火龙果•整理
uml.org.cn

上海交通大学

硕士学位论文

基于总线通道技术的SOA架构实现

姓名：刘斌

申请学位级别：硕士

专业：软件学

指导教师：吴刚

20060701

火龙果•整理
uml.org.cn

表目录

表 4-1 银行业务种类 23
表 4-2 认证服务类型列表 30
表 5-1 两种 ESB 技术比较 49

图目录

图 3-1 SOA 服务结构 18
图 3-2 SOA 的发展趋势 19
图 3-3 SCA 编程模型 21
图 4-1 eMuSC 运用体系 24
图 4-2 总线通道软件配置 25
图 4-3 Facade 设计模式 26
图 4-4 多种协议内部实现 27
图 4-5 服务骨架程序 29
图 4-6 服务绑定机制 30
图 4-7 服务的多对多关系 31
图 4-8 通道内部结构 32
图 4-9 通道定义规则 33
图 4-10 ejb_jar.xml 定义片断 34
图 4-11 soa.esb.iiop.BusConnectorIIOPLocalBean 实现类 34
图 4-12 扩展配置依赖关系图 35
图 4-13 流量控制 37
图 4-14 系统内部数据交互方式 39
图 5-1 分层结构图 41
图 5-2 自定义前后端总线通道模型 43
图 5-3 横向通道部署结构 44
图 5-4 纵向分层部署结构 45
图 5-5 异构平台构成图 46
图 5-6 分布式管理控制方式 47

火龙果•整理
uml.org.cn

上海交通大学
学位论文原创性声明

本人郑重声明：所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。对本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。本人完全意识到本声明的法律结果由本人承担。

学位论文作者签名：刘斌

日期：2006 年 10 月 17 日

火龙果•整理
uml.org.cn

上海交通大学
学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权上海交通大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

保密□，在___年解密后适用本授权书。

本学位论文属于

不保密□。

（请在以上方框内打“√”）

学位论文作者签名：刘斌　　指导教师签名：吴　刚

日期：2006 年 10 月 17 日　　日期：2006 年 10 月 17 日

火龙果•整理
uml.org.cn

基于总线通道技术的 SOA 架构实现

摘 要

随着软件产业的不断发展，企业对计算机技术依赖程度越来越高。软件也从单机的软件工具，发展为分布式，网络化，集信息自动化、数据存储、企业管理、企业策划等越来越复杂功能为一体的软件系统。企业的发展促成各种技术的软件不断涌现，但始终有一个问题困扰着企业，就是怎么使各个系统能整合在一起，安全的、规律的、流畅的、统一的工作。在这种需求越来越强烈时，软件学术界和工业界适时地提出了 SOA（Service Orient Architecture）面向服务的软件架构，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。通常接口是采用中立的方式进行定义，且独立于实现服务的硬件平台、操作系统和编程语言。服务之间是松耦合的，以使构建在系统中的服务可以以一种统一和通用的方式进行交互。

本文研究了目前分布式软件中 SOA 的应用现状，以及如何通过 ESB 技术平台来实现 SOA。文中介绍了 SOA 中使用的 WebService、BPEL、ESB、MOM 等关键技术，分析了 IBM 的 ESB 技术平台，并结合当前工作中正在开发的实际项目，给出了一个基于总线通道技术的 SOA 解决方案，其中详细分析了前后总线通道技术的细节，包括总线通道如何自定义多通道，如何实现多种传输协议的转换，如何灵活的部署和使用 JMX 动态监管运行状态。文中还从 SOA 架构角度阐述了总线通道技术对于 SOA 架构特点的满足性，并根据实际项目开发经验分析了两种有代表性技术方案和通道技术的异同点，努力使论文能清楚阐述产品集成和项目定制两种技术方案各自的优势和劣势。用户可以对两种不同的方案进行取长互补，并根据企业的实际情况来平衡两者的关系。

关键字：总线通道，SOA，ESB，SIBus，SCA，BPEL

火龙果•整理
uml.org.cn

THE IMPLEMENT OF SOA UNDER MULTICHANNELBUS

ABSTRACT

As the software's development, enterprise has more attention and action to computer science for working more efficient. As the demand the software from simple electric tool to distribute, internet, automatic, storage, enterprise resource plan system. With the IT technology more progress, it has a problem yet about how to integrate the inside and outside system with security, smooth, unify. Because of it’s critical demand. The software academic and industry announce the Service Orient Architecture which converts application to service unit then defines the loose and standard interface and contact to service. Therefore the interface adopts independence hardware, operation system, language to realize SOA system.

In the thesis, I will introduce the SOA distribution system how to use ESB design SOA software and describe the core technology about WebService, BPEL, ESB and MOM. Then I will present the IBM SOA implement solution with WebSphere Business Integration 5(WBI5) architecture and WebSphere 6 SIBus. According to CIS(Channel Integration System) Bank project, I will focus on how to use thinking of SOA to implement multi channel , how to switch multi-protocol ,how to deploy and use JMX to monitor and control the dynamic system. Trace back to the phase, about the challenge to developing of project to follow the rule of SOA, about the achievement to goal. Based on upon three SOA solution analyzes the ESB which is base infrastructure in SOA. To discuss the three styles of ESB advantage and disadvantage, and how to balance

火龙果•整理
uml.org.cn

ESB usage according to finance cost, business request, deployment. At last I will introduce how to use service orient method and analysis to guarantee the SOA project successful.

KEY WORDS: multichannelbus, soa, esb, cis, sibus, sca, bpel

火龙果•整理
uml.org.cn

第1章 引言

1.1 应用背景

SOA(Service Oriented Architecture)称为面向服务的软件架构。SOA 不是一种语言，也不是一种具体的技术而是一种软件系统架构，它尝试给出在特定环境下推荐采用的一种架构，从这个角度上来说，它更像一种模式(Pattern)。因此它与很多已有的软件技术比如面向对象技术，是互补的而非互斥的。它们分别面向不同的应用场景，用来满足不同的特定需求。它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。SOA 促进了可重用性，提供了接口和实现之间的抽象级别以最小化依赖关系，将业务需求与 IT 功能结合，从而可以为您提供用于将业务需求转换为编程服务来实现流程自动化的机制，以及当前竞争激烈且快速变化的业务环境中所必需的灵活性。以下这些相关因素通常会让企业建议采用面向服务的体系结构：

1．利用现有的资产

SOA 提供了一个抽象层，通过这个抽象层，企业可以继续利用它在 IT 方面的投资，将这些现有的资产包装成提供企业功能的服务。组织可以继续从现有的资源中获取价值，而不必重新从头开始构建。

2．更易于集成和管理复杂性

在面向服务的体系结构中，集成点是规范而不是实现。提供了实现透明性，并将基础设施和实现发生的改变所带来的影响降到最低限度。通过提供针对基于完全不同的系统构建的现有资源和资产的服务规范，集成变得更加易于管理，因为复杂性是隔离的。当更多的企业一起协作提供价值链时，会变得更加重要。

3．更快的响应和应用速度

从现有的服务中组合新的服务的能力为需要灵活地响应苛刻的商业要求的组织提供了独特的优势。通过利用现有的组件和服务，可以减少完成软件开发生命周期(包括收集需求、进行设计、开发和测试)所需的时间。这使得可以快速地开发新的业务服务，并允许组织迅速地对改变做出响应和减少应用准备时间。

4．减少成本和增加重用

通过以松散耦合的方式公开的业务服务，企业可以根据业务要求更轻松地使用和组合服务。这意味资源副本的减少、以及重用和降低成本的可能性的增加。通过 SOA，企业可以未雨绸缪，为未来做好充分的准备。SOA 业务流程是由一

火龙果•整理
uml.org.cn

系列业务服务组成的，可以更轻松地创建、修改和管理它来

5．满足不同时期的需要

SOA 提供了灵活性和响应能力，这对于企业的生存和发展来说是至关重要的。但是面向服务的体系结构决不是灵丹妙药，而迁移到 SOA 也并非一件可以轻而易举就完成的事情。

1.2 问题提出

作为一个很有前景的应用系统架构，SOA 尚处在不断的发展中，肯定存在许多有待改进的地方。所以 SOA 架构的软件依然有多方面需要研究。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：如何使用 SOAD 方法来设计开发 SOA。第八：如何在松耦合和敏捷性之间达到更好权衡优化。就以上的几个问题，都是在 SOA 架构的实际软件项目中所需要解决的。在论文中将针对第一、四、六点，从技术角度研究在实际项目的中实现过程。并在此基础上分析第七、八点，从实际项目开发中总结开发经验和体会。

1.3 研究内容

论文中主要关注 SOA 的实现问题。通过对 SOA 架构的介绍，研究如何使用 ESB 技术平台建立 SOA 架构的基础平台。以 ESB 技术平台的角度为切入点，研究目前可参考的代表性解决方案。同时通过目前正在研发的解决方案，阐述开发基于 SOA 架构软件的困难以及实现的具体过程。

针对第 1.3 节提出的几个本文需要解决的问题，在论文中将研究目前分布式软件中 SOA 的研究现状及如何通过使用 ESB 来实现 SOA。介绍 SOA 中使用的 WebService, BPEL, ESB，MOM 等关键技术，分析 IBM 的 ESB 技术平台方案。并根据目前工作中正在开发的实际项目,给出一个基于总线通道技术的解决方案。详细分析前后总线通道技术的细节。深入研究总线通道如何自定义多通道，如何实现多种传输协议的转换，如何灵活的部署和使用 JMX 动态监管运行状态。然后从 SOA 架构角度阐述总线通道技术对于 SOA 架构特点的满足性及其体现的 ESB 技术平台。并根据实际项目开发经验分析两种有代表性技术方案和通道技术的异同

火龙果•整理
uml.org.cn

点。努力使论文能清楚阐述产品集成和项目定制两种技术方案各自的优势和劣势，进一步介绍如何对两种不同的方案进行取长互补。根据企业的实际情况来平衡两者的关系。

1.4 论文结构

在文中的第二章中将介绍 SOA 架构的理论研究，并介绍 SOA 架构的技术体系，讲述 SOA 的组成部分和 SOA 的特点。在第三章中描述 ESB 的演变进化过程以及 ESB 技术一路走来所经历的几个阶段。并介绍如何使用 ESB 技术平台实现 SOA 方案，分析一个 IBM 的产品集成型 ESB 技术平台。在第四章中将通过一个实际的前后端总线通道技术的 ESB 方案进行详细描述。研究项目定制型 ESB 技术平台实现中碰到的问题和如何克服困难实现的过程。在第五章中将验证前后端总线通道技术对于 SOA 的满足性。从 SOA 的几大特点的角度，分析前后端总线通道技术对 SOA 的满足性。并对文中阐述的两种方案进行比较，分析产品集成和项目定制的优势及缺点。讲述如何通过服务的软件开发方式来进行 SOA 的开发。第六章中总结全文，回顾前文对 SOA、ESB、前后端总线通道技术的研究，展望将来，思考不足之处及将来的研究方向。

火龙果•整理
uml.org.cn

第2章 SOA 架构的理论研究

SOA 架构的出现是软件发展的一个必然趋势，在本章节中将描述 SOA 的整体概要和 SOA 架构拥有基本特性以及 SOA 架构中所使用的核心技术。

2.1 SOA 架构概述

面向服务的体系结构 SOA(Service Oriented Architecture)是一个软件架构模型，它将应用程序的不同功能单元(称为服务)通过服务之间定义良好的接口和契约联系起来。接口是采用中立的方式进行定义的，它应该独立于实现服务的硬件平台、操作系统和编程语言。这使得构建在各种各样的系统中的服务可以以一种统一和通用的方式进行交互。这种具有中立的接口定义(没有强制绑定到特定的实现上)的特征称为服务之间的松耦合。松耦合系统的好处有两点，一点是它的灵活性，另一点是，当组成整个应用程序的每个服务的内部结构和实现逐渐地发生改变时，它能够继续存在而不受变化的影响。而另一方面，紧耦合意味着应用程序的不同组件之间的接口与其功能和结构是紧密相连的，因而当需要对部分或整个应用程序进行某种形式的更改时，它们就显得非常脆弱。而随着企业的发展对松耦合的系统的需要来源于业务应用程序需要根据业务的需要变得更加灵活，以适应不断变化的环境，比如经常改变的政策、业务级别、业务重点、合作伙伴关系、行业地位以及其他与业务有关的因素，这些因素甚至会影响业务的性质。我们称能够灵活地适应环境变化的业务为按需业务，在按需业务中，一旦需要，就可以对完成或执行任务的方式进行必要的更改。虽然面向服务的体系结构不是一个新鲜事物，但它却是更传统的面向对象的模型的替代模型，面向对象的模型是紧耦合的，已经存在二十多年了。虽然基于 SOA 的系统也需要使用面向对象的设计来构建单个服务，但是其整体设计却是面向服务的。由于它考虑到了系统内的对象，所以虽然 SOA 是基于对象的，但是作为一个整体，它完全不是面向对象的。以下就是基于 SOA 的软件系统具备以下特点：

- 设计松耦合性：服务需要完全独立，通过公开的服务接口进行交互。同时接口的协议可以支持多种传输格式
- 服务可重用性：服务是定义完整的功能实体，合理的使用可以减少重复功能开发，降低成本。
- 部署灵活性：在对硬件没有任何限制的前提下，可以灵活在异构系统上部

火龙果•整理
uml.org.cn

署。

- 系统兼容性：可以把遗留系统与遗留数据库无缝的集成到 SOA 架构。
- 技术多样性：可以使用各种软件开发语言来是实现 SOA 系统。
- 传输稳定性：可以在服务之间建立稳定安全的数据传输通道。
- 架构扩展性：可以根据企业的发展，调整服务，扩展服务功能。

其实 SOA 并不是一个全新的概念。SOA 系统原型的一个典型例子是通用对象请求代理体系结构 CORBA(Common Object Request Broker Architecture)，它已经出现很长时间了，其定义的概念与 SOA 相似。然而，现在的 SOA 已经有所不同了，因为它依赖于一些更为先进的技术，这些技术是以可扩展标记语言 XML(eXtensible Markup Language)为基础的。通过使用基于 XML 的语言(称为 Web 服务描述语言 WSDL(WebService Definition Language)来描述接口，服务已经转到更动态且更灵活的接口系统中，非以前 CORBA 中的接口描述语言 IDL(Interface Definition Language)可比了。当然 SOA 也并不是仅仅使用 WebService。同时是为了建立体系结构模型，需要的并不只是服务描述。同时需要定义整个应用程序如何在服务之间执行其工作流。尤其需要找到业务的操作和业务中所使用的软件的操作之间的转换点。因此，SOA 应该能够将业务的商业流程与它们的技术流程联系起来，并且映射这两者之间的关系。例如，给供应商付款的操作是商业流程，而更新您的零件数据库，以包括进新供应的货物却是技术流程。因而，工作流还可以在 SOA 的设计中扮演重要的角色。此外，动态业务的工作流不仅可以包括部门之间的操作，甚至还可以包括与不为您控制的外部合作伙伴进行的操作。而这就是 SOA 的优势所在。

最后必须提到的是，所有这些都必须处于一个信任和可靠的环境之中。因此，安全、信任和可靠的消息传递应该在任何 SOA 中都起着重要的作用。面向服务的体系结构所涉及的技术至少包括 CORBA、DCOM 和 J2EE。面向服务的体系结构也包括非常重要的中间件技术-基于消息传递系统(如 IBM WebSphere MQ)创建过他们自己的面向服务企业体系结构。所以说面向服务的体系结构并没有摒弃其他技术，而是通过一定的规范把各种可用的技术融合在一起。但由于 SOA 还是一个比较新颖的概念，所以在项目具体实施上还需要不断的研究，有各种各样的实施方案来尝试实现 SOA。在论文的以下部分就将分析其中有代表性的几种实现 SOA 的方案，结合实际项目进行探索。

2.2 SOA 架构的技术体系

SOA 主要包括几种核心技术，WebService、WSIF、MOM、BPEL。这四种是

火龙果•整理
uml.org.cn

SOA 标准中定义的核心技术。WebService 是服务的容器，通过其定义的传输协议可以用来匹配面向服务体系的接口。WSIF 则定义了 WebService 接口和其他技术的标准绑定方法。MOM 则为面向服务体系提供标准稳定的传输通道。BPEL 是对面向服务进行流程控制语言。通过 Business Process Choreography 使用者可以方便使用可视化定义 BPEL 流程定义控制，并提供了整合的智能客户端。为面向服务体系在开发和使用上提供完整的 SOA 开发。

2.2.1 WebService 技术及其兼容优化

WebService 是构建互联网分布式系统的一种方法，它可以使不同程序和不同系统平台上开发出来的程序具有平台无关的接口。WebService 使用 WSDL 定义详细接口说明。WSDL 是一种描述网络服务(network service)的 XML 格式，网络服务是能对面向文档类型的信息和面向过程的信息进行操作的端点(endpoint)的集合。对操作和消息的描述是抽象性的，并在定义端点时，将消息和操作绑定到具体的网络协议和消息格式上。一个 WSDL 文档将服务定义为一个网络端点的集合，或者说端口的集合。在 WSDL 里面，端点及消息的抽象定义与它们具体的网络实现和数据格式绑定是分离的。这样就可以重用这些抽象定义：消息，需要交换的数据的抽象描述；端口类型，操作的抽象集合。针对一个特定端口类型的具体协议和数据格式规范构成一个可重用的绑定。一个端口定义成网络地址和可重用的绑定的联接，端口的集合定义为服务。

在 WebService 的初期，开发者一般都使用 SOAP 作为传输协议来实现 WebService。并且都对其基层的 SOAP 进行繁琐的编程。但随着 WebService 的普及，开发者逐渐发现，WebService 加 SOAP 并不一定是最有效的方法，且当开发者需要对 WebService 进行重构时，就需要修改整个代码。有时开发者也需要把 EJB 修改成 JMS 面向消息，这种情况开发者也要修改代码。虽然 SOAP 支持对 EJB 的开发，带对于非 JAVA 的项目就会带来很大的问题。

为了解决这个问题，在实际开发使用 WebService 中就提出了 Web 服务调用框架 WSIF(WebService Invocation Framework)的调用方案，WSIF 提供一组 Java API，在不管 WebService 后台是由什么技术驱动前提下，统一的触发 WebService。WSIF 允许开发者通过 WSDL 和后台的服务进行交互，而不再需要直接的和传输协议交互。这样开发者就可以专注于对程序模型的开发，而不再关心 WebService 是怎么实现的。

WSIF 目前支持 SOAP，JMS，EJB 甚至是.NET 的软件开发。WSIF 使用 WSDL 来描述接口信息，并把 WSDL 和具体的实现分离。这样对于开发者就只要考虑如何设计 WSDL，而不需要面对具体的技术方案。在使用 WebService 访问时就不需要关心后台的实现细节。甚至可以不使用 SOAP 来传输数据，而使用更加高效的 RMI/IIOP 来传输数据。WSIF 对于 WSDL 的扩展可以是无限的，随着技术的发展，

火龙果•整理
uml.org.cn

可以对 WSIF 的所需要的邦定对象进行追加，而这种设计模式正是适合于 SOA 的开发。系统可以通过 WSIF 的机制来连接 Legacy 系统，也可以对系统将来留出足够的空间。这种方式的出现就为 SOA 松耦合的特点提供更好的保障。

2.2.2 MOM 方式的数据传输

在企业级应用中，面向消息的中间件系统 MOM 技术 (Message Orient Middleware)为不同类型的分布式系统提供消息通道。MOM 应用提供了一种公共的、可靠的方式，使这些应用能够轻松地创建、交换和处理消息，而无需考虑消息传递客户机的实现细节。消息被发送到服务器目的地，而不是发送到物理地址。消息传递客户机只需简单地声明对某个特定的域和目的地感兴趣，提供适当的安全性机制以获得访问该域的权限，然后通过那个目的地与消息传递服务器进行交互。消息的发送方只负责使用正确的包装，提供正确的地址与消息的安全可靠的传递相关的所有问题，而不管出现在他们面前的网络信号，数据完整等异常情况。

在一个 MOM 系统中，连接两端之间的完全是松耦合，这允许它们不必强调网络的稳定性、堵塞情况便能维持服务的最佳质量。利用其松耦合的特性的使维护和伸缩性将变得更加易于管理。系统可以在一天中的任何时候将应用离线，更新应用，或者作为例行的维护工作刷新应用。所有的数据将通过 MOM 中的缓冲队列对数据进行保持，等待系统恢复以后再顺序同步。

由于使用 MOM 技术有众多优点，所以其在 SOA 架构中占有重要的地位。作为 SOA 的一个强而有力的保证，在 SOA 架构中发挥着不可代替的作用。

2.2.3 BPEL 形式规约语言及其引擎

BPEL(Business Process Execute Language)是一门用于自动化业务流程的形式规约语言。使用 XML 文档写入 BPEL 中的流程能在业务之间以标准化的交互方式得到精心组织。通过这些流程能够在任何一个符合 BPEL 规范的平台或产品上执行。为了能在 Web 服务之间使用 BPEL，在其基础上制定了 Web 服务的业务流程执行语言 BPEL4WS(BPEL for WebService)规范，它是针对 Web 服务方面提供的流程标准。

可以说 BPEL4WS 是一种具有程序语言特点的工具，类似于用来表达算法的流程图。流程的每一步称为一个活动。存在以下一些基本活动：调用某个 Web 服务上的操作(invoke)，等待一条消息来响应由某人从外部进行调用的服务接口的操作(receive)，生成输入／输出操作的响应(reply)，等待一段时间(wait)，把数据从一个地方复制到另一个地方(assign)，指明某个地方出错了(throw)，终止整个服务实例(terminate)，或者什么也不做(empty)。

通过使用语言所提供的任何结构化活动，可以将这些原语活动组合成更复杂的算法。这些结构化活动提供的能力有：定义一组步骤的有序序列(sequence)，使用现

火龙果•整理
uml.org.cn

在常见的“case-statement”办法来产生分支(switch)，定义一个循环(while)，执行几条可选路径中的一条(pick)，以及指明一组步骤应该并行地执行(flow)。在并行地执行的一组活动中，您可以通过使用链接(link)来指明执行顺序方面的约束。

在 SOA 架构中，BPEL 具有很重要的角色，使用 BPEL 能把 SOA 中的松耦合的服务单元体有机的整合在一起并通过流程引擎进行管理。目前 IBM WebSphere 的 Business Process Choreography 就是在对 BPEL 的实现，它把 BPEL 通过可视化的界面提供给开发者使用。开发者通过图形界面设计好所有的流程之后，就可以通过 WebSphere 应用服务器的流程查看器查看、激活、停止、重启、完成一系列的步骤控制整个 BPEL 执行的流程。在 BPEL 中 WebService 就是其活动节点上的一个部分。在此基础上，开发者就可以方便的 WebService 接口进行 BPEL 的编排。

火龙果•整理
uml.org.cn

第3章 使用 ESB 技术平台的 SOA 解决方案

ESB(Enterprise Service Bus)技术平台是一种构造 SOA 架构的重要方案。对于 ESB 来说其实早于 SOA 架构前就有其雏形，但却一直没有形成一个固定的概念。随着 SOA 架构的出现，ESB 技术平台才慢慢显现其有武之地和重要性。随之而来就出现了各种形式的 ESB 技术平台。在本章节中将阐述 ESB 技术平台的发展过程和在 ESB 技术平台中如何融入前一章介绍的几种 SOA 关键技术，给出一个产品集成性 ESB 技术平台。

3.1 ESB 技术平台的发展

在网络技术的早期，传统的企业应用系统之间使用以下的连接结构，每个应用之间是通过一种简单的一对一的连接方式进行互相访问的。这种连接方式基本解决了信息孤岛的问题。但随着技术的进步，及企业对信息整合性要求的不断提高，随着节点的越来越多系统之间的互连会越来越复杂，逐渐造成维护性的越来越差，系统也将不堪重负。

由此产生了 EAI(Enterprise Application Integration)的概念。它为应用服务系统提供一个集成的中间层。由它充当整个 EAI 架构的中央管理器的作用。通过 EAI 确实现有 IT 系统的重新利用，延长这些应用系统的生命周期。但 EAI 仍然需要考虑信息格式，互联方式，异构环境等各种复杂的可能性。最重要的是这种 EAI 的实现方式是封闭式的，架构师需要在开发前就完全了解所用的系统的详细情况，依此来构建这个集成器。所以在 SOA 出现前 EAI 依然相当复杂且充满风险。

至此，慢慢地在其企业级应用中产生了 SOA 的概念如图 3-1 SOA 服务结构。它比单一 EAI 的中间集成器的形式更开放。在系统构建前无需了解服务的内部结构，总线将提供丰富的标准接口，以供服务选择使用。总之，在 ESB 中总线和服务是平等的，两者都有着无限扩展的可能。为此就需要总线发挥系统中枢的作用，首先它必须了解被它中介的两个端点：1)服务的请求者以及请求者对服务的要求，2)服务的提供者和它所提供服务的描述；其次，它必须具有某种机制能够完成中介的任务。我们把这两类考虑归纳为 ESB 的两个基本功能：面向服务的原数据(MetaData)管理功能 和中介(Mediation)功能。 作为 SOA 的重要构成部分，ESB 承担的重任还包括怎样将企业架构中已存在的业务服务连接到总线上来，我们称之为适配器(Adapter)功能。尽管服务本身已经用公开的接口来描述，但具体的实现还是运行在不同的环境中。因此，ESB 需要提供对服务底层协议的支持，譬如应

火龙果•整理
uml.org.cn

用协议 J2EE，.Net， 通讯协议如 Http，JMS 等等。还应该对具体应用中涉及到的服务加以管理，如性能，可靠性，安全性等等。以下就是 ESB 为保障 SOA 系统的运行所提供了最基本的功能：

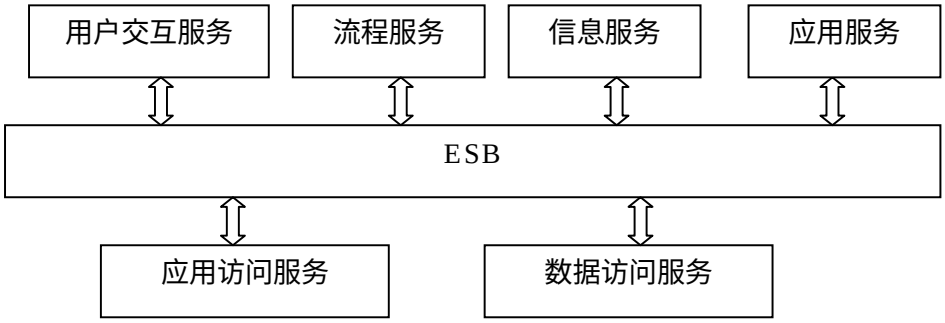

图 3-1 SOA 服务结构

Fig. 3-1 SOA Service Structure

- 在总线范畴内对服务的注册命名及寻址管理功能 - 服务的 Meta-data 管理
- 面向服务的中介功能
- 提供位置透明性的服务路由和定位服务
- 多种消息传递型式(请求/响应，单路请求，发布/订阅等等)
- 支持广泛使用的传输协议(Http，JMS，MQ 等等)
- 支持多种服务集成方式，比如 JCA、Web 服务、Messaging、Adaptor
- 对服务管理的支持，如服务调用的记录、测量和监控数据的提供

3.2 ESB 技术平台的特点

通过图 3-2 SOA的发展趋势所示可以看到SOA的发展进程，是在继承前者的基础上逐渐完善的。

火龙果•整理
uml.org.cn

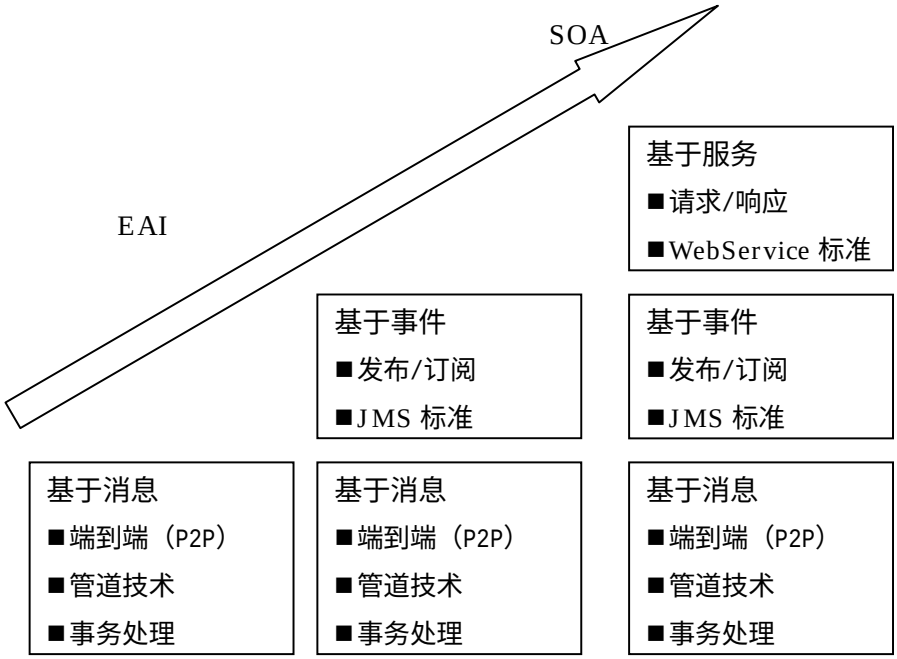

图 3-2 SOA 的发展趋势

Fig. 3-2 SOA Evolution

- 事件驱动的架构 - 应用之间异步地产生和接收消息
- 面向消息的架构 - 应用之间通过ESB发送和接受消息
- 面向服务的架构 - 分布式的应用由可重用的服务组成

ESB在SOA架构中是实现服务间智能化集成与管理的中介。ESB在逻辑上与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。可以这样说，ESB是特定环境下(SOA架构中)实施企业集成的方式：首先，在ESB系统中，被集成的对象被明确定义为服务，而不是传统系统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。通过这些定义完整就可以连接到ESB总线上去，而一切关于通信的工作也全部转移给ESB总线来完成。

ESB允许不同的消息传递方式。通常服务之间传递的消息有两种形式，一种是调用(Call)， 即请求/回应方式，这是常见的同步模式。还有一种我们称之为单路消息(One-way)，它的目的往往是触发异步的事件， 发送者不需要马上得到回复。考虑到有些应用服务是长时间运行的，因此，这种异步服务之间的消息交互也是ESB必须支持的。

ESB明确强调消息(Message)处理在集成过程中的作用，这里的消息指的是应

火龙果•整理
uml.org.cn

用环境中被集成对象之间的沟通。以往传统的EAI实施中碰到的最大的问题就是被集成者都有自己的方言，即各自的消息格式。作为基础架构的EAI系统，必须能够对系统范畴内的任何一种消息进行解析。因此尽管消息处理本身很重要，但消息的直接处理不会是传统EAI系统的核心。ESB系统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，直接面向消息的处理方式成为可能。如果ESB能够在底层支持现有的各种通讯协议，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。这样对消息的处理就会成为ESB的核心，因为通过消息处理来集成服务是最简单可行的方式。这也是ESB中总线(Bus)功能的体现。

3.3 基于产品集成的 ESB 平台

目前在软件工业界有两种参考模型，一种使用产品集成配置 SOA 项目；另一种则采用项目定制开发 SOA 项目。在本节中将主要研究使用产品集成的 SOA 架构方案。随着 SOA 架构的理论研究越来越深入，ESB 的集成产品的功能也越来越强大。在下文中就将研究 ESB 平台的前沿技术。通过对前沿技术的研究找出前沿技术的意义，并为后面第四章节项目定制的 SOA 架构方案起到借鉴的作用。

3.3.1 产品集成概述

在 SOA 的发展史上，最初的 ESB 概念是从 EIA 系统继承下来的。在这个基层上产生了 WBI(Websphere Business Integration)参考模型的概念，最初的体系完全是建立在一个一个分散的环境体系中的它们都能作为独立的中间件使用，也可以组合在一起使用。

WebSphere MQ： WebSphere MQ 为 ESB 提供了安全、稳定、可靠的传输支持。其通信程序可以在不同时刻运行，并提供统一的开发接口，使得程序与网络的复杂性相隔离。IBM WebSphere Business Integration Server Foundation：WBI Server Foundation 以及 WebSphere Application Server，WebSphere Application Server 本身是一个稳定的 J2EE 应用服务器。IBM WebSphere Business Message Broker：WBI Message Broker，提供了消息的路由、转发、发布/订阅等功能。

但随着随着企业的要求不断提高，有的中间件已经不能完全满足企业的要求，有的中间件虽然还发挥着重要的作用但其集成维护的成本太高，需要更好的整合才能满足要求。为此在产品集成的角度产生了WPS(WebSphere Process Server)整合产品。它是建立在WebSphere应用服务器上的新一代业集成平台。它支持面向服务的应用架构SCA（Service-Component Architecture）和企业服务总线ESB（Enterprise Service Bus），含有符合业界标准的业务流程引擎，使用统一的服务调用和业务表

火龙果•整理
uml.org.cn

现模型，并遵循业界的开放标准。

3.3.2 IBM 的 ESB 集成产品平台

目前 SOA 的概念还是显得相对抽象的，为了使客户能够更加简单的实现向这种面向服务架构的转变，在最新的 SOA 构架中提出了一种明确的，可实现的集成方法。面向服务组件的架构 SCA（Service Component Architecture）。这是一种全新的、跟语言无关的编程模型，它提供了一种统一的调用方式，从而使得客户可以把不同的组件类型，比如 POJO, EJB, 流程组件，人工交互组件等都可以通过一种标准的接口来封装和调用。结合 SDO 的数据模型，这种服务组件的编程模型可以大大的简化客户的编程，提高应用的灵活性。在 SCA 中，一个服务可以称为一个组件模型，一个组件模型中可以有多个模块。论文将从以下中介代理 Mediation、服务组件两个方面进行分析研究。

中介代理

在解决方案中，通过 MOM 解决了服务之间不同协议的总线传输的功能。但 SIBus 所做的工作是机械的把数据从一端通过 ESB 传输到另一端。这对于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受到的数据进一步的加工，在 SCA 框架中就提供 Mediation 中介功能。为了解决上述问题，在 SCA 模块中引入了两个连接端点，一个是导入（Import），它的作用是使得模块中的服务组件可以调用模块外部的服务。另一个是导出（Export），它的作用是使得模块外部的应用可以调用模块中的服务组件。通过这两个端点就可以和外界以及 SIBus 进行通讯。

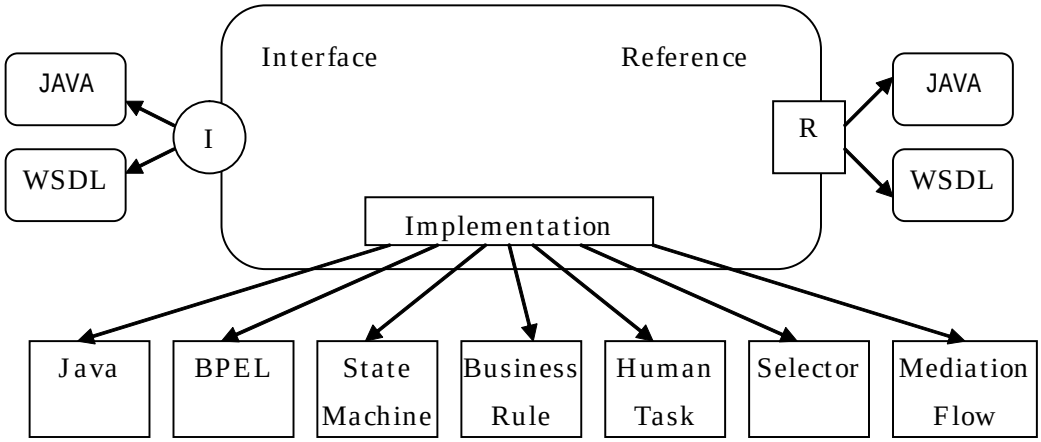

图 3-3 SCA 编程模型

Fig. 3-3 SCA Programming Model

火龙果•整理
uml.org.cn

服务组件是 SCA 中的基本组成元素和基本构建单位，也是具体实现业务逻辑的地方。在应用中可以非常容易地把传统的 POJO，无状态会话 BEAN 等包装成 SCA 中的服务组件。 SCA 服务组件的主要接口规范是基于 WSDL（Web Service Description Language）的，另外为了给 Java 编程人员提供一个比较直接的接口，SCA 的部分服务组件也提供了 Java 接口。因此，使用服务组件的客户端可以选择使用 WSDL 接口或 Java 接口。服务组件提供给别的服务调用的入口叫 Interface（接口）。而服务组件本身可能也需要调用别的服务，这个调用出口叫 Reference（引用）。无论是接口还是引用，其调用规范都是 WSDL 或 Java 接口。

服务组件

有了 SCA 组件模型的支持，为 SOA 架构提供了优秀的编程模式。可以说 SCA 不仅提供了中介模式用来帮助整合不同类型的服务单元，还提供了四种功能强大的服务组件，在 ESB 的基础上帮助业务流程整合。

业务流程引擎(Business Process Engine)是用来执行业务流程语言的运行环境。它把 BPEL 通过可视化的界面提供给开发者使用。开发者通过图形界面设计好所有的流程之后，就可以通过 WebSphere 应用服务器的流程查看器查看、激活、停止、重启、完成一系列的步骤控制整个 BPEL 执行的流程。在此基础上，开发者就可以方便的 WebServices 接口进行 BPEL 的编排。

人工任务管理器（Human Task Manager）模块实现了与人工任务相关的功能，它表示业务流程中有人的参与。让用户启动业务流程或者其他 Service 组件、实现业务流程中的 Staff 活动、流程管理(Administration)、动态创建含有与人工或者 Service 交互的任务。

业务状态机（Business State Machine）能够更好地描述由事件驱动的业务应用。业务状态机的描述能被自动构建并转换成 BPEL 实现，两者之间的关系类似于源代码和二进制代码的关系。从编程模型上来看，业务状态机是一种特殊的 SCA 组件。从语义上讲它继承于 UML2.0 的状态机模型。

业务规则（Business Rule）把传统应用中的嵌在系统各处的业务规则提取出来，进行独立的开发管理，是业务规则模块的基本思想。一个显而易见的好处是，可以动态更新业务规则从而使运行的业务更加灵活。

火龙果•整理
uml.org.cn

第4章 基于总线通道技术的 ESB 实现

在对上述可参考的 ESB 技术平台分析后，在本章节中将给出一个金融软件如何通过总线通道的 ESB 技术实现 SOA 架构的方案。可以说本章节中的总线通道技术是对 ESB 技术平台的具体实现。在项目架构中根据项目的需求以及 SOA 架构的特点来分析设计，将着重分析总线通道中几个关键技术的实现，最后还将阐述如何把几个关键技术进行融合。

4.1 背景知识

本文作者在 IBM ISSC 工作，作者在项目中主要参与基础架构方面的工作，项目的全称为 e-business Multi-Channel Solution Center。它是面向金融行业业务的一个多通道业务，基于 IBM 随需应变的思想，使用面向服务的设计开发的一套金融软件系统。在业务设计上这个系统可以通过电话，电脑，移动设备客户端来操作业务。它所针对的业务可以是银行，证券，保险等行业。同时 eMuSC 项目是跨地区的分布式系统，它拥有两个数据存储中心，一个是 IBM 的数据储存中心，另一个是银行的数据储存中心。两个存储中心分布分别处理相应的业务，定时通过 MQ 中间件同步两者之间的数据。并提供多通道以供系统和外部的金融机关进行通信。

项目规划 16 个银行参与使用这套系统。终端用户可以通过电话，电脑，移动设备客户端来访问这套系统。用户可以分为个人管理维护系统，企业管理维护系统，个人网上银行系统，法人网上银行系统，电话银行系统。不同用户可以通过不同的角度来访问这套系统。在功能上支持和在线支付业务的连接，以及和金融行业 Legacy 系统的整合，并且为将来预留接口。

表 4-1 银行业务种类

网上银行个人系统	网上银行企业系统
本币业务（定期，活期）	企业本币业务（定期，活期）
外汇业务（定期，活期）	企业外币业务（定期，活期）
信托业务（股票，证券）	工资支付业务
缴费业务（水，电，煤）	企业转帐业务
基金业务（房屋基金，保险基金，投资基金	企业税金业务

火龙果•整理
uml.org.cn

eMuSC 银行项目中可以使用计算机以及三种类型的手机来访问 eMuSC 网上银行。三种手机分别是 imode，Vodafone，EZWeb。用户可以使用同一个账号在上述的几种设备上操作网上银行，所有的操作会通过 eMuSC 服务器访问数据库。在任何一个客户端的操作，都完全可以通过其他三种客户端来反映。

- PC 使用 HMTL 标记语言
- Imode 使用压缩 HMTL 标记语言(cHTML)
- EZWeb 使用手持设备标识语言(HDML)
- Vodafone 使用手机标识语言(MML)

开发者根据不同的标记语言开发，并通过三种手机在 PC 机上的模拟程序来编写，调试相应的标记语言。虽然可以使用模拟器来开发多种客户端，但同时需要开发者把同样的业务完全正确的反映到四种客户端上也需要付出巨大的努力。

eMuSC(e-business Multi-channel Solution Center)介绍

1. 面向金融企业的多通道的电子商务服务系统
2. 支持多种客户端(电话设备，网络设备，移动设备，ATM)
3. 使用 WebService, EJB, MQ、JMS 开发系统构架
4. 使用不同的通道(Channel)来传输不同金融数据
5. 使用 SOA 思想，ESB 模式整合系统

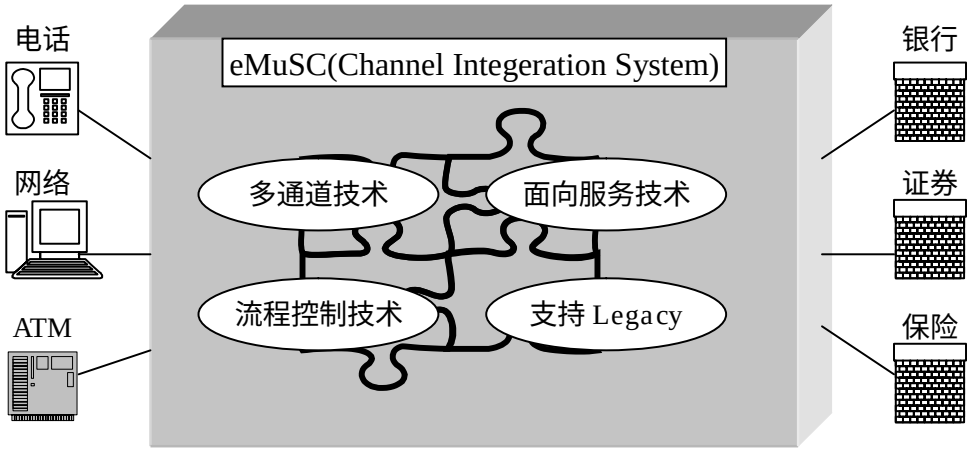

图 4-1 eMuSC 运用体系

Fig. 4-1 eMuSC Solution System

火龙果•整理
uml.org.cn

4.2 总线通道软件配置

下面这个总线通道软件配置图是目前的项目的实际基本设计。在软硬件的设计上使用了三台 RS6000 工作站，在硬件上分别作为网站服务器，数据服务器，中间件服务器。网站服务器上安装了 Websphere Application Server。中间件服务器上安装 IBM MQ Series 用来和银行之间的 MQ 进行数据同步处理。数据服务器上安装了 IBM DB2 软件，后台数据处理程序，IBM Tivoil 数据保障软件。

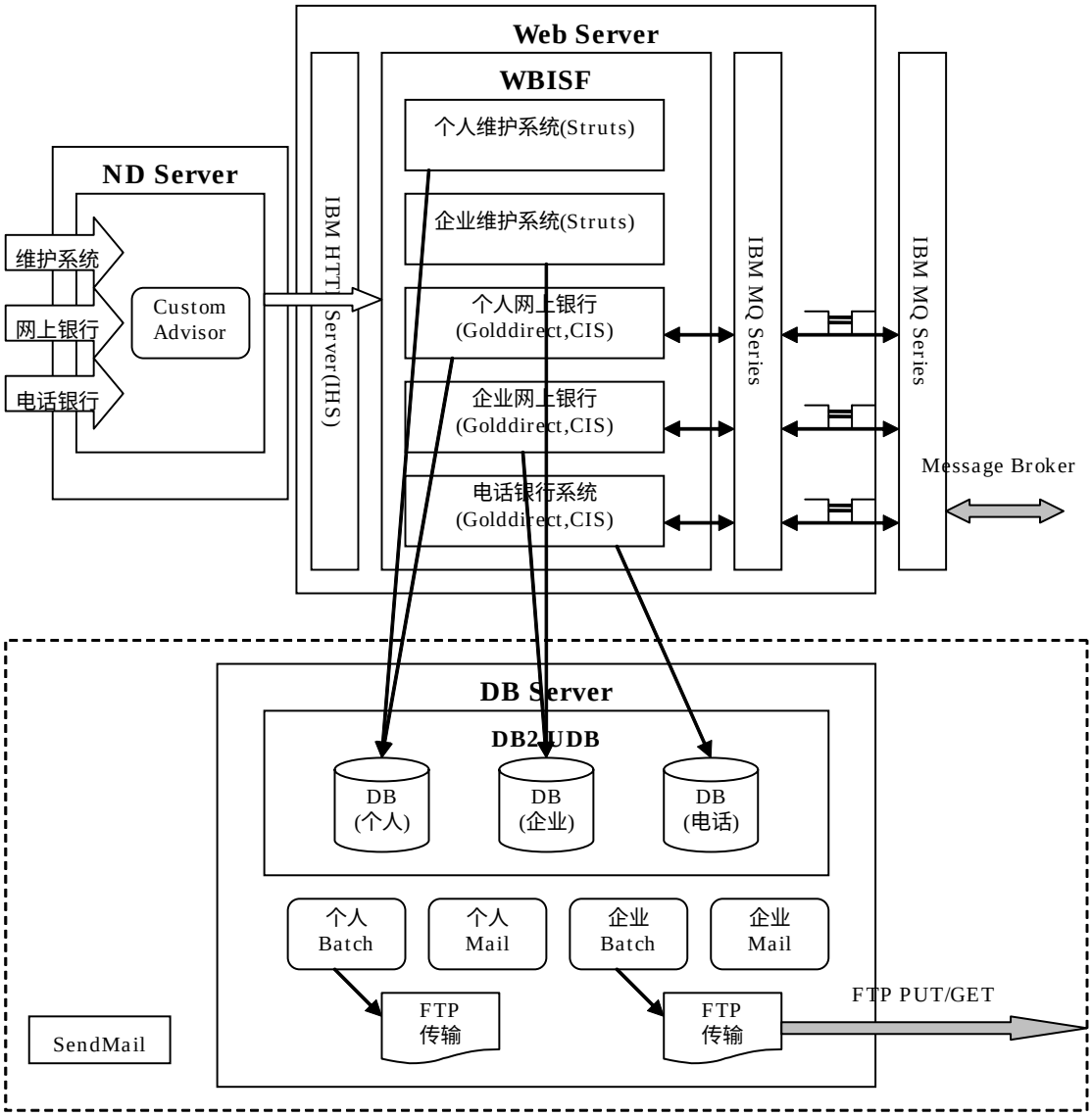

图 4-2 总线通道软件配置

Fig. 4-2 Multiple-Channel Software Configuration

火龙果•整理
uml.org.cn

4.3 多种传输协议的实现

在传统的系统中都使用单一传输协议进行数据传输，并把业务和传输协议进行了紧耦合的绑定。这在实际的使用中往往会严重影响系统的扩展性以及与外部系统的互联。而扩张性和互联性在新系统中都是重要的系统特性。为此在进行系统设计时，首先就需要对上述问题进行一个系统级的考虑和研究。这也是架构设计中的第一个课题。

由于 SOA 技术的不断完善，并且在出现了 WebService 等先进的技术。参考了 WSIF,所以经过研究在架构中预先设计实现了三种协议的实现。由于考虑到以后可能有更多的传输协议的扩展所以在实现不能简单对传输协议进行枚举式的硬编码。

- IIOP(Internet Inter-ORB Protocol)
- MOM(Message-Oriented Middleware)
- SOAP(Simple Object Access Protocol)

在开发中发现通过使用多种传输协议确实可以兼容多种格式的数据，但在使用多种传输协议时需要使用不同的协议的传输接口，尤其是多种传输协议的复杂性。涉及到协议封装、协议格式、协议版本以及异构平台消息转换。所以在总线通道接口的设计上引入了 Facade 的设计模式。由于 Facade 模式具有屏蔽子系统细节，方便客户访问子系统的功能，并且可以消除冗长的复杂代码。

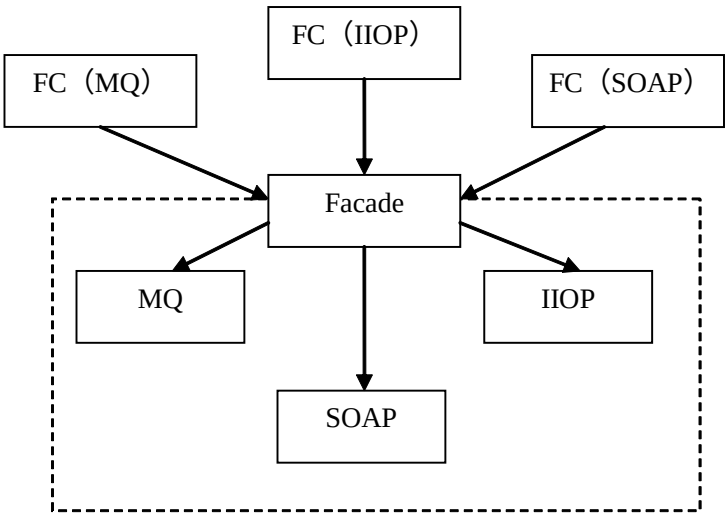

图 4-3 Facade 设计模式

Fig. 4-3 Facade Design Patten

火龙果•整理
uml.org.cn

在实现 Facade 模式时首先用抽象类实现 ProtocolConnector，利用其特点把问题的涉及面降低到最低。用其子类去对应具体各种协议，这样可以进一步降低对协议依赖的耦合度。同时也可以使用不同的协议对象去配置 Facade，可以方便地替换及扩展其他协议类。在具体的实现上如图 4-4 多种协议内部实现所示，根据传输协议的要求编写一个具体协议类。并在后面 4.4 节通道定义规则配置文件中进行描述后，ProtocolConnectorFacade 协议连接工厂就可以根据两者的定义进行。传输协议松耦合的匹配。

```
//协议转换工厂类
Public Class ProtocolConnectorFacade {
        getProtocalConnectorInstance(Channel);//实例化协议类
        getProtocol();//获得当前协议
        putRequest(RequestBean);//设定输入参数
        ResponseBean getResponse();//获得输出参数
        setPCContext();//设定协议环境变量
}
//MQ-XML 方式的传输协议
ProtocolConnectorMQXML extends ProtocolConnector
//WebService 方式的传输协议
ProtocolConnectorSoapMsg extends ProtocolConnector
//EJB/IIOP 方式的传输协议
ProtocolConnectorIIOP extends ProtocolConnector
```

图 4-4 多种协议内部实现

Fig. 4-4 Multiple Protocol Switch

通过使用 Facade 设计模式来整合不同的协议接口时，开发者不需要直接面对传输协议进行开发。这样就很好的降低系统间耦合度，一旦对 Facade 后面的总线机制进行修改，开发者也不需要修改任何关于传输协议的代码。这种实现方式后，就完全透明于上层应用开发。开发者无需关心在总线通道中正在使用的协议是哪一种。因为在项目中的总线通道中传递的数据只是一组 Request/Response 对象。对于开发者只需把 Request/Response 对象传递给前后端总线通道，而当总线接受到数据时，就会在总线内部使用于传输协议相配合的数据适配转换器，根据与其相对于的传输协议来处理数据。提高了总线通道在整个架构中的灵活性。与常见的单传输协议相比设计多种传输协议有以下几个优点：

火龙果•整理
uml.org.cn

- 传输协议的丰富可以提高整个系统的松耦合度。整个系统不再局限于某种唯一的传输协议。
- 提供多种通传输协议为系统提供足够的接口以便将来业务的扩展。
- 提供多种传输协议更好兼容 legacy 系统。
- 提供同步和异步的传输方式。可根据不同的业务，动态的响应数据传输要求。
- 帮助确定系统边界，划分业务服务功能范围。
- 协议的使用完全透明于开发者

4.4 基于依赖关系的松耦合服务

在设计系统时，无论何时都要面临一个问题就是业务需求的不断变化。而当业务发生变化时，都需要对 IT 架构进行相应程度的修改而能够快速响应变化的 IT 基础架构就是最适应的软件体系。所以在设计上考虑如何更容易地将业务流程变化映射到应用系统中就是一个重要的课题。所以必须为灵活建模、创建、修改、测试、维护业务服务提供一套完整的架构，方便适应服务的变化。如图 4-5 服务骨架程序所示，在项目中所有的服务单元都是通过这种继承方式进行构造实现，从图中所示所有的参数都是使用同一个抽象类 CommonHeader 继承下来的业务数据对象。然后使用五个基类定义的重载函数进行业务编码。对于认证服务来说也是同理，在一个认证服务的基类上进行各种扩展。

火龙果•整理
uml.org.cn


```
public abstract class AbstAuLogon extends AbstApplication {
    private static final String CLASS_NAME = AbstAuLogon.class.getName();
    protected void startProcess(CommonHeader request)
        throws EMUSCAppException, EMUSCSysException {
    }//服务启动
    protected void checkRequestObject(CommonHeader request)
        throws EMUSCAppException, EMUSCSysException {
    }//数据对象检查
    protected CommonHeader executeOffline(CommonHeader request)
        throws EMUSCAppException, EMUSCSysException {
    }//执行 Offline 状态
    protected CommonHeader executeOnline(CommonHeader request)
        throws EMUSCAppException, EMUSCSysException {
    }//执行 Online 状态
    protected void endProcess(CommonHeader response)
        throws EMUSCAppException, EMUSCSysException {
    }//服务结束
}
```

图 4-5 服务骨架程序

Fig. 4-5 Service of Skeleton Source

同时设计上采用注入模式 DI (Dependence Invert)控制倒置方法，将业务代码依赖关系从架构中脱离出来，使用 setters 来设置依赖的服务。把依赖的服务作为一个实例对象，通过 setters 方法来动态设置依赖服务。同时 IoC 的容器提供相应的注入方法，以保证依赖关系通过 setter 方法来注入。

在设计服务种类上，项目定义两种类型的服务：认证服务、业务服务。认证服务是用来保证服务的安全性。业务服务则是用来实施定义核心业务。和一般的注入模式的不同点，在于为满足服务的松耦合性时，还定义服务上建立里一种有依赖关系的机制。认证服务和业务服务之间的关系可以根据不同的情况进行关联。所有业务可以根据需要对认证进行选择。进行多对多的关系匹配。

在具体实现上如**错误！未找到引用源。**，相应的业务代码类会定义到配置文件中，在两种文件中每一个业务都会被定一成唯一标示符。同时认证服务和业务服务会通过同一个关键字关联在一切，从而方便的把服务注入到相应的业务中，使开发者能更专注于业务中去。最后通过 JAVA 的反射机制来获得核心业务服务。这样也可以把核心业务代码和基础架构分离的目的。

火龙果•整理
uml.org.cn

根据项目框架的设计，系统可以通过关键字映射调用相应的服务。在调用时 Request 会传递相应的核心业务的关键字来获得具体的核心业务。当相应业务对象被触发并访问数据库后，则返回一个 Response 对象给前端用户界面。

```
＃APP_BIND_TABLE.properties
appAuLogonOn=com.ebank.cistrx.ib.business.auth.AuLogon
appCzInsert=com.ebank.cistrx.ib.business.auth.CzInsert
appKtBanksrhBsrh=com.ebank.cistrx.common.search.KtBankSearch
appUmPwdChg =com.ebank.cistrx.ib.business.auth.UmPwdChg

＃AUTH_BIND_TABLE.properties
authAuLogonOn=com.ebank.cistrx.ib.business.authorizer.KtAuthorizerLogon
authCzInsert0= com.ebank.cistrx.ib.business.authorizer.KtAuthorizerFirstCode
authCzInsert1= com.ebank.cistrx.ib.business.authorizer.KtAuthorizerSecondCode
authCzInsert2= com.ebank.cistrx.ib.business.authorizer.KtAuthorizerThirdCode
authKtBanksrhBsrh=com.ebank.cistrx.common.authorizer.KtAuthorizerEtc
authUmPwdChg=com.ebank.cistrx.ib.business.authorizer.KtAuthorizerFirstCode
```

图 4-6 服务绑定机制

Fig. 4-6 Service of Binding Mechanism

在整个处理流程中，数据首先通过认证服务，认证服务会从数据中获得相应的鉴定信息进行用户认证，经过认证后再通过关键字映射由核心服务处理。而当认证不通过时则会直接返回用户认证失败。在整个认证体系中所有的认证服务是完全独立的，通过同一个关键字可以和每一个核心业务服务进行绑定，如果需要甚至可以为每个核心业务定制多次的认证服务方法。绑定机制会根据关键字的访问规则按认证顺序为同一个核心业务依次认证。

表 4-2 认证服务类型列表

用户认证服务	注释
KtAuthorizerTimeStamp	根据时间戳认证
KtAuthorizerFirstCode	第一确认密码认证
KtAuthorizerLogon	用户登陆认证
KtAuthorizerSecondCode	第二确认密码认证
KtAuthorizerSecondCodeSessionDelete	第二确认密码修改后删除认证

火龙果•整理
uml.org.cn

在项目中使用多种认证模式表 4-2 认证服务类型列表，以满足现有系统的需求。实际上认证服务和业务服务都是继承同一个服务的基类(AbsAppService)。可以自由定义认证的逻辑结构以满足需求。根据多对多的关联，以图 4-7 服务的多对多关系就可以方便的建立两者之间的关系，两者之间的关系以同一个标示符绑定。通过这种工作关系，不仅可以使服务之间达到完全的独立、松耦合，也可以方便的在两种不同类型的服务之间建立必要的关系。在实际工作就可以灵活的进行调整和扩展以及复用。

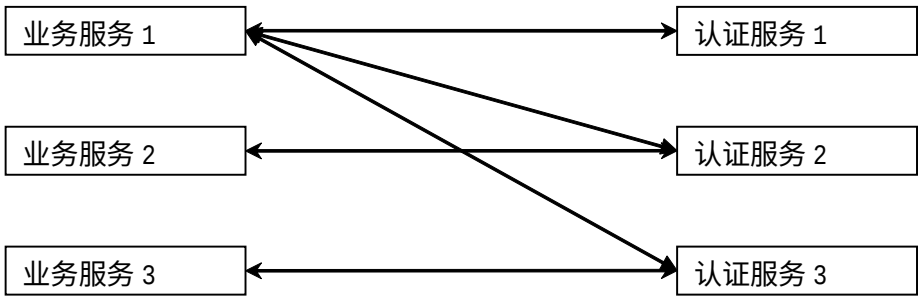

图 4-7 服务的多对多关系

Fig. 4-7 Service of Many to Many Relationship

4.5 传输通道的定义规则

随着中间件技术的完善，用户对数据传输的实时性，可靠性，安全性的需求越来越高。因此把中间件技术更好的融入系统中是一个很重要的系统设计。目前项目要求是对不同的金融业务数据进行隔离式的数据传输，而且在传输数据时选择不同的传输协议。所以在开发设计时就需要结合 4.2、4.3 节的设计成果以及 SOA 的特性来实现。

根据的 4.2 节、4.3 节的分析，项目可以做到可切换的传输协议以及多对多的松耦合的服务，如果只是为了实现项目的要求，有了以上两点就能够满足要求。但要考虑到项目的可扩展性，就需要对在设计传输通道时制订相应的定义规则，通过定义规则来实现自定义传输通道。开发者可根据不同的业务数据定义的数据交互方式，根据通道所使用的传输协议，为业务定制专用数据传输通道。在使用

火龙果•整理
uml.org.cn

前面介绍的 Facade 设计模式的基础上，开发者只需给定传输通道标示和传输通道协议就可以在不影响其他数据传输的同时，方便的扩展整个系统的架构。

在具体的实现上，如图 4-8 通道内部结构中所示的每个通道，在内部定义时都会有一个通道标示符 ChannelID 和通道协议符 ChannelType。通过这两个标示符的匹配就可以定位一条数据传输通道。然后在总线通道内部会有一个消息转换适配器(Adapt)把不同协议的数据转换成对象。在绑定相应的服务认证符 ServiceAuth 通过安全认证后，传输通道就可以通过服务标示符 ServiceID 绑定到应用服务。通过这样的定义规则，一个完整的传输通道就可以进行数据通信。

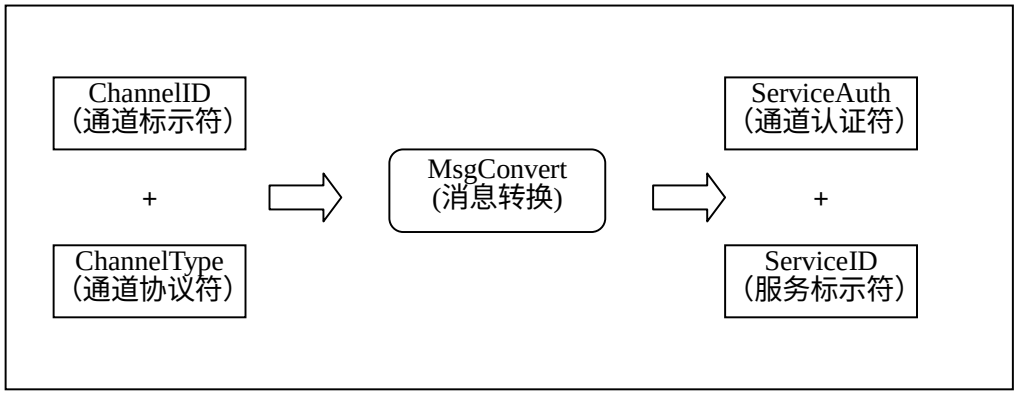

图 4-8 通道内部结构

Fig. 4-8 Channel of Inside Symbol

在实际的项目中如图 4-9 通道定义规则所示，四种标示符先分别在各自的定义文件中配置，然后通过 EMUSC_UBC.properties 再把关键字一起绑定为一个整体。使架构可以通过预先配置的规则进行数据传输，这时数据通道就会根据上述的绑定对数据进行加密，加固保证数据的正常交互安全。把数据从互联网上独立出来。使业务之间数据完全隔离。保证数据不会因为互联网的堵塞而使数据不能正确传输。

火龙果•整理
uml.org.cn

//CH_BIND_TABLE.properties ChannelID 通道标示符
chAuLogon=WIKICH
//PROTOCOL_BIND_TABLE.properties ChannelType 通道协议符
ctAuLogon=ProtocolConnectorMQXML
//APP_BIND_TABLE.properties ServiceAuth 服务认证符
appAuLogon=com.ebank.cistrx.ib.business.auth.AuLogon
//AUTH_BIND_TABLE.properties ServiceID 服务标志符
authAuLogon= com.ebank.cistrx.ib.business.authorizer.KtAuthorizerLogon

//EMUSC_UBC.properties
AuLogon=chAuLogon;ctAuLogon;appAuLogon;authAuLogon

图 4-9 通道定义规则

Fig. 4-9 Channel of Definition

4.6 使用 EJB 技术衔接前后端总线通道

通过前几节分析，从局部说总线通道的设计基本完成了。但对于项目来说还有一个最重要的问题等待解决。由于项目需要图 5-2 自定义前后端总线通道模型结构的一组总线通道。不同的前端服务通道用户可以同时访问数据，不同的后端服务通道保证和外界第三方数据接口进行后台数据同步和传输。如何来组建前后端总线通道，使用什么技术来衔接前后端总线通道。怎么很好的使用现有的研究成果都是一个重要的问题。

为了保证数据传输的事务性，并发性，正确性。为此在 J2EE 的技术平台上，就选择 EJB 来是实现其功能。在实现的过程也是相当痛苦，也遇到过许多具体实现的技 术壁垒，但是考虑到金融软件的安全性以及分布式事务处理等问题，最后依然决定使用 EJB 技术。如图 5-2 自定义前后端总线通道模型就是在反复研究后所确定下来的模型需要在前端 FC Bus 和后端 BC Bus 上使用 Sessionless EJB 进行封装，它是由 BusConnectorFactory 的 Bean 工厂类，BusConnectorBean 的实现类以及 ejb-jar.xml 配置文件组成。

由于考虑到项目是一个庞大的工程不能简单通过硬编码来拼凑整个架构所以在设计前后端总线通道时，并没有把相应的两端生硬的连接在一起。因为生硬把

火龙果•整理
uml.org.cn

两端连接在一起，也就没办法实现多通道和多协议共同存在，所以必须有一个中介来协调两种功能而这个就是EJB技术。如图 4-10 ejb_jar.xml定义片断在ejb-jar.xml中就可以根据相应的协议指定相应的协议的Bean的类型。并在ejb-jar.xml中通过环境变量定义绑定的通道。其中soa.esb.iiop.BusConnectorIIOPLocalHome就是指定通信协议对象。buschannelID和下面代码中的GENARATER_CLASS_NAME相对应，而FCH_CGIK就是要说的通道识别描述符。

```
<session id="Bus_Channel">
    <ejb-name>WikiChannel</ejb-name>
    <local-home>soa.esb.iiop.BusConnectorIIOPLocalHome</local-home>
    <local>soa.esb.iiop.BusConnectorIIOPLocal</local>
    <ejb-class>soa.esb.iiop.BusConnectorIIOPLocalBean</ejb-class>
    <session-type>Stateless</session-type>
    <transaction-type>Container</transaction-type>
    <env-entry>
        <description></description>
        <env-entry-name>buschannelID</env-entry-name>
        <env-entry-type>java.lang.String</env-entry-type>
        <env-entry-value>AuLogon</env-entry-value>
    </env-entry>
</session>
```

图 4-10 ejb_jar.xml 定义片断

Fig. 4-10 ejb_jar.xml Fragment

```
GENARATER_CLASS_NAME =    "java:comp/env/ buschannelID ";
……
InitialContext iniCtx = new InitialContext();
String    className = (String)iniCtx.lookup(GENARATER_CLASS_NAME);
ResponseObjectrespGene=(ResponseObject)Class.forName(className).newInstance();
```

图 4-11 soa.esb.iiop.BusConnectorIIOPLocalBean 实现类

Fig. 4-11 soa.esb.iiop.BusConnectorIIOPLocalBean Implement

从图 4-11 soa.esb.iiop.BusConnectorIIOPLocalBean 实现类代码可以看出，通过

火龙果•整理
uml.org.cn

这种配置方式，在开发时只需要在 ejb_jar.xml 上述规则按套进行通道追加就可以充分满足自定义的要求。所以在技术的选择上，虽然 EJB 有很多性能底下的问题，但考虑到安全性和分布式环境及配置管理，最后决定使用 EJB 作为一个中介来连接接口两端。目前在项目中的前端总线采用了 IIOP 协议的通道，后端总线采用了 MQ 技术的通道。当然根据目前的结构也完全使用 WebService 技术来作为通道的传输协议，而不需要修改任何的架构代码。可以说这就是采用 SOA 设计思想的最好的一种体现。

对于整个项目来说 4.5 节的 EJB_JAR.xml、4.4 节的 EMUSC_UBC.properties、4.3 节的 APP_BIND_TABLE.properties 和 AUTH_BIND_TABLE.properties 以及 4.2 节的协议扩展类之间都有很重要的联系。如图 4-12 扩展配置依赖关系图所示，通过 APP_BIND_TABLE.properties 和 AUTH_BIND_TABLE.properties 中的定义关键字会通过 EMUSC_UBC.properties 映射，同时扩展协议类也会通过 EMUSC_UBC.properties 实例化。而在 EJB_JAR.xml 中的传输通道关键字就会通过 EMUSC_UBC.properties 的定义在相应的 EJB 中使用 ProtocolConnectorFacade 来调用相应的传输协议。

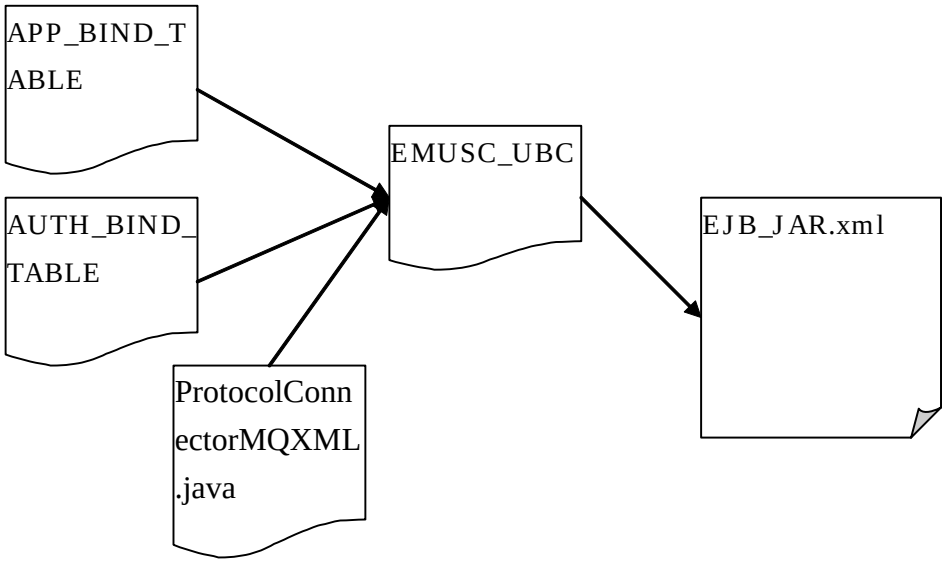

图 4-12 扩展配置依赖关系图

Fig. 4-12 Configuration Dependency Relationship

4.7 总线通道技术对 ESB 的体现

火龙果•整理
uml.org.cn

在项目中 ESB 模式可以说是整个系统最重要的组成部分。由于采用分层的体系结构把系统分成多个组成部分。所以就需要对总线通道在其中起到桥梁的作用，承上启下的功能。在 SOA 发展的现阶段 ESB 拥有许多种不同的模式，有的是多层级联，有的是中心控制，所以需要根据项目的具体情况处理。为此就需要运用 ESB 模式来定义属于自己的分层结构方案。以下三种模式就是在总结 ESB 思想和项目的实际情况进行互为补充以实现 SOA 架构。

1．交互模式：在系统的中所有的数据的交互是 Request/Response 对象进行交互。在项目中 Request/Response 定义为每一个业务的数据对象。该数据对象主要有两部分组成。在共用部分存放运行时的动态信息，它包括有 Session 信息，Channel 通道信息，以及为后面中介模式所需要的路由信息，另一部分则定义了业务数据对象。

2．中介模式：项目中的总线通道是通过 FC Bus 和 BC Bus 实现的。在建立在 Request/Response 对象交互模式的基础上，FC Bus 和 BC Bus 完成对不同数据传输协议的转换，允许服务请求者使用各种交互协议或 API(如 SOAP/HTTP、JMS 和 MOM)发送其消息，将请求代码转换为目标服务提供者的格式。并为数据提供相应的包装和加密。在完成以上数据处理后，根据数据信息选择其相应的服务的处理路由，高效的在服务提供者和服务请求者之间建立同通道。并使用中介模式的优势对总线中传输的数据进行统一的数据流量控制，以及加密，日志记录，管理的功能，使总线的路由定位传到相应的服务个体上去。

3．部署模式：在自主的 ESB 结构上使用桥接的部署模式，系统可以根据需要把 FS、TRX、BS 应用服务，分别部署在不同的应用服务器上然后通过强大的总线通道功能连接在一起。通过这种部署模式系统可以灵活的根据企业的需求进行部署。企业可以根据成本，稳定，业务需求进行不同组合的系统部署。由于采用了桥接的部署模式的部署方法。整个系统将获得随需部署的灵活性。通过这种方式可以方便的使系统中的服务具有更好松耦合性。

对于 ESB 方案来说，可以有各种不同的拓扑模式。而以上三种模式是根据项目的实际情况而制定方案。

4.8 前后端总线通道的应用优势

通过 EJB 的衔接，整个架构基本已经完成了。可以说整个过程是具有挑战性，不仅需要创造性的思维，更需要对 SOA 架构思想的不断完善。在目前来说 SOA 架构完全没有固定先例可以借鉴，更多的是对 SOA 架构的尝试。可以说一路走来都是在反复推敲尝试的努力。在搭建好整个架构后，项目逐渐出现了成果所带来的好处，一些非功能性需求也在无形中得到了很好的解决，体现出其优势。

火龙果•整理
uml.org.cn

4.8.1 流量控制机制

经过对上述几小节的介绍，几个关键的技术难点基本都已经解决了。但对于大流量的网上业务来说都需要考虑一个不可避免的问题。当出现网络堵塞或是网上银行使用过多时产生的问题，或者有人恶意攻击网上银行都是分布式系统需要考虑的问题。在项目系统中我们从三个方面来考虑这个问题。

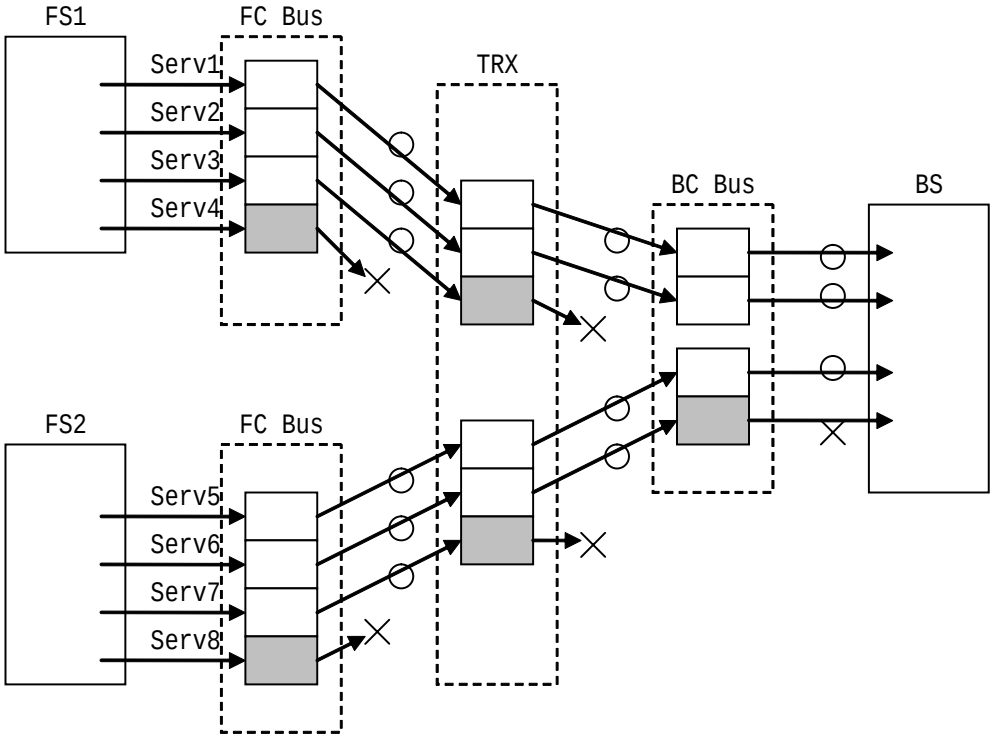

图 4-13 流量控制

Fig. 4-13 Flow Control

首先，从硬件上使用，设置一个 AIX RS6000 系统组从硬件上保证对 eMuSC 软件的负载。并使用 WebSphere 的负载平衡冗余技术。一旦其中一台服务器出现故障，另一台可以及时恢复系统的正常运行。

其次，从软件上使用，IBM Tivoil 监控管理软件，用来管理服务器上网络运行情况的，过滤各种恶意攻击数据。监控数据库存储良好。

最后，也是最重要的就是通过前后总线通道软件系统来进行流量控制。虽然从硬件上可以保证服务器有足够的冗余能力来负载系统，但是硬件上不能对恶意攻击进行保护。所以在系统使用了专业的监控管理软件 Tiviol 从数据类型上来管理

火龙果•整理
uml.org.cn

网络银行。所以在总线通道系统中使用了属于自己特有的流量控制机制。

项目中把流量控制功能集成到两个前后端总线通道上。如图 4-13 流量控制所示，项目通过分层结构的设计方法，采用前台层 FS，业务层 TRX，后台层 BS。在这三个层次之间可以使用 FC Bus 和 BC Bus 来做流量控制。所有的数据都会经过上图所示的 FC Bus 和 BC Bus, 在开发时，开发者通过定制相应的流量控制类 Flow Control，在该流量控制类中读入相应服务 ID 的流量控制数量来限制相应的网络请求。并把相应的流量控制类通过相应的设置动态插入到 FC Bus 和 BC Bus 总线上。系统维护者则可以根据服务 ID，以及每个服务的流量限制数量。并统计当前服务的流量数并加以控制。由于服务是松耦合的所有的流量控制都可以根据每一个通道服务的业务要求在流量配置文件进行灵活的定义。由于每个服务都有一个唯一的标示符，所以流量配置文件就可以针对这个标示符进行控制。一旦数据经过总线时就可以通过业务标示符判断当前业务的数据流量，根据服务服务器的负载能力进行相应的管理。这也是对 SOA 架构很好的体现。

4.8.2 日志记录机制

从上述介绍中我们可以看到 FC Bus 和 BC Bus 在对于流量控制中所起的至关重要的作用。同时 FC Bus 和 BC Bus 还提供各层之间统一日志记录功能。在目前大多数的软件系统中的开发者经常要同时考虑业务逻辑、性能、同步，日志和安全等问题，兼顾各方面的需要，这样就会导致相应关注点的实现元素同时出现，引起代码混乱，并且也会造成大量的代码重复，相关实现也就得散布在这些代码里。杂乱的代码同时也掩盖了代码中隐藏的问题。一旦需求改变后，就需要修改现有的代码。而为了重构系统可能会带来数据不一致，而且还需仔细的测试来保证这些修改不会带来错误。所以对于日志记录功能需要能尽量和业务代码分离。

火龙果•整理
uml.org.cn

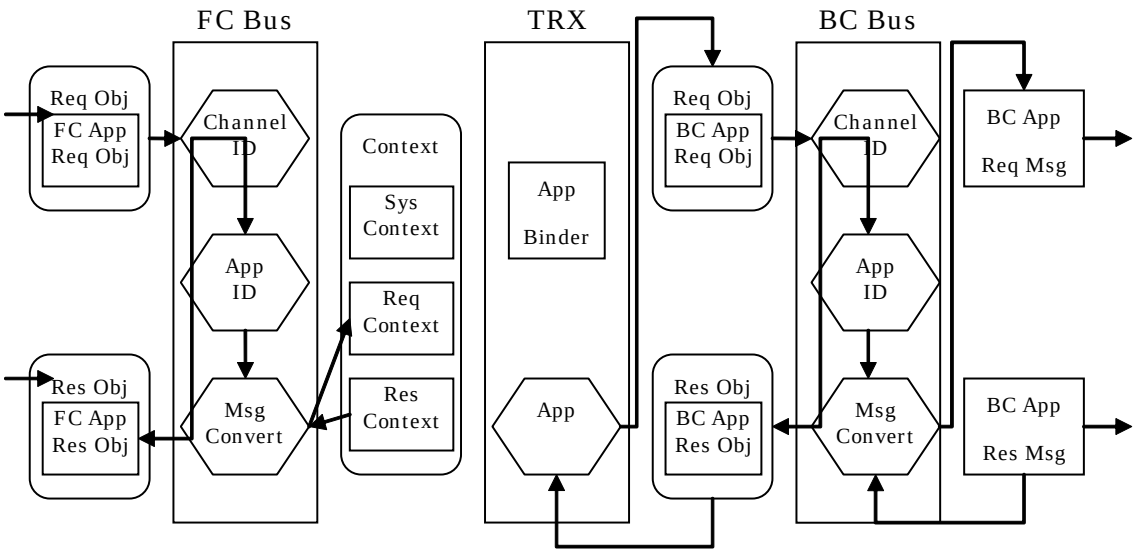

图 4-14 系统内部数据交互方式

Fig. 4-14 System Internal Data Interaction

从图 4-14 系统内部数据交互方式可以看出通过所以的 Request/Response 都是使用 FC Bus 和 BC Bus 这两个总线通信。这样项目就可以一劳永逸的在 FC Bus 和 BC Bus 插入相应的日志记录功能就可以一劳永逸的管理通过总线传输的所以数据。而不再需要再在代码中写入比核心业务更多的而和业务关系不大的日志记录代码。在目前的项目中，数据通过 Request/Response 在系统中流动，当从一个分层过渡到另一个分层后，系统都会根据需要来记录相应的历史纪录。在开发设计中这个功能是非常重要的，由于牵涉调试整个业务流程是否能够正常的运行。就需要依靠这些数据，察看当前的数据路由纪录。在实际的运行中更需要察看所有的日志纪录，用来判断问题范围，以及问题所涉及的逻辑服务。

火龙果•整理
uml.org.cn

第5章 总线通道技术对 SOA 架构的验证

通过第 4 章节对前后端总线通道技术的实现，整个架构中都是以 SOA 架构和 ESB 模式为指导进行设计开发。在本章节中将从 SOA 架构的角度验证项目定制型前后端总线通道技术。并和第三章节中提到的产品集成型 SOA 方案进行比较，阐述项目研发中的心得体会。

5.1 总线通道技术对 SOA 架构的满足性

在第 4 章中阐述了项目中的几种技术实现，在本节中将分析这几种技术如何体现 SOA 架构思想。

5.1.1 分层结构的松耦合性

为了充分体现 SOA 的松耦合性，在项目中开发了分层结构的总线通道技术并组成前后通道。通过实践证明在其应用中确实发挥了分层结构的作用，为此就可以充分体现系统的松耦合性。它在把系统分成三部分的基础上，为各种解决方案提供有力的保证。在前端 FS 上系统可以支持不同表示层技术(Struts、JSF、.Net、Php)来访问系统。在总线通道 FC Bus 和 BC Bus 上则可以支持多种传输协议 IIOP/RMI，MQ，SOAP/WebService 等，用来支持不同的面向服务的业务。TRX 核心业务则可以以即插即用的方式定义和开发。而对于后端 BS 端则可以扩展连接各种异构环境下的系统。在重要的耦合部分都可以通过配置单元来管理。

火龙果•整理
uml.org.cn

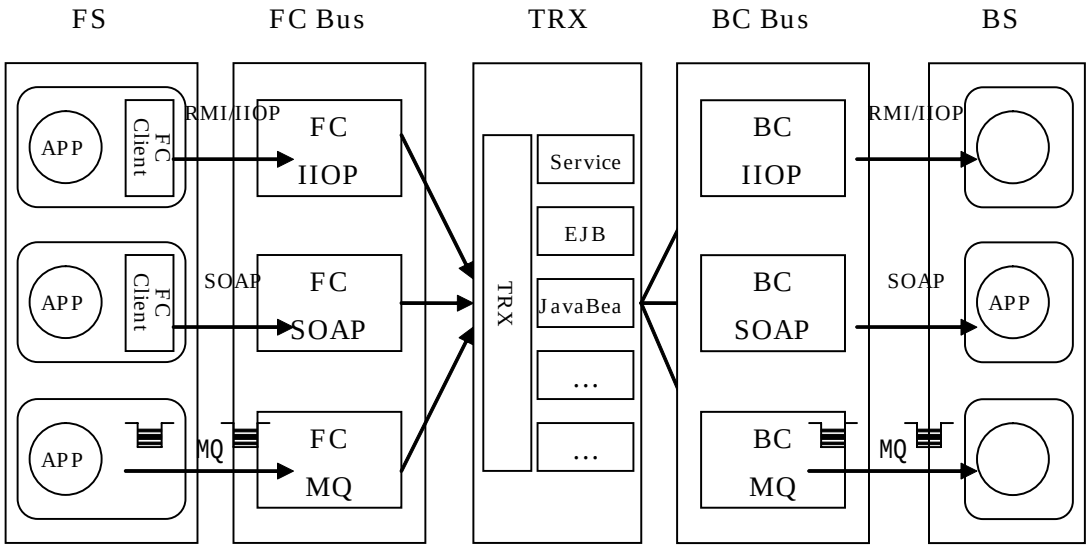

图 5-1 分层结构图

Fig. 5-1 Multiple Layers Structure

在分层的体系结构上主要有 FS,FC Bus, TRX , BC Bus, BS 五个部分组成。通过五个分层就可以更灵活的组织整个系统。

FS 是 Front-End Server 前台服务端，这一层用来表示数据，使用者可以通过 PC, Imode, Vodafone, EZWeb 多种客户端来访问 eMuSC 网上银行。FS 可以使用 Struts, GoldDirect, JSF 等技术来开发。在项目中选用了自行开发的 GoldDirect 技术，这种技术类似于 Struts 中的自定义标签技术。所有 html 上的动态数据都通过 pagevariable 类来表现。而这些 pagevariable 可以同时运用到 PC 终端及三种手机客户端上。

FC 是 Front-End Connector Bus 前端连接总线，这一层用来连接 FS 和 TRX。该层的特点就是可以使用 RMI/IIOP、SOAP、MQ 多种传输协议来连接以达到松耦合的目的。对于前端总线来说，由于考虑到 EJB 接口的方便，所以针对 RMI/IIOP 进行了相应的开发配置。

TRX 是 Transaction Servers 用来定义核心业务逻辑层。开发者可以定义具体的应用服务，使用 IoC, DI 面向接口技术来开发程序。对于这个业务层，可以说是最核心的业务处理中心。

BC 就是 Back-End Connector Bus 后端连接总线，这一层用来连接 TRX 和 BS。该层和 FC 一样可以使用 RMI/IIOP、SOAP、MQ 多种传输协议来连接以达到松耦合的目的。在实际项目中主要考虑到和外界的数据库接口是 MQ 的通道，一种采用 C 的 API 连接方式，另一种则采用 Java 的 Message Driver Bean 的连接方式。

火龙果•整理
uml.org.cn

BS 是 Back-End Server 后台服务端。该层定义了一个方案用来和第三方进行后台数据的同步和传输。在整个 CIS 软件中使用了大量的配置文件，目的是为了降低整个 CIS 系统的耦合性。在其配置属性中，开发者可以为整个 CIS 系统制定总线通道，根据不同的应用服务设定专用通道。根据总线通道的设置情况来为通道建立相应的 MQ 队列。根据总线通道的设置来调整整个 FS，FC，TRX，BC，BS 的软件体系结构，并为总线通道设置相应的流量控制，访问权限。

通过这种分层结构加上后面要介绍的异构平台和集成管理，也就是总线通道的最有特点技术。

在提供解决方案时，松耦合所带来的协议无关性使系统需要异步操作就可以通过服务器端使用消息驱动 Bean(MDB)和 JMS，配合 MQ 方式的传输通道，通过其远程接口调用表示服务的 EJB 组件，以使上述位置独立性最大化。当然在设计通道时也可以使用 SOAP/WebService 组合方式来实现通道。更可以使用 J2EE 的 EJB/IIOP 来实现通道。所以说在总线通道的设计实现上完全根据不同的需求定制松耦合的总线通道。

利用已经封装好的各种服务来构建业务流程。而这些服务也是完全透明的。就算在各个分层上发生变化的时候(变化协议、变化技术、甚至变化介质)，也只需要在本层内部进行修改。对于其他分层也不会产生严重的影响。

5.1.2 自定义通道的扩展性

可扩展性是指在不影响现有系统功能的基础上，为系统添加新的功能或修改现有功能的能力。在项目中通过重组协议，通道，服务的关系开发出自定义通道。传统项目在刚配置好的时候，很难衡量它的可扩展性，直到第一次有需求必须去扩展系统已有功能的时候，才能真正考验整个系统的可扩展性。这就为项目的维护带来一定的风险。而通过自定义通道的定义规则这种方法，可以看到自定义通道的扩展性已经在对业务开发中隐式的得到验证。所有的通道会根据业务的要求选择传输协议，通道类型，而无需破坏原有结构的稳定性。就能很好的按业务需求进行通道。因为 SOA 架构中的不同服务之间本身就保持了一种低依赖的松耦合关系，服务本身是通过标准的接口定义来描述具体的服务内容，并且很好地封装了各层的具体实现。在这里我们也可以从一个方面看到根据 SOA 架构所带来的好处。

火龙果•整理
uml.org.cn

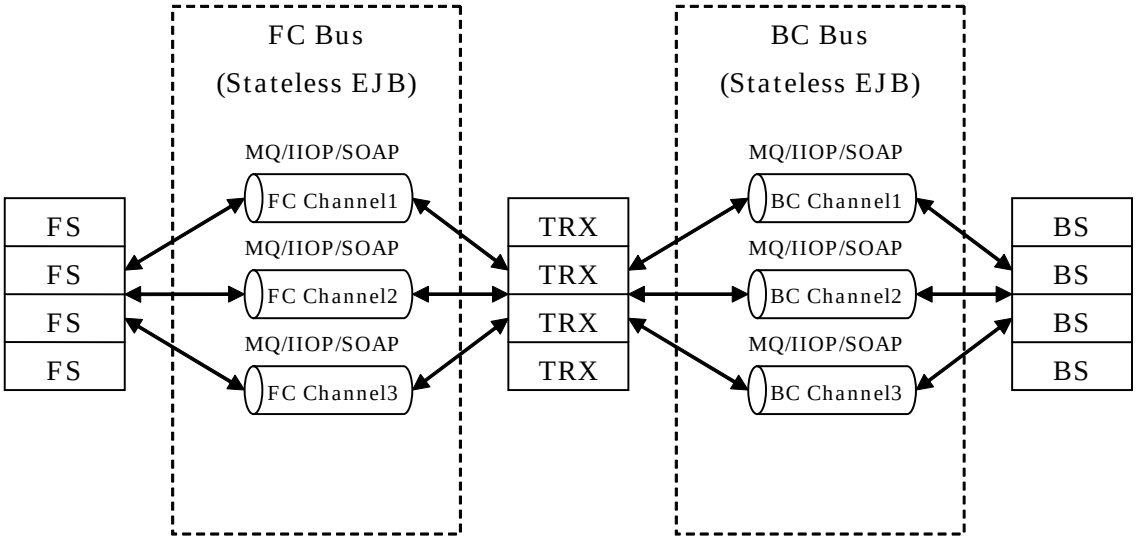

图 5-2 自定义前后端总线通道模型

图 5-2 Customize Front-Back Bus Channel Model

5.1.3 分层和通道的平衡性

在完成对总线通道设计后，需要充分的利用目前所具有的功能。首先从分层结构角度来说，整个系统可以划分成纵向部署结构。对于这种结构来说每一个分层都是独立，当需要替换和更改其中一个分层时，完全可以在不影响其他分层的前提下进行。其次从通道的角度来说，整个系统可以划分成横向部署结构。对于这种结构来说每一个通道就是一个独立的服务，完全可以把不同业务部署到不同的系统环境中。

为了能做到分层和通道的纵横的松耦合性，首先需要纵向对每个分层之间的接口有充分的灵活性。这个问题就可以借助 4.3 多种传输协议的实现多种传输协议来实现，通过多种传输协议分层之间就有了足够了方式进行自由的连接。完全不受到操作系统，传输方式的影响。对于横向来说，就需要使用 4.4 基于依赖关系的松耦合服务以及 4.5 传输通道的定义规则的技术。在设计系统中，每一个服务都是具用原子性的，可以完全独立地作为一个部件存在。只需要通过相应的标示符进行映射，这种技术类似于 Web Services 一样可以方便的被定位寻址。利用这一个特点就可以在传输通道中把前端，业务，后端连接为一个横向的整体。完成一个业务流程的定制。

火龙果•整理
uml.org.cn

从图 5-3 横向通道部署结构图 5-4 纵向分层部署结构中可以看出在总线通道中每一个部分都是独立的。在这种完全松耦合的系统中，任何一个组成部分都可以作为一个零件进行拆装使用。正是由于整个系统的灵活，在建立整个系统的结构时就可以根据以下几个方面的因素，一个是系统的高效稳定，一个是低成本以及方便维护更改。综合几方面来优化平衡总线通道的系统。

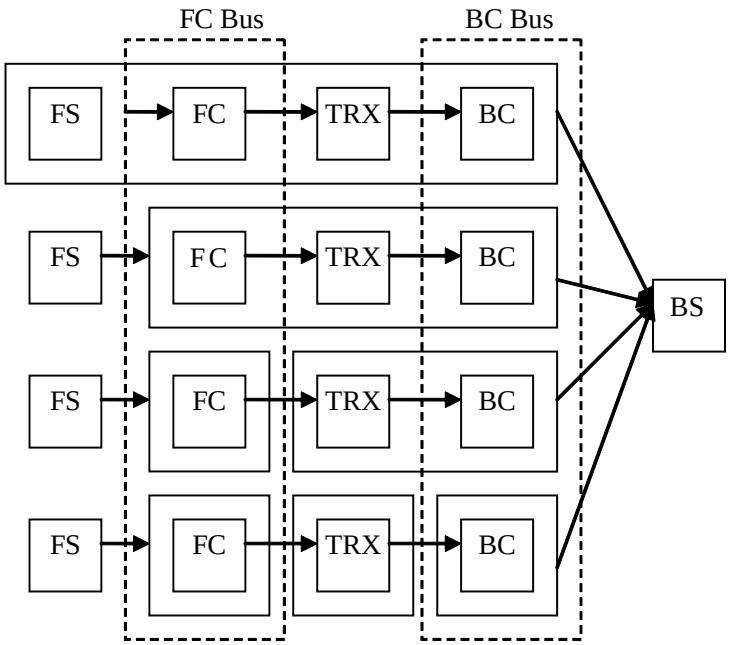

图 5-3 横向通道部署结构

Fig. 5-3 Horizontal Channel Deploying

火龙果•整理
uml.org.cn

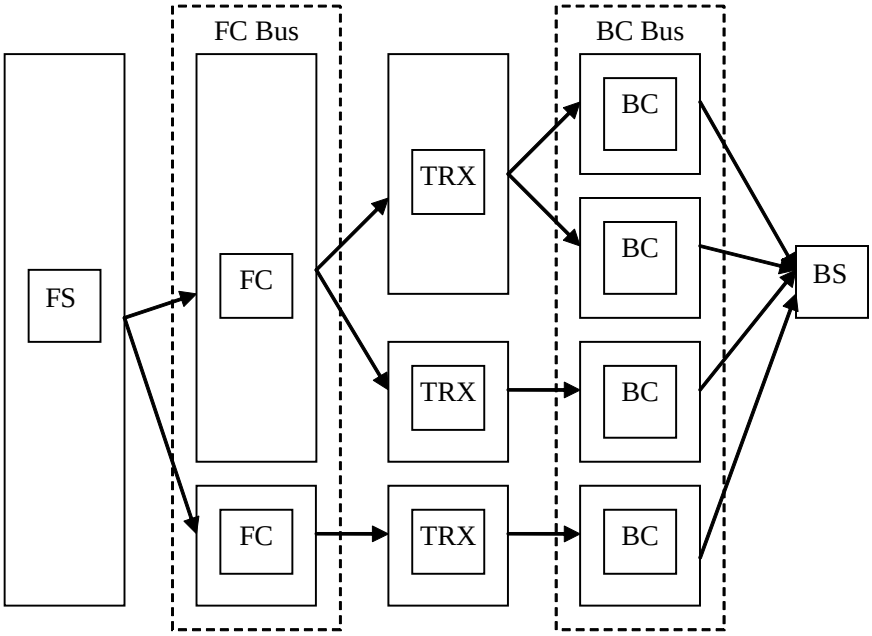

图 5-4 纵向分层部署结构

Fig. 5-4 Vertical Channel Deploying

5.1.4 异构平台部署的灵活性

异构平台部署是指开发服务可以不依赖于操作系统底层的环境。由于整个系统采用分层结构，体系部署上也完全可以根据企业对业务的需求伸缩的部署系统。从**错误！未找到引用源。**上所示每一个 FS,TRX,BS 都是松耦合的，各种组成部分可以理解为 SOA 中的一个服务，每一个服务都能以分布式的形式部署到不同的环境平台。前端 FS 是数据展示服务，中间 TRX 是核心数据处理服务，后端 BS 是外部数据终端服务。方案可以根据不同的平台采用不同的技术并使用相应的传输协议加以组合以总线通道平台为中介进行数据交互。在实际开发上在 Windows、Unix、Linux 平台上自由的选择相应的.Net 技术，J2EE 技术甚至是使用 C,C++开发的 legacy 系统，通过前后总线通道分别部署在异构平台上。

火龙果•整理
uml.org.cn

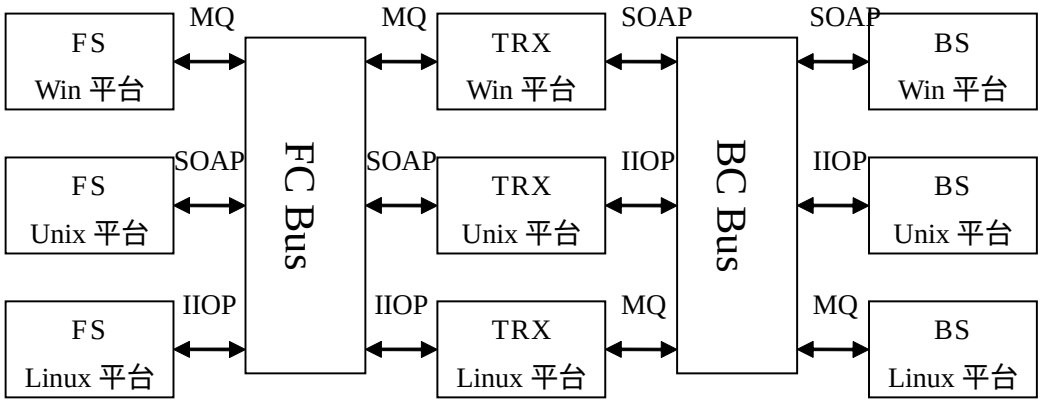

图 5-5 异构平台构成图

Fig. 5-5 Heterogeneous Environment Deployment

以上是从操作系统的角度说灵活性的体现，从业务角度说这种分层模式也大大提高了部署的灵活性。在制定方案时如果需要强健，稳定的部署模式就可以在多个平台上采用负载平衡的方式。这种方式要求有足够硬件和多实例下高负载性能下工作。如果需要低成本的部署模式也可以完全集中在一台服务器上用来减少系统维护的要求。而这些在原先很难同时达到的要求。在前后端的总线通道就可以很灵活的完成。

5.1.5 集成平台管理的控制性

在通过使用Websphere应用服务器来管理应用系统时，应用服务器确实可以提供重复的底层运行信息以方便管理。但对于分层结构的项目来说还不能完全满足要求，所以项目根据应用服务器的规范开发了基于J2EE规范的JMX扩展管理。JMX (Java Management Extensions)，即Java管理扩展,是一个为应用程序、设备、系统等植入管理功能的框架。它可以跨越一系列异构操作系统平台、系统体系结构和网络传输协议，灵活的开发无缝集成的系统、网络和服务管理应用。JMX作为一个基于J2EE标准的管理接口，用户只需装备一次即可通过诸如SNMP、HTTP、RMI、CORBA、SOAP和JMS等多种协议执行管理。JMX强调的是一个管理概念。在管理上使用者可以通过软件的JMX管理平台公布的MBean接口通过分布式的方式管理，部署，控制整个系统。

在实际项目中由于采用了分层的结构，所以FC、TRX、BC灵活部署到应用服务器节点上图 5-6分布式管理控制方式。正如前面研究的整个总线通道技术是完全松耦合的，可以横向纵向地进行拆分整合，但这样对于整个应用服务的集成管理

火龙果•整理
uml.org.cn

就有了一定的难度。根据现在的分层结构在每个节点上插入管理服务模块(AdminService)，并使用J2EE中JMX标准的MBean接口来访问应用服务器。依此应用服务器上的FC、TRX、BC运行状态就可以通过CISAdminService进行收集，并通过管理模块的客户端WebAdmin进行管理控制。而WebAdmin则可以通过SOAP协议统筹连接每一个节点，达到对整个项目系统的管理，控制和动态部署。

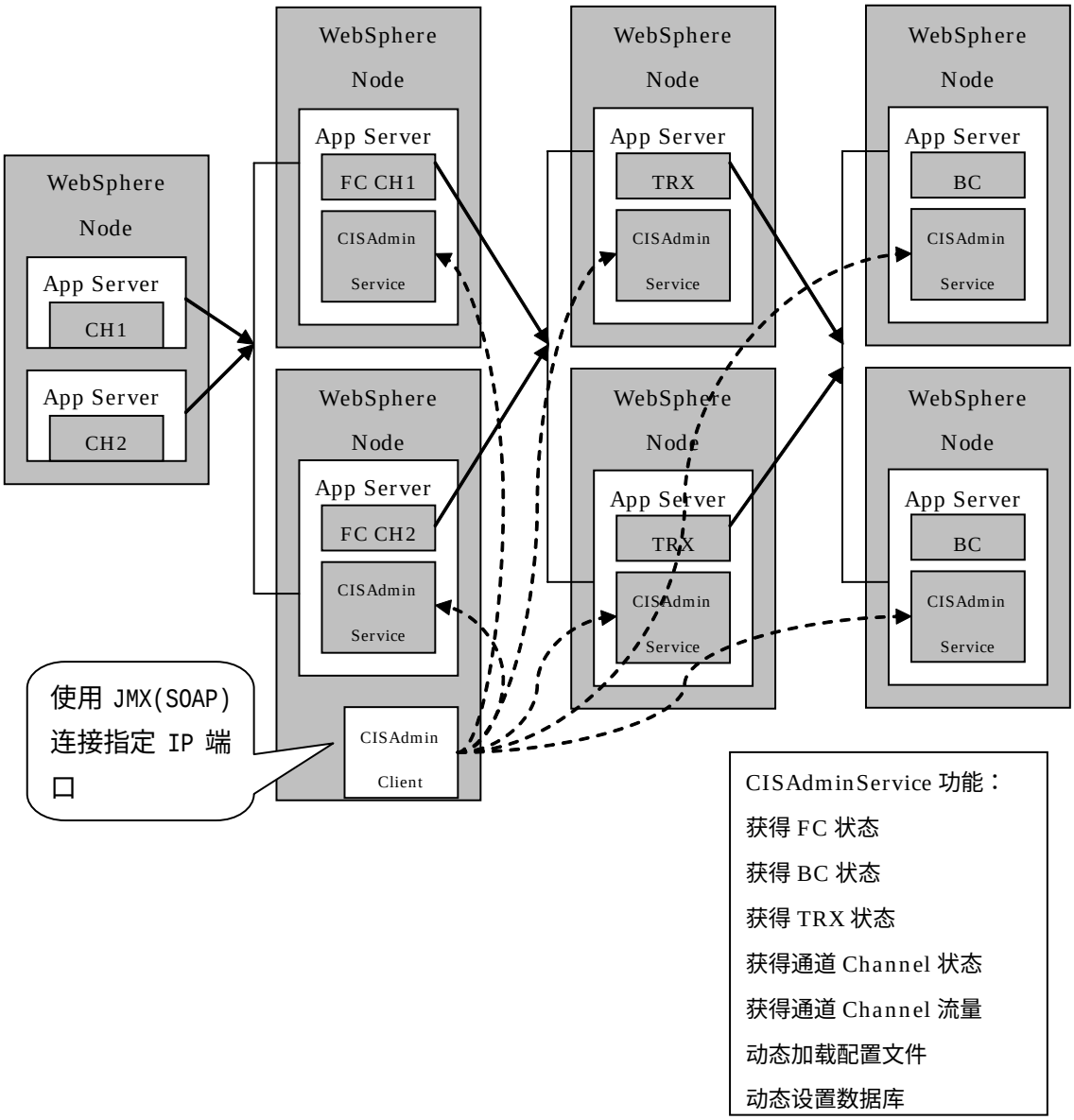

图 5-6 分布式管理控制方式

Fig. 5-6 Distributed Environment Management

火龙果•整理
uml.org.cn

5.2 SOA 开发和配置的心得体会

经过项目的开发，对于 SOA 架构来说有了深刻的理解。本节将通过两个部分来说明自己的心得体会。首先会比较产品集成和项目定制两种不同 ESB 技术平台的 SOA 架构用来阐述各自的优缺点。其次会根据项目开发经验讲述针对面向服务的软件开发。最后还会把实际开发过程中所遇到的一些问题进行梳理，回顾一下开发过程。

5.2.1 两种不同方案 SOA 架构的比较

在基于 SOA 架构的系统中，具体应用程序的功能是由一些松耦合并且具有统一接口定义方式的服务组合构建起来的。ESB 技术平台作为 SOA 架构的本质是用来整合 SOA 项目最合适的工具，它可以用于调节 SOA 中的调用者及服务提供者的机制。使应用和事件驱动服务以一种松散耦合的方式紧密地联系在 SOA 中。并且使得它们能够彼此独立运行使用。所以 ESB 作为 SOA 项目的切入点，可以很好的体现 SOA 的设计思想。根据第 3 章节的 IBM 的 ESB 产品的分析以及第 4 章的节总线通道技术，大致可以把 SOA 架构方案分成两类：产品集成型和项目定制型。

产品定制型：在 SOA 发展的初期，为了解决企业应用中的集成问题提出了以稳定的产品如 WS MQ，WBI Message Broker 等为代表的 WBI5 解决方案。但随着 SOA 的发展，SOA 研究的不断深入。在其最新的 WebSphere 软件中提出了以 WAS6 SIBus 为代表的专用 ESB 平台。WAS6 中提供了崭新的消息服务平台 WPM(WebSphere platform messaging)，并基于这一平台提供了 ESB 的一个具体实现- SIBus(Service Integration Bus) 它可以支持同步和异步的消息通信。总线(Bus)通过互联的消息引擎管理消息源。同时支持基于 Web 服务和 JMS，MQ 格式的消息交互。

项目定制型：目前项目中则是研发了基于总线通道的软件结构的使用 FC Bus 和 BC Bus 为 ESB 技术平台的软件产品方案。通过 EJB 衔接构成前后 ESB 总线结构。在总线上预设了 IIOP、MQ、WebService 三种传输通道方式，通过相应的配置灵活的自定义通道帮助构建业务。定义基于依赖关系的松耦合服务，并在前后的分层结构的 ESB 平台上加入了日志、流量、权限功能。

可以说通过对 SOA 的研究，上述的两种方案都有各自的代表性，可以选择基于成熟 WBI5 产品实现 ESB，可以选择最新 SIBus 技术。但所有的产品都有各自的优点和缺点以及局限性。更可以根据 ESB 模式开发属于自己的 SOA 架构。在纷繁复杂的方案中取其优点，在其基础上进行更为高级的 SOA 实现。

可以说项目就是对 ESB 模式的理解基础上的开发。目前项目所采用的项目软

火龙果•整理
uml.org.cn

件框架和 IBM WebSphere6 中提出的 SIBus 有着异曲同工的功能。不同点在于后者是集成在应用服务器上的。相同点在于两者都有通道 channel 的概念。SIBus 使用 WebService 作为所有服务的 OutBound，把其他的数据接口统一的转换为 WebService，用 WSDL 描述每一个服务的外部接口，再通过 SOAP 协议传输。并在 SIBus 上由 BPEL 流程来动态的管理整个面向服务的软件。当然 SIBus 并不是 ESB 的全部。首先 SIBus 需要完全通过 WebService 的 SOAP 协议作为统一的接口来连接面向服务的应用(EndPoint)。对于使用 RMI/IIOP 和 MQ 的等协议就有相应的限制。而总线通道在流程控制，以及性能上和对服务器的整合上却没有 SIBus 的优势。可以说总线通道和 SIBus 可以互相补充。很多时候，很难界定哪些功能是应该由 SOA 的基础架构(infrastructure)提供的，而哪些应该放在 ESB 的范畴内来解决。放大或突出 ESB 在 SOA 架构中的地位并不很恰当。比较合理的解释是：ESB 应该构筑在完善的 SOA 架构上，做它应该做的事-服务集成。至于怎样集成，应该根据系统上下文环境，考虑有哪些 SOA 的基础设施可供使用，然后再基于 SOA 的基础架构来实现 ESB 设计。

表 5-1 两种 ESB 技术比较

	总线通道(项目定制型 ESB)	SIBus(产品集成型 ESB)
移植性	好，和平台无关	一般
整合性	一般	好，集成在 WebSphere
复杂性	一般	简单
高效性	好，横向和纵向	一般
支持现有资产	好，支持 MOM	好，需要 WebService 作配置
扩展新业务	一般	好，以 WebService 为基础
支持流程编排	通过传输通道定义	BPEL
对业务的控制	好，完全控制	一般
传输协议	IIOP、MQ、WebService 可扩展	WebService 支持 WSIF

从表 5-1 两种 ESB 技术比较分析，在使用两种不同的 ESB 技术来实现 SOA 软件可以说是各有优势和劣势。从移植性上说总线通道采用了 J2EE 实现系统，所以不存在任何的限制。而 SIBus 而言，虽然 IBM 提供对商业级操作系统的支持但移植上依然有很大的风险。从整合性上说 SIBus 是 WebSphere 上最新的 ESB 产品，其内部可以整合 MQ 作为总线通道可以说基本包括了一般 ESB 的所有特性，而总线通道则需要开发者和部署者进行复杂的系统整合。从复杂性上说 SOA 项目是庞大的系统需要考虑方方面面，所以都较为繁琐。从高效性上说总线通道支持多种

火龙果•整理
uml.org.cn

数据传输协议(RMI/IIOP、SOAP、MQ)所以可以直接高效的对接传输。在系统支持横向和纵向的系统结构优化，而 SIBus 则是需要把其他协议转换为 WebService 然后再和 JMS 进行连接，在 SIBus 内部进行定位寻址，所以在效率上相对较低。从支持现有资产上来说总线通道使用了 MOM 直接通过编程和外部系统进行整合，而 SIBus 使用多种 Adaptor 连接器才能够连接各种现有资产系统，同时需要复杂的配置。相反在扩展新业务上 SIBus 由于使用最新技术可以说对新业务有着极大的优势。从对 BPEL 流程控制语言来说 SIBus 是最新的 SOA 架构产品，而总线通道则是通过总线通道的定义来编排业务流程。从业务的控制程度说，由于总线通道的 ESB 模式是为项目系统开发，所以在其调用上是完全无缝连接的。

总线通道同时利用其优势解决了 ESB 产品的一个问题缺点。由于在定义 WebService 中，开发者会发现对业务进行组建时，需要划分业务的颗粒度，当颗粒度过细时，对应的业务则太细致，开发时 Web 服务接口过多，很难管理。当颗粒度过粗时，对应的业务则太大，开发时 Web 服务不灵活，不能充分发挥 Web 服务的效用。而总线通道在这方面提供了一个很好的解决方案。

在项目的前后端系统总线中，就考虑了这个问题并提出了相应的解决方案。在开发时，首先可以通过总线通道从整体上划分业务服务。在划分好相应的业务服务，再根据总线通道的定义在业务服务的内部定义每一个具体的核心服务。通过两层业务划分能够很好的管理每一个通道中的业务服务，而当需要对具体的核心业务进行流程控制和具体调用也可以灵活的从内部获得。

可以说任何的可配置功能都是有局限性的，任何的集成也是有不足之处的。所以说总线通道、SIBus 两者之间都有着各自的优势和劣势。在开发 SOA 项目中不能孤立的认为那一种技术更优秀。在架构 SOA 软件系统初期时需要对两种 ESB 实施方案进行评估取众家之长。根据项目的需要特点来选择组合两种技术。从不同上述列表中的不同角度来架构系统，而不是单纯的选择其中的一种。这样才能达到 SOA 软件项目的最终目标：架构松耦合，灵活，整合，可扩展的软件系统。

5.2.2 适合 SOA 架构的软件开发

面向服务的架构(SOA)不仅需要开发出来的产品体现 SOA 软件的特点，更需要在软件开发中体现面向服务的软件开发。在传统的软件开发中经验，由于过度的强调集成技术的先进性，而忽略了软件开发过程。这就导致很多系统从局部上说是正确而一旦把所有的系统整合在一起时却不能很好的协同工作。所以从软件开发的角度，首先要从全局出发，制定整个企业集成的策略。这种策略包括业务目标，集成路线图，技术规范，和对开发管理提出的新要求。

为此也就提出了面向服务的软件开发方式SOAD(Service Orient Architecture Development)。它是以业务和服务为导向，应用SOA创建解决方案的企业，要重新考虑系统种类、定义团队成员协作、设计解决方案等组件对业务的影响。这种更

火龙果•整理
uml.org.cn

广泛的面向服务的环境就是为面向服务的设计和开发就是 SOAD方法。SOAD 的过程，包括服务的发现(Identification)、服务的描述(Specification)和服务的实现(Realization)三个步骤。

1．服务的发现分成以下三种形式
   - 从业务流程中去抽取服务
   - 利用现有的遗留系统来构建服务
   - 通过目标服务建模抽象服务
2．描述服务：把服务功能规范定义清楚。
3．实现服务：利用传统的面向对象分析设计方法开发实现。

可以说SOAD为项目的需求捕获提供了理论上的依据。同时需要结合项目实际情况。第一：传统金融项目的业务在SOA的架构中都可以直接转化为服务的形式出现，这就使用了“从业务流程中去抽取服务”的方法来抽象。第二：对于一些项目边界外系统以及遗留系统，就可以使用使用“利用现有的遗留系统来构建服务”的方法采用自定义通道的方式来整合。第三：对于金融业务的不断创新，金融衍生产品不断出现。则可以使用“通过目标服务建模抽象服务”的方法来和金融业务共同增值，体现SOA架构的优势。使用以上方法对于服务具体化指出了一条清晰的方向。但这只是面向服务的软件开发的第一步。更为重要的是要让每一位差参与设计开发的成员包括(软件开发人员，业务涉及人员，软件配置人员，管理人员)都知道面向服务的要点以及自己的工作的范围，通过软件编程，平台配置，业务编排等一系列高效的最适合的面向服务的手段完成任务。以服务接口、业务流程的角度解决问题。

5.2.3 SOA 项目实践经验

面向服务的构架不仅需要开发出来的产品体现SOA软件的特点，更需要在软件开发中体现面向服务的软件开发。在传统的软件开发中经验，由于过度的强调集成技术的先进性，而忽略了软件开发过程。这就导致很多系统从局部上说是正确而一旦把所有的系统整合在一起时却不能很好的协同工作。所以从管理的角度，软件开发首先要从全局出发，制定整个企业集成的策略。这种策略包括业务目标，集成路线图，技术规范，和对开发管理提出的新要求。

为此也就提出了面向服务的软件开发方式。通过将若干已有系统的相关功能转化为服务来进行集成。随着这些项目的进行，可重用的服务越来越多，最终，新的集成需求将绝大多数可以通过已有的服务来完成。所以，我们可以从当前重要的集成需求开始来封装已有系统的功能和开发必要的新服务，以渐进的方式逐步地扩展到整个企业范围内的集成。更重要的是，由于服务的灵活性，即使已有系统迁移到新的技术平台，甚至是被替代，都不会影响到依赖于这个应用所提供

火龙果•整理
uml.org.cn

功能的那些应用，从而保证了集成对变化的适应能力，使得业务灵活性有一个坚实的基础。这也意味着从传统规模大、投入大、周期长、见效慢、风险高、专有技术，转向小步快跑、见效迅速、风险低、基于标准的集成方式。

从目前的eMuSC项目可以说就是一个面向服务的开发模式。16个银行的业务是有不同需求的，在开发的时候，我们把银行的业务进行细分，分成若干个服务模块，一旦该银行拥有这项业务就把该业务相应服务加载到系统中，或者在它的基础上再定制。开发者需要做的就是把所有的界面根据银行的要求修改，再选择一种或多种传输协议，配置相应通道连接银行的数据。eMuSC中的每一部分都可以是一个服务模块，也就是说每一部分都是可以替换的，对于在目前的项目FS使用的GoldDirect软件框架，要是银行有其他要求我们也可以灵活的改成Struts或者最新的JSF技术。甚至使用.Net传输SOAP数据。对于BS现在项目使用Message-Driver Bean，也可以使用C语言来写相应的MQ传输数据。对于服务模块也可以根据银行的要求在既用业务上进行二次开发。

所以从开发人员的角度来说，他们使用的工具必须需要以SOA为导向，允许开发人员有效地使用SOA对象。把设计SOA模型、开发服务和服务对象以及测试SOA应用程序这些过程包括进来并组成一个整体。

由于eMuSC是一个规模相当大的项目，同时有行内管理，网上银行，电话银行三大块业务同时开发。并且三块业务分别由三个异地的开发小组实现。还要考虑和Legacy系统的连接，这就牵涉到如何协调完成的问题。这也是对SOA开发实现提供了一个机会，使用SOA面向服务的开发技术，细分模块，归纳数据，整合业务。为了能使开发可以并行开发，服务之间，模块之间的松耦合，在开发中对前台表示层FS，核心业务层TRX，后台同步层BS分别使用来两套开发机制。

首先，对于前台表示层FS，开发者可以通过一整套XML数据来模拟数据库，完成对用户界面的开发，可以在系统开发的初期迅速完成一套系统原型，形成相应的业务轮廓。为核心业务层固定下数据库结构打下扎实的基础。

其次，对于核心业务层TRX，开发者可以完全不关心，表示层界面是如何。只需要根据业务的需求，来访问数据库并配合JUNIT来完成对核心业务的测试。

最后，对于后台的同步层BS，开发者需要考虑如何去把核心业务层的数据和银行数据库进行同步数据。并反馈数据。

项目有一个项目有一个需求，由于eMuSC项目是由IBM集中管理的系统，这就和造成数据并不是集中式储存的，而是分布式储存的。这就牵涉到数据的同步，共享的问题。在项目中使用了两种互为补充的解决方案。对于实时性，稳定性，准确，安全性要求高的业务数据，项目采用IBM MQ中间件技术来传输数据，来保证数据完整性。同时对于一些并不是特别关键的数据，则采用在数据通讯不繁忙的晚上进行批量的数据同步。

所有上述的业务功能都是分地分工分人进行开发的，在开发中会不断发现完

火龙果•整理
uml.org.cn

成的东西需要修改，有的是可以弥补的，有的却是困难重重。正如前一章节介绍的新的业务被不断的发现，旧的功能也能进行合并，复用。和文档的一致性尤为重要。不同的开发人员都是需要以文档和业务模型，完全一致，这也是一个非常繁琐的工作。实际工作中，许多在单体测试中通过的程序，在集成测试中会完全失败。慢慢的开发人员开始适应SOA的项目开发，重视文档，以及对模拟数据的严格要求都是SOA项目的成功的重要因素。

火龙果•整理
uml.org.cn

第6章 结束语

6.1 研究成果

本文以前后端总线通道为切入点，阐述SOA的特点并解释了SOA软件架构对软件发展所带来的影响。分析使用ESB技术来实现SOA架构的方法，使用产品集成构成ESB。同时给出了一个实际项目的ESB解决方案。描述前后端总线通道作为一种ESB技术在整个系统中起到的至关重要的作用。通过对前后端总线通道技术的分析，验证其设计中的独到之处以及如何在实际的项目通过ESB技术体现SOA的设计思想。最后通过两种ESB技术的比较，得出本文对使用ESB架构SOA架构的研究成果。在比较的过程中发现关于产品集成和项目定制的ESB技术其实不是完全对立的，而为了使企业集成能够具有更好的集成功能就需要使两种ESB之间有一定的兼容扩展的能力。在实践的过程中就可以使用SCA和SDO技术来在两种ESB之间建立一个桥梁作用。而这也是有待进一步研究的工作。

6.2 有待进一步开展的工作

在本文中主要讲述了基于SOA的软件架构总线通道的技术方案，并得出相应的研究成果。但对于SOA的架构的研究才刚刚开始。本文没有描述非ESB形式的技术来实现SOA架构的情况。但与ESB技术相比也有其特色的部分。因此在将来的研究和工作将观察SOA的进展。但可以肯定的是每一种技术都有其适用的一面。没有一种技术是万能的，也没有一种技术是毫无作用的。所以可以根据需要正确的使用每一种技术的优点，把各种技术的优点有机的整合在一起就能发挥其最大的功效。

在文中分析的WSIF技术和项目中的传输协议转换可谓是有异曲同工之处。所以在接下去的工作中将研究如何把WSIF引入项目中，更好的提高程序结构的内聚性，同时也可以更好在项目更好发挥WebService的特长，并在此基础上把项目的结构和标准的ESB模型组合发挥更好的效能。对于SCA和SDO技术来说，虽然在本次的论文研究中没有涉及，但对于这两种基于XML技术的SOA技术标准，完全可以通过Wrapper的设计模式把所用的业务接口合数据类型包装为统一的技术标准，以便以两种技术的整合。同时，在文中没有对于SOA架构中的网络安全问题进行研

火龙果•整理
uml.org.cn

究，对于网络安全问题来说一直是分布式软件中一个重要的问题。因此在SOA的环境中也需要有对网络安全有重点考虑。

所以说SOA架构的研究依然存在很大的空间，需要在项目开发中不断的总结研究。在进一步的工作将努力尝试对产品集成型和项目定制型ESB技术平台进行融合渗透，把各自的优势更好的体现在将来的项目中。

火龙果•整理
uml.org.cn

参考文献

[1] 李文军，周晓聪，李师贤.《分布式对象技术》：机械工业出版社，2002.7
[2] 欧阳光，安锦.《JAVA 分布式程序设计》：国防工业出版社，2003.6
[3] 黄刚.《O'Reilly:Java 与 SOAP》：中国电力出版社，2004.5
[4] Scott Seely.《SOAP：XML 跨平台 WebService 开发技术》：机械工业出版社，2003.5
[5] Deepak Alur,John.《J2EE 核心模式(原书第 2 版)》：机械工业出版社，2005-3
[6] Dale R. Nilsson.《J2EE 应用与 IBM WebSphere》：电子工业出版社，2004-8
[7] Howard Kushner .《IBM WebSphere Studio J2EE 应用开发》：机械工业出版社，2004-4
[8] Elisabeth Freeman,Eric Freeman. 《深入浅出设计模式》：东南大学出版社，2006-1
[9] 张洪斌.《WebSphere Server 平台的 J2EE 实开发指南》:机械工业出版社，2006-1
[10] Richard Monson-Haefel .《J2EE WebService 高级编程》:清华大学出版社，2005-4
[11] 刘晶炜，闫健卓.《IBM 信息集成技术原理及应用》：电子工业出版社，2004-5
[12] JAMES SNELL,DOUG TIDWELL.《SOAP WEB 服务开发》:中国电力出版社，2002-9
[13] 科拉夫兹格，本克，斯拉姆.《Enterprise SOA 中文版》:清华大学出版社，2006-7
[14] Rahim Adatia.《EJB 编程指南 Professional EJB》：电子工业出版社，2002-1
[15] 罗时飞.《精通 SPRING》：电子工业出版社，2005-4
[16] MATJAZ B.JURIC.《J2EE EAI 编程指南》：电子工业出版社，2002-9
[17] 李志伟.《IBM DB2 应用开发指南》：清华大学出版社，2003-12
[18] 甘荃，娄丽军.《IBM WEBSPHERE MQ 基础教程》：电子工业出版社，2004-1
[19] Eric Newcomer,Greg Lomow.《Understanding SOA with Web Services 中文版》：电子工业出版社，2006-7
[20] Holt Adams.《选择 SOA 的原因和时机.http://www.ibm.com/developerworks/cn/WebService/ar-itio1/，2006.2
[21] Andre Tost, Rachel Reinitz.《使用 WebSphere Application Server V6 构建企业服务总线 — 第 1 部分》：http://www.ibm.com/developerworks/cn/websphere/techjournal/0501_reinitz/0501_reinitz.html，2005.3
[22] Andre Tost, Rachel Reinitz.《使用 WebSphere Application Server V6 构建企

火龙果•整理
uml.org.cn

业服务总线 — 第 2 部分》：http://www.ibm.com/developerworks/cn/websphere/techjournal/0502_reinitz/0502_reinitz.html，2005.4

[23] Andre Tost, Rachel Reinitz.《使用 WebSphere Application Server V6 构建企业服务总线 — 第 3 部分》：http://www.ibm.com/developerworks/cn/websphere/techjournal/0504_reinitz/0504_reinitz.html，2005.5

[24] Bobby Woolf.《通过服务模拟来简化 SOA 开发》：http://www.ibm.com/developerworks/cn/WebService/ws-mocks/，2006.3

[25] 娄丽军.《WebSphere Business Integration Message Broker 循序渐进(一) 》：http://www.ibm.com/developerworks/cn/websphere/library/techarticles/loulijun/0404_WBIMB/part1.html，2004.4

[26] 娄丽军.《WebSphere Business Integration Message Broker 循序渐进: 第二部分：Message Broker 的应用系统开发》：http://www.ibm.com/developerworks/cn/websphere/library/techarticles/loulijun/0404_WBIMB/part2.html，2004.4

[27] 娄丽军.《WebSphere Business Integration Message Broker 循序渐进: 第三部分：Message Broker 技术纵深》：http://www.ibm.com/developerworks/cn/websphere/library/techarticles/loulijun/0404_WBIMB/part3.html，2004.4

[28] 李珉.《企业服务总线解决方案剖析，第 1 部分: 企业服务总线的基本概念》：http://www.ibm.com/developerworks/cn/WebService/ws-esb1/，2005.8

[29] 周志荣.《企业服务总线解决方案剖析，第 2 部分: 利用 WebSphere 6 2005.9 中的 SIBus 实现》：IBM ESBhttp://www.ibm.com/developerworks/cn/WebService/ws-esb2/，2005.9

[30] 张柯.《企业服务总线解决方案剖析，第 3 部分: 利用 WBI 5 实现 ESB》：http://www.ibm.com/developerworks/cn/WebService/ws-esb3/，2005.9

[31] 张思博，吴宇.《利用 WID 和 WESB 构建基于 Mediation Flow 的 ESB Service》：http://www-128.ibm.com/developerworks/cn/websphere/library/techarticles/0606_zhangsibo，2006-8

[32] 王惇，薛江波.《使用 WebSphere Business Modeler 和 WebSphere Integration 加速企业流程建模和实现》：http://www-128.ibm.com/developerworks/cn/websphere/library/techarticles/0607_wangdun，2006-9

[33] Fuhwei Lwo.《Service Data Objects(SDO) 2.0：创建和读取基于 XML 模式的 XML 文档》：http://www-128.ibm.com/developerworks/cn/webservices/ws-sdoxmlschema，2006-7

[34] IBM,Oracle,SAP.《规范：Service Component Architecture (SCA) 和 Service Data Objects (SDO)》：http://www-128.ibm.com/developerworks/cn/webservices/specification/ws-scasdosumm，2005-11

[35] 方国伟.《SCA（Service Component Architecture）编程模型入门》：http://www.

火龙果•整理
uml.org.cn

ibm.com/developerworks/cn/webservices/ws-sca/，2006-1
[36] 李传峰，刘昕鹏.《WID 服务组件与 JMS 系统的交互方法》：http://www-128.ibm.com/developerworks/cn/websphere/library/techarticles/0602_liuxp1，2006-2
[37] Anh-Khoa Phan，Eric Herness.《Java SCA 调用样式》：http://www-128.ibm.com/developerworks/cn/webservices/ws-soa-scajava/，2006-9

火龙果•整理
uml.org.cn

致谢

本论文在上海交大的导师吴刚副教授的谆谆教导和鼓励下完成的。导师严谨治学的态度和干练务实的作风给我留下深刻的印象。无论是在理论学习阶段，还是在论文的选题、资料查询、开题、研究和撰写的每一个环节，无不得到导师的悉心指导和帮助。我愿借此机会向导师表示衷心的感谢！

回顾两年半的工程硕士学习期间的每个日日夜夜，自己在工作的同时，还需要静心钻研，潜心研究。我要向关心和支持我学习的所有领导、同事和朋友们表示真挚的谢意！感谢他们对我的关心、关注和支持！

在即将毕业之际，我要感谢每一个共同学习同学，和同事给予我的关心和帮助以及学业上的切磋和指点，感谢上海交大软件工程硕士 2004 级工程硕士生全班同学的帮助和勉励。同窗之谊和手足之情，我将终生难忘！

我的妻子，王吉婷，在论文写作期间给与很大的帮助。特别是在论文最关键阶段，由于本人出差，不能提交论文，只能让她代为打印，制作。没有她的热爱和支持，论文就无法按时完成。

我愿在未来的学习和研究过程中，以更加丰厚的成果来答谢曾经关心、帮助和支持过我的所有领导、老师、同学、同事和朋友。

火龙果•整理
uml.org.cn

攻读学位期间发表的学术论文目录

本人出生于1979年12月3日，27岁。于2002年在上海东华大学计算机学院计算机应用专业获得计算机工学学位。并与2004年开始攻读上海交通大学软件学院在职工程硕士学位。目前，就职于IBM ISSC公司，从事软件开发工作。主要研究方向：分布式软件的开发，J2EE架构，IBM WebSphere产品线的研究。目前正积极参与研究SOA架构的软件开发。

2006 年 4 月在计算机工程增刊上发表《前后端总线通道技术在 SOA 架构中的应用》