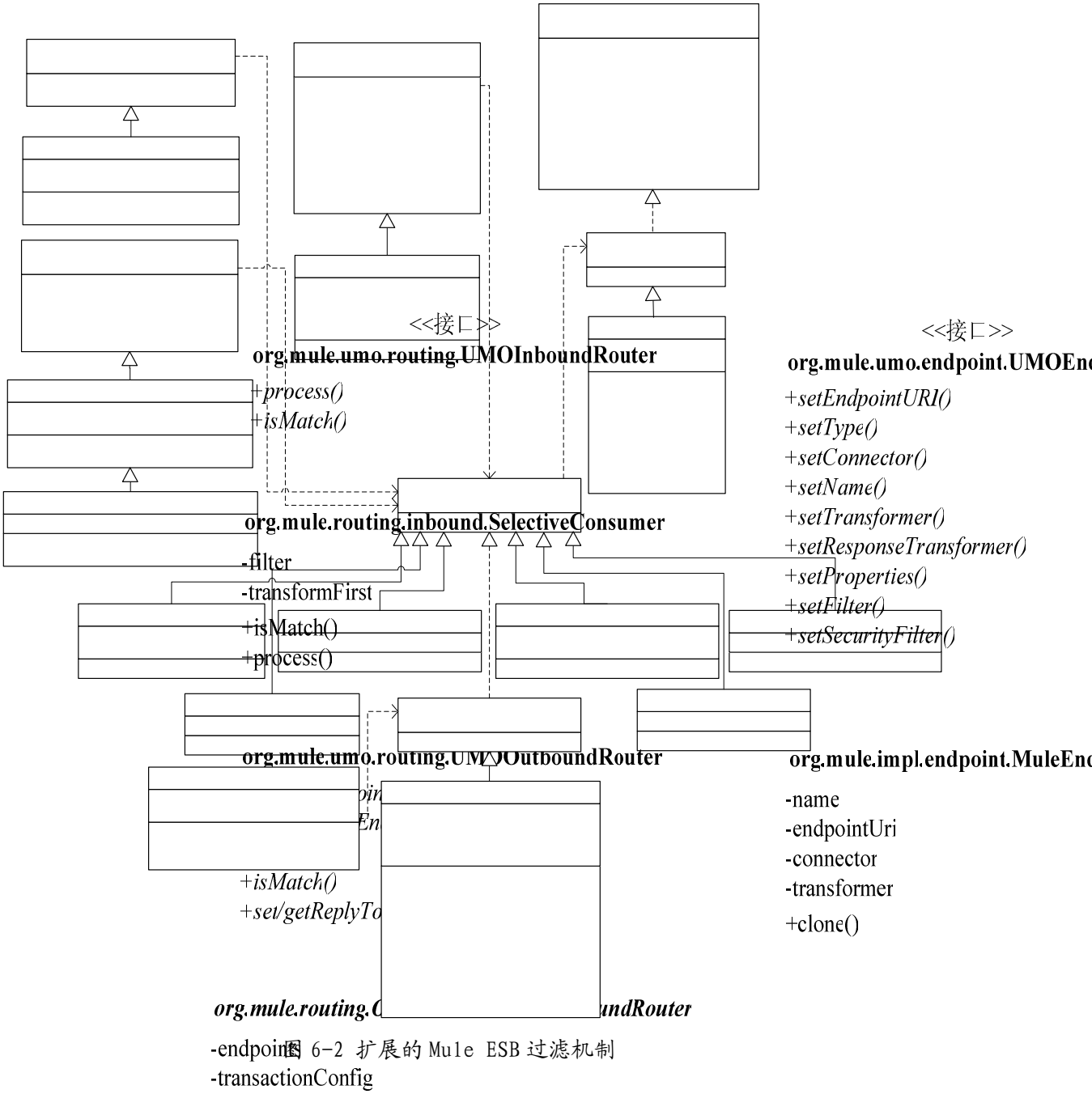


图 6-2 扩展的 Mule ESB 过滤机制

Mule ESB 中采用不同的 Filter 来过滤消息，并且 Mule 支持过滤机制的扩展。它提供了一个接口 UMOFilter，如果想要实现不同的 Filter，需要继承该接口，同时需实现它里面的重要方法 accept()。如图 6-2 所示，org.mule.umo.UMOFilter 是 Mule ESB 用来过滤消息的基类，ExceptionTypeFilter、MessagePropertyFilter 等过滤器都继承它。这些过滤器中的 accept(UMOMessage message)方法根据不同的过滤准则来过滤消息，如果消息和过滤准则匹配则返回布尔值 true，否则返回 false。

我们在对 Mule ESB 优先级过滤机制进行扩展时，遵循了 Mule 的过滤扩展

机制。由于基于优先级过滤需要对接收到的所有消息进行优先级计算和排序，这实际上是一个异步过程，而 Mule ESB 里面的其他过滤机制都只需在消息到达时，直接采用过滤器进行过滤，这是同步的过程。因此，我们首先让 org.mule.umo.AsynFilter 接口继承 UMOFilter 接口，然后再让 PriorityFilter 类继承 AsynFilter，接着引入了调度引擎－ScheduleEngine。ScheduleEngine 首先得到队列中的全部消息即 eventsGroup，然后使用 getFilter()方法得到的 AsynFilter 过滤器，完成 eventsGroup 中所有消息当前优先级的计算，最后经过 isMaxpriority()方法来验证该消息的当前优先级是否是最高优先级，如果是的话，则返回该消息，它即是我们使用基于优先级过滤机制所选出的入站消息。

6.3 电子政务并联审批原型系统的设计

6.3.1 总体结构

该原型系统基于 CreatorSOIF 而构建，这里给出了电子政务并联审批应用总体架构，如图 6-3 所示。它的功能是完成规划局的建设工程规划许可行政审批服务。

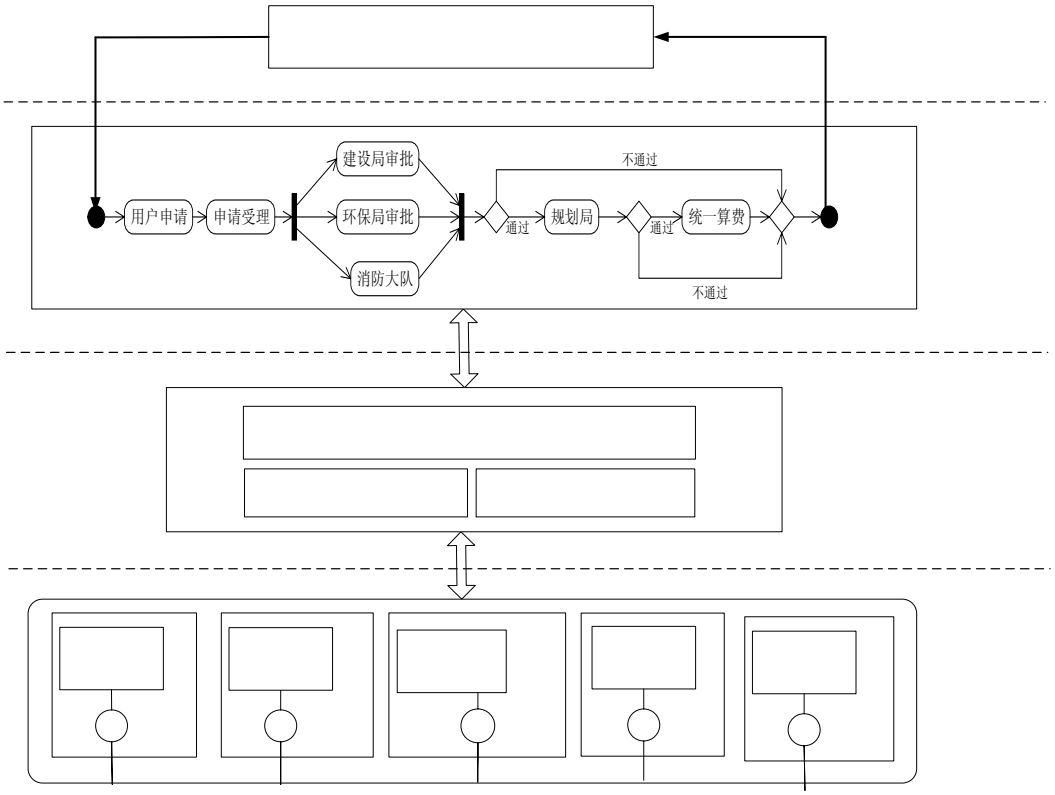


图 6-3 电子政务并联审批原型系统总体架构

服务资源层中涉及到的部门包括规划局、环保局、建设局、消防大队和财政部门，这些部门的业务系统是相对独立建设的，不存在一个统一的标准。我们采

规划局的建设工程规划许可行

用 Mule ESB 将它们通过各种适配技术连接到 ESB 上来。

ESB 服务层中有 Mule ESB，Tomcat Web 服务器以及 JBoss 应用服务器。该层的核心是 ESB，它能够实现到何种程度也就意味着最终能够实现多大程度的应用集成。针对上节提出的 Mule ESB 的不足之处，我们对其进行了扩展。 它可以对建设局、环保局及消防大队等所提供的政务服务进行注册管理，对于每一个需要进行审批项目的用户来言，这些服务是透明的，基于 CreatorSOIF 而构建的系统中的 Mule ESB 会自动完成相应服务的查找。

业务服务流层是该原型系统实现的建设工程规划许可行政审批服务的具体流程。基于 Mule ESB 的业务流程是通过流程配置文件来进行配置的，该配置文件基于 XML，具有丰富且灵活的表达方式，不仅支持 Web 服务的集成，而且也支持其他服务如 EJB，POJO 等的集成。

客户应用层对请求审批用户展现了一个建设工程规划许可行政审批服务。审批请求用户通过使用该服务完成相应的审批过程，具体的服务实现细节如服务位置，访问协议等对请求审批用户而言都是透明的。

6.3.2 系统组件介绍

建设局、消防大队、环保局以及规划局的业务系统通过提供服务或者 EJB 等方式插入到 ESB 上，如图 6-4 所示。

在该系统中的组件包括：

受理代理（Admission Broker）：接收审批请求，将受理请求转发给审批部门网关，以及收集计费结果并返回给审批请求者；

审批部门网关（AudDep Gateway）：验证审批请求，并可能发送到审批部门业务服务系统；

审批部门服务（AudDep Service）：接收审批部门网关发过来的请求，基于审批请求者请求审批的资料，进行审批请求者的审批；

部门认证网关（ColAud Gateway）：分派各审批请求者的请求信息到相应的部门进行审批；

规划局网关（PlanBureau Gateway）：收集各部门结果，基于优先级过滤机制来编排总线与规划局服务之间的请求；

规划局服务（PlanBureau Service）：根据审批请求进行相关项目的审批；

计费网关（Toll Gateway）：编排总线与计费应用之间的请求；

计费服务（Toll Service）：对审批请求者的请求进行计费。

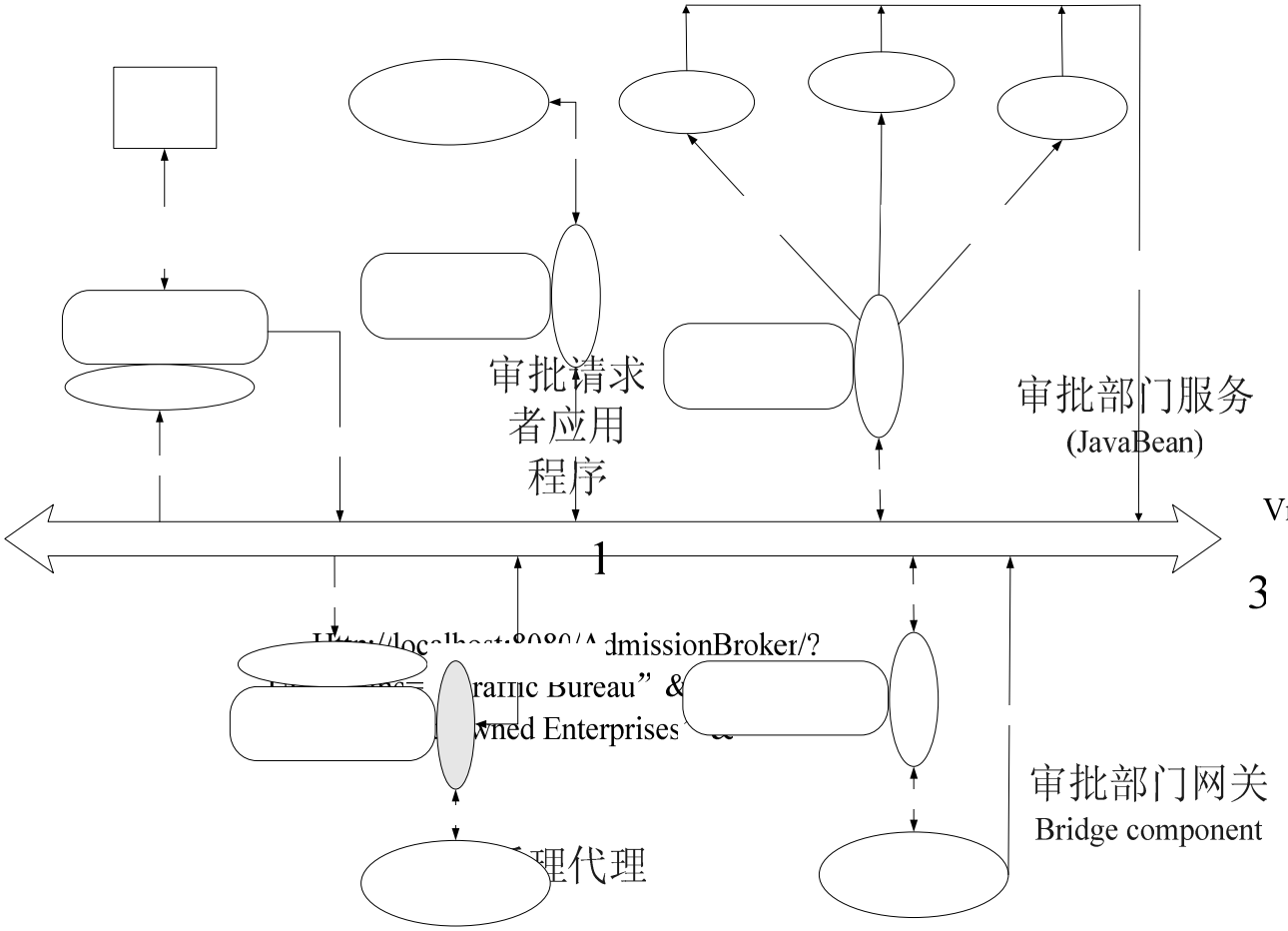


图 6-4　电子政务并联审批原型系统中的组件

规划局在对审批请求者进行审批时，它需要依据审批请求者的轻重缓急进行一一审批。因此，我们采用在 6.2.4 中描述的经扩展的优先级过滤机制。规划局网关的路由器配置文件代码如图 6-5 所示。

```
<filter className=“org.mule.routing.filters.logic.AndFilter”>
    <left-filter expectedType=“java.lang.String”
        className=“org.mule.routing.filters.PayloadTypeFilter”/>
    < right-filter expression=“auditPriority”
        className=“org.mule.routing.filters.PriorityFilter”/>
</filter>
```

图 6-5 规划局网关的路由器配置文件

在这段过滤器的配置文件中，我们采用到逻辑过滤中的 AndFilter，即与路由器，它意味着只有同时满足 left-filter 和 right-filter 所表示的过滤条件情况下，才能通过此 AndFilter 进行过滤。我们首先要判断审批请求者的类型是否为字符串型，然后再选择具有最高优先级的审批请求进行规划局审批操作。配置文件中的 expression=“auditPriority”表示审批请求消息中代表优先级的属性的名称为

auditPriority，PriorityFilter 可以根据 auditPriority 先得到初始优先级，然后再计算当前优先级。经过该 AndFilter 的过滤后，所得到的是消息类型为串行，并且当前优先级为最高的消息。

6.3.3 系统交互模型

在扩展的 Mule 的支持下，通过事件传递各个组件得以联系到一起，完成并联审批流程。在 6.3.2 小节中，图 6-4 中的数字代表了具体的请求事件流，现描述如下：

客户应用程序发送客户审批请求到受理代理；

受理代理产生审批请求信息；

审批请求信息被发送到 Mule ESB 上，Mule ESB 通过 JMS 发送信息到审批部门网关；

审批部门网关运用 VM 传输方式调用审批部门服务；

处理完成后的请求信息被发送到 Mule ESB 上，Mule ESB 通过 JMS 发送到部门认证网关；

部门认证网关运用 SOAP 调用审批部门，这里的审批部门有三个，分别是建设局、环保局和消防大队；

各个审批部门将审批请求信息关联到部门审批流程中，根据规划局网关提供的地址（ReplyTo：jms://esb.audit），将审批结果提供给规划局网关；

规划局网关运用 EJB 方式调用规划局服务，规划局服务中的响应路由器接收 ReplyTo 地址上的信息，选择各审批部门通过的客户，并进行相关项目等的审批；

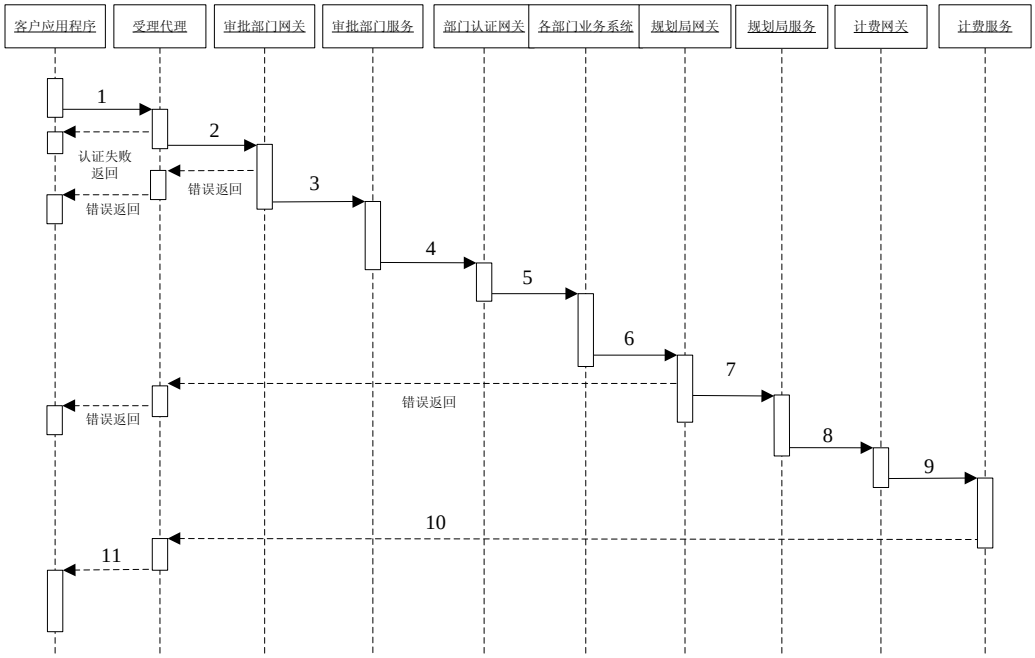
上一步传来的信息中携带了规划局服务的处理结果，Mule ESB 通过 JMS 发送审批结果到计费网关；

计费网关运用 VM 传输方式调用计费服务进行计费操作；

计费服务将审批请求信息关联到计费模块中，然后根据受理代理提供的地址（ReplyTo：jms://esb.tol），将计费结果提供给受理代理；

受理代理服务中的响应路由器接收 ReplyTo 地址上的信息，将计费结果等信息提供返回给客户。

整个系统交互图如图 6-6 所示。



1、http://localhost:8080/AdmissionBroker/?UnitsName=”Traffic Bureau”&UnitsType=”Stated-owned Enterprises”&…
2、jms://esb.audPro.service
3、vm://audPro.service
4、jms://esb.audDepartments
5、axis:http://localhost:X0080/mule/?method=getaudresult
6、reply-to jms://esb.audit
7、ejb://localhost:1099/AuditHome/getAudResult
8、jms://esb.toll.service
9、vm://toll.service
10、reply-to jms://esb.toll
11、返回算费结果

图 6-6 系统组件交互时序图

6.4 系统运行情况及分析

6.4.1 运行情况分析

审批请求者打开申办人基本信息界面，如图 6-7 所示。

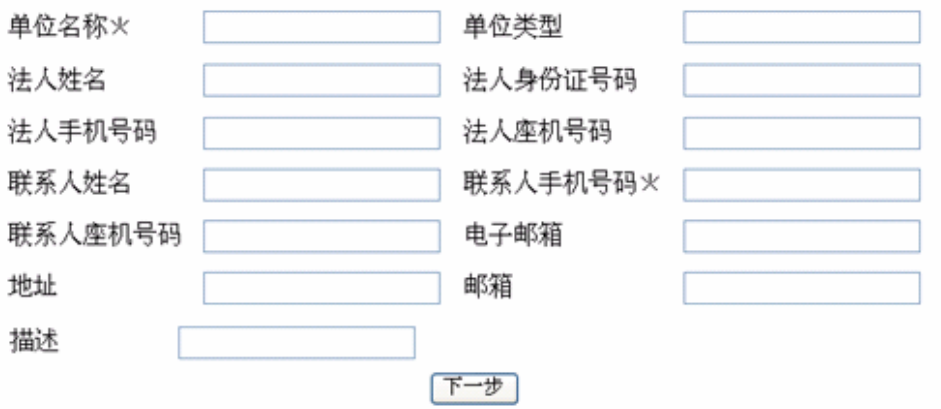


图 6-7 电子政务并联审批原型系统申请受理界面

请求者输入的信息，经受理代理转换消息类型，此消息通过 jms 传送到审批代理，审批代理通过 VM 方式调用审批部门服务，此服务依据请求者请求的审批类型来确定具体审批单位。建设局的执行情况如图 6-8 所示。

根据电子政务并联审批原型系统总体架构，在扩展 Mule 基础上，我们实现了此并联审批原型。对于业务用户，在初始化客户端时，按照配置文件来执行相应的服务。通过访问规划局网关访问规划局服务的审批请求的先后秩序有区别，这里是根据优先级来进行消息的过滤。通过过滤后的消息再发往规划局服务进行具体的审批。这样的设计也考虑了政务并联审批的实际需要，实用性也更强。

图 6-8 审批部门服务调用建设局审批服务的执行界面

6.4.2 优势与不足

传统的电子政务并联审批系统中，如果业务用户的服务需求发生了更改，我们就要对整个流程进行相应的变更，这就涉及到一个系统的重新设计或重构等问题，相应地也会带来软件开发周期延长及软件代价增加等问题。在我们基于 CreatorSOIF 实现了的电子政务并联审批原型系统中，其优势有：

1．ESB 是 CreatorSOIF 的核心，业务服务对用户来言是透明的，用户只需利用统一的接口如 Web 服务接口来调用服务，无须知道服务的地址及变更情况等。

2．各个审批部门提供的服务为“即插即拔”型，我们可以按照需要灵活且轻松地实现政务审批流程的更改，即如果此应用中需要增加或者减少职能部门的审批环节，只须将要增加或删除部门的相关服务通过 ESB 进行注册或通过 ESB 进行删除，无须对整个审批流程进行更改。这可以快速地开发新的业务服务，并缩短开发时间。

3．系统灵活的路由过滤机制的采用，也使得其更具实用性。

4．基于网络的、轻便的 Mule ESB 更容易逐步升级。

5．Mule ESB 已经在一些集成领域得到了不同程度的应用，开源的本质突出了它的成本优势。

另外，通过此并联审批系统，可以规范政府职能部门的各项工作流程，提高办事效能及服务质量，增加政府行政的透明度，大大提高了政府公信力；另一方面可以积累相关数据，为政府职能部门绩效考核体系提供确实而有效的评估依据。

但是该原型系统也有一些不足之处。在设计该原型系统时，我们着重考虑的是该系统对电子政务业务服务的支持程度，但是像安全性和事务性等因素并没有加以考虑，所以如果要想增强该系统的实用性则必须在这几个地方有所突破。此外，该系统中采用的 Mule ESB 还不能支持图形化界面的流程编排。由于 Mule 的开源特性，和其他商业产品相比，它的用户文档和资料较少，不易收集，不便于开发人员上手，企业的培训、管理和维护成本也要相应增加。

6.5 本章小结

基于对开源 Mule ESB 源码和工作机制的分析，针对 Mule ESB 在分布式应用开发中的不足，对其进行了扩展，即根据服务请求的轻重缓急，加入了基于优先级的过滤机制，并将其应用到了 ESB 的入站路由器中。本章的重点是实现了一个电子政务并联审批原型系统，它可以实现规划局的建设工程规划许可行政审批服务，它基于扩展的 Mule ESB 来构建，具有明显的优势。

第七章 总结与展望

7.1 本文总结

SOA 和 Web Services 技术与企业服务总线技术的结合为解决异构环境下分布式应用集成提供了有效的途径。企业服务总线可以有效克服长期以来传统企业应用集成的不足。本文在基于 ESB 的企业应用集成框架、基于服务执行引擎的企业服务总线参考模型、服务适配网关、消息路由器及其在电子政务并联审批系统中的应用等方面进行了一定的探讨，为企业应用集成提供了新的思路和方法。

本文的工作归纳起来有以下几点：

1．提出了一种基于 ESB 的企业应用集成框架 CreatorSOIF

通过对企业应用集成、面向服务架构以及关键技术如 Web Services 和 ESB 的分析，在此基础上，提出了一种基于 ESB 的企业应用集成框架－CreatorSOIF。这是标准化的集成框架，具有较好的可扩展性和动态性。CreatorSOIF 为面向服务的集成和电子政务集成系统的研究提供了问题背景和研究内容。

2．提出了一种基于服务执行引擎的企业服务总线参考模型 SEE-ESB

在对现有 ESB 模型深入分析和研究的基础上，通过分析它们的优势与不足，提出了一种基于服务执行引擎的企业服务总线参考模型 SEE-ESB。同时对参考模型的核心－服务执行引擎进行了研究，并提出了其体系结构。这些为面向服务的集成的进一步研究提供了基础。

3．对 SEE-ESB 中的服务适配网关的研究

服务适配网关可以实现服务请求者和服务提供者之间的松散耦合、位置透明、协议独立等特征。本文提出了其工作模型，它主要通过各种协议的适配器来调用各种异构的服务，可以屏蔽各种通信协议的差异性，为 ESB 提供统一的服务调用接口。

4．对 ESB 中消息路由机制的研究

消息路由器对消息进行转发和过滤。本文设计了消息路由器的体系结构，同时对进行转发和过滤消息的过滤机制进行了分析和研究，提出了一种基于优先级的过滤机制和一种基于规则的过滤机制。这些过滤机制和 ESB 中原有的过滤机制丰富了 ESB 中的消息路由机制，并能够满足企业集成复杂性和实际业务复杂性增加的需要。

5．提出了一种基于 Mule ESB 的电子政务并联审批系统原型

在扩展开源 Mule ESB 的基础上，结合本文的研究对象 CreatorSOIF、SEE-ESB、服务适配网关和消息路由器等，实现了一个电子政务并联审批应用系

统的原型。

7.2 进一步研究方向

企业应用集成中的企业服务总线技术研究目前是一个热点。我们在研究它的过程中，又发现了一些新的问题，这些问题的解决可以进一步改善和提高本文已有的工作，有的则可以作为企业服务总线新的研究方向。

1．ESB 中消息路由的进一步研究

目前现有的消息过滤机制种类不是很多，再加上本文提出的两种消息路由过滤机制后也有可能仍然不能满足日益复杂和多变企业集成需求。基于语义的 Web 服务发现和组合机制的研究成为了当前的研究热点，因此可以考虑在消息路由过程中采用基于语义的过滤/适配机制，这样能够使得 ESB 中的路由更具智能性。

2．ESB 的安全性研究

长期以来，企业应用集成的安全问题一直是用户关注的重点之一，本文提出的 SEE-ESB 模型时对安全性也并未过多论述。实际上，基于 ESB 进行企业应用集成不能不重视安全问题，对流经 ESB 的消息进行加密、数字签名，对用户进行安全认证、访问控制等方面的研究是一个有待进一步研究的方面。

3．ESB 集成标准的研究

关于 ESB，目前还没有被一致接受的标准，每一个软件供应商对它都可能有不同的理解和实现的策略。本文提出的 SEE-ESB 参考模型也是经过深入分析和对比目前主流 ESB 模型的基础上得出的一个模型。目前出现的 JBI 标准是第一个 ESB 标准，但是它没有得到大多数厂商的承认，从这几年来看，它有逐渐退出历史舞台的迹象。但是 JBI 毕竟也有一部分市场，而且对它的研究也有利于最终统一的 ESB 标准的推出。和 ESB 相关的标准还包括 WS-*协议，但大多数 WS-*协议仍处于起步阶段，这些 WS-*本身还处于频繁变化的阶段。

4．基于 Mule 的电子政务并联审批系统原型的实用化

目前在该原型系统中还没有完成对基于规则的过滤机制的实现，同时，开源 Mule ESB 的功能还不是很完善，比如它没有提供可视化的流程编排功能，部分制约了该原型的实用性。

参考文献

[1] Steve Graham. 用Java构建Web服务. 北京: 机械工业出版社, 2003. 4~6

[2] Lauri Jaakkola. Applying Service-Oriented Architecture to Geographically Distributed Industrial Information Systems: [硕士学位论文]. Helsinki: Helsinki University of Technology, 2005

[3] 吴昊, 刑桂芬. 基于Web Services的企业数据集成的研究. 计算机工程与设计, 2005, 26(10): 2725~2727

[4] 徐罡, 黄涛, 刘绍华. 分布应用集成核心技术研究综述. 计算机学报, 2005, (4): 433~443

[5] 东方通科技. 企业应用集成的现状和发展趋势. http://www.tongtech.com/js-qy/yqxwview.asp?id=475, 2005

[6] Barry & Associates. Service-oriented architecture definition. http://www.service-architecture.com/web-services/article/service-oriented_architecture_soa_definition.html, 2005

[7] 黄普. 图解ESB/SCA. 程序员, 2006, (1): 90~91

[8] Apache. serviceMix. http://incubator.apache.org/servicemix/home.html

[9] Apache. PXE-Process Execution Engine. http://incubator.apache.org/ip-clearance/ode-2-intalio-pxe.html

[10] Codehaus. Mule ESB. http://mule.codehaus.org/

[11] Ron Ten-Hove. JBI Components: Part 1. Sun Microsystems Inc., 2006, 3(9): 1~12

[12] JavaEye. SOA之2006年终回顾以及2007展望. http://www.javaeye.com/topic/39348, 2006

[13] 梁爱虎. 精通SOA: 基于服务总线的整合应用开发. 北京: 电子工业出版社, 2007. 227~228

[14] ObjectWeb. Celtix. http://celtix.objectweb.org/

[15] Zhaohui Wu, Shuiguang Deng, Ying Li. Introducing EAI and Service Components into Process Management. In: Proceedings of the 2004 IEEE International Conference on Service Computing, 2004

[16] Fred A Cummins. Enterprise integration: An architecture for enterprise application and system integration. 北京: 机械工业出版社, 2003

[17] Hammer K. Web services and enterprise integration. EAI Journal, 2001, (11): 12~15

[18] 陈志刚, 曾志文, 王红燕. 多层客户/服务计算模型及实现技术. 长沙: 湖南科学技术出版社, 2002. 42~97

[19] 陈松乔, 任胜兵, 王国军. 现代软件工程. 北京: 北京交通大学出版社, 2002. 140~152

[20] 李建华, 陈松乔, 马华. 面向服务架构参考模型及应用研究. 计算机工程, 2006, 32(20): 100~102

[21] 范玉顺, 李慧芳. 企业集成技术的研究与发展趋势. 中国制造业信息化, 2003, 32(1): 59~61

[22] 范玉顺, 王刚, 高展. 企业建模理论与方法学导论. 北京: 清华大学出版社, 2001

[23] 夏敬华. 走向未来的企业应用集成. http://www.amteam.org/k/Theory/2002-3/0/443600.html, 2002

[24] 张林平. 基于J2EE平台的面向服务体系结构的研究. 计算机工程, 2004, 30(12): 103~105

[25] Gartner Research. Service-Oriented Architecture Scenario. http://www.iturls.com/English/SoftwareEngineering/SE_SDA.asp, 2002

[26] Michael Huhns. Munindar Singh. Service-Oriented Computing: Key Concepts and Principles, IEEE Internet Computer, 2005, (1): 45~49

[27] 杨渊, 邵贝恩. ASP模式中面向服务的应用系统设计方法及实践. 计算机工程, 2005, 31(22): 218~221

[28] Loosely Coupled. Define meaning of SOA-Service Oriented Architecture. http://www.looselycoupled.com/glossary/SOA, 2006

[29] IBM. SOA术语概述. http://www.ibm.com/developerworks/cn/webservices/ws-soa-term1/index.html, 2007

[30] 技术中国. 架构设计师与SOA. http://www.technetchina.com/html/newsest/20070225/3730.html, 2007

[31] IBM. 用SLA保证第二代Web服务应用程序. http://www.ibm.com/developerworks/cn/webservices/ws-wssla/, 2004

[32] 李慧盈, 李德昌, 段羽. 利用J2EE构建XML Web Services的研究. 计算机工程与应用, 2004, (5): 128~130

[33] 杨涛, 刘锦德. Web Services技术综述－一种面向服务的分布式计算模式. 计算机应用, 2004, (8): 1~4

[34] W3C. XSL Transformations(XSLT)Version 1.0. http://www.w3c.org/TR/xslt, 1999

[35] 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. http://www-128.ibm.com/developerworks/cn/webservices/ws-esbscen2.html, 2004

[36] 李晓东, 杨扬, 郭文彩. 基于企业服务总线的数据共享与交换平台. 计算机工程, 2006, 32(21) : 217~221

[37] Gartner. Enterprise Service Buses Are Taking Off. http://www.gartner.com/DisplayDocument?ref=g_search&id=419193, 2003

[38] Michal P., Papazoglou, Paolo Traverso, Schahram Dustdar, Frank Leymann, Bernd Kramer. Service-Oriented Computing Research Roadmap. In: Dagstuhl Seminar Proceedings, 2006. 1~29

[39] IBM. ESB企业服务总线解决方案剖析. http://www.ibm.com/developerworks/cn/webservices/ws-esb4/, 2006

[40] Liisel Murre. Enterprise Service Bus and Web Services. In: Proceedings of the 2004 IEEE International Conference on Service Computing, 2004

[41] 罗海驰. 基于Web Services的电子政务体系结构及其应用. 计算机工程与应用, 2006, (32): 229~232

[42] 陶海燕, 曹书涛. EAI技术在电子政务集成中的应用. 计算机技术与发展, 2006, 16(2): 4~6

[43] 科拉夫兹格, 本克.斯拉姆. Enterprise SOA中文版－面向服务架构的最佳实战. 北京: 清华大学出版社, 2006

[44] 中和威公司. InterESB. http://www.intervision.net.cn/default.asp, 2005

[45] Dave Chappell. Enterprise Service Bus. Sebastopol: O'Reilly, 2004

[46] 许周毅, 金心宇. 基于JMS和MOM的信息交换系统研究. 计算机应用研究, 2005, (7): 199~201

[47] John Harby. ESB的选择. http://www.realesoft.com.cn/soa_forum/files/ESB%E7%9A%84%E9%80%89%E6%8B%A9.pdf, 2005

[48] William Tay. Routing Secured SOAP Messages Through Multiple SOAP Intermediaries Using WSE 2.0. http://msdn2.microsoft.com/en-us/library/ms977358.aspx, 2005

[49] Sylvanus A. Ehikioya, Suresh Jayaraman, Jose A. Rueda. Intelligent Content-Based Routing for Enhanced Internet Services. International Journal of The Computer, 2006, l(14): 523~531

[50] Yi-Min Wang, Lili Qiu, Chad Verbowski, Dimitris Achlioptas, Gautam Das, Paul Larson. Summary-based Routing for Content-based Event Distribution

Networks. ACM SIGCOMM Computer Communications Review, 2004, 34(5): 59~73

[51] German Sakaryan. WS-Eventing and content-based routing for integrating multiple applications via Web services. http://goliath.ecnext.com/coms2/gi0-199-3403565/Beyond-point-to-point-WS.html, 2004

[52] 金源, 李松年. 内容发布订阅服务网络中的路由策略. 计算机工程与应用, 2006, (12): 171~174

[53] M.T. Schmidt, B.Hutchison. The Enterprise Service Bus: Making service-oriented architecture real. IBM SYSTEM JOURNAL, 2005, 44(4): 46~51

[54] Pamela Zave. Requirements for Routing in the Application Layer. AT&T Laboratories-Research, 2006, 1~7

[55] Abdelkarim Erradi, Piyush Maheshwari. A Broker-based Approach for Improving Web Services Reliability. In: Proceedins of the IEEE International Conference on Web Services, 2005

[56] Colombe Herault, Gael Thomas, Philippe Lalanda. Mediation and Enterprise Service Bus-A position paper. http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-168/MEDIATE2005-paper5.pdf, 2005

[57] IBM. 使用服务组件体系结构构建SOA解决方案. http://www.ibm.com/developerworks/cn/webservices/sca-theme/sca.html, 2006

[58] Gregor Hohpe, Bobby Woolf. Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. New Jersey: Addison-Wesley, 2004

[59] Paul Browne. Give Your Business Logic a Framework with Drools. http://www.onjava.com/pub/a/onjava/2005/08/03/drools.html, 2005

[60] 尹天明, 李也白. 电子政务中构建工作流开发平台技术的研究. 计算机工程, 2006, 24(6): 275~278

致 谢

在我的课题和硕士论文完成之际，谨向在攻读硕士学位过程中曾经指导过我的老师，关心过我的朋友，关怀过的领导，和所有帮助过我的人们致以崇高的敬意和深深的感谢。

首先，要感谢我的导师李建华教授，感谢他三年来在学习、生活和科研工作中给予我的关怀、指导和照顾。李老师治学严谨、积极工作、以身作则、以及不断学习新知识的精神使我在学业和为人上受益匪浅。同时，感谢李老师给予我参与项目实践的机会，使我的动手能力有了较大的提高，这将对我以后的工作产生积极的影响。李老师对本论文的选题和研究提出了许多有益的建议，使我能够顺利完成论文。

感谢高琰师姐、马征师姐、马华师兄和徐海军师兄，他们在我的项目实践、论文研究及撰稿期间提出了很多宝贵的意见。感谢郝丽波、许甸、曹龄兮、夏媛、杨萍、何毅俊和李金同学，感谢师弟张慧、师妹李永军、曾慧琼和李桂林的支持和帮助，跟他们在一起学习、工作、讨论将是我终身怀念的事情。

同时，感谢我的家人为我的成长所付出的心血，以及多年对我的教育和培养；感谢其它亲人和朋友对我的鼓励和关心，感谢帮助过我的所有人们！

攻读学位期间主要的研究成果

已发表的学术论文：

[1] 丁昭华，李建华. 企业服务总线在企业应用集成中的研究与应用. 计算机应用与软件，已录用，拟刊在2008年3月（编号：611318）

[2] 丁昭华，李建华. 基于Struts和动态表单的MVC设计模式. 计算机时代，2006（9）： 41~42