面向服务的网构软件中间件研究*

周相兵[1,2]，杨小平[2]，谢成锦[3]

(1. 阿坝师范高等专科学校 计算机科学系，四川 郫县 611471；2. 四川省软件重点实验室，成都 610068；3. 四川银海软件有限责任公司 研发部，成都 610021)

摘 要：网构软件是近年来提出的一种具有自底向上的自主性、协同性、反应性、演化性、多态性等特征的软件状态，主要用来解决 Internet 下分散的构件体的共享、集成和复用，但网构软件还处于发展初始阶段。结合 SCA 简化 SOA 所构建的业务应用程序的设计和集成，以及 SDO 和 DAS 的数据访问模式来构建一种网构软件中间件框架，并用 UML 对各个阶段建模；最后结合 Apache 下 SCA 系统的 Tuscany 开源包的一个远程教育集成系统来应用该网构软件中间件。

关键词：网构软件中间件；面向服务体系结构；服务组件体系结构；服务数据对象；远程教育集成系统

中图分类号：TP313 **文献标志码**：A **文章编号**：1001-3695(2008)10-3006-04

Research of service-oriented internetware middleware

ZHOU Xiang-bing[1,2], YANG Xiao-ping[2], XIE Cheng-jin[3]

(1. *Dept. of Computer Science, Aba Teachers College, Pixian Sichuan* 611471, *China*; 2. *Software Key Laborotary of Sichuan, Chengdu* 610068, *China*; 3. *Dept. of Research & Development, Yinhai Software Limited Liability Co., Chengdu* 610021, *China*)

Abstract: Internetware was a kind of software's state which was brought out these years, it was architected as bottom-up, and also with the symptom: independently, cooperatively, reactively, fast-evolvement and multi-polymorphism, it was mainly used in dealing with the sharing, integrating and reusing of the dispersive components. But from the present seeing, it was still in the initial phase of the developing, the involved theory, technique and also the methods were still in discovering and application phases. This paper combined the SCA and simplified the SOA's architecture of the business application programs' design and integrity, and the data's access model of SDO and DAS architect a internetware's middleware framework, and it was using the UML to model in every phase, at last it combined the Apache's SCA system's open source packet which was named Tuscany, to function in the distance learning integrating system, that's just the application of the internetware.

Key words: internetware middleware; SOA; SCA; SDO; REIS

0 引言

网构软件(internetware)是直接由 Internet 催生而得[1,2]，是适应日益增长的信息量、大量不可再用信息、众多信息单一功能和分散等缺陷而诞生的。目前国内外都沿这个方向从理论、技术和方法角度进行研究。国外主要的机构如 IBM、微软的 Web 服务工厂等；国内以北大青鸟研发团队、南京大学吕建团队、上海普元公司等，他们都提出了与网构软件相关概论、基础理论，以及发展方向等[3,4]。但因网构软件是具有自底向上的自主性、协同性、反应性和演化性等特征的软件形态，与传统的自上而下的方法正好相反，这就对软件体系结构的形成、软件开发的方法和使用什么样的技术带来了巨大的挑战；虽然基于构件的软件开发技术已经成为当今软件开发的一种有效方法，但对网构软件的开发还是显得力不从心。由于网构软件强调从最初的服务状态通过自动动态配置、组合、组装成一种按需求和策略的软件系统，本文就从网构软件的体系结构出发提出一种面向服务的网构软件中间件，采用 SCA 简化 SOA 的设计和集成[5]，并结合 SDO 和 DAS 数据方法[6]建立一种层次结构的网构软件结构，并用 UML 逐一分析其中所涉及的组件和相关服务，同时用 ESB 联结各组件实现集成；最后结合一个远程教育集成系统进行实践应用。

1 网构软件中间件的理论和技术基础

1) SOA(service oriented architectrue)参考模型 实行基础是组合服务提供者和服务消费者各个业务功能和流程，实现复杂的业务应用程序和流程，其相对于粗粒度的业务组件被作为服务公开，也将 IT 资产构造为一系列可复用的服务，而且这些服务是松散耦合且与平台和实现无关；但 SOA 解决方案设计为服务组合的编排和排列，并为基于构件的组装提供良好的接口和契约进行连接。在 2006 年 10 月由 OASIS 组织制定和发布了 SOA 参考模型 1.0 版本中[7]，使用了概念、关系和语义来描述服务，并使用了 capabilities 这一方式进行描述 SOA，如图 1 所示是 SOA 参考模型整合图。

定义 1 服务组合，α 称满足下面条件的一个十四元组 $SC = \langle Sp, Ss, Uddi, Wsdl, Pr, Su, Cm, On, In, Ar, Pe, T, Out\rangle$ 为一个服务组合：

a) $Sp = \{sp_1, sp_2, \cdots, sp_m\}$ 是一个有限服务提供者集合，$Sp(Wsdl) \Rightarrow Uddi$，且它们可能存在如下关系：

收稿日期：2007-11-27；修回日期：2008-03-10 基金项目：四川省考试学院“十一五”科研规划科研课题(SCZKY11528II001)；四川省教育厅自然科研课题(07ZS002)；四川省科技厅应用基础资助项目(2006J13-051)；阿坝师专校级重点基金资助项目(ABA07-08)

作者简介：周相兵(1980-)，男，四川仪陇人，讲师，主要研究方向为分布式系统集成、语义(3dsmaxmaya@163.com)；杨小平(1973-)，男，副教授，博士，主要研究方向为人工智能、语义；谢成锦(1984-)，男，软件工程师，主要研究方向为分布系统.

继承关系: $sp_{i-1} \infty sp_i \rightarrow sp_{i+1}$;

并集关系: $sp_{i-1} \cup sp_i \rightarrow sp_{i+1}$;

交集关系: $sp_{i-1} \cap sp_i \rightarrow sp_{i+1}$;

互斥关系: $P\ sp_{i-1} \cap P\ sp_i =$ 。

b) $Ss = \{ss_1, ss_2, \cdots, ss_n\}$ **是一个有限服务消费者集合,且同样可能存在 a) 中的关系,则此时可定义一个服务是一个二元组**: $S = \langle Sp, Ss \rangle$。

c) Pr、Su **分别代表服务直接前驱和后继,且** $S(Pr)$ o $S(Su)$ **表示一个前驱与后继的依赖关系**:

事件依赖: $En \rightarrow S(Pr)\ \Theta S(Su)$;

信息依赖: $Inf \rightarrow S(Pr) \cdot S(Su)$;

任务依赖: $Task \rightarrow S(Pr) \otimes S(Su)$;

语义依赖: $Se \rightarrow S(Pr) \oplus S(Su)$;

接口依赖: $Int \rightarrow S(Pr) \cdot S(Su)$,**则服务依赖集合为** $DeO = \{\langle En, Inf, Task, Se, Int\rangle, context\}$。

d) $Cm = \{Cm_1, Cm_2, Cm_3, Cm_4\}$ **是服务组合流程**:

顺序流程:

$Cm_1 = \{S_{1n} \rightarrow S_{2n} \rightarrow \cdots \rightarrow S_{mn}\} \cup \{S_{m1} \rightarrow S_{m2} \rightarrow \cdots \rightarrow S_{mn}\}$;

并行流程:

$Cm_2 = \{S_{1n} \cup S_{2n} \cup \cdots \cup S_{mn}\} \cup \{S_{m1} \cup S_{m2} \cup \cdots \cup S_{mn}\}$;

串联流程:

$Cm_3 = \{S_{1n} \infty S_{2n} \infty \cdots \infty S_{mn}\} \cup \{S_{m1} \infty S_{m2} \infty \cdots \infty S_{mn}\}$;

选择流程:

$Cm_4 = \{S_{1n} \lrcorner S_{2n} \lrcorner \cdots \lrcorner S_{mn}\} \cup \{S_{m1} \lrcorner S_{m2} \lrcorner \cdots \lrcorner S_{mn}\}$。

e) On **是基于本体的语义**: $On = \langle C, R, H^C, rel, A^O\rangle$。

其中: **概念层次** $H^C \subseteq C \times C$ **是一个有关向关系**, $H^C(C_1, C_2)$ **表示** C_1 **是** C_2 **的子概念**; **关系函数** $rel: R \rightarrow C \times C$; **公理集** A^O **是本体中的概念** C **和** R **必须满足的约束**。

f) **接口** $In \in S$, $f(In)$ **是一个服务组合流程映射** f: $f(In, On) \rightarrow Cm$。

g) Ar、Pe **表示服务组合流程采用的编排和排列的方式**:

Ar: $Cm(f)_1 \sum Cm(f)_2 \sum \cdots \sum Cm(f)_m$;

Pe: $Cm(f)_1 \prod Cm(f)_2 \prod \cdots \prod Cm(f)_n$。

h) T **是变迁集表示基本服务**。

$In \cap T = \varnothing$, $In \cup T \neq \varnothing$; $Cm \subseteq (In \times T) \cup (T \times In) \leftarrow Ar \cup Pe$ **是流程关系**。

i) **输出** Out: $Ar \cup Pe \rightarrow S'$。

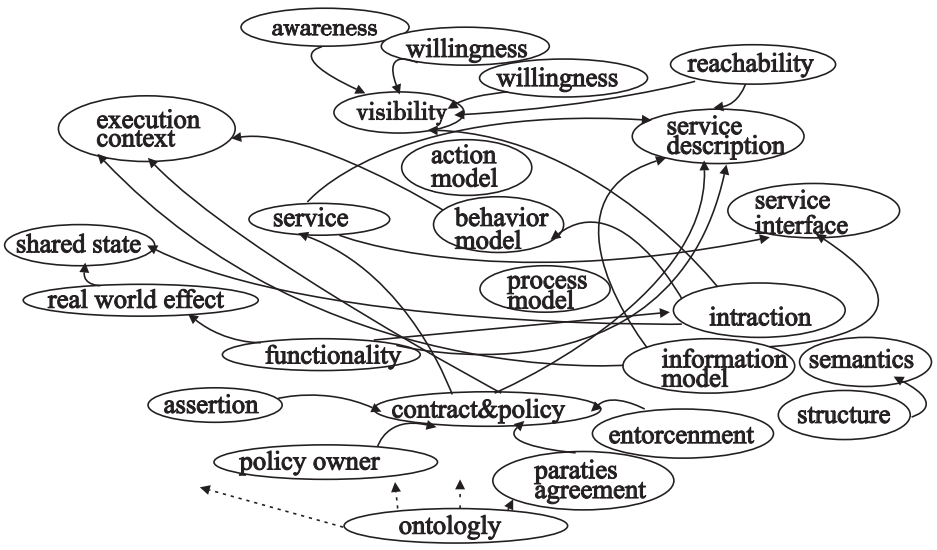

图1　SOA参考模型

2) SCA(service component architecture)　**它将业务流程从业务逻辑中分离出来,从而实现简化 SOA 构建的业务应用程序的创建和集成,且提供了构建由细粒度到粗粒度组装的组件机制,如图 2 所示,主要表现在**:

a) **简化业务组件开发**;

b) **简化作为服务网络构件的业务解决方案的组装部署**;

c) **提高可移植性、可复用性和灵活性**;

d) **通过屏蔽底层技术变更来保护业务逻辑资产**;

e) **满足 SDO 数据访问**;

f) **提高可测试性**。

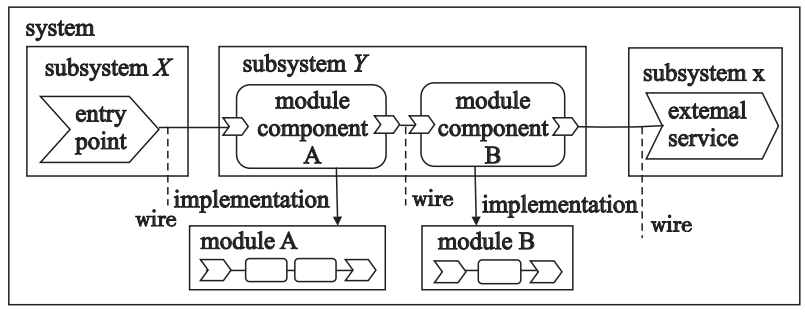

图2 服务组件体系结构

SCA **在构建面向服务的实体时,以实现提供服务和其他服务的组件和组装组件。在具体实施时,服务实现是业务逻辑的具体实现。**

定义 2　SCA **系统组件组装,** β **称满足如下条件的一个十一元组** $CSACA = \langle Ser, Qu, At, Ent, Out, Co, ScaIn, Scam, Subs, Scas, Conf\rangle$ **为一种 SCA 系统组件组装成应用实体。**

a) $Ser = \{S, AS, QS\}$, $AS = \{MAS, SAS\}$。Ser **由基本服务、外部服务(模块级外部服务和系统级外部服务)和引用服务组成,且** $QS \rightarrow \langle AS_{i-1} \langle AS_i\rangle$, $QS \subseteq S$, $AS \rightarrow key(AS)$。

AS **与 QS 合成**: $AS \quad QS = \{\langle x, y\rangle \mid \exists z(xASz \wedge zQSy)\}$。

QS **在** S **上的限制**: $QS \quad S = \{\langle x, y\rangle \mid xQSy \wedge x \in S\}$。

b) **引用** $Qu^{Ob} \rightarrow Binding(QS, AS)$,**通过外部服务的惟一性连接机制的目标。**

c) **属性** $At \rightarrow Set(Qu, Value)$。

d) **入口点** $Ent = \{Moet, Syent\}$,**且** $Moet \rightarrow Key_1(Ent)$, $Syent \rightarrow Key_2(Ent)$,**表示模块级和系统级入口点,均保持入口点的惟一性**, Declare: $AS \rightarrow AS(Ent)$。

e) **组件** $Co = \{Out, \langle QS, At\rangle, AS, CoIn, Sp\} \cup \{\langle componet_1(xml), \cdots, component_n(xml)\rangle\}$; **组件接口** $CoIn = \{CoIn_1, CoIn_2, \cdots, CoIn_m\}$,**组装** CoA: $Co(As) \rightarrow \langle Co, <\rangle$, < **表示组装执行偏序**。

f) **模块接口** $ScaIn = \{ScaIn_1, ScaIn_2, \cdots, ScaIn_n\}$。

g) SCA **模块** $Scam = \{Co, AS, Ent, Sp, ScaIn, m\}$,**模块组装方法** $m = \{m_1, m_2, m_3, m_4\}$ **指衔接各部分的机制**:

消息衔接: $m_1 \rightarrow \langle Scam, \alpha\rangle$;

任务衔接: $m_2 \rightarrow \langle Scam, \beta\rangle$;

语义衔接: $m_3 \rightarrow \langle Scam, \gamma\rangle$;

事件衔接: $m_4 \rightarrow \langle Scam, \lambda\rangle$。

h) **子系统** $Subs = \{Subs(Scam, Moet), Subs(component, Moet), Subs(co, CoIn), Subs(sp, In)\}$,

$Subs(Value) \rightarrow \langle Case, As\rangle$。

i) SCA **系统** $Scas = \{\{\langle Subs, Syent\rangle\} \cup \{\langle Scam, Moet\rangle\}, M\}$, M **是连接机制**。

j) **各阶段配置方式**: $Conf = \{Conf_1(CoA, f_1), Conf_2(Scam, f_2), Conf_3(Subs, f_3), Conf_4(Scas, f_4)\}$,**配置是一组 XML 文件,** f **为映射关系**。

3) SDO(service data objects) **与** DAS(data access service)　SDO **是一种数据编程体系结构和 API,可对异类数据源的统一数据访问,且提供多种不同种类数据的公共方法。目前的 SDO2.1 规范描述了静态和动态 API 创建 SDO 的两种方法,如图 3 所示,主要表现在**:

a) **简化 J2EE 数据编辑模型**;

b) 抽象 SOA 中的数据;

c) 统一数据应用程序的开发;

d) 支持和集成 XML;

e) 提供统一的数据访问框架。

SDO 由 SDO 编程模型和 API、data 中介服务、数据源、数据对象、数据图、变更摘要、属性、类型和序列组成。

DAS 可以采用任何持久化机制实现,允许采用标准方式将 SDO 保存到各种后端系统,从而提升企业应用程序内的互操作性和标准,如图 4 所示。

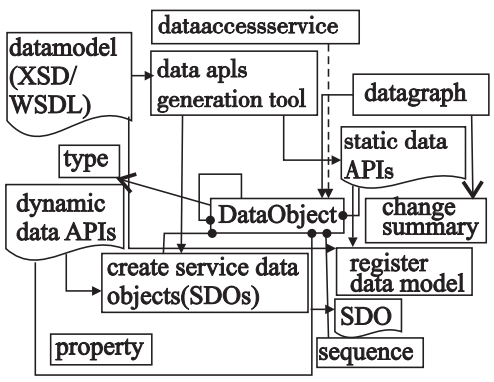

图3 SDO、静态与动态API用法

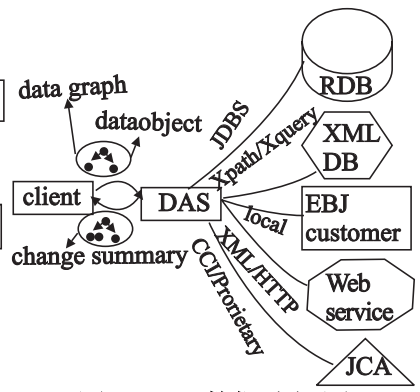

图4 DAS数据访问图

定义 3 数据访问, γ_0 称满足下列条件的一个十一元组

DA = 〈Sdom, Api, Ds, Dms, Das, Do, Dg, Ca, Att, Sdot, Dd〉

是一个网构软件数据访问的实现。

a) Sdom = { $sdom_1$, $sdom_2$, …, $sdom_k$} 为一个有限编程模型个体集。

b) SDO 数据 Api = { StaticApi, DynamicApi} 且 Api∈ Sdom, Api: Sdom→ π, π 表示 API 在 Sdom 上的连接运算方式。

c) 数据源 Ds = { Ds_1, Ds_2, …, Ds_r} ,表示数据源有限个体,且 Sdom 不访问 Ds。

d) 数据中介服务 *f*: Dms→ create(view) , change(*f*) → Ds (Das) 。

e) 数据对象 ∏ Do∈ Sdom, Public: *g*: Do→create(view) Λ *f*∧ Dms(*g*, Λ) , Λ 表示一种反射方法。

f) 数据图 dg: container(Dg) =〈 *V*, *E*〉, *V* ={ v_1, v_2, …, v_t} 且 $v_1 \cap v_2 \cap \cdots \cap v_t = \varnothing$, *E* ={ 〈$v_{i-1}$, v_i〉} , i∈ t_0。

Make(MDS) →dg, Update(dg) ⊗ ergodic(*V*) → Ds ′。其中: ⊗表示遍历修改树,更新数据源,并定义如下映射关系:

函数映射 *u*: Ds(Do) →dg;

XML 直接交换 *v*: Ds(Do) →dg[*f*(xml)] 。

g) 变更摘要 initial(Ca) = ∅, Ca∈ dg, return: Ca(dg) →Dms ↑ Dms→ Ds。

h) Do(Att, Sdot) →Dd, Att(Sdot) = { int, float, char, …} ∪ { date, string, …} 。其中: Dd 是访问序列; Att 是属性; Sdot 是类型。

命题 面向服务的网构件软件中间件的实现的服务、组装、数据访问由一个三元组可完成描述:

$$IWMW = \langle \alpha, \beta, \gamma \rangle$$

2 网构软件中间件体系结构

网构软件中的实体目前主要以构件形态存在、分布在 Internet 中,国内主要以北大青鸟的 ABC 方法,国外的 OMG 的 CMM 等方法加以简单的聚合和推导,而因网构软件固有的特征,这些方法很难满足具体的需求。但在文献[3] 提出了一个网构软件的基本层次结构,即控制层和目标层;并从体系结构的角度加以了研究。本文在此基本的层次结构上增加一个数据整合层,并结合 SOA 层次结构建立网构软件中间件的体系结构,使用 ESB 和具有 ESB 的服务交互点(SIP) 来实现网构软件的具体的模式。图 5 是网构中间件层次体系结构图。

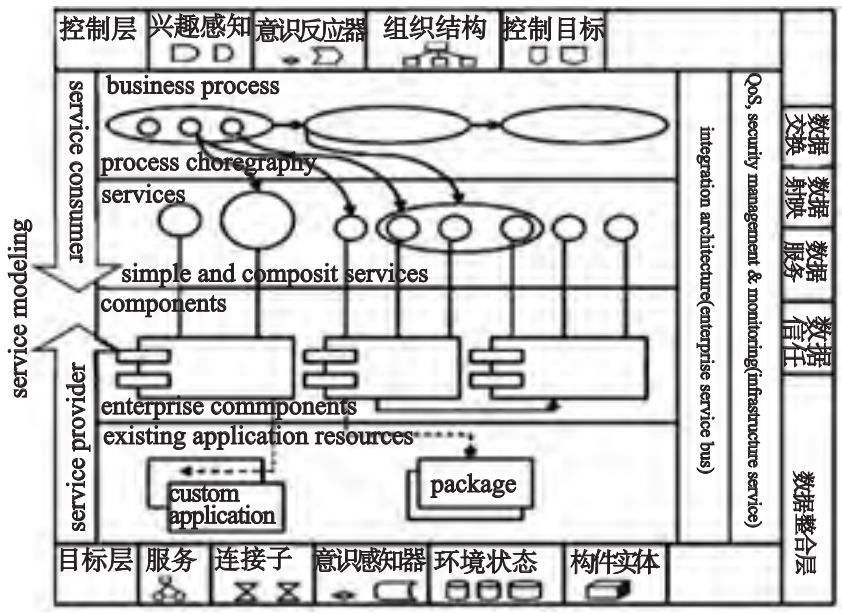

图5 网构软件中间件体系结构图

1) 企业服务总线(ESB)

提供了请求与响应、数据转换、基于内容的路由、各种优化、各种服务通用连接、虚拟化和监视等功能,同时还支持 SCA 简化模型。具体的 ESB 使用服务注册中心和存储组件作为动态查找机制来提供服务端点信息,如图 6 所示。

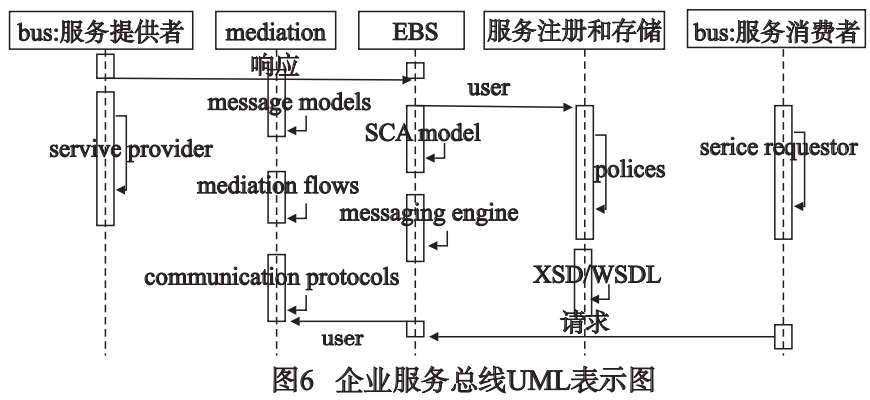

图6 企业服务总线UML表示图

2) 具有 ESB 的服务交互点(SIP)

1) 交互服务 使用门户(portlet) 或 WS-* 和 Web 技术,提供服务内容和交互数据的能力和功能。

2) 流程服务 根据业务流程和业务流,管理消息流和多个服务之间的交互的控制功能。

3) 信息服务 联合、整合、复制和转换不同数据源的功能。

4) 外部服务 将外部服务 EDI 和遗留系统集成到共同的企业体系结构中的功能。

5) 本地服务 业务应用程序服务调用服务使用者的功能。

6) 应用程序和数据访问数据 将核心应用程序与外部数据存储以及打包的应用程序进行集成的功能,如图 7 所示。

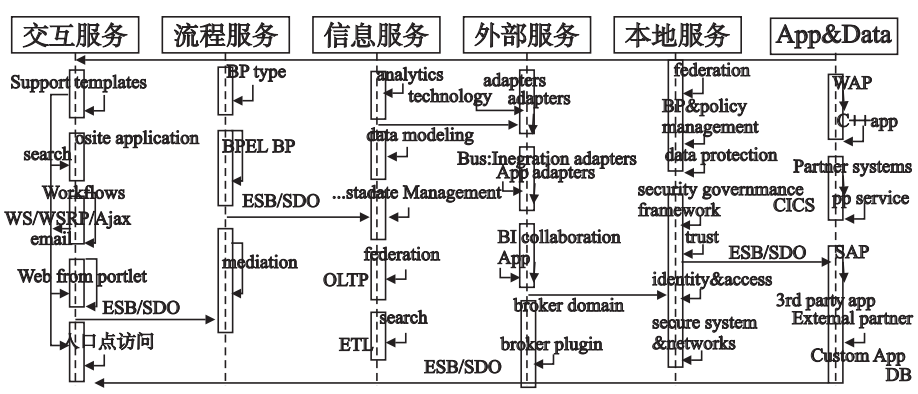

图7 具有ESB的SIP UML表示图

3 基于 Tuscany 的网构软件中间件应用

本文以一个远程教育集成系统(REIS) 为例来应用分析。

目前,远程教育已进入各个教育机构的各个层次,但在数据交换、资源共享、实时更新、按需要组装变化、访问效率等问题上一直面临各种问题,如对遗留系统集成不完整、资源存在局部性强等;致使远程迅速发展带来极大的阻碍。首先将系统分为资源系统(resource systems) 、业务系统(business systems) 、门户框架系统(protlet systems) 、工作流系统(workflows systems) 和智能计算系统(intelligence systems) 五大基础系统,接着使用本文的网构软件中间件体系结构进行设计,使用 SCA、SDO 等软件模型进行实践,并结合 Apache 下 SCA 系统的 Tuscany 开源包作为基础编程模型,采用 J2SE(JDK1.5 +) 作为开发支持平台,并且 Tuscany[9] 开源项目包含的组件实现类型有: Spring、Groovy、JavaScript 等,绑定类型有 Axis、CXF、AMQP、ActiveMQ、JXTA 等;数据绑定类型有 JAXB、SDO、XmlBeans 等;接口绑定类型有 WSDL、Java 等。图 8 是远程教育集成系统关系图。

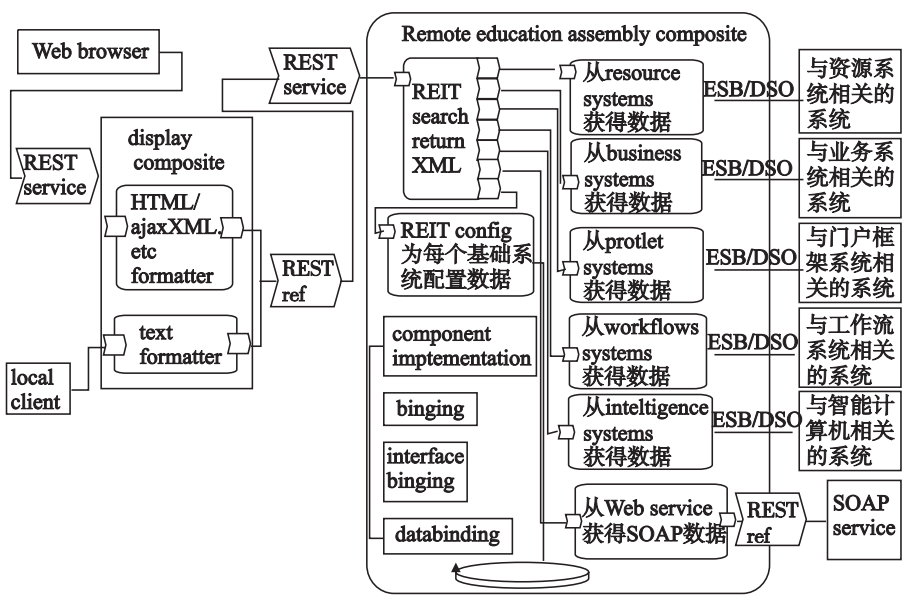

图8　REIS组成结构图

Display composite 提供所需的功能来将 REIT assembly composite 提供的 XML 转换为可识别的格式,如可在 Web 浏览器中显示的 HTML。组合包含以下组件:

a) HTML formatter 组件。从 REIT assembly composite 检索配置 XML 和最新返回 XML,然后基于此数据生成 HTML 表。可以从网页通过对公开此组件的 REST 服务进行 Ajax 调用来检索此 HTML。

b) Text formatter 组件。也从 REIT assembly composite 检索 XML 数据,并将其转换为可供人阅读的文本,供本地客户机访问。

c) REIT search 组件。基于 REIT config 组件提供的配置数据管理对各个基础系统的组件调用。基础系统组件所返回的 XML 数据聚合为单个 XML 文档,并使用 REST 服务绑定向客户机公开。

d) REIT config 组件。管理应用程序的配置,并保留相关应用系统的信息。

e) Web service 组件。调用指定的 Web 服务,并将返回的数据转换为简单的 XML 格式。

下面程序片段是 SCA 定义了一种 XML 语言服务组件定义语言(service component definition language, SCDL) 和 SDO (DAS) 请求配置方法。

```
〈composite xmlns = " http: //www. osoa. org/xmlns/sca/1. 0 " name = " REIT. reitsubbaseSystem" 〉
    〈service name = " reitsubbaseSystemService" 〉
    〈binding. rest/〉
    〈reference〉REITSearchComponent 〈/ reference〉
    〈/service〉
    〈component name = " REITSearchComponent" 〉
    〈implementation. python module = " REITSearchImpl" scope = " composite" /〉
    〈reference name = " REITConfigService" 〉REITConfigComponent 〈/ reference〉
    〈reference name = " resourceSystems Service" 〉ResourceSystemsComponent 〈/reference〉
    〈reference name = " businessSystems" 〉BusinessSystemsComponent 〈/reference〉
    …
    〈/component〉
    〈component name = " ResourceSystemsComponent" 〉
    〈implementation. python module = " ResourceSystemsImpl " scope = " composite" /〉
    〈/component〉
    〈component name = " BusinessSystemsComponent" 〉
    〈implementation. python module = " BusinessSystemsImpl" scope = " composite" /〉
    〈/component〉
    …
    〈component name = " REITConfigComponent" 〉
    〈implementation. python module = " REITConfigImpl" scope = " composite" /〉
    〈/component〉
    〈/composite〉
  〈Config〉
  〈ConnectionInfo dataSource = " java: comp/env/jdbc/bigbank" /〉
  〈Command name = " getAllResource" SQL = " select * from ResourceList" kind = " Select" /〉
  〈Command name = " getResourceType" SQL = " SELECT * FROM Resourceaccounts where ResourceDetail = ?" kind = " Select" /〉
  〈/ Config〉
```

4　结束语

使用面向服务的方法建立了一种网构软件中间件的具体实现过程,其首先分析了网构软件中间件的理论和技术,主要研究了 SOA 中服务组合、SCA 系统组装、SDO(DAS) 数据访问的方法;接着建立了网构软件的体系结构和内部实现机理;最后结合 SCA 系统的 Tuscany 开源件在一个远程教育集成系统中进行应用,且详细介绍了应用的具体过程。该中间件方法的顺利应用将从很大程度上解决现阶段资源共享、构件复用和系统集成。

参考文献:

[1] 杨芙清. 软件工程技术发展思索[J]. 软件学报, 2005, 16(1): 1-7.

[2] 杨芙清, 梅宏, 吕建, 等. 浅论软件技术发展[J]. 电子学报, 2002, 30(12A): 1901-1906.

[3] 吕建, 马晓星, 陶先平, 等. 网构软件的研究与进展[J]. 中国科学 E 辑: 信息科学, 2006, 36(10): 1037-1080.

[4] 梅宏, 黄罡, 赵海燕, 等. 一种以软件体系结构为中心的网构软件开发方法[J]. 中国科学 E 辑: 信息科学, 2006, 36(10): 1100-1126.

[5] SCA service component architecture-assembly model specification [EB/OL]. (2005-11). http: //download. boulder. ibm. com/ibmdl/pub/software/dw/specs/ws-sca/SCA_AssemblyModel_V09. pdf.

[6] Service data objects for Java specification[EB/OL]. [2006-11]. http: //www. ibm. com/developerworks/library/specification/ws-sdo.

[7] OASIS. Reference model for service oriented architecture 1.0 [S/OL]. [2006-12] http: //docs. oasis-open. org/soa-rm/v1. 0.

[8] 曹宝香, 刘阳. 基于中间件的企业计算模型[J]. 计算机应用研究, 2007, 24(2): 69-72.

[9] tuscany[EB/OL]. (2003-2007). http: //incubator. apache. org/tuscany.