

上海交通大学

硕士学位论文

SOA的开源实现平台

姓名：马中游

申请学位级别：硕士

专业：软件工程

指导教师：杨旭波;钟友良

20060501

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：马中游

日期： 2006 年 10 月 26 日

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：马中游

指导教师签名：杨旭波

日期：2006 年 10 月 26 日

日期：2006 年 10 月 26 日

SOA 的开源实现平台

摘要

SOA (Service-Oriented Architecture , 面向服务的体系结构) 是近几年在企业应用集成领域备受业界追捧的软件体系结构。目前全球主要的软件提供商都纷纷推出了自己的 SOA 解决方案 , 与此同时也涌现出大量开放源 SOA 项目。然而目前还没有一个开源项目能够提供一个具有普遍性的、相对完善的 SOA 实现方案。

针对以上问题 , 本文提出基于开源技术来构造一个完整的 SOA 实现平台——平台的构建过程从 SOA 的实现角度出发 , 基于 JBI(Java Business Integration)规范进行设计 , 力求体现 SOA 的通用性及灵活性特点。

本文的主要论述思路如下 :

首先对 SOA 开源实现平台的现状进行了综述 , 提出了研究目标 ; 通过分析 SOA 相关的实现技术和具体业务场景 , 明确目标平台的基本模块组成。并对各个模块相应的开源产品进行了选择和评估 , 通过整合所选的开源组件构建出 SOA 的实现平台。

然后对实现平台的架构及开发框架和开发流程进行了分析。并通过静态 WEB 页面整合和 WEB 服务复合两个示例分别介绍如何针对具体领域对 SOA 的开发平台进行扩展 , 以及如何利用这个平台实现 SOA 的业务流程。

最后对全文进行总结 , 分析了平台的现状和不足 , 以及未来可能遭遇的问题。并结合这些问题对下一步的工作进行了展望。

目前平台已经过测试和评估 , 并且作为一个开源项目发布给用户试用 , 同时提供了相应的文档和在线技术支持。

关键词 : 面向服务的体系架构 , WEB服务 , 业务流程执行语言 , 企业服务总线 , SOA实现平台

OPEN SOURCE SOA PLATFORM

ABSTRACT

SOA (Service-Oriented Architecture) is the most advanced and popular software frame structure in the field of enterprise application integration in recent years. Currently, the world's leading software providers have launched their own SOA solutions, while SOA has emerged in a large number of open-source projects. However, a few items have not been able to provide a universal, relatively sound SOA Implementation.

To solve the above problem, this paper intent to implement open source technology-based structure to achieve a complete SOA platform -- from the process of building a platform for the realization of SOA perspective, based on the JBI (Java Business Integration) standard design, every effort has been made to reflect the overall SOA and flexibility features.

First of all, we will analyze the status of existing SOA open source platforms, which lead to the study objectives of this article. We will also look at the related technology with open-source tools to implement SOA solution, combine with the analyses on SOA modules and practical case study, and eventually confirm the basic modules of our target platform. Then we will compare and analyze the open-source products for each module based on our module selection standards, and complete a SOA solution platform using open-sourced pattern technology.

Secondly, we will analyze the structure of the SOA solution platform; provide the standard development frameworks and processes. We will discuss how to expand the SOA solution platform according to different application area via 2 case studies on static Web page integration project and BPEL project, then further develop and operate SOA solution process on this platform.

Finally, we will sum up the important points that have been studied in the article, and analyze the insufficiency and shortcomings of the current solution and the possible problems would occur in the future. And based on those analyses, we will present the prospect on the next objectives of our research.

Now the platform had been developed and tested, and it had been published online as an open-source project for free downloading. Users can also have related technology documents and online support.

Keywords: Service-Oriented Architecture, Web Services, Business Process Execution Language, Enterprise Service Bus, SOA Platform

第一章 概述

1.1 引言

目前在企业级应用开发领域，越来越多的厂商在寻求一种灵活的 IT 解决方案，来应对快速变化的业务需求。这使得 SOA(service-oriented architecture, 面向服务的体系架构)技术成为业界追捧的热点。众多的软件厂商对它趋之若鹜，由此掀起新一轮企业架构浪潮。各种 SOA 技术和产品如雨后春笋般涌现出来，它们来自不同的厂商和阵营，遵循不同的协议和标准，覆盖了 SOA 相关的各个领域。面对如此多的技术和产品使得用户希望构建 SOA 系统时很难做出选择。因此提供一个统一和通用的 SOA 实现平台的需求已变得十分迫切。通过它用户应当即可以基于标准的开发架构进行 SOA 开发，又可以部署及运行 SOA 系统。

1.2 研究背景和动机

SOA 的主要目标是在 SOI(service orient integrate, 面向服务集成)领域实现跨系统、跨平台的企业应用整合。它作为未来企业整合技术的主流发展方向，自然受到业界的追捧。目前，全球主要的商业软件提供商 IBM、BEA、ORACLE、微软等都纷纷提供了自己的 SOA 解决方案。

同时大量活跃的开放源 SOA 项目也纷纷涌现出来，如基于 Java 语言的 WEB 服务引擎 Axis (Apache eXtensible Interaction System)、BPEL 引擎 PXE (Process eXecution Engine)以及基于 JBI (Java Business Integration)规范的开源 ESB(Enterprise Service Bus) 产品 ServiceMix 等。

这些开源项目大都专注于 SOA 的某一领域，在遵循业界规范的基础上提供前沿的技术实现方案。但遗憾的是，开源软件往往只专注于某个技术局部，针对于解决 SOA 系统中的某些特定问题。要想真正解决复杂的企业级应用，需要对多方面的组件进行组合。而开源组件间的协调性不如商业软件，往往需要第三方进行整合和加工。目前还没有一个开源项目能够提供一个具有普遍性的，相对完善的 SOA 解决方案。

本文的目标就是希望提供给用户一个基于开源技术的 SOA 系统实现平台。这个实现平台应该既可以辅助开发人员构建企业的 SOA 应用，又可以作为 SOA 系统的部署和运行平台。

1.3 研究目的和意义

与商业 SOA 软件相比较，一个开源 SOA 实现平台的价值在于，它可以更好的实现 SOA 的设计基本原则：

按照业务需求自顶向下分析服务，根据现有系统功能自底向上提供服务，通过两者相互结合提供实现。

遵循这一原则，SOA 的系统实现应该由多层次实现来组合形成。而目前，各种层次不同的 SOA 需求却都要依靠商业软件来实现，这对于低层、简单的需求来说代价太贵，门槛太高。

而且商业软件在设计思路可能会趋向于厂商自身对 SOA 架构的理解，带一些厂商自己的风格，实现上往往和厂商自己的其他产品或技术捆绑在一起，由于 SOA 技术正处在不断完善和发展的过程中，这可能存在技术壁垒的风险，有可能会在以后整合 SOA 系统时带来一些不良影响。

此外，SOA 通常需要很多组件来实现。事实上一个完整的 SOA 解决方案会用到多种中间件技术。商业软件的厂商通常会自行开发某个或某些方面的组件，而其他组件通常通过购买获得，这些组件未必都适合用户的需求。而由开源组件组成的 SOA 平台则可以很容易地选择各个领域中最成熟的组件，将它们组合成相对稳定、满足用户技术及业务需求的 SOA 解决方案。随着技术的发展还可以灵活的替换某些组件。

本文的主要目的是：

- 提供一个尽可能完善的 SOA 解决方案，以满足业界对可执行的 SOA 架构的组合应用和事件驱动的需求。
- 作为一个通用的运行平台，尽可能采纳开源软件，特别要考虑版权的限制情况。
- 充分满足企业对于松耦合设计、基于业界标准和充分兼容的需求。

1.4 研究思路和文章结构

论文的研究思路是遵循软件工程的思想进行分析和设计，主要包括了技术研究、模块分析、组件分析、组件评估、平台整合、平台扩展、开发示例、系统部署及测试等。

整篇论文的构架有两条主线，一条是基于如何构建 SOA 实现平台。首先介绍了 SOA 的相关技术。然后结合 SOA 实现模式对具体案例进行分析，明确了 SOA 实现平台的模块结构，再根据我们组件选择的标准对每个模块相关的开源产品进行选择 and 评估，明确平台的构成，最终在此基础上进行整合和扩展，组建起 SOA 的实现平台。

另外一条主线是介绍如何使用及扩展平台，通过一个对 WEB 系统表现层进行整合的组件扩展模块以及一个业务流程系统的示例，来介绍如何使用平台提供的开发流程来开发 SOA 系统。

本文最后还分析了平台当前的急需解决的不足和以后可能遭遇的问题。并基于这些问题对下一步要做的工作和平台以后的扩展方向做了大胆的构思。

从文章的结构看全文共分八章：

第一章概述阐明了研究 SOA 开源实现平台的背景和动机、研究目的和意义以及论文的结构。

第二章介绍了 SOA 的相关实现技术的概念和作用。

第三章对 SOA 的通用场景进行了分析，进而对 SOA 系统的结构组成进行了探讨，明确了我们目标实现平台所需的基本模块组成。

第四章在上一章的分析结果的基础上，对各个模块相应的开源产品进行了选择和评估。确定出平台最终所需的开源产品。此外还介绍了对这些组件进行的整合工作。

第五章介绍了 SOA 实现平台的架构，开发和运行平台的特点，开发框架和开发流程等。

第六章主要阐释了针对静态 WEB 页面整合领域做了一些扩展与尝试，介绍了如何针对具体领域对 SOA 的开发平台进行扩展。也就是对标准开发框架进行的扩展。

第七章主要利用一个业务流程处理系统的示例来展示如何使用开发平台和开发框架。

第八章主要对全文做了总结，介绍了系统现状及特点，对开发过程所遭遇的问题以及以后可能会碰到的问题进行了分析，总结和展望。

1.5 本章小结

本章主要介绍了当前 SOA 实现平台的背景，阐述了文章的具体目标，研究目的和意义以及研究构架。此外还介绍了论文的组织结构。

第二章 相关技术

要构建一个 SOA 实现平台，我们必须明确什么是 SOA？为什么要用 SOA？如何实现 SOA 等，只有了解了这些才能明确对实现平台的需求。下面我们介绍与 SOA 实现平台相关的一些思想及技术。

2.1 SOA

SOA[1] 的核心思想是创建一个组件模型，把每个功能单元定义为一个服务，通过服务之间定义良好的接口和契约，对现有功能进行重用和扩展以及构造新的应用，把它们组织起来构建起整个应用系统。

SOA 有三个基本原则，它们是松耦合、粗粒度、异步。松耦合指服务接口的定义独立于实现服务的硬件平台、操作系统和编程语言。系统的各个组件之间的松耦合消除了某一组件需要更改时对系统其它组件的影响。粗粒度指在相对较粗的粒度上对应用服务或业务模块进行封装与重用。异步的意思是指业务流程的异步执行(Asynchrony)。

SOA 通过将遗留系统包装为服务的方法，把历史的资源提供给其它系统，使得企业资源得到充分利用。这些资源的结构可能是面向对象、面向服务、面向事件、面向过程等。因此 SOA 适用于较为复杂的业务环境，比如企业应用集成，事实上应用集成可以说是 SOA 技术发展的主要推动力。利用 SOA 框架企业 IT 系统既可以充分利用现有系统的功能，又可以在未来遇到问题时灵活的做一些改变来应对。

SOA 的思想并不是突然冒出来的，以前的 CORBA(Common Object Request Broker Architecture，通用对象请求代理体系结构)、DCOM(Distributed COM)等就是基于类似服务的观念，但与现在意义上的 SOA 不同的是这些过去的面向服务架构都存在某些致命缺陷。首先它们并没有成为业界真正认同的标准，技术上受到某些厂商的垄断和约束。属于 Microsoft 阵营的 DCOM 自不必说了。CORBA 虽然看上去是标准化的，但实际上还是受到很多厂商的影响，如果想真正实现它往往需要和某个厂商对规范的具体实现相结合[2]。最重要的是它们的组件之间并不是松耦合的。

今天的 SOA 已经发生了显著变化，它基于已被业界广泛接受的 Web 服务标准，以

XML(eXtensible Markup Language , 可扩展标记语言)为基础, 通过使用基于 XML 的 WSDL(Web Services Definition Language , Web 服务描述语言)来描述服务, 实现了服务描述与实现的松耦合。因此 SOA 代表了软件架构的发展趋势 ,Gartner Group 预测到 2008 年, SOA 将成为占有统治地位的软件架构[3]。

2.2 WEB服务

目前业界最为广泛接受的 SOA 技术就是 WEB 服务(Web Services)[4]。把 WEB 服务技术和以前的分布式计算技术进行对比的话, 它具有更好的可靠性、扩展性、易用性以及协议开放性[5]。事实上 WEB 服务并非直接等同于 SOA, 也并非是实现 SOA 的必需组件 [6]。但由于它所具有的上述优势使其在实现 SOA 时具有非常重要的地位, 实现 SOA 过程中往往被优先考虑采用。

Web 服务是建立在 SOAP(Simple Object Access Protocol) [7], WSDL [7]和 UDDI(Universal Description , Discovery and Integration)[7]三个 XML 协议之上。WEB 服务通过 WSDL 语言描述将服务交互所需的全部细节, 包括消息格式、传输协议和位置定义为接口。该接口隐藏了服务实现的细节, 允许通过独立于服务实现、独立于硬件或软件平台、独立于编写服务语言的方式使用该服务。WEB 服务通过 SOAP 协议采用真正与平台无关的方式来描述任何数据格式, 以跨系统交换数据, 实现了 SOA 的松耦合原则。

WEB服务可以看作实现SOA的基础。要想真正实现一个复杂的企业应用集成系统, 仅靠WEB服务技术还远远不够, 如何协调WEB服务之间的关系, 管理数据的交换, 控制系统的安全、可靠性、事物等多方面问题都需要解决。这就需要用到其它相关的技术如WEB复合技术和企业服务总线技术等。

2.3 WEB服务复合技术

单一的WEB服务只能提供有限的功能。Web服务的发展趋向于Web服务的复合, 复合的本质就是通过流程定义使一组独立的Web服务相互协作, 组织成新的上层的服务实现应用的集成。从而满足复杂的业务需求。由此可以看出Web服务复合其实质就是BPM (Business Process Management , 业务流程管理)[8]。

目前随着SOA, WEB服务技术的逐步成熟, 涌现出不少的BPM相关业界标准, 比如WfMC's XPD (XML-based Process Definition Language) 、 BPMI's BPML (Business

Process Modeling Language, 商业流程建模语言) [9]、ebXML's BPSS (Business Process Specification Schema)、BPEL4WS (Business Process Execution Language for Web Services, WEB服务的业务流程执行语言)等。

这么多的标准使得我们在选择时有点无所适从,目前从许多迹象显示,显然是 BPEL 较占上风,也许随着 BPEL2.0 的正式推出,BPEL 将结束目前这种比较混乱的状态。所以我们将 WEB 服务复合技术的关注点放在 BPEL 上。

2.3.1 BPEL

BPEL 是基于 WEB 服务技术的工作流语言,由 Microsoft, IBM 和 BEA 在 2002 年共同推出,在 2003 年推出了 BPEL4WS 1.1[10] 版后被 OASIS(结构化信息标准促进组织)实现标准化,并组建了 Web 服务业务流程执行语言技术委员会(WSBPEL TC) [11]。现在 BPEL2.0[12]版本也即将正式推出。

BPEL 为指定基于 Web 服务的业务流程行为定义了一种表示方法。作为一种可执行流程的实现语言,BPEL 能够将现有的多个服务复合起来,对它们加以编排和协调,使它们按照一定业务流程组成新的服务。BPEL 可以指定多个 WEB 服务的调用顺序,这些 Web 服务间共享的数据,业务流程涉及的伙伴、角色,并进行逻辑约束、异常处理、并行处理、数据转换等,还允许 Web 服务间长期运行的事务处理。

BPEL 所表示的流程通过使用 WEB 服务接口来访问流程[13]。它的应用方式有两种[14]。一种是使用抽象流程,BPEL 流程可以定义业务协议角色。每个角色都有自己的抽象流程。抽象流程对待数据处理的方式反映了描述业务协议公共部分所需的抽象程度。也就是说抽象流程仅处理有关协议的数据。BPEL 提供了把有关协议的数据识别为消息属性的方式。另一种就是定义可执行业务流程,通过流程的逻辑和状态来控制 Web 服务的执行顺序和交互[15]。

由于 BPEL 通过 XML 语义来定义 WEB 服务的业务流程行为,脱离了开发平台的约束。利用合作伙伴之间的交互实现流程与服务之间的调用。合作伙伴可作为服务的提供者,也可作为流程的请求者,或者参与到流程的双向交互中。利用 BPEL 可以灵活的构造复杂的分布式系统,系统可以由不同架构不同平台的 WEB 服务组成,根据运行时的状态动态的决定流程的走向。

但 SOA 框架并不局限于 WEB 服务,实际系统结构复杂的多,往往有很多不同结构的组件和遗留系统组成如消息驱动组件、EJB、CORBA 组件、WEB 站点等。为了提供对这些组件和消息的透明支持和安全性、策略、可靠性等方面的要求,在 SOA 体系

结构中引入了一个新的软件概念。ESB(Enterprise Service Bus , 企业服务总线)。

2.4 ESB

ESB[16]是SOA架构中的一组基础功能组件，它以一种标准、可靠的总线交互模式实现异构环境中的服务、消息以及事件的通信、仲裁和控制机制。在ESB的内部，活动是运行在分布环境中，基于消息的通信。

ESB就如同企业应用系统中的一条骨干神经，服务交互双方并不直接交互，而是通过ESB整合在一起，由它按照业务需求逻辑访问相应的服务和控制消息的流转。由于ESB提供了统一的信息交换平台和信息格式，降低了系统间的转换的数量和复杂度，提高了复用度，从而降低系统整合的成本。因此ESB成为一种标准简捷的企业应用整合SOA架构方案。

ESB框架主要有三部分组成：总线、集成在总线上面的服务和负责逻辑控制的底层服务。它的核心是总线，总线提供高吞吐量，高可靠的信息传输服务。[17]

ESB能够作为一种EAI(Enterprise Application Integration，企业应用集成)解决方案满足企业客户的复杂需求[18][19]。

2.5 JBI

由于在SOA中采用ESB可以降低系统整合的成本，因此采用ESB技术组建SOA系统成为一种标准简捷的SOA解决方案。ESB只是一种框架，并不是系统层次的成果。于是，业界又提出了ESB的实现规范JBI[20]。

JBI可以看作是一个用于规范化信息服务和路由器的标准容器，这个容器包含了一些用于部署集成服务的组件和管理模型。作为SOA的基础构架可以部署包含业务逻辑和功能的SE(Service Engine, 服务引擎组件)以及提供通信协议转换的BC(Binding Component, 绑定组件)[21]。

SE可提供内部服务如业务流程 bpel，数据转换 xslt、业务逻辑 EJB 等。BC为访问外部服务及给外部提供服务进行通信协议转换，它即可作为 soap，AS2，ebXML File，XML-over-HTTP，EDI 等协议的远程服务代理又可以将内部的服务提供给外部。

部署在JBI中的组件即可以作为服务提供者又可以作为服务请求者或者两者兼而有之。作为服务提供者时可通过WSDL格式将服务的消息、操作和消息交换模式、服务端点描述出来在NMR(Normalized Message Router，规范消息路由器)中注册。

JB1 的核心是 NMR。NMR 作为规范化的消息容器，控制交换的状态和信息，对交换进行调停。消息的交换基于 WSDL2.0 的消息交换模式。交互模式为 In-Only、Robust In-Only、In-OutIn、Optional-Out 四种。交换的消息必须转化为规范化的消息，该消息为 WSDL 格式并附有消息的属性。消息属性中可包含支持的相关协议信息、安全信息、转换信息、以及其他组件可以识别的数据。

JB1 提供给用户一个使用基于开放标准的扩展模式组合应用系统的开发途径。组件提供者可以基于 JB1 开发可应用到任何 JB1 平台的组件，而集成商则可以基于 SOA 原则利用这些组件便捷的整合服务。

2.6 SOA实现方式

为了理解如何实现SOA。就必须分析SOA的实现方式，本节将对通用的SOA的实现模式[16]和服务访问方式进行介绍，看看SOA系统究竟如何实现。

2.6.1 实现模式

SOA 的实现模式主要有七种，它们是 SOA 实现逻辑的基础，通过对它们进行组合就可以实现复杂的 SOA 系统逻辑。

- 基础 WEB 服务模式：

如图 2-1 所示，这是最简单和基础的方式，可以很简单的实现点对点的 WEB 服务请求模式。

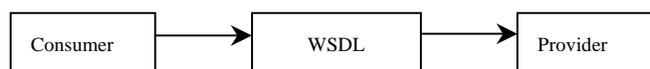


图 2-1 基础 WEB 服务模式
Fig 2-1 Basic Web Services Pattern

- 抽象远程服务模式：

通过将服务提供者的服务描述抽象的方法，使得同一个服务描述可能对应多个服务的实现，从而降低服务提供者发生改变时对服务请求方的影响。如图 2-2 所示。

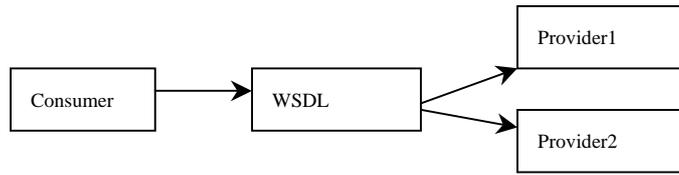


图 2-2 抽象远程服务模式

Fig 2-2 Abstract Remote Service Pattern

- 服务包装模式

提供一种将现有系统转化为 SOA 的机制，把现有功能包装为一个服务。使得服务请求者可以访问到非 SOA 的遗留系统而不必关心数据格式或传输协议的不同。如图 2-3 所示。



图 2-3 服务包装模式

Fig 2-3 Service Adaptor Pattern

- 服务代理模式

这种模式服务请求者不直接访问服务提供方的服务，而通过代理访问服务。这样服务请求者不必关心服务提供者的具体位置信息，而且还可以通过代理选择最佳服务。如图2-4所示。

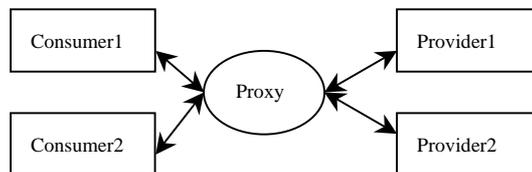


图 2-4 服务代理模式

Fig 2-4 Service Proxy Pattern

- 虚拟服务模式

如果服务提供者没有直接提供请求者想要的服务，可通过组合包装服务和代理的方法实现一个虚拟服务，它通过一个服务代理描述所需服务，通过包装服务将现有系统的功能转化为服务，利用 Façade 模式实现两个服务之间差别的转换。如图 2-5 所示。

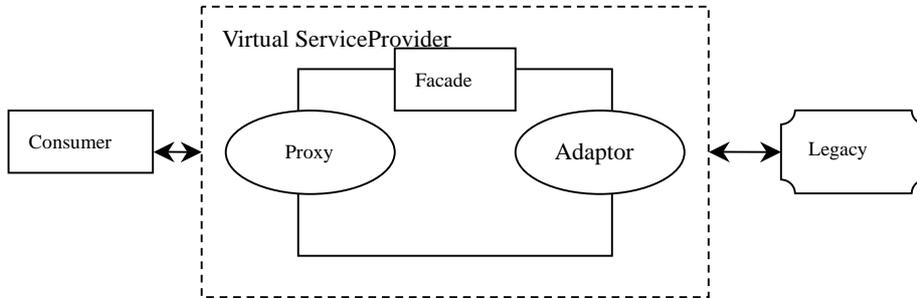


图 2-5 虚拟服务模式

Fig 2-5 Virtual Service Pattern

- 整合服务模式

整合现有服务的功能，按照业务流程进行组合形成新的服务。BPEL 就是它的一种实现。它是对 WEB 服务的一种进化，采用了工作流的概念，将流程用到的服务组织成一个逻辑上新的服务，以前需要在客户端处理的逻辑判断和数据格式转换放到服务端。客户可以对服务流程提出请求，而流程将访问多个服务。即将本来需要的多个点对点操作简化为 1 次操作。如图 2-6 所示。

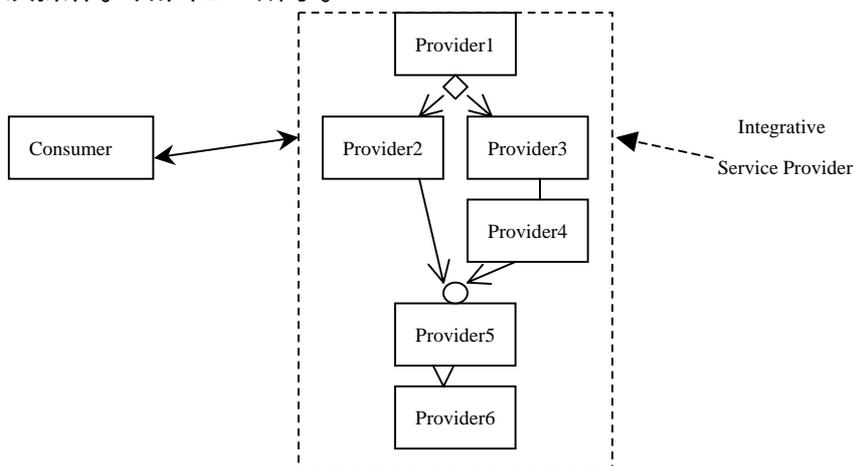


图 2-6 整合服务模式

Fig 2-6 Integrative Service Pattern

- ESB 模式

通过公用的底层框架提供服务通信、调停、转换和整合功能。同时对服务管理、安全、监控进行集中控制。它主要适合企业有大量日益增加的服务需要一个通用的管理和操作模式进行控制。ESB 模式的特点主要是：

支持大量的可管理的服务交互。提供高级的服务交互能力(事务、持久、底层服务、安全、质量等)。支持多种交互形式同步、异步处理、消息、事件。提供可管理的分布式整合底层框架支持。支持服务路由和协议转换和其它消息处理。支持 WEB 服务和其它传统的 EAI 通信标准。如图 2-7 所示。

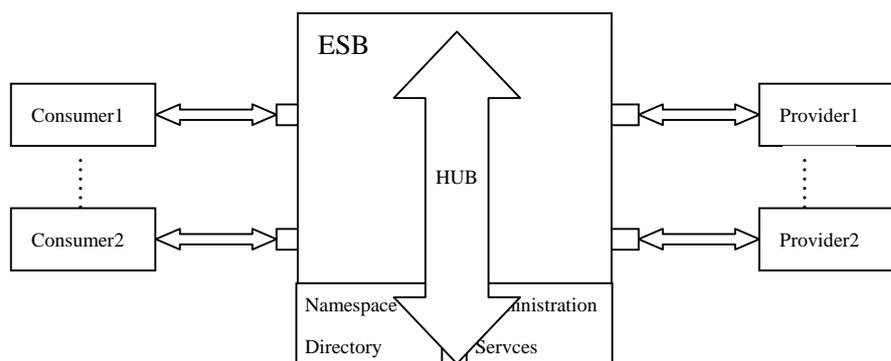


图 2-7 ESB 模式

Fig 2-7 ESB Pattern

各种模式之间相互联系，高层模式通常建立在低层模式之上，可能由几种底层模式组合而成。但并不是高层的模式就是最适合的模式，往往需要结合具体的业务需求对它们进行选择 and 组合。

2.6.2 访问方式

访问方式指的是服务请求者如何访问提供者所提供的服务。笼统划分的话，访问方式可简单分为两种，同步与异步访问。同步是指请求者发送服务请求后必需要等到响应返回后才进行下面的流程。而异步则是指请求者访问服务时不需要阻塞线程等待响应信息，而是继续执行其它流程，通过另一个单独的线程来接收响应。如果结合 SOA 实现模式对服务访问方式进行细分的话，访问方式又可分为同步直接访问、同步代理访问、异步代理访问三种。

- 同步直接访问

同步直接访问方式类似于执行函数调用，请求者需要得到端点的地址，并直接调用

它。比如通过 SOAP 直接调用 Web 服务的方式就是一种同步直接访问。每一 Web 服务对应特定的 URI (Uniform Resource Identifier)。当同一类型的服务有多个提供者时，每个提供者的端点必须不同，以便请求者可以进行选择。

- **同步代理访问**

同步代理调用通过引入一个代理，作为访问服务的中介。请求者不再直接访问服务，而是通过服务代理来调用服务提供者提供的服务。由于代理地址相对稳定，请求者只需查找一次代理的地址，以后就可以直接使用它。而选择提供者工作由代理负责，从而减轻了请求方的负担。

- **异步代理访问**

异步代理访问是通过消息传递系统实现的，消息传递系统使用消息队列来发送请求和接收响应。由消息传递系统负责选择服务的提供者，从而简化了请求方的工作。而能够提供消息处理的代理实际上就是 ESB 功能的一部分。ESB 可以支持同步和异步的代理访问方式。

2.7 本章小结

本章主要介绍了 SOA 框架的概念以及 SOA 的一些相关技术和实现模式。

第三章 平台组成分析

本章将介绍针对SOA实现平台所进行的模块分析工作。这也是平台实现首要的工作部分。只有明确了平台的组成结构，才能具体实现各个模块，进而提供给用户一个完善的产品。然而要准确分析出SOA实现平台的具体模块组成，必须充分理解实现SOA系统的业务需求，因此本章对几个具有普遍意义的业务场景进行了分析，并结合SOA的实现模式和访问方式，提出了相应实现方案，通过这些实现方案来确定平台的必要模块。

3.1 业务需求分析

- 第一种场景比较简单，某企业想对外界提供一个查询产品的WEB服务。

这是最为简单和普遍的业务需求，利用WEB服务给企业内部或外部的用户提供某项功能。解决方案是如果企业中已有系统有相似功能，可通过服务包装，服务代理等受限模式将现有系统的功能包装为WEB服务既可。如果没有相似功能的话，过程更加简单，只需创建一个WEB服务既可。当然实现时可以通过抽象服务，服务代理等模式优化系统结构。

- 第二种业务场景是企业想把现有的多个WEB系统中的功能转化为WEB服务，并进行整合。

现在企业内部网中通常会有很多独立的WEB系统，如雇员信息管理，个人时间管理，会议室预定，财产管理等等。这些WEB系统通常都是独立开发的，可能是由HTML，ASP，JAVA，JSP，PHP等技术实现的，数据库也可能不同。因此可能会对用户操作带来很多麻烦，例如当某人想要计划一个会议，他首先要进入个人时间管理系统，去查找可能的空闲时间作为候选，然后他需要进入会议室预定系统去查看候选时间内是否有空的会议室，如果有的话就进行预定，预定之后还要再次进入个人时间管理系统，填写时间安排。

毫无疑问这种事情会经常发生，你不得不在多个系统之间来回反复。可能每次进入系统还要重新登陆。而且不同的系统间通常会有很多重复信息和重复工作，这些问题导致了企业信息系统效率的不断降低。

理想的情况下把这些系统整合成一个新系统,使得用户可以在一个界面完成一次任务,比如用户可以同时看到个人时间安排和会议室安排,并且可以同时选择会议室和填写时间安排。

解决方案是这种场景是一种企业应用的整合。把现有的一些 WEB 系统功能整合起来,实现一些新的功能。结合 SOA 具体的实现模式,可先通过服务包装,服务代理等模式将现有的 WEB 系统的功能都转化为一个个的 WEB 服务,然后通过整合服务模式根据业务需求对所要用到的 WEB 服务进行选择 and 编排,将编排后的结果作为一个新的功能服务提供给用户。

- 第三种场景是企业外部合作伙伴支持系统。

某企业有一个合作伙伴支持系统,运行方式如下:

企业将想要采购的产品列在网站上,这些产品希望由合作伙伴提供实现。合作伙伴都有自己产品信息的网站。合作伙伴浏览到产品信息后,将会提交建议书或报价单。目前所提交的建议书或报价单需要通过电子邮件、电话或会面的方式提交。而不是直接通过系统提交。

企业还有一个合作评估系统,对每个合作伙伴,公司都会在数据库中保留合作信息,当有产品需要采购时,企业都会根据以往的合作记录来进行评估。当产品采购完成时,公司会把合作结果保存到评估系统中。评估系统是一个由 C++ 语言编写的 CS 结构的程序。

这几个系统是相互独立的。如何能够把这几个系统整合成一个系统,使得企业既可以发布需求产品的信息,又可以查看合作伙伴所提供的产品信息及合作伙伴的评估信息,并且可以对它们进行操作,合作伙伴也可以通过这个系统直接提交报价。如果采购确认还需同时更新需求信息和合作伙伴的供给信息。

这个场景就必较复杂了,涉及企业间的应用集成,先通过服务包装,服务代理等模式将现有系统先转化为 WEB 服务,将相关企业的流程定义为合作伙伴服务,通过整合服务模式来整合,BPEL 可以支持合作伙伴服务,而这种复杂的企业间跨平台的集成通常需要 ESB 作为服务网关因此需要采用 ESB 模式[22]。

3.2 模块组成分析

通过上面的例子可以看到,不同的业务场景对 SOA 的使用层次需求不同,一些只需要 SOA 的某些基本功能,比如可能提供一些功能性的 WEB 服务或在这之上实现某些简单的业务流程的 WEB 服务(采用整合服务模式),还可能需要将一些遗留系统的

功能转换为 WEB 服务(采用包装服务或虚拟服务)。通常采用某种或几种底层实现模式组合实现的 WEB 服务就足以满足需求。

这种通过 soap/http 和 WSDL 访问 WEB 服务的方式搭建的系统实现起来相对容易,结构上是请求方和提供方或代理方点对点连接。但它的缺点是控制服务寻址和路由的功能被分散到客户端,系统没有统一控制点。而且当替换服务提供方实现时需对请求方或代理方改动代码,这使得请求和提供方被服务调用绑定在具体的协议和地址之上。随着业务逻辑的复杂,当系统服务的数量和相互间的交互增加到一定程度后将很难管理和维护。

因此,当企业希望 SOA 系统框架能够提供更强大、易于管理的底层功能以支持复杂的海量的交互操作时。我们还需要一个中间层,能够帮助实现在 SOA 架构中不同服务之间的智能化管理。由它充当整个 SOA 架构的中央管理器的作用,这个中间层就是 ESB。

ESB 克服了点对点连接的缺点,服务请求者不必再关心是谁提供的服务,服务采用的哪种协议,此外 ESB 可提供调停、路由、转换等支持功能。因此 ESB 是构建复杂的 SOA 系统必不可少的组成部分。

通过分析我们已基本明确目标平台的基本组成,对于复杂的企业应用需要 ESB 作为实现 SOA 系统的核心模块。此外 WEB 服务间的业务流程处理模块也是必不可少,当然还有 WEB 服务的实现模块。这几个模块将作为我们实现平台的基础。

3.3 本章小结

本章通过对业务场景接合 SOA 实现模式进行了分析,了解了企业对 SOA 实现平台的具体需求,并对目标实现平台的基本模块组成进行了分析。

第四章 平台组成

本章中将对平台的各个组成模块的实现产品进行选择 and 评估。下面首先介绍组件的调选标准。

4.1 组件选择标准

现在我们已明确平台将由一组开源中间件组成，包含 ESB 模块，WEB 服务间的业务流程处理模块，以及 WEB 服务的实现模块等。然而目前 SOA 的各种开源项目可谓多如牛毛，如何从中选择相应模块的开源组件呢？

我们主要从平台的目标用户进行考虑，要考虑的是我们的平台将提供给他们什么，他们所关注的又在哪些方面，这将决定我们对组件的选择。平台将面向两类用户，一类是开发人员，他们利用这个平台开发他们的 SOA 系统，因此他们关注的是系统开发的便捷，能否满足他们的技术需求，以及能否适应未来系统的扩展等方面。

另一类是系统管理员他们使用平台运行和管理服务。他们对系统的关注点是系统的性能、稳定性、可管理性等方面。详细标准如表 4-1

表 4-1 开发人员和系统管理员关注的基本标准对比

	开发人员	系统管理员
功能性		
性能		
可靠性		
扩展性		
便捷性		
操作性		
安全特色		
开源情况		

表 4-2 评比标准

评估标准	标准内容	评估目标
功能性	它可以做什么	设计文档
性能	是否可以高效工作	测试报告、用户回馈
可靠性	它是否可靠	测试报告、设计文档
扩展性	是否可以增加新功能	设计文档
便捷性	是否便于使用	开发指南、代码示例、IDE
操作性	是否容易操作	管理员手册
安全特色	是否提供安全服务	设计文档
开源情况	是否是活跃的开源项目，是否存在很多用户案例	项目文档

我们对组件选择的评比标准就是由这些方面构成。下面对具体模块的实现组件进行对比和评估。从而确定出我们 SOA 实现平台的具体构成。评估标准如表 4-2 所示。

4.2 ESB 组件

4.2.1 ESB 产品介绍

开源的 ESB 组件是我们平台的基础与核心，其它的组件或者嵌入 ESB 中，或者要和它进行整合。因此 ESB 组件的选择至关重要，一旦确定选择哪个组件后就可以将它作为基准组件，在选择其它模块组件时，它们与基准组件之间的兼容性就将成为一个非常重要的选择标准。因此我们精选了 4 种 ESB 相关产品，对它们作了全面的对比。

ServiceMix[23]是 LogicBlaze 提供的一个建立在 JBI (JSR 208)之上的开源 ESB 产品，目标是提供 SOA 及 EDA(Event Driven Architecture)。它的核心是一个完整的 JBI 容器，支持跨越分布式松耦合服务间的基于服务的 SOA 流程执行，以及基于事件驱动的处理。它现在已成为一个 Apache 的子项目遵循 Apache license。

Mule[24]是一个 ESB 消息框架，它是一个可升级、高效的分配代理，通过前沿的传输和消息机制可以无缝的处理服务和应用交互。

Glassfish[25]是 Sun Java System Application Server PE 9 之中的一个开源项目。这个项目是对发展中的基于 J2EE1.5 标准的高性能应用服务器平台提供架构处理。目标是成为 Sun 的下一代应用服务器。

OpenESB[26]也是一个 ESB 实现，它将基于 J2EE 1.4 且与 JSR 208 同步。OpenESB 提供与 JMS、JCA、JDBC、WEB 服务集成的能力。

4.2.2 对比

下面根据我们的组件选择标准对它们进行详细对比。如表 4-3 至 4-11 所示。

表 4-3 4 个 ESB 产品概要

ServiceMix	Mule	Glassfish	OpenESB
允许用户和卖方扩展，轻量级	是一个轻量级消息框架和性能分配代理	目标成为一个支持 JAVA 企业应用特性如 java 1.5, EJB, JMS, JTS 的应用服务器	将基于 J2EE 1.4.
提供完整的 SOA 实现，基于 JBI 标准.	提供 EDA 和 SOA. 支持 JBI 标准.	并不遵循 SOA 模式	基于 JBI 标准.
有一些文档	有一些文档	文档和资料很少	---

表 4-4 功能对比

		ServiceMix	Mule	Glassfish	OpenESB
架构	JBI Container	JSR 208	JSR 208		JSR 208
	Bindings	HTTP, JMS, JavaMail, WSIF, Jabber	HTTP, JMS, SOAP, SMTP, etc.	HTTP, JavaMail, JMS	JMS
	Services	WS-Notification, WS-Reliable Messaging, WS-Eventing, WS-ResourceProperties, WS-Distributed Management			---
	Components	ActiveMQ, ActiveSOAP, Mule, Groovy, Drools, PXE, Agila	PXE		---
Routing					---
BPEL					---
Cache					---
Scripting					---
Java Connector Architecture					

XSLT 转换				---
JAXP1.3 和 XMLSchema 或 RelaxNG 验证				---
Enterprise timer integration				
XSQL				
Axis support				---
POJO support				---
Xfire support				---
JBoss integration				---
Spring support				---
Geronimo integration				---
Workflow				

表 4-5 性能对比

	ServiceMix	Mule	Glassfish	OpenESB
Routing				---
Caching				---
Transforming				---
SOAP				---

表 4-6 可靠性对比

	ServiceMix	Mule	Glassfish	OpenESB
Cluster				---
Transaction				---

表 4-7 扩展性对比

	ServiceMix	Mule	Glassfish	OpenESB
Open architecture				---
Plugin support				---

表 4-8 便捷性对比

	ServiceMix	Mule	Glassfish	OpenESB
Eclipse plugin				---
Jboss deployer				---
WAR deployment				---
Maven JBI plugin				---
Maven SAR plugin				---
Developer's manual				---
Code examples				---

表 4-9 可操作性对比

	ServiceMix	Mule	Glassfish	OpenESB
Administrator's manual				---
Management GUI				---
Distributed management				---

表 4-10 安全性对比

	ServiceMix	Mule	Glassfish	OpenESB
Authentication				---
Authorization				---
Confidentiality				---
Anti-tamper				---
Non-repudiation				---
DoS				---

表 4-11 开源状态

	ServiceMix	Mule	Glassfish	OpenESB
Released	Yes	Yes	No	No
Recent version	V2.02	V1.1.1	Build 19	---
Team size	25	25	Over 100	---
Use cases				---

通过对比从各方面综合考虑最终确定选择 ServiceMix 作为 ESB 组件，由于 ESB 组件 SOA 平台的重要性，所以我们又对 ServiceMix 进行了进一步的评估。

4.2.3 评估

在选择 ServiceMix 作为我们平台的核心组件之后，我们又对 ServiceMix 做了详细的评估。

首先我们下载了 ServiceMix 的源码，并在 Eclipse 环境中进行编译和调试，以便我们能够进一步了解 ServiceMix 的内部结构。

接着我们研究了 ServiceMix 自带的几个示例程序，来进一步理解 ESB 及 JBI 容器的内部机制，以及在这个容器中如何实现 SOA。这几个示例分别是展示 ServiceMix 如何对文档操作的 File Binding Example，展示 ServiceMix 如何控制 Http 请求与 WEB 服务交互的 Http Binding Example，以及展示 ServiceMix 如何协调 Http, JCA(J2EE Connector architecture)和 WEB 服务的 Basic Example。研究的细节工作包括分析它们的类图、时序图、源码等，并在 Eclipse 环境中调试运行。

此后我们又对这些例子进行了扩展，进一步了解 ServiceMix 的扩展机制和部署机制。

通过对 ServiceMix 的进一步研究和调查，以及具体示例的扩展工作，加深了对 ServiceMix 的理解。也更加让我们坚定了选择 ServiceMix 的信心。明确了 ServiceMix 作为 SOA 平台核心的位置。

4.3 WEB服务组件

WEB 服务组件我们主要考虑两方面需求，一方面是提供 WEB 服务支持的 WEB 服务引擎，另一方面是开发 WEB 服务的工具。

4.3.1 WEB 服务引擎

因为我们项目的原则是基于开源技术、面向 JAVA，J2EE 领域为主的企业应用，所以我们很自然的选择了 Apache 的 Axis[27]作为 WEB 服务引擎。Axis 是基于 JAVA 语言 SOAP 规范(SOAP 1.2)和 SOAP with Attachments 规范(来自 Apache Group)的开放源代码实现。它本质上就是一个 SOAP 引擎，通过 Axis 可以很容易的将 JAVA 代码转化为 WEB 服务，并发布到 WEB 容器中。

4.3.2 WEB 服务开发工具

WEB 服务的开发工具是为了辅助开发人员方便快捷的开发 WEB 服务，我们的选择

标准如下：

首先要支持 JAVA，即可以轻松的将 JAVA 代码转换为 WSDL 文件，又可从 WSDL 文件转到 JAVA 代码。其次可以自动产生已存在 WEB 服务的客户端代码。并且还应支持客户端和服务器的调试。此外还要具有良好的兼容性，可以和主流的 IDE 轻松整合。以及提供图形化的 WSDL 生成及编辑界面等。

基于以上要求我们主要考察了来自 Systinet 的 Wasp(web application and service platform) [28]和 Eclipse 的子项目 WTP(web tools platform) [29]两个产品。

表 4-12 WEB 服务开发组件比较

	Eclipse 插件	易用性	示例和文档	Tomcat 支持
Wasp developer	Yes	Yes	Yes	No
WTP	Yes	Yes	Yes	Yes

如表 4-12 所示。这两个项目都对 WEB 服务开发作了支持，而且都算是比较成功的项目，有很多的用户。从它们与 Tomcat 的兼容性，以及可扩展性考虑，我们最终选择了 WTP 项目。

4.4 BPEL 组件

BPEL 组件的选择我们主要从两方面考虑，一方面是 BPEL 引擎，另一方面是 BPEL 的图形化建模工具。

4.4.1 BPEL 引擎

BPEL 引擎的作用是解释执行用户定义好的业务流程。我们主要从对 BPEL 标准的支持程度，同 ESB 容器的兼容性，开源项目状态及支持程度等几方面考虑，着重考察了 PXE[30]、Twister[31]和 ActiveBPEL LLC 提供的 ActiveBPEL[32]三个开源项目。

表 4-13 BPEL 引擎比较

Standard	PXE	Twister	ActiveBPEL
Service Describe(fun)	WSDL	WSDL	WSDL
Service Describe(non fun)	None	None	None
License	CPL,MIT	LGPL	GNU
Support	short	short	more
OS	Windows XP, Linux, MacOS X, Solaris, and AIX.	Windows XP, Linux, MacOS X, Solaris, and AIX.	Windows XP, Linux, MacOS X, Solaris, and AIX.
JVM	1.5	1.4 or 1.5	1.4 or 1.5
Database	Hibernate HypersonicSQL	Hibernate HSQL-DB Xindice	Tamino a persistence manager
Container	standalone, Tomcat... any	Tomcat... any	Tomcat... any
Standard	WS-BPEL2.0, 1.1	WS-BPEL1.1	WS-BPEL1.1
Service Discover and Public	None (support by ServiceMix)	None	wsdlCatalog.xml
Service Select	None	None	None
Released VersionDatabase	Version 1.41 (Currently all BPEL activities are supported)Hibernate HypersonicSQL	Version 0.3 (flow, Compensation, scopes not supported)Hibernate HSQL-DB Xindice	ActiveBPEL 2.0 (variables, message property aliases, Suspend....)Tamino a persistence manager
Service Composition	Service Provider, Orchestration	Engine, Orchestration	Engine, Orchestration
Released Version	Version 1.41 (Currently all BPEL activities are supported)	Version 0.3 (flow, Compensation, scopes not supported)	ActiveBPEL 2.0 (variables, message property aliases, Suspend....)
Graphic Model	None	None	None

具体比较如表 4-13，下面对比较结果进行了概括。

- PXE：支持 ServiceMix、支持所有 BPEL 活动，支持 BPEL4WS1.1 和 BPEL2.0、复杂、有源码和许可、不直接支持 WS 寻址、没有图形化建模工具不过支持其他工具生成的流程。
- Twister：第一个开源 BPEL 引擎，支持用户交互、角色和组，类似一个工作流引擎，目前只支持 BPEL 基础活动，SOA 架构，没有图形化工具，有源码和许可，需要同 ServiceMix 整合。
- ActiveBPEL：架构类似于 twister，有一个可插入的表达式语言框架，支持附加的 WS 规范，只支持一部分 BPEL4WS1.1 活动，没有图形化工具，有源码和许可，需要同 ServiceMix 整合。

基于我们的调查和评估，特别考虑到规范的支持度，灵活性以及同 ESB 组件的兼容性，我们最终的意向是采用 PXE 作为 BPEL 的引擎组件。

4.4.2 BPEL 建模工具

BPEL 建模工具方面我们主要考虑的是图形化定义、编辑、部署、监控和测试 BPEL 流程的工具，目的是用来辅助用户定义 BPEL 业务流程。此外还应当提供 BPEL 流程的监控。它既可以是 IDE 插件的形式，也可以是单独的应用程序。调查的产品有：IBM 的 BPEL4J[33]，Cape Clear Orchestrator[34]，ActiveWebflow Designer[35]，Oracle BPEL Design[36]以及 Java studio Enterprise TPR-orchestration[37]。

通过对BPEL建模工具的调查，发现目前还没有满足我们需求的开源项目。

BPEL4J项目包含一个BPEL流程的运行平台和验证BPEL4WS文档的工具，此外还包括一个Eclipse的插件提供简单的图形化编辑器来创建和编辑BPEL文件。但这个插件不是开源产品只可作为评估版试用，而且已经很久没有更新只支持Eclipse2.x。

Sun的Java studio Enterprise TPR-orchestration虽然对PXE有所支持，但它只能作为SUN的java studio enterprise technology preview release的一个模块运行，与我们以Eclipse作为开发平台的想法违背。而其他项目最多只支持对BPEL流程文件进行创建、编辑而且还会有license的限制。特别是它们都缺乏流程监控能力，而这一点是我们认为在业务流程处理中非常重要的，因此我们决定先选择Eclipse的WTP项目中的XML编辑器作为BPEL建模工具的替代。目前先给用户简单的XML编辑功能，以后在Eclipse WTP项目的基础上自主开发BPEL建模工具。

4.5 J2EE容器

J2EE 服务器是 SOA 实现平台的可选模块,用户可将开发好的 SOA 系统部署到 J2EE 服务器中,以谋求更强大的性能和更稳固的系统支持。ServiceMix 声称支持 Jboss 和 Apache Geronimo[38]等多个 J2EE 应用服务器的集成。然而 Servicemix 本身作为 Geronimo 的 1 个子项目,毫无疑问有更好的兼容性。在 Servicemix 上开发的 JBI 组件可以直接部署到 Geronimo 之上。而且最近 IBM 刚刚收购了 Geronimo,在它基础上建立了 Websphere Application Server Community Edition。使其成为 Websphere 体系的一个开发和低端应用的开源环境,它和 Websphere Application Server 之间将可以容易的迁移。这使得 Geronimo 项目的前景比较明朗,因此我们选择了 Geronimo 作为 J2EE 应用服务器。

4.6 消息组件

ServiceMix内置了ActiveMQ[39]来提供基于JMS1.1标准的消息服务,而且ActiveMQ可以无缝的整合到J2EE 1.4 容器,轻量级容器及JAVA程序中。

4.7 IDE

我们的 IDE(integrated development environment)选择了以 Eclipse[40]为基础。因为 Eclipse 是目前最成功的开源 IDE。Eclipse 是一个开放源代码的、基于 Java 的可扩展开发平台。本质是提供了一个可移植、可扩展、开放源代码的框架,用于控制附加的插件组件,由它们共同构建开发环境。Eclipse 本身提供了插件开发环境(Plug-in Development Environment, PDE),通过它软件开发人员可以方便的构建与 Eclipse 环境集成的组件。

由于 Eclipse 良好的扩展性,我们选择它作为 SOA 实现平台的开发环境,并开发了一组相应插件来辅助开发 SOA 应用。

4.8 组件整合

通过对组件的选择和评估,我们已经明确了系统的构成成分。我们的平台应该包括一个 ESB 组件 ServiceMix、业务流程处理模块 PXE、WEB 服务的引擎 Axis、WEB 服务和 BPEL 的开发工具 WTP、J2EE 应用服务器 Geronimo、Web 容器 Tomcat、消息组件 ActiveMQ、以及开发环境 Eclipse。这些组件是我们的 SOA 实现平台的基础部分,其他

组件可根据需求进行扩展。

下一步的工作就是把这些组件整合起来形成一个 SOA 的实现环境。进而可以在这个平台上开发和部署 SOA 系统。

我们首先做的整合工作是将 ServiceMix 与 WEB 容器进行整合。ServiceMix 默认的运行方式是 ServiceMix 自己作为一个容器,WEB 容器 Jetty 内嵌在一个部署在 ServiceMix 中的组件里。当 ServiceMix 启动时会实例化这个组件,通过这个组件启动 WEB 容器。这种运行方式不太普遍,而且 Jetty 作为 WEB 容器功能太过简单。因此我们把 ServiceMix 和 Tomcat 做了整合,因为 ServiceMix 是建立在 Spring 技术之上,所以我们将 ServiceMix 作为普通的 WEB 程序嵌入 Tomcat 中。这样当 Tomcat 启动后会自动加载 ServiceMix 组件。

在完成 WEB 容器整合后,我们开始考虑 ServiceMix 与 J2EE 应用服务器 Geronimo 进行整合。由于 ServiceMix 和 Geronimo 之间的关联关系使得它们之间的整合变得很简单。ServiceMix 的组件可以直接部署到 Geronimo 中。而 ServiceMix 容器本身我们首先将其与 Tomcat 整合后生成 WAR(web application archive)文件,然后将这个 WAR 文件部署到 Geronimo,由 Geronimo 内嵌的 Tomcat 或 Jetty 来运行。这样的整合方式是考虑与其它 J2EE 服务器整合的一致性。

ServiceMix 与 PXE 的整合是给我们带来困难最多的部分。现在看起来原因其实很简单, PXE 的运行方式分为两种,一种是单独作为 BPEL 容器运行,另一种是嵌入到 JBI 容器中运行。当 PXE 嵌入 ServiceMix 中运行时需要有一个 PXE-Install 的 JAR 文件来将 PXE 引擎进行部署。而 PXE 是一个非常活跃的项目,几乎每一两周都会更新。很难找到一个稳定的版本可以和 ServiceMix 匹配起来。此外 PXE 开放的源码也存在表结构与代码不匹配的问题,最终不得不自己重建数据库的结构。不过这些都是开源项目经常会碰到的问题,通过对数据库的重建,我们也将 PXE 本身采用的内嵌 HSQL 数据库替换为性能更强的 PostgreSQL。

ServiceMix 与 ActiveMQ 以及持久化模块的整合就非常简单了,只需修改 ServiceMix 提供的配置文件即可。

4.9 本章小结

本章介绍了我们构建平台的基础工作。根据 SOA 所需的组成模块,对相应的开源产品进行了选择和评估。最终确定了平台的各个组成部分,并对它们进行了整合。

第五章 实现平台介绍

通过对平台组件的选择和评估以及组件之间的整合工作，实现平台的框架已经基本形成。这一章中将对平台的架构、扩展模式及开发框架进行介绍。

5.1 实现平台架构

我们提供的平台目的主要在于提供一个开发和运行 SOA 系统的环境。用户可通过扩展自己业务相关的组件构建起自己的 SOA 系统。

下面具体介绍一下 SOA 实现平台的架构。该平台是由一组开源软件组合而成，以一个 ESB 容器 ServiceMix 为核心。下面图 5-1 是该平台的结构示意图。

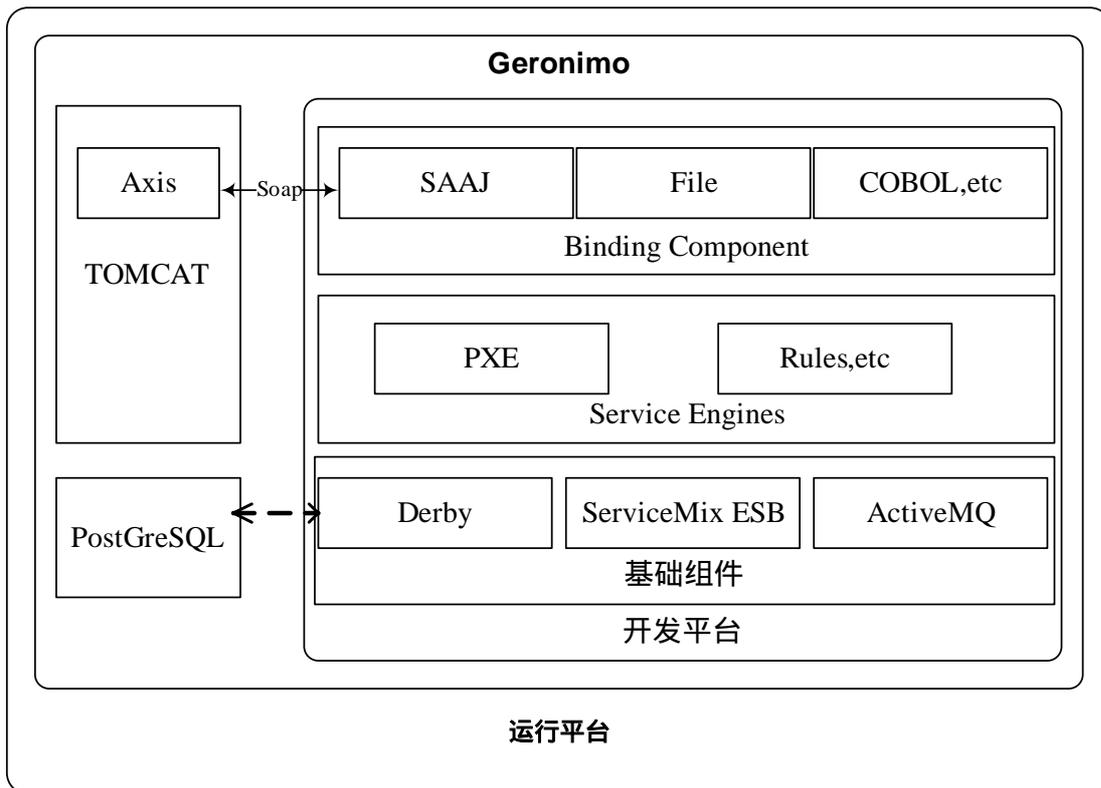


图 5-1 : SOA 开源平台的结构示意图

Fig 5-1 Open Source SOA Platform

平台结构上分为三个层次，底层主要是由 ESB 容器 ServiceMix，消息处理的组件 ActiveMQ，以及负责持久化的 Derby 数据库组成，提供路由、服务部署、管理、事件驱动等底层支持。

中间层是部署在 ServiceMix 上的 SE 组件，它们主要负责业务流程处理、数据转换以及业务逻辑等。

最上层是提供通信协议转换负责同外界的服务进行交互的 BC 组件。

这些组件以 ServiceMix 作为容器组合在一起形成一个基本的 SOA 运行环境。但这个运行环境并不等于我们所要实现的 SOA 实现平台。我们的目标是提供给用户一个既可以辅助开发又可以作为部署和运行 SOA 系统的平台。因此在这个运行环境基础之上我们又分别从便于用户开发的角度及基于性能和负载能力的角度作了相应扩展，最终形成的实现平台包括两部分，分别是运行平台和开发平台。

5.2 开发平台

5.2.1 开发平台介绍

开发平台是在运行环境的基础上添加开发环境 Eclipse，以及一些基于 Eclipse 的图形化插件工具如组件部署、BPEL 流程建模等。在这个开发平台上我们还将提供一些扩展的 ServiceMix 组件来辅助用户构建 SOA 系统。

开发平台主要是辅助软件开发人员在 SOA 实现平台基础上开发 SOA 系统。基于我们对 SOA 系统的各种实际案例的分析结合对 SOA 实现模式的研究，最终决定以 ESB 模式为基础，围绕 ServiceMix 针对不同需求构建起了一组标准开发框架，并通过 Eclipse 插件的方式提供给开发人员。对于通常的 SOA 需求开发人员只需在框架之上扩展自己的业务功能模块就可以实现。

目前的开发框架有三种：

标准框架：以 ServiceMix 作为核心，在此基础上通过扩展 SE 组件和 BC 实现 SOA 系统的 ESB 实现模式。这套框架主要应用于几个不同类型的服务通过 ServiceMix 相互集成。如 WEB 服务和 Http 协议、JMS 协议的程序之间集成。

业务流程框架：是在标准框架基础之上扩展了业务流程处理，从实现上说也就是 ESB 模式加 WEB 复合模式。这套框架可以处理多个 WEB 服务根据业务流程相互组合的需求，并且也可以通过 ServiceMix 和其他类型协议的服务集成。

WEB 整合框架：对遗留的 WEB 系统页面进行整合的框架。也是在标准框架基础上

进行了扩展，具体内容将在下一章中详细介绍。

目前 Eclipse 插件方面我们主要针对 WEB 服务整合、业务流程处理及 WEB 整合方面做了一组相应插件，包括项目模版、编辑和部署工具等。

WEB 服务整合采用了标准框架，主要针对现有 WEB 服务，通过获取 WSDL 文件自动将服务整合到系统中。业务流程处理是基于业务流程框架，通过 BPEL 引擎 PXE 将现有的 WEB 服务进行编排，提供业务流程处理。我们将这两个框架作为模版程序，嵌入到我们的 Eclipse 插件中，开发人员通过 Eclipse 插件的项目模版就可以创建框架程序，在创建项目的过程中还可以对框架进行初始化设置。

除此之外我们还对 WEB 服务和 BPEL 开发做了一些扩展支持。如可将现有的 WEB 服务自动部署到 ServiceMix 中，并生成相应 BC 以及 Client 组件。这样可以将现有的 WEB 服务方便的集成到系统中。对于 BPEL 开发，我们在 PXE 的基础上提供了一些流程描述文件的模版，通过 WTP 插件可对这些描述文件进行编辑，并提供了编译和部署的功能。开发人员只需完成对 BPEL 业务流程文件的定义，Eclipse 插件可将 BPEL 流程自动编译并部署到 ServiceMix 中，完成对业务流程的集成。插件还提供了对流程的测试功能。

5.2.2 开发流程

如何利用我们的平台开发 SOA 系统呢？通常构建 SOA 系统很少是从头开始设计实现每一个服务，而是首先将企业现有业务系统的功能转化为服务，再将它们整合到 SOA 系统中。因此利用我们的平台构建 SOA 系统的流程可以概括为 3 个部分。

首先是服务的创建过程，是指根据业务需求将企业现有系统的功能封装为服务或者基于现有的开发方式创建出新的服务。

其次是服务的连通过程，是指通过标准或用户自扩展的 JBI 组件，使来自于不同应用的服务以松耦合的方式通过 ServiceMix 内嵌的消息路由器进行交互，也就是将这些来自于不同协议、位置和实现技术的服务通过 ServiceMix 和 JBI 组件整合在一起。只要服务的接口描述不发生变化，服务具体实现的改变不会对系统造成影响。ServiceMix 提供了一些通用的组件，用户也可以基于 JBI 规范自定义针对某些领域的组件。

最后是服务的组合过程，是指将已部署在系统中的服务根据业务逻辑进行组合来创建新的服务或者系统，也就是创建业务流程。当业务逻辑发生变化时，常常只需调整服务组合的方式既可以满足需求。

通过这种开发模式可以重用粗颗粒的服务，因而可以减少系统投入和实现时间。

5.2.3 开发框架

从根本上说,我们的平台是基于 JBI 规范的灵活可插拔的架构,由核心的 ServiceMix 容器和一些符合 JBI 规范的组件组成。组件间通过容器的 NMR 消息路由器进行通信。从架构上分析组件是通过一个基于 WSDL2.0 的抽象服务模式进行交互,具体来说就是组件间以一种标准化的消息交互方式和消息格式进行通信。

服务调用的具体过程是服务请求者发送一个标准格式的消息给 NMR,NMR 把消息请求转送给服务提供者。如果服务提供者需要返回响应信息给请求方,他也会发一个标准格式的消息给 NMR,由 NMR 转回请求方,如果返回的是一个错误信息,它也是通过 NMR 来转发。

企业整合最常见的需求就是基于不同协议和消息格式的业务功能之间的交互。比如两个现存系统 A,B。A 作为服务请求者它产生 XML 格式的请求消息,只支持特定的传输协议 SOAP,而 B 作为服务的提供者它的消息格式是文本格式,传输协议为生成一个文件。为了让它们能够进行交互通常需要利用适配器将其中一个系统的消息格式和通信协议转化。

这样的话如果需要整合的系统使用的消息格式和传输协议比较多,相互的逻辑关系又很复杂,将会使整合工作变得非常庞大和复杂。

而在我们的平台下通过两个分别支持 File,SOAP 协议的 BC 组件就可以实现适配器的工作,它们将不同格式的消息转换为标准的消息格式,并由 NMR 负责传送的对应的组件进行处理。而处理工作,可通过 SE 组件来实现。

5.2.4 架构的特点

我们的 SOA 实现平台架构实质上是面向企业应用整合领域提供 SOI 解决方案。用户可以利用平台提供的框架进行 EAI 系统开发。与传统的企业应用整合技术相比我们的平台架构存在什么优势呢?这要从企业应用整合技术的发展过程说起。

针对于企业应用整合领域的实现技术经历了长时间的发展演化,目前有多种方案并存。最为简单和原始的方法是通过程序接口、数据库、共享文件、端口及特定传输协议等来进行应用程序间的数据交换和功能调用。这种情况下应用之间大多是通过点对点的连接来进行整合,接口通常是硬编码的,所使用的协议也是非标准的,因此功能的提供者 and 使用者之间是紧耦合的,重用性和扩展性比较差,而且随着需要整合的节点增多,节点逻辑的复杂度增加,系统实现的困难度和复杂度会成倍增长。

为了解决复杂的企业集成需求,业界又提出了一些基于分布式技术和消息中间件的

集成方案。利用消息中间件来提供消息的传输、转换、路由和分发等，从而实现分布式，跨平台的程序交互及数据交换。这一类集成方式的实现机制是通过一个类似于 HUB 的集成基础设施来简化应用之间连接的拓扑结构，利用它来实现不同格式数据和功能调用方式之间的转换。

这种基于消息中间件的集成方式缺点是没有统一的标准，因此，各个厂商都有各自不同的 EAI 解决方案，存在着各种各样的中间件平台。当异构平台之间进行集成时必须花费大量精力在各个不同的中间件的协调上，因此这种解决方案通常都比较笨重。此外这种集成方式都是以一个 HUB 组件为中心的，利用 HUB 进行信息交换和控制，是一种星形的拓扑结构，所有的请求都需要经过中央 HUB 进行处理和转发，因此 HUB 组件的责任过重，不但系统的性能会随着连接者的增多而愈来愈差，而且一旦 HUB 出错，整个系统都会可能崩溃。

而我们的平台所提供的是一种以服务为中心的集成架构，不同应用系统可以在一个标准、可靠、安全、可管理的环境下，以松散耦合的方式相互交互，根据业务需求动态地进行企业应用集成，提供高度灵活的应变能力和重用能力。从拓扑结构上看，我们的平台是一种以 ESB 为核心的总线结构。部署在 ESB 上的服务可以平等的进行通信。不同格式的数据和消息首先通过部署在 ESB 上的组件转化为标准的消息格式，然后通过 NMR 负责传送。因此它比 HUB 的方式更开放，有更好的性能和扩展性。

此外，这种以服务为中心的集成框架可发挥服务的重用性，从而实现渐进式集成。企业实现 SOA 系统时可首先将现有系统的相关功能转化为服务来进行集成，随着项目的进行，可重用的服务越来越多，最终绝大多数新的集成需求将可以通过已有的服务来完成。而且由于服务的灵活性，即使现有服务的实现已发生转变也不会影响到依赖这个服务的那些应用，从而保证了集成对业务逻辑变化的适应能力。

同以往的 EAI 集成方式相比较，我们的平台架构克服了以往集成方式规模投入比较大、周期长、风险高、技术壁垒等缺点，因此利用我们的平台进行企业应用集成是一种渐进、灵活的集成方式。

5.2.5 组件开发

在我们的平台之上开发 SOA 系统最重要的部分就是如何开发和利用这些组件。如何具体应用将在以后章节中通过具体实例详细介绍，这里首先解释一下组件的开发模式

ServiceMix 支持的组件结构上分为两种一种是标准的符合 JBI 规范的组件如 PXE 等，

另外一种轻量级组件如 SAAJ 等。

轻量级组件相当于单个 JBI 端点，不需要通过服务单元的部署，而直接通过 ServiceMix 的配置文件进行配置。它们通常被用来实现消息格式转化，作为 BC 组件来使用。开发它们的过程相对简单，只要继承 ComponentSupport 类，并根据需求通过一个辅助类来实现现有的消息格式和标准化的消息格式之间的转化，然后将转化后的消息发送给 NMR 即可。消息的路由可通过配置文件来描述。

而创建一个标准 JBI 组件的过程就复杂得多，由三个阶段组成：组件代码的开发，组件的安装,部署服务定义[20]。

开发阶段就是传统的 java 代码开发过程，实现特定的接口，创建 java 类等。最终实现组件的核心功能。对于服务引擎组件来说就是创建一个特定类型服务定义的容器。

安装阶段需要一个 xml 格式的组件结构描述，它包含组件的 id 和很多组件内部的属性描述。将这个 xml 描述文件和编译过的组件类组成一个 JAR (java application archive) 格式的压缩包，放到 ServiceMix 对应的组件安装目录即可。

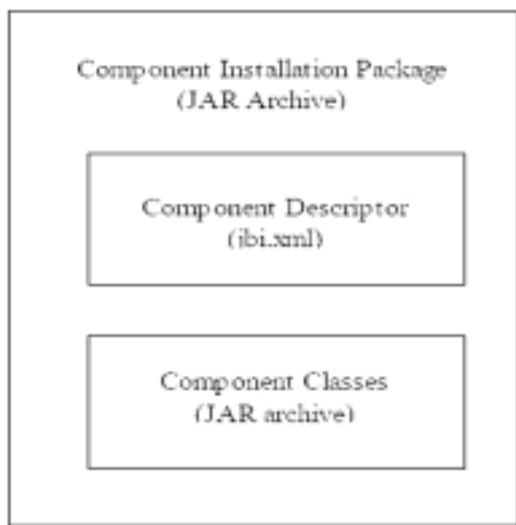


图 5-2：组件安装包结构

Fig 5-2 Component Installation Package structure

服务部署的阶段，是通过已安装的组件部署一个具体的服务定义。它需要创建一个服务单元，它是以 JAR 文件的格式，里面包含具体的服务实现和用来将服务单元部署到组件中的服务描述信息以及对应的组件信息。而服务单元还要被包含在服务的装配包中，服务的装配包中包含描述服务实现结构的装配信息以及 JAR 格式的服务单元，最终在 ServiceMix 环境中将服务装配包的内容部署到组件中。部署过程也可通过将服务装配包放到 ServiceMix 对应的服务部署目录，由 ServiceMix 自动部署。



图 5-3：服务部署包结构

Fig 5-3 Service Assembly package structure

轻量级组件也可以打包成为服务装配包，由 ServiceMix 提供的 lw-container 组件进行部署运行。

5.3 运行平台

ServiceMix 容器可以有 3 种运行模式，分别是独立运行模式、嵌入模式及 WAR 模式。最轻量级的独立运行模式优点是可以快速的启动或停止容器，这种模式下 JBI 组件和服务可以动态部署及运行，便于开发时对组件进行调试。嵌入模式是最普遍的应用模式，将 ServiceMix 容器和相应的 JBI 组件嵌入到应用程序中。使得应用程序可以提供基于这些 JBI 组件的功能。WAR 模式是将 ServiceMix 容器象一个 Web 应用程序那样嵌入到 J2EE 应用服务器中。

从性能、管理、稳定性及负载能力等多方面考虑我们决定以 WAR 模式作为 SOA 运行平台的架构。ServiceMix 可以支持 JBoss、Geronimo 等多种 J2EE 应用服务器。相对而言由于 ServiceMix 本身作为 Geronimo 项目的子项目，Geronimo 对 ServiceMix 有更好的支持。所以我们选择将运行环境部署到 Geronimo 中构成一个性能更为强大的 SOA 的运行平台，同时为了满足更高的持久化要求所以将 Derby 替换为的 PostgreSQL 数据

库，此外运行平台还包括 WEB 容器 Tomcat 及提供 WEB 服务的 Axis。

5.4 本章小结

本章首先阐释了 SOA 实现平台的架构。然后从面向开发人员的角度详细介绍了开发平台的特点，开发框架和开发流程以及组件的开发模式。最后对运行平台也进行了介绍。

第六章平台扩展

本章主要是介绍如何针对具体领域对 SOA 的开发平台进行扩展。也就是对标准开发框架进行的扩展工作，这里将通过 WEB 整合框架来进行介绍

6.1 WEB整合项目介绍

WEB 整合项目是对 SOA 实现平台的一个扩展示例。它针对于企业遗留 WEB 系统的整合。企业系统整合的途径可以分为三种分别是数据层、应用层及表现层整合。如图 6-1

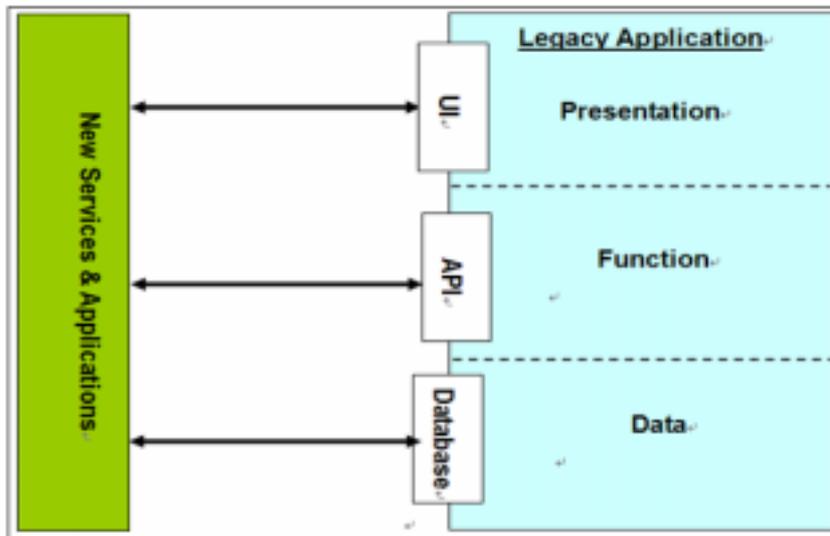


图 6-1：系统整合层次

Fig 6-1 System integration layer

通常业界的系统整合思路主要集中在对系统的数据层和应用层进行整合。早期的 EAI 系统大多关注于数据层的整合。而目前随着 WEB 服务及 BPM 的快速发展业界更关注于应用层的整合。

而 WEB 整合项目的目标定位在表现层整合，也就是对浏览器中显示的页面中的信息进行整合。其本质是通过对浏览器中所显示的 Html 文件进行分析，抽象出原有系统

的业务逻辑和数据结构，最后通过对它们进行处理从而实现整合目标，这样使得整合的工作脱离原有系统的代码和数据的约束，上升到了 UI(user interface)的层次。如图 6-2 所示

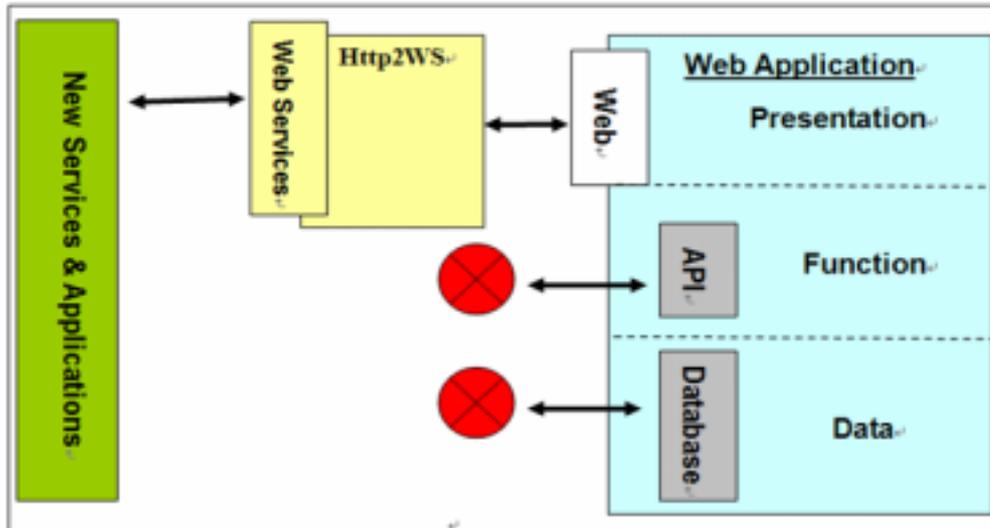


图 6-2：表现层整合

Fig 6-2 Presentation layer integration

为什么要进行表现层整合呢？目前很多企业内的遗留系统随着人员和技术的变更，已经很难维护和更新了。然而这些系统还具备一定的功能，依然担当某些职责。直接替换掉它们无疑是一笔浪费，特别是这些遗留系统中保存的某些珍贵的历史性数据由于格式的原因可能已无法导出。可是随着业务需求的改变，扩展和整合这些系统无疑又是迫在眉睫。这些系统的源码可能已很难了解，或许根本就不存在了，我们所能得到的信息只有浏览器中显示的 WEB 页面。

WEB 整合的目的就是通过对遗留系统的业务功能进行分析，抽象出一些相应功能。然后根据某一功能对浏览器中显示的相应页面信息进行解析，提取出有用的数据信息及一组对应操作信息，然后通过我们的系统模拟这些 HTTP 操作，访问 WEB 服务器。进而实现功能。最终将这个功能转化为 WEB 服务。这样不管系统是由什么语言开发的，系统的具体结构是什么样的，也不管系统的页面是 HTML, ASP, PHP, 还是 SERVLET, JSP 创建的，只要系统通过 WEB 浏览器和用户交互，并生成 HTML 页面，都可以通过我们的系统进行功能封装，转化为 WEB 服务，最终利用我们的实现平台将它们与其它系统集成。

6.2 相关产品和技術

6.2.1 Kapow

根据我们的目标我们对目前市场上的相关技术和产品进行了调查。目前市场上类似的产品非常少,我们只找到一家美国的 Kapow 公司的 RoboSuit 产品有类似的从浏览器而不是程序层次进行整合的思想。RoboSuit 可以从浏览器获取信息,并且可以同 WEB 服务器进行交互。目前 RoboSuit 已经在全世界 100 多个企业中应用。目前 RoboSuit 宣称也提供 WEB 服务支持。但是因为是商业产品具体资料很少,RoboSuit 开发和运行在 Kapow 自己的环境中,而不是在 SOA 的中间件中。

6.2.2 HttpClient

通过我们的调查 Http Client(Apache 基金会的 1 个子项目)[41]看起来对我们的工作非常有帮助。

Http Client 是一种类似客户端或浏览器同 WEB 服务器交互的技术。尽管 java.net 包中提供了通过 Http 协议访问 WEB 服务器的基本功能,但相对比较简单。而 Http Client 组件提供了高效,不断更新、功能强大的工具包实现了最新的 HTTP 标准和建议的功能。

Http Client 除了支持基本的 HTTP 协议,还提供了良好的扩展性,通过对它扩展的项目几乎涉及到任何 HTTP 相关的客户端应用系统,不论是 WEB 浏览器,WEB 服务的客户端,甚至是扩展 HTTP 协议到分布式通讯领域。

我们首先以企业内部的遗留系统为基础,用 Http Client 尝试开发了一些示例原型,包括取得网页信息,以及提交信息等。通过这些示例得出结论:

可以通过 Http Client 访问 WEB 服务器并取回相应信息,通过 Http Client 也可以和 WEB 服务器交互。

然而也遇到了一些问题,首先是 Session 的问题。当用户访问 WEB 程序时,WEB 服务器会为每次连接创建一个 Session,在页面间切换时通常会有一些信息要保存到 Session 中,最为常见的就是用户的授权信息通常会保存在 Session 中。而 Http Client 没有提供 Session 处理的功能。

除此之外 Http Client 是一个比较底层的工具对 WEB 服务器访问只能得到响应的 HTML 页面。为了取得 HTML 页面中的具体元素我们必须将 HTML 的元素进行建模。

这将是一个非常复杂的工作。

6.2.2 HttpUnit

通过对Http Client的研究,我们考虑到HTML元素建模的重要性,因此我们又调查了另一种开源工具Http Unit[42]。

Http Unit最初是一个Java编写的自动测试WEB程序的框架,它仿效了浏览器行为的相关部分包括表单提交,JavaScript,基础认证,cookies,页面自动重定向等。

除此之外还对HTML元素进行了建模,可以具体获得表单,表格,连接的内容。而且Http Unit可以轻松解决Session的问题。

我们在Http Unit基础上也构造了原型系统。对两个相关联的企业内部遗留系统进行了整合。这两个系统为会议室安排系统和个人时间安排系统。通过整合提供一个新系统可同时对两个遗留系统的数据进行处理。

6.3 WEB整合框架

虽然通过试验证明在Http Unit上可以整合WEB系统,可实际中的问题还很多,例如要求开发人员必须很了解HttpUnit,并且熟悉HTML语言,此外开发中对多个系统同时操作还会遇到线程及事务问题。基于以上考虑我们决定提供一个通用开发框架来解决WEB整合的问题。框架基于HttpClient和HttpUnit等开源项目,由五个模块构成:

1. Captor

Captor模块本身相当于一个Http代理服务器,捕获浏览器发出的Http请求和WEB服务器返回的相应信息,记录用户的操作流程。使用Captor首先要设定浏览器通过Captor指定的Http代理服务器来访问网络,开始捕获操作时Captor会监听代理服务器的端口,

当接受到浏览器发送的请求后Captor会记录下请求信息并将请求转发给远方的WEB服务器,同样当相应信息返回后Captor也会先进行记录然后转发给浏览器。这样用户通过浏览器对WEB服务器的所有操作和得到的结果都被Captor所记录。

2. Modeller

Modeller模块的作用是根据Captor捕获的信息和用户定义的信息对流程进行建模。用户可以将需要替换的输入域,如Url的参数,表单提交的域等定义为变量,也可以定义希望得到的页面输出域变量。定义过程中所需的页面信息是通过浏览器中显示的页

面解析后自动捕获的。Modeller最终定义的输入和输出信息是遵循业务逻辑对所定义的域变量的封装。

3. Engine

根据Modeller生成的建模文件运行流程。Engine模块模拟浏览器对远程WEB服务器进行操作。运行时用户可根据Modeller中定义的输入信息格式进行输入，Engine模块可自动对建模时定义的域变量进行添值。

4. Wrap

Wrap 模块的作用是将流程和 Engine 转换为服务并进行部署。用户只需要调用服务就可以得到所期望的 WEB 系统的信息或实现对 WEB 系统所期望的操作。

Wrap 的转换可分为两种一种是为了通用性和便于用户的访问考虑将操作流程转换为 WEB 服务，部署到 Axis 服务器中，可通过 ServiceMix 提供的标准 BC SAAJ 进行整合。

另一种是自定义一个标准的 JBI 组件将服务直接部署到 ServiceMix 中。

5. BPEL

根据业务逻辑对 WEB 系统转化的服务进行编排，也就是将遗留 WEB 系统的功能进行整合。可通过 SOA 实现平台中包含的 PXE 组件来实现业务流程处理。

框架的开发思路是首先对目标系统进行分析，根据需求划分成若干功能，如查询时间，添加会议等。每个功能则对应了一组页面操作。对于某个具体的功能，如会议室的安排查询，对应了实际WEB系统的三个页面操作的组合，如图6-3所示。

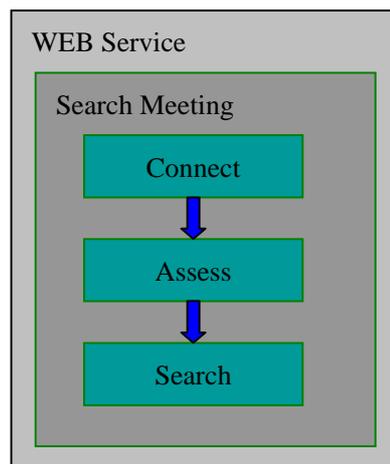


图 6-3 WEB 操作流程

Fig 6-3 WEB operate process

Connect 操作对应的是浏览器通过 URL 访问 WEB 系统的操作。

Assess 操作对应了用户通过输入用户名，密码进行 Basic 认证的操作。

Search 操作对应了用户输入查询条件进行会议室系统查询的操作。

然后通过浏览器对这些功能进行访问，根据Captor捕获的信息分别进行建模，将功能操作包装为WEB服务，并部署到ServiceMix中，最后通过BPEL根据业务流程来整合这些服务，如图6-4所示。

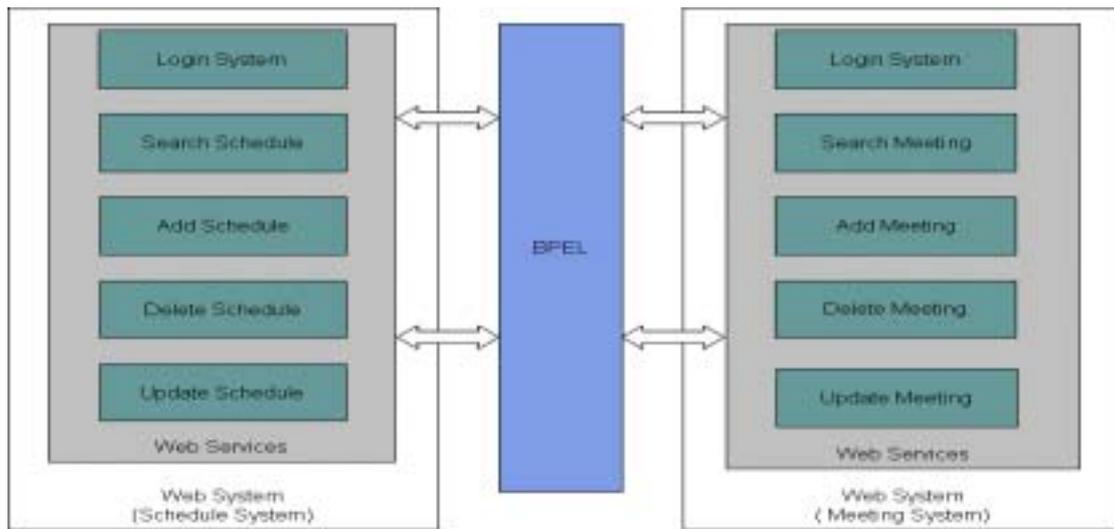


图 6-4 业务流程整合

Fig 6-4 Business Process integration

这种框架的优点在于我们可以充分利用SOA实现平台的优点。如利用实现平台进行消息格式转化等，此外通过BPEL对包装成WEB服务的功能进行业务流程处理时，可通过BPEL的特色解决很多整合问题，象BPEL的补偿功能可以用来解决多个系统事务处理的问题。BPEL的并行处理及异步执行的特色可从性能上大大提升整合系统的执行效率。

6.4 本章小结

本章主要介绍了我们对开发平台的一部分扩展工作，针对静态 WEB 页面整合领域做了一些扩展与尝试，提供了一个开发框架。目标也是为了针对具体领域对 SOA 的开发平台进行扩展。也就是对标准开发框架进行的扩展工作

第七章 开发示例

本章中将通过创建一个简单的示例来展示如何利用开发平台实现 SOA 系统的业务流程处理。在这个例子中将介绍如何进行 WEB 服务编排，如何将业务流程和 ESB 进行整合，以及如何利用 JMS 模板来发布和预订 ActiveMQ 的主题。

7.1 场景

现有 3 个 WEB 服务分别是 OddOrEven, SayOdd, SayEven。

OddOrEven 服务的作用是当用户输入一个整数时对它进行取余。如果输入为奇数则返回 1，偶数返回 0。

SayOdd 服务输出字符 “the number is Odd”

SayEven 服务输出字符 “the number is even”

目标是需要将这 3 个独立的 WEB 服务组合为一个流程，并通过事件驱动流程运行，由 JMS 客户端发送一个消息启动流程。流程接受到消息后将先调用 OddOrEven 服务，如果消息中包含一个奇数时调用 SayOdd 服务，反之调用 SayEven 服务。

7.2 WEB 复合框架

为了便于开发业务流程系统，SOA 开发平台中提供了一个 WEB 复合的基础框架，可通过项目向导自动生成框架结构。它包括了 BPEL 项目所需的组件、配置及模板程序。主要有 ESB 容器 ServiceMix, 消息代理 ActiveMQ, BPEL 引擎 PXE, 及定义业务流程的配置文件模板等。

下面是这个例子中各个组件间的调用关系

(1) 这个例子中有一个 JMS 的客户端程序 JMSClient, 它发送一个 SOAP 格式的请求消息到一个部署在 ServiceMix 上的 JMS BC。

(2) JMS BC 将请求消息转给部署在 PXE 上的 BPEL 服务, 然后等待返回信息

(3) BPEL 服务接收到请求后进行业务流程处理并利用 SAAJ BC 调用外部的 WEB 服务

(4) JMSClient 和 JMS BC 之间通过 JMS 在 ServiceMix 外部进行通信，而 JMS BC, BPEL 服务和 SAAJ BC 之间则通过 ServiceMix 内部的 NMR 进行交互。

7.3 开发流程

7.3.1 创建 BPEL 项目

通过 Eclipse 插件创建一个 BPEL 模版项目，创建过程中需设定流程名称。项目生成后会自动创建好 BPEL 流程所需的目录及文件结构。

7.3.2 定义流程及相关的服务

创建流程所要用到的 3 个 WEB 服务，OddOrEven, SayOdd, SayEven。它们作为流程的合作伙伴关系。通过 Apache Axis 可以将 java 类转换成 WEB 服务并通过 Tomcat 发布，因此只需要实现 3 个 JAVA 类，由 Axis 进行发布，具体实现可通过 Eclipse 的 WTP 插件完成。发布成功后可通过浏览器访问 WSDL 描述的方式来查看接口定义。

流程相关的 WEB 服务成功发布后下一步工作是根据业务流程进行 WEB 服务编排，就是通过 BPEL 流程定义将这 3 个 WEB 服务复合为一个服务。因此我们首先要定义流程服务的 WSDL 描述。插件已根据流程名称创建了流程的基础 WSDL 文件，只需在这个文件上进行编辑。

第一步要定义命名空间，然后通过 import 标签将 3 个 WEB 服务的 WSDL 描述导入以便新服务访问。最后定义流程服务的消息类型，操作、端口、绑定、服务以及伙伴类型等。

7.3.3 定义和编译 BPEL 以及部署文件

完成服务描述后开始定义描述流程的 BPEL 文件，BPEL 文件中第一步将流程要用到的命名空间、公共消息、伙伴类型进行定义，第二步开始定义流程。通常流程最外层为一个顺序流程，顺序流程内首先定义一个接收操作，用来接收外部输入消息以启动流程。然后需定义赋值操作将输入的消息转换为 OddOrEven 服务所需的输入消息。下一步定义调用 OddOrEven 服务的操作。在调用 OddOrEven 服务之后定义一组分支流程，如果 OddOrEven 服务调用成功将根据输入值返回 0 或者 1。根据返回值进行 Switch 分支操作，当值为 0 时这一分支将包含一个调用 SayEven 服务的操作和一个将返回值赋

给 BPEL 流程的输出消息的操作，值为 1 时包含调用 SayOdd 服务的操作及赋值操作。最终两个分支合并起来通过一个 reply 操作将流程的输出消息返回。

因为我们选用 PXE 作为 BPEL 的 Engine，因此还要定义 PXE 自己的配置文件 pxe-system.xml。在这个文件中我们要首先定义流程要用到的命名空间，通常包括流程服务及合作伙伴服务命名空间的引用。下一步定义流程所需的通道，根据需求有一个访问流程的通道以及 3 个访问合作伙伴服务的通道。最后定义使用通道的虚拟服务及控制流程的虚拟服务。要注意的是 PXE 即可以作为 ServiceMix 上的一个 Engine 通过 JBI 对部署在 ESB 上的 WEB 服务进行编排，又可以独立作为一个 BPEL 的运行环境直接通过 SOAP 协议调用 WEB 服务。因此虚拟服务的提供者可为 JBI 或 SOAP，这里我们需设定为 JBI。

到这里流程已经定义完毕，可以编译流程，并部署在 PXE 上运行了。通常为了将流程整合到 ServiceMix 中，通过部署到 ServiceMix 的 PXE Engine 组件解释执行，所以需要修改 JBI 组件统一的部署描述文件 JBI.xml。在这个文件中需描述服务如何装配及目标的 Engine 等，这些工作已由插件自动完成了。

7.3.4 部署 BPEL 流程到 SOA 平台

完成对 BPEL 及部署文件的定义后可以运行 PXE 提供的 Ant 支持将流程编译成一个可部署的压缩文件。然后就可以部署 BPEL 流程到 ServiceMix 了。ServiceMix 部署组件的过程完全基于 JBI 的规范。只要将要部署的压缩文件丢到部署目录即可。在 ServiceMix 启动后会自动监控安装目录及部署目录，发现有新组件会自动部署。我们的插件在编译的过程中已自动将 BPEL 流程进行了部署。

当然我们首先要确保 PXE Engine 已经部署到 ServiceMix 之上才能运行 BPEL 流程。需将 PXE 源程序编译后生成的 PXE-install 文件放到 ServiceMix 的安装目录中。我们的 BPEL 项目模版在创建过程中已将 PXE Engine 部署到 ServiceMix。

最终需要做得是配置 ServiceMix 要用到的组件，需要将流程服务添加到 ServiceMix 的配置文件中，因为希望通过 JMS 消息的方式启动流程所以定义了一个处理 JMS 消息的 BC 目的服务为 BPEL 流程服务。

此外流程需要访问 3 个 ServiceMix ESB 之外的 WEB 服务所以还要定义 3 个处理 WEB 服务的 SAAJ BC。

7.3.5 运行示例

现在这个例子已经完成了。将 ServiceMix 容器启动，然后通过发送一个 JMS 消息唤起部署在 ServiceMix 之上的 BPEL 流程，流程通过 ESB 和 SAAJ BC 访问 3 个部署在外部 Tomcat 容器中的 WEB 服务。最终将结果以 JMS 消息的形式返回。

7.4 示例总结

这个例子用到一个处理 BPEL 流程的 PXE 服务 SE 组件，一个处理 JMS 消息的 BC 以及 3 个访问外部 WEB 服务的 SAAJ BC 组件，具体结构如图 7-1 所示。

虽然逻辑很简单也只是采用了 ServiceMix 本身提供的一些组件。但它的意义在于模拟了企业集成中最常见的场景：通过 SOA 平台将几个孤立的 WEB 服务组合起来按照一定的业务逻辑形成业务流程，通过事件驱动启动流程执行。流程与启动流程的外部程序或服务异步执行，流程返回结果也以消息的形式返回调用者。当然实际场景复杂的多，往往需要整合的并不是 WEB 服务而是遗留系统，需要先将其转化为 WEB 服务或通过自定义组件扩展实现遗留系统与 ServiceMix 之间的交互。我们的平台主要是提供了一种基于 JBI 的可扩展的 SOA 实现方式，用户将自定义的组件部署在我们的平台之上共同搭建 SOA 系统。

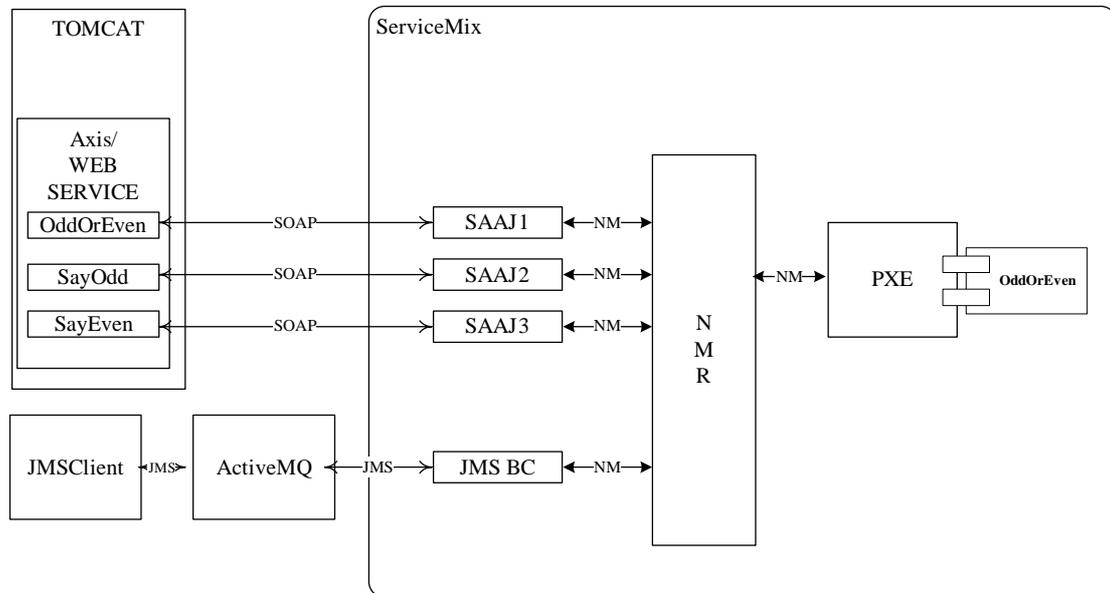


图 7-1：BPEL 示例

Fig 7-1 BPEL Example

7.5 本章小结

本章主要介绍了如何利用开发平台构建一个业务流程处理系统。通过这个简单的例子展示了开发 BPEL 程序的流程。从首先定义 BPEL 服务描述、流程描述到修改 PXE 和 JBI 的配置文件最终编译、部署及运行的全过程。

第八章 总结与展望

8.1 平台现状

SOA 实现平台从如何实现 SOA 出发，基于 JBI 规范进行设计，是一种灵活可插拔的架构。从平台的结构上分析，它是一种以 ESB 为核心的总线结构。以开源 ESB 产品 ServiceMix 为基础，辅以 BPEL 引擎 PXE 和消息系统 ActiveMQ 及一些符合 JBI 规范的开源组件组成。

平台还提供了一组具有普遍性的标准开发框架和开发流程，开发人员既可以基于它们轻松的开发 SOA 系统，又可以针对具体业务逻辑对它们进行扩展。

目前这个 SOA 实现平台已初步完成，通过了内部测试，并利用平台开发了一组 SOA 的应用系统，对平台各方面进行评测。

- 针对遗留 WEB 系统、文件系统及事件驱动系统等方面开发的示例，利用 BC 组件将各种遗留系统包装为服务，整合到平台中。展示了平台具有良好的适应性，充分体现了 SOA 与平台无关的特性。
- WEB 复合系统的示例主要测试了平台的业务流程处理能力，以及根据业务来整合不同服务的能力。
- 而通过 WEB 页面整合模块的开发，则充分体现了平台本身的良好扩展性，可针对具体业务领域进行功能扩展。
- 这些示例最终都部署到运行平台上进行测试，来检验平台的性能，稳定性及负载能力。

通过对平台进行的测试和评估工作，最终结论是平台具有良好的适应性和扩展性，并且具备一定的性能及稳定性。因此平台足以满足普通的 SOA 应用，同时也可以作为企业实现复杂 SOA 系统的开发和过渡平台。

现在 SOA 实现平台作为一个开源项目已经发布到网络上提供给用户下载试用，并提供了相应的文档支持和在线技术支持。根据最终用户的反馈以及随着我们对 SOA 思想理解的不断加深和相关开源技术的不断发展，SOA 实现平台的组件和框架也会不断更新和发展。

[paul1]8.2 平台的特点

作为一个基于开源技术组建的 SOA 系统实现方案，我们的实现平台同现有的相关产品相比有哪些优势呢？它的存在价值究竟体现在哪些方面呢？平台的特点可概括为以下几个方面：

- 完整的 SOA 实现支持

目前各种开源的 SOA 项目非常多，然而功能相对完备的 SOA 开源项目目前还没有。我们的平台整合了针对不同领域的多个成功开源 SOA 技术，并在此基础上进行了扩展，目标是提供给用户面向 SOA 各个领域的全套解决方案。

- 先进的架构

同以往的 EAI 技术相比，我们的平台作为一种针对企业应用集成领域的 SOI 解决方案，部署在平台上的应用可以在一个标准、可靠、安全、可管理的环境下，以松散耦合的方式进行交互，并可根据业务需求动态的进行企业应用集成，因此使用平台开发的系统具有灵活的应变能力和良好的重用能力。

- 良好的扩展性

平台建立在开源技术和业界规范之上，各个组件间的整合遵循业界标准以及侧重松耦合原则，因此可以根据需求灵活的替换及增加组件，扩展系统的功能。

- 简化的开发模式

开发平台基于最通用的 IDE Eclipse，提供标准的开发框架和模板，以及一组图形化的辅助开发工具，降低了开发过程的难度和复杂度。此外还提供了示例和文档，并提供技术支持。

- 低成本的 SOA 实现

平台基于开源技术，因此对用户来说可以大大降低 SOA 系统的实现成本。用户既可以单独利用我们的平台实现 SOA 系统，又可以与商业软件相互结合实现。此外还可用作学习 SOA 技术的实践和过度平台。

[paul2]8.3 当前的问题

相对于最初组建这个 SOA 实现平台的目标，目前还可能存在一些问题及不足。主要有以下几点：

8.3.1 功能问题

目前 SOA 开发平台的功能还比较单薄，相对商业软件来说对开发人员提供的支持

还比较少，还缺少一些图形化的工具。如缺少 BPEL 的图形化建模工具。这使得平台对开发人员的技术要求比较高，有一些局限性。此外 SOA 方面还有一些领域我们的平台还没有涉及到，如 UDDI，网格计算等。需要我们作进一步研究。

8.3.2 稳定性问题

稳定性主要包括两个方面系统稳定性和模块稳定性。系统稳定性是指由于目前平台主要由开源组件构成，所以平台的稳定性还需进一步测试。系统稳定性主要针对组件本身的稳定性，以及这些组件整合后的兼容性。模块稳定性主要是由于开源组件非常活跃需要经常进行升级，此外还会遇到项目停滞或被收购的问题，可能会进行组件替换。

8.3.3 性能问题

性能及负载能力也是我们关注的重点，目前这方面所做的工作是通过替换数据库及部署到 J2EE 服务器进行性能扩展。由于目前在平台上开发和运行的示例本身系统复杂程度和企业级应用还有很大差距，因此还需要进一步测试和评估。

8.4 下一步工作

平台下一步要做的工作首先是针对目前遇到的问题，做出适当的解决。此外还要从以下几方面进行扩展。

8.4.1 功能扩展

在功能方面，我们还要根据客户的具体的反馈做出相应扩展，目前主要考虑的是从平台的易用性上进行扩展。通过基于 Eclipse 的图形化工具，对 SOA 系统建模、编辑、和部署考虑，面向开发人员尽可能提供友好的开发及部署支持。

8.4.2 可靠性扩展

可靠性也是我们下一步工作的重点，这方面的工作主要针对系统的管理人员的需求。比如系统中的状态监控、业务流程的监控、事物补偿方面要做更多扩展。特别是业务流程的监控，当面对复杂的业务逻辑，通过 BPEL 往往需要调用复杂的业务流程，可能还需嵌套调用其它 BPEL 流程，如何跟踪流程调用的服务，流程当前的状态，并以友

好的界面展现给用户，这都是我们迫切要解决的。

8.4.3 长远目标

SOA 提倡的设计原则之一是自顶向下分析和定义服务，这个分析过程实际上是根据业务流程逐步细化服务的过程。谁最适合做这部分工作呢？毫无疑问是业务人员，只有他们最了解和明确具体的业务需求。而目前 SOA 的整合技术如 WEB 服务、BPEL 即便有再友好的用户界面对业务人员来说也太复杂了，有太多的技术性细节。因此实际上都是技术人员根据与业务人员的沟通进行分析和定义服务。根据软件工程的理论，这种沟通越多，项目失败的风险越大。能否在现有 SOA 技术上进一步抽象，达到业务人员能够掌握和明确描述服务的程度呢？这也是我们下一步要研究的方向。

具体来说服务本来就是抽象的概念，能否提供给业务人员一个图形化的易操作的平台以及一组通用的描述符号。通过针对某些领域让业务人员可以根据自己的业务需求自顶向下定义出一组虚拟的服务，以及虚拟服务间的操作流程。这样开发人员就可以根据这些服务的描述自底向上实现它。

譬如针对银行领域我们事先将一些通用的流程或者服务定义为符号。业务人员可以用这些符号以及业务流程符号一起描述系统组成。根据最终描述自动生成系统所需的 WEB 服务描述和业务流程文件描述。

下一步将对这种针对业务人员分析服务的设想的可行性做进一步研究和探讨。

8.5 总结

毫无疑问 SOA 技术的发展趋势不可逆转，企业越早的理解 SOA 以 SOA 的思想构建系统，意味着越少的后期业务变更或系统整合的投入。

我们的 SOA 实现平台具有良好的扩展性和技术先进性，在正处于面向对象思想向面向服务思想转化中的现阶段，它可以给想要构建 SOA 系统的用户提供灵活的解决方案，减少盲目投入的风险。用户可通过这个平台逐步熟悉和理解 SOA 技术，以 SOA 的思想构建系统，真正进入面向服务领域。

我相信随着对平台的不断扩展和更新，我们的 SOA 实现平台将会越来越成熟和稳定。能够对推动 SOA 技术的发展做出贡献。

8.6 本章小结

本章主要对全文做了总结，介绍了系统的现状，对开发过程所遭遇的问题以及可能会碰到的问题进行了分析，并根据它们提出了下一步的工作目标。

参考文献

- [1] Hao He , What Is Service-Oriented Architecture, <http://www.xml.com/pub/a/ws/2003/09/30/soa.html> , September 30 , 2003.
- [2] 面向服务架构(SOA)的原则 [EB] , <http://www.csdn.net>. 2003 , 3.
- [3] SOA :构建下一代 Web 服务的技术架构[EB], <http://www.hebeiasp.com/NewsContent.aspx?type=Library&id=14>, 2004 , 4.
- [4] Web Services standard, <http://www.w3.org/2002/ws/>
- [5] David Chappell, Tyler Jewell, Java Web Services. O'Reilly Press, March 2002.
- [6] Kishore Channabasavaiah , Kerrie Holley , Edward M Tuggle Jr, 迁移到面向服务的体系结构, <http://www-900.ibm.com/developerWorks/cn/webservices/ws-migratesoa/index.Shtml>, 2004.
- [7] 李安渝 , Web Service 技术与实现 [M] , 北京 : 国防工业出版社, 2003.
- [8] D. Jack Elzinga , Tomas Horak , Chung-Yee Lee , Charles Bruner, Business Process Management: Survey and Methodology, IEEE Transactions on Engineering Management, 1995.
- [9] 盛戈散 , 杨宗凯 , 李达, 基于 BPML 的商务流程定制工具的开发与应用. 计算机工程与应用, 2003
- [10] BPEL4WS 1.1 , Business Process Execution Language for Web Services Version 1.1
- [11] Matjaz Juric , BPEL 实例教程
http://www.oracle.com/technology/global/cn/pub/articles/matjaz_bpel1.html
- [12] WS-BPEL 2.0 , Web Services Business Process Execution Language Version 2.0
- [13] 中国科学院软件研究所软件工程技术中心, 面向服务的计算和应用集成, www.ios.ac.cn, 2003.
- [14] Frank Leymann , Dieter Roller , Business processes in a Web services world , August 2002 , <http://www-900.ibm.com/developerWorks>
- [15] Chris Peltz , Hewlett Packard , Co, web services orchestration , January 2003
- [16] Martin Keen, Amit Acharya, Patterns: Implementing an SOA Using an Enterprise Service Bus [ibm.com/redbooks](http://www.ibm.com/redbooks)
- [17] Nick Simha, Understanding ESB , http://dev2dev.bea.com/blog/nsimha/archive/2005/09/understanding_e_1.html , September 03 , 2005
- [18] Francisca Losavio , Dinarle Ortega , Maria Pérez. Modeling EAI. In: Proceedings of the XII International Conference of the Chilean Computer Science Society (SCCC'02) , Copiapo , Chile , November 2002
- [19] 潘顺 EAI 模型与构架设计研究 电信科学, 2004 , 20(6):15-18
- [20] JBI Standard JSR 208: Java™ Business Integration <http://www.jcp.org/en/jsr/detail?id=208>
- [21] Daniel Rubio 重造 Java ESB : 细述 JBI 与 ServiceMix

- <http://fanqiang.chinaunix.net/program/java/2005-11-25/3878.shtml>
- [22] ibm.com/redbooks Patterns: Service-Oriented Architecture and WEB Services
- [23] [Servicemix](http://servicemix.org/) <http://servicemix.org/>
- [24] [Mule](http://mule.codehaus.org/) <http://mule.codehaus.org/>
- [25] [Glassfish](https://glassfish.dev.java.net/javaee5/jbi-se/ServiceEngine.html) <https://glassfish.dev.java.net/javaee5/jbi-se/ServiceEngine.html>
- [26] [OpenESB](https://open-esb.dev.java.net/) <https://open-esb.dev.java.net/>
- [27] [Axis](http://ws.apache.org/axis/) <http://ws.apache.org/axis/>
- [28] [Wasp](http://xml.coverpages.org/Systinet2003.html) <http://xml.coverpages.org/Systinet2003.html>
- [29] [WTP](http://www.eclipse.org/webtools/) <http://www.eclipse.org/webtools/>
- [30] [PXE](http://pxe.fivesight.com) <http://pxe.fivesight.com>
- [31] [Twister](http://www.smartcomps.org/twister/) <http://www.smartcomps.org/twister/>
- [32] [Activebpel](http://www.activebpel.org/code/validator/) <http://www.activebpel.org/code/validator/>
- [33] [BPEL4J](http://www.alphaworks.ibm.com/tech/bpws4j) <http://www.alphaworks.ibm.com/tech/bpws4j>
- [34] [Cape Clear Orchestrator](http://www.capeclear.com/products/orchestrator.shtml) <http://www.capeclear.com/products/orchestrator.shtml>
- [35] [ActiveWebflow Designer](http://www.active-endpoints.com/products/index.html) <http://www.active-endpoints.com/products/index.html>
- [36] [Oracle BPEL Design](http://www.oracle.com/technology/global/cn/products/ias/bpel/index.html) <http://www.oracle.com/technology/global/cn/products/ias/bpel/index.html>
- [37] [Java studio Enterprise TPR-orchestration](http://developers.sun.com/prodtech/javatools/jsenterprise/tpr/reference/docs/bpel_gsg.html)
http://developers.sun.com/prodtech/javatools/jsenterprise/tpr/reference/docs/bpel_gsg.html
- [38] [Apache Geronimo](http://geronimo.apache.org/) <http://geronimo.apache.org/>
- [39] [ActiveMQ](http://www.activemq.org/) <http://www.activemq.org/>
- [40] [Eclipse](http://www.eclipse.org/) <http://www.eclipse.org/>
- [41] [Httpclient](http://jakarta.apache.org/commons/httpclient/) <http://jakarta.apache.org/commons/httpclient/>
- [42] [Httpunit](http://httpunit.sourceforge.net/) [http:// httpunit.sourceforge.net/](http://httpunit.sourceforge.net/)

致 谢

本文是在很多人的帮助下才得以完成的，特别是杨旭波老师诲人不倦的指导，不断的在我迷失时、不知所措时用他那智慧的眼光帮我找到前进的道路，指明研究的方向，使得我终于能在如此短的时间完成这篇论文。

此外还要特别感谢我的企业导师和老板钟友良博士，他以严谨踏实、一丝不苟的态度对待我的工作和研究，在他的严格要求和耐心指导下，我的理论知识和实践能力都到了极大的提高。同时他给我一个自由的空间，让我能够充分发挥我的能力，他教给我很多做人的道理，也将使我受益终生。

还要感谢的是交大的任瑞老师、毛桂琴老师等对我耐心的帮助。

最后，对我的父母表达深深的谢意，没有他们的教导和支持我不可能一直坚持进行下去。

马中游

2006年05月

攻读硕士学位期间已发表或录用的论文

- [1] 马中游. 构建一个 SOA 的开源平台.《计算机工程》发表于 2006 年 12 月[paul3]

SOA的开源实现平台

作者：[马中游](#)
学位授予单位：[上海交通大学](#)

相似文献(10条)

1. 学位论文 [刘洋](#) [基于Web Services面向服务架构的虚拟企业应用集成研究](#) 2005

虚拟企业作为一种企业间的协同合作模式,其成员企业是动态的、不确定的,并且各成员企业运行模式所依赖的底层基础架构、通信协议以及对外交换的数据格式都有所差异,整个动态联盟处于一种异构的、分散的环境中。如何保障在成员之间方便地、低代价地实现无缝集成和互操作,从而提供一个动态的企业间的应用集成环境,是构建虚拟企业的关键问题之一。

目前,多数的虚拟企业应用集成方案一般都是“独立”地针对某个特定应用来设计,并且集成双方之间实现的是紧耦合机制,难以在使用期进行再定制。而WebServices技术作为一种新型的分布式对象技术,具有完好封装、松散耦合和高度可集成能力等特点,能够很好地满足虚拟企业成员的动态性要求,并解决成员之间应用集成的架构异质问题,从而使虚拟企业的应用集成环境具有良好的可扩展性和易维护性。

本文首先对企业应用集成的内涵进行了深度和广度的阐述;结合虚拟企业的特征,分析并总结了虚拟企业应用集成在系统体系结构和集成机制两方面的特性需求;进而对现有的虚拟企业应用集成方案进行了不足分析,并详细地分析了WebServices技术在虚拟企业应用集成构建中的优势所在。在上述分析的基础上,提出了一个基于WebServices技术的面向服务架构(SOA)的虚拟企业应用集成模型,详细的设计和阐述了集成体系架构中的核心企业方、成员企业方以及客户方的各自功能和交互,并深入探讨了如何利用WebServices接口集成调用合作伙伴所提供的应用服务,以及私有UDDI和统一身份认证服务的创建。最后,将模型应用于一个虚拟企业的分布式集成订单处理系统,具体地实现了成员之间的应用服务松散耦合的集成及调用。从而验证了基于WebServices的虚拟企业应用集成模型的可操作性和正确性,为进一步实现复杂虚拟企业应用系统做了必要的理论与实践准备。

2. 期刊论文 [马华](#) [李建华](#) [MA Hua](#) [LI Jian-hua](#) [面向服务架构在动态企业应用集成中的应用](#) -[计算机工程与设计](#)2006, 27 (13)

现代企业间的竞争日趋激烈,为迅速实现业务过程重组并敏捷地响应市场变化,企业产生了进行动态企业应用集成(EAI)的需求。粗粒度的松耦合的面向服务架构(SOA)是一个支持动态EAI的具有较好健壮性的架构。通过对SOA的分析,提出了一个通用的SOA基础模型,分析了基于SOA的动态EAI技术和实现方式。最后结合医学资源网项目,开发了一个基于SOA的动态EAI平台。

3. 学位论文 [马奔](#) [基于面向服务架构\(SOA\)的企业应用集成研究](#) 2007

自从国家提出以信息化带动工业化以来,大中型企业都相继建立了相对比较丰富的各类企业应用系统。由于企业信息化过程是渐进的、分散的,在这个过程中使用的各种应用系统通常运行于不同的系统平台、由不同的开发工具开发而成,其后果是形成了一个“信息孤岛”。

伴随着企业规模的扩大,企业信息共享已成为其竞争资本。因此,如何将企业现有的应用系统集成起来,实现数据层、应用层和业务层的共享成了当前企业应用集成的核心问题。

本文首先在比较了传统集成方案和面向服务架构集成方案的管理复杂度、可控度、敏捷性以及可重用性的基础上,结合娃哈哈公司当前的信息化水平、业务需求和未来的信息化策略,提出了娃哈哈公司基于面向服务架构的企业应用集成框架。该框架以企业应用集成深度为轴线,涵盖了企业数据层、应用层和业务层集成范围。然后,文章详细论述了企业数据层、应用层和业务层集成的技术方案和实现方法。最后,分析了娃哈哈公司的当前信息化困境及其今后的发展方向,在娃哈哈公司基于面向服务架构的集成框架的基础上提出了娃哈哈基于面向服务的企业应用集成J2EE解决方案。在整个过程当中,作者重点研究了以下内容:

(1)提出了娃哈哈基于面向服务架构的集成框架,该框架以企业应用集成深度为轴线,涵盖了企业数据层、应用层和业务层集成;在文章的最后,提出了娃哈哈基于面向服务的企业应用集成J2EE解决方案,论述了新系统与SAP/R3系统及其它应用系统如何通信的问题。

(2)在数据层集成问题上,提出了利用基于XML数据模型的虚拟数据库和数据仓库来共同实现数据集成,不但解决了企业数据集成问题并可大幅度提高企业大数据量报表的查询性能,而且为企业开展商务智能提供了数据基础。并且在研究如何将XQuery转换成标准SQL的问题上提出了将最小生成树(MST)算法用于解决查询计划的优化问题;

(3)在应用集成问题上,提出了基于面向服务的用户角色访问控制模型(SOA-RBAC),并将以前的权限控制扩展到了数据级控制,该模型可提高数据访问的安全性。

(4)在业务层集成问题上,提出了基于语义的Web服务发现和合成匹配算法:细粒度匹配算法(根据请求者需求匹配已发布的服务实现 Web服务发现)和粗粒度匹配算法(根据最优目标匹配候选服务列表实现Web服务合成,文章提供了多选择背包算法和多属性决策分析),弥补了BPEL4WS的缺陷。

4. 期刊论文 [李建华](#) [陈松乔](#) [马华](#) [LI Jianhua](#) [CHEN Songqiao](#) [MA Hua](#) [面向服务架构参考模型及应用研究](#) -[计算机工程](#)2006, 32 (20)

面向服务架构(SOA)是一种粗粒度、松耦合的系统结构,它支持动态的企业应用集成。提出了一种以服务执行引擎为核心的面向服务架构参考模型,并结合遗传医学资源网项目,在该参考模型的基础上设计了一个基于SOA的动态企业应用集成支撑系统。

5. 学位论文 [顾莉娜](#) [面向服务的企业应用集成的研究与实践](#) 2005

文章首先总结了企业应用集成及面向服务架构的概念,然后分析了面向服务架构在企业集成中的应用模式、对企业各方面的影响,其目的是给出一个企业如何使用面向服务架构进行应用集成的全局观。在此基础上,文章初步探讨了面向服务的企业应用集成策略,研究其可取与不足之处,并在其中注入了新兴的企业服务总线与企业流程管理理念,分析出了对相对全面的面向服务的企业应用集成一框架模型。通过一个简化的供应链系统的案例分析,说明了这一集成框架在实际应用中的价值。最后,关于进一步工作的方向进行了简要的讨论。

6. 会议论文 [王海林](#) [企业应用集成与面向服务架构](#) 2007

企业的信息化建设处在一个从多个应用解决方案过渡到构建企业整体信息架构和应用阶段,企业应用集成(EAI)是当前信息化建设面临的一个复杂的课题。传统的企业信息化无架构或架构不完整导致信息化应用缺乏可扩展性和灵活性,企业应用集成正快速从传统架构向新兴的面向服务架构(SOA)演进。文章回顾了EAI的发展过程,探讨了SOA的有关理念和特点,通过江苏省电力公司EAI与SOA的实际案例比较,深入分析了SOA与EAI的异同。

7. 学位论文 [杜志文](#) [基于动态工作流的面向服务企业应用集成研究](#) 2007

企业应用集成(EAI)是指通过企业内部网络连接在一起的应用程序和数据来源来实现数据和业务的自由共享。由于早期开发的软件系统从规划和技术架构上都没有从整个企业的信息架构上考虑,因此产生了很多信息孤岛。面向服务架构(SOA)利用服务来封装遗留系统的业务功能,并通过服务组合来灵活的构建企业的业务流程,具有很好的灵活性和扩展性,由此面向服务的企业应用集成成为了一个重要的研究方向。随着电子商务的不断发展,需要将不同业务功能的Web服务基于业务流程整合起来,来满足复杂的商业逻辑需要。而传统的服务组合技术因为在设计阶段就需要将服务硬编码到业务流程中去,使得业务在运行时不能根据实际的商业环境进行动态的改变。

本文分析了现有企业应用集成的模型,并在此基础上提出了一种面向服务的企业应用集成模型。该模型保留了面向服务企业架构的松散耦合性等特点,引入工作流技术来取代传统的服务编排,利用工作流的样式定制服务编排的规则,支持传统的服务组合,并支持更加复杂的流程编排和人机交互工作流活动。为了解决传统企业应用集成在设计阶段就需要将服务硬编码到业务流程中去的缺陷,该模型引入了抽象服务的概念。利用抽象服务结合工作流来编制业务流程,用服务代理机制实现工作流的动态性,使得构建的业务流程能够在运行时根据上下文动态地选取实际的Web服务。而QoS参数定制使得业务流程可以根据用户的需要选择可用的服务。除此以外,本文还分析了目前已有的几种基于工作流样式的服务组合算法,比较了这些算法的时间性能,并根据其不足提出了一种网格环境下的改进型服务选择算法,实验结果证明了该算法的有效性。与此同时,根据企业实际的应用环境,分析了该改进型算法的适用范围,在此基础上提出了一种企业环境下服务选择算法的选择策略。最后,结合一个实际的企业应用集成实例,给出了基于动态工作流的面向服务企业应用集成模型的分析、设计和构造过程。

8. 期刊论文 [唐跃中](#) [曹晋彰](#) [郭创新](#) [曹一家](#) [韩祚祥](#) [TANG Yuezhong](#) [CAO Jinzhang](#) [GUO Chuangxin](#) [CAO Yijia](#) [HAN Zhenxiang](#) [电网企业基于面向服务架构的应用集成研究与实现](#) -[电力系统自动化](#)2008, 32 (14)

结合电网企业数字化和信息化的发展方向,针对电网企业目前信息化的发展现状,提出采用企业应用集成(EAI)技术解决现存信息孤岛的问题。指出基于面向服务架构(SOA)的集成技术是实现电网EAI的重要模式,提出基于SOA的EAI框架,并从数据层和业务流程层集成设计相应的实现方案,使该集成平台可以提供基于公共信息模型(CIM)语义的数据和完成典型业务流程的自动化运行,最后介绍上海市电力公司的基于SOA的EAI实践。

9. 学位论文 [唐乐竟](#) [基于SOA的企业应用集成技术研究](#) 2009

企业应用集成EAI是企业信息化应用的一种延伸,它将企业内部若干异构信息系统通过一定的技术手段有效地组织起来,使其成为一个逻辑上的整体,从而更好地发挥出企业信息系统的作用。传统的EAI技术在系统实现成本、耦合度和开发周期等方面都存在问题,限制了EAI应用的推广和发展。面向服务架构SOA作为一种新的计算模型,它通过服务这种统一的模式,为支持异构系统在分布式环境下的松耦合集成提供了一种有价值的技术途径。

论文研究工作主要围绕运用SOA架构及其相关技术支持EAI应用这个主题进行,重点研究基于SOA的企业应用集成的架构问题。论文主要开展了以下研究工作:

论文首先研究了传统企业应用集成模式及其关键技术,分析了传统EAI技术存在的不足,并将SOA架构引入EAI应用领域,探讨了SOA的服务封装、松散耦合的交互方式在EAI中的应用模式。然后对目前应用于EAI的SOA技术——企业服务总线ESB进行了研究,分析了ESB所采用的集中式数据交互方式对面向服务集成的系统可靠性和可扩展性方面存在的局限性。

运用P2P技术思想,提出了一种面向服务集成模型SOIM。该模型通过“分布式通信,集中式管理”的架构,采用P2P重叠网络的方式来进行服务的部署、组织以及服务数据的交互,从而消除了集中通信方式的容量限制和性能瓶颈,具备良好的系统伸缩性,为基于SOA的企业应用集成提供了一个开放、可靠、可扩展的集成基础平台。

论文针对服务承载节点动态负载均衡对系统性能造成的不利影响,在SOIM模型基础之上融入了Web服务动态迁移的思想,通过对Web服务的动态部署和迁移机制来实现节点服务器间的负载均衡,优化系统性能。重点讨论了服务迁移的触发条件、服务迁移的策略和迁移服务的一致性等问题,并通过仿真实验,验证了服务迁移策略中对等节点负载均衡算法对系统性能的影响。

10. 学位论文 吕鸣剑 基于SOA企业应用集成与设计的研究 2007

面向服务架构(SOA)是新一代的架构思想,用于分布式软件开发。由于SOA具有良好的松耦合、与平台无关等特性,很好的解决了系统的灵活性和互操作问题,因而具有广泛的应用。目前,作为企业新系统架构和企业应用集成的主要解决办法,SOA正在成为研究领域的热点。在未来的软件开发世界,SOA将成为软件体系结构领域的统治者。

本文对企业应用集成EAI传统的实现技术进行了学习研究,分析了面向服务的软件体系结构出现的必然性,及其特点和优势。为了选择SOA的实现技术,特别对SOA在Internet环境下的实现技术—Web Service进行了学习研究,主要包括Web Service的基础知识和协议规范,并将Web Service技术与传统的分布式计算技术进行了比较,更加突出Web Service的与平台无关、易于扩展、易于集成等特点。

本文结合SOA的设计理论知识,综合运用DCOM、Web Service等技术,研究提出了基于SOA的企业应用集成的一般模型。该框架模型可以在现有各种异构企业平台上构建一个平台无关、语言无关的中间层,以支持不同平台应用彼此无缝地连接和集成,并以一种松散的服务捆绑形式,快速、低代价地开发和绑定各种企业应用。另外,本文对Web Service互操作、旧有系统的Web Service转化问题做了研究和讨论。针对Web服务的安全问题,提出了利用SOAP头条目进行验证的解决办法;对于企业私有UDDI注册中心的组建,提出了实用、快捷的实现方法。

最后,结合中石油股份公司的基金项目管理系统集成案例,根据该企业自身的特点,在.NET开发平台下,实践了基于SOA的企业应用开发。

本文链接: http://d.g.wanfangdata.com.cn/Thesis_D031078.aspx

授权使用: 上海理工大学(shlgtsg), 授权号: b954302d-30d3-4b53-b9c1-9e7900d31a01

下载时间: 2011年1月28日