分布应用集成核心技术研究综述

徐 罡 黄 涛 刘绍华 叶 丹

(中国科学院软件研究所软件工程技术中心 北京 100080)

摘 要 分布应用集成技术已成为构建大型信息系统,特别是 SOA 的关键技术.该文在分析分布应用集成的基本特征和集成层次的基础上,阐述了其关键问题,包括数据多样性、传输载体和应用访问、协同管理、可重配和自适应、建模和形式化,进一步论述了分布应用集成与 Grid,SOA 及 B2B 等技术的联系与区别.从不同的角度对这些问题作了全面的概述,既分析了存在的问题,又论述了已有技术在处理这些问题上的优势和不足并探讨了相关的新兴技术,展望了未来发展方向.

关键词 分布应用集成;中间件;面向服务的体系;B2B;网格

中图法分类号 TP311

Survey on the Core Techniques of Distributed Application Integration

XU Gang HUANG Tao LIU Shao-Hua YE Dan

(*Technology Center of Software Engineering, Institute of Software, Chinese Academy of Sciences, Beijing* 100080)

Abstract Distributed Application Integration(DAI) techniques are becoming the key enabling techniques to construct large information systems. But the researches about DAI mainly exist in industry, not in academy. The fact results in the shortage of consistency and integrality in using and expressing DAI's related techniques. In order to enhance the consistency and integrality in DAI, the authors summarized those basic characteristics of DAI and furthermore divide the integration levels of DAI to help categorizing the related techniques. On the base of the analyzed characteristics and levels, the key problems of DAI are presented, including Data Diversity, Transferring Carrier, Application Access, Collaboration Management, Reconfiguration and Adaptation, Modeling and Formalization, and so on. Furthermore, the existing and new emerging techniques to resolve those problems are presented and compared to point out the advantage and disadvantage of each other. Those techniques mainly come from four domains, namely software architecture, distribution computation, component techniques and middleware. Simultaneously, the relationships between DAI and Grid, Web Service, B2B are discussed. Although the premises and techniques between them are different, the trend of the complementing and fusing is obvious. How to efficiently utilizing and referencing each other's techniques are analyzed and presented. At last, the future developments of DAI are discussed.

Keywords distributed application integration; middleware; service-oriented architecture; B2B; grid

收稿日期:2004-12-05;修改稿收到日期:2005-01-25.本课题得到国家"八六三"高技术研究发展计划项目基金(2001AA113010,2002AA413610,2003AA413010,2003AA115440)和国家"九七三"重点基础研究发展规划项目基金(2002CB312005)资助.**徐 罡**,男,1973 年生,博士,主要研究领域为软件工程、分布式计算、企业应用集成、面向服务的体系.E-mial: xugang@otcaix.iscas.ac.cn.**黄 涛**,男,1965 年生,博士,研究员,博士生导师,主要研究领域为软件工程、分布式计算.**刘绍华**,男,1976 年生,博士研究生,助理研究员,主要研究领域为分布式计算、工作流技术、服务协作.**叶 丹**,女,1971 年生,博士,副研究员,主要研究领域为软件工程、分布式计算、虚拟企业.

1 引 言

近年来,软件系统的开发方法正由面向功能的开发方法转向面向服务的、面向流程的开发方法;由基于模块的构造方法转向基于组件的构造方法.而这些面向服务和流程的、基于组件的开发方法都是以分布应用集成为基础的.在现实中,而又往往忽略对分布应用集成技术的研究,例如,在面向流程的方法中的工作流管理系统,它通常包括过程建模、工作流机及过程监控系统,而工作流管理系统要实际运行首先需要有效地集成低层的信息系统;在面向服务的应用示范中,独立的、来自不同开发商的系统或COTS(Commercial Off-The-Shelf)产品被视为可以动态共享的服务,使用这些共享的服务同样要求能够屏蔽系统与服务间的差异;在基于组件的方法中,不同的组件采用不同的打包方式、通信方式,同样需要分布应用集成技术来屏蔽这些差异性.此外,构成e-Business的两大关键技术也是接口和集成,而目前e-Business的绝大多数研究都集中在接口而忽略了对集成的研究.在对网格技术的研究中,也偏重于对资源协调的研究,而忽略了对不同资源的集成机制的研究.

从分布应用集成技术的发展来看,产业界对分布应用集成技术的研究要比学术界更加深入,并推出了多种产品和标准,如各种对象请求代理、集成代理、集成中间件.分布应用集成技术的原动力主要是来自于实际分布式集成系统的开发及集成遗留系统,工业界的经验和方法需要归纳总结并提升为理论.从分布应用集成的核心技术来看,很多关键问题还有待于进一步地解决和规范,特别是随着新环境、新需求的产生,还有更广阔的研究空间.

本文首先概述了分布应用集成的基本构成和特征,接着分析了当前分布应用集成核心技术,在此基础上引出分布应用集成的关键问题及其解决方案,然后论述了分布应用集成技术的新发展,并展望了未来的工作.

2 分布应用集成的基本概念

分布应用集成的目标是无约束地连接分布的应用程序,并实现应用程序间的数据和功能的共享,这种共享是以不对应用程序本身作大的修改为前提条件的.因此,具有在对应用程序代码和数据结构不做大的修改的前提下,集成其它应用程序的能力是分布应用集成技术的基本特征.

分布应用集成的技术和方法涉及的范围较广,主要集中在分布式系统、组件方法、中间件平台、软件体系结构四个方面.分布应用集成在借鉴了以上四方面方法的同时,其解决问题的重点又与上述研究领域不同.

对组件方法的研究集中在组件的接口层次,包括接口命名、输入/输出参数类型及交互协议.中间件平台集中在对组件运行和交互的基础架构研究,提供组件运行容器(container 或 dock)及附加的服务和通信中间件.文献[1]列出了多种基于连接器的软件体系风格,软件体系风格描述体系结构及构成软件体系结构的组件交互行为,基于连接器的软件体系风格包括管道体系风格、对象抽象体系风格、通信过程体系风格等.分布式系统通常更注重对系统的性能、可扩展性、可靠性、适应性的研究,此外,分布式系统还解决并发控制、事务等问题.

对分布应用集成而言,首先它本身集成的是独立的应用系统,具有更大的集成粒度.其次,复杂的分布应用集成需要集成中间件作为传递数据和控制的载体,它不像组件方法处于同一命名空间.集成的应用系统具有更多的多样性,例如,依据软件体系结构来说,应用系统所采用的软件体系结构各不相同,其使用的连接器不同,其对外所提供的集成点也各不相同,并且在一般的集成场景中,对应用系统的体系结构往往是未知的,所有这些都增加了分布应用集成的困难度.面对不同的集成问题,分布应用集成所采用的方法也有所不同,通常可以将集成问题划分为多个层次,不同层次的集成问题采用不同的集成方法和不同的集成组件.图1描述了分布应用集成的相关问题和方法.从较高的抽象层次来看,分布应用集成主要解决数据转换、通信连接、访问介入三大问题.

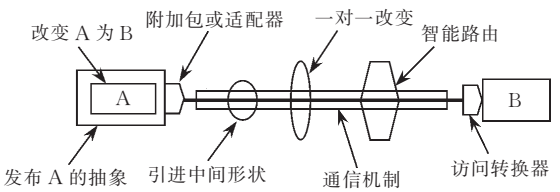

图1 分布应用集成技术和问题

图2给出了分布应用集成通常采用的拓扑结构.图2(a)为简单的应用系统集成方式,图2(b)为较复杂的应用系统集成方式.在简单的应用系统集成方式中,采用直连的方法集成两个应用系统,这种

方式适于应用系统不多的情况，具有很好的可集成性，其缺点是整个系统柔性差、耦合性高. 在复杂的应用系统集成方式中，通常需要集成多个应用系统，使用集成中间件完成其中的绝大部分集成工作，对应用系统的集成能力要求不高，整个系统柔性好、耦合性低，其缺点是较为复杂.

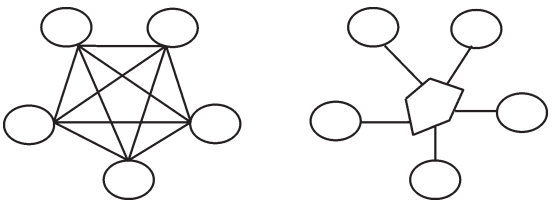

图 2　分布应用集成的拓扑结构

3　分布应用集成的集成层次

对于分布应用集成来说，由于集成的是独立的信息系统，而各个信息系统(特别是遗留系统)往往解决特定的领域问题，因此导致所采用的软件体系结构、实现语言、对外提供的集成点(集成点是指外部系统通过它能够进入应用系统从而能够访问数据和调用功能)及交互协议各不相同，这就导致了分布应用集成的复杂性. 解决分布应用集成的复杂性问题，需要将集成问题划分为多个层次，不同层次的集成问题采用不同的集成方法. 我们依据集成点的不同，通常将集成层次从低层到高层分为传输机制、数据集成、接口集成、过程集成.

3.1　传输机制层

传输机制层是分布应用集成的基本层次，简单地说，它提供在两个或多个集成点间连接和移动数据的传输渠道. 分布应用集成可以利用建立在协议层上的两个系统共同定义的契约(或协议)来交换数据或利用更复杂的传输机制，如消息队列等. 传输机制层的前提是在传输层上连接多个系统，其方法包括 IP，FTP 及特定的通信中间件(如 Message Queue，Event Middleware). 例如，在 B2B 中，采用 HTTP 协议封装 SOAP 协议作为传输机制，HTTP 本身就是在 TCP/IP 连接基础上建立的一种请求/应答通信方式.

分布应用集成的传输机制可分为同步和异步两种类型. 同步传输要求通信进行前建立和维护通信渠道，在这种模式下，数据传输是与之相适应的请求/应答模式，这种模式符合实时通信对接收者和发送者始终是可提供的要求. 在传输机制层，使用同步

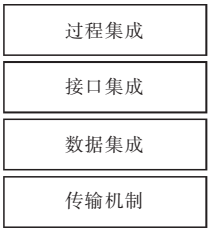

图 3　分布应用集成的层次

通信的典型技术包括事件中间件、HTTP、SOAP、TCP/IP、FTP，此外又包括接口集成层中的 RPC，RMI. 而异步传输机制允许信息的发送者和接收者间的松耦合，发送者可以不间断地发送消息而没有堵塞，接收者在另一端间断地接收消息，这种能力适用于发送者与接收者间断的连接. 典型的异步传输机制是消息中间件，消息队列作为集成系统的中介者，与同步传输机制不同的是消息队列更适合多对多的集成，并且能够管理会话、路由、审核数据流等功能，而同步传输机制多是实现点对点的集成.

3.2　数据集成层

数据集成层是建立在传输机制层之上，超越了只是简单地在两点间传输数据，数据传输和转化只是数据集成层的基本功能. 此外，要成功地实现两个系统间的数据集成，还必须解决应用系统访问、应用句法和应用语义三方面的问题.

在数据集成层中，应用系统访问是指能够从应用系统的数据存储中抽取和插入数据的能力，同时也包括抽取元数据的能力. 抽取和插入数据依赖于应用系统的特性和数据访问方法的不同而不同. 例如，SAP-ERP 系统通过 RFC 提供对数据的访问，数据库管理系统可以采用 API 和 SQL 语句两种方式提供对数据的访问，而绝大部分主机系统(mainframe)则不提供对数据的访问.

应用句法是指应用系统特定的数据结构，不同的应用系统具有特定的应用数据结构，有效地实现数据集成需要改变应用句法. 例如，集成基于 Web 的应用系统与 OS/390 系统就要求将 XML 文档转换为特定的 ADABAS 数据.

应用语义对于复杂的分布应用集成是必要的，意味着当源数据从一个应用系统到另一个应用系统进行转换和传递时，数据应用的上下文环境必须考虑. 例如，在 ERP 系统中的订单系统与客户关系系统的集成，订单系统中的客户信息要比客户关系系统中的客户信息相对简单，因此客户信息在向客户关系系统转化时需要补起必要的信息，如需要补足客户的级别、订货次数等信息.

数据集成层的目标是解决上述三个主要问题，采用的技术可以归纳为数据集成适配器、数据转换框架、数据改变服务及数据集成规则 4 方面. 数据集成适配器的功能是完成对应用系统的访问，适配器可以利用多种方式来实现访问，包括直接访问数据库、应用系统 API 及其它非传统方法，如面向主机系统的屏幕刮擦方法(screen scraping). 数据转换框架实现数据句法的转换，包括数据类型间的转换、文本文档间的转换. 数据改变服务实现格式转换功能，也被称为格式转换器，如改变服务通常对 N 个输入消息重新格式化为 M 个输出消息. 严格来说，数据改变服务要比格式转换更宽广，它还可以包括操作计算，例如，除格式转换外，改变服务还可以包含从一个消息中抽取一个元素，对其操作并将其写入新格式的消息中的操作. 在数据集成层中，数据集成规则正确地解释应用数据的语义，数据规则与业务规则不同，业务规则通常被规则引擎使用，其本身更加详细、更富规则特征，而数据集成规则则倾向于简单、基于条件的，多采用布尔逻辑序列进行处理，可以说数据集成规则提供了数据转换条件和映射参数.

数据集成代理是在数据集成层的一种集成中间件，它将数据集成层的基本功能结合为一个统一的单元，并使用图形界面来配置和驱动.

3.3 接口集成层

目前的分布应用系统通常由包括表示层、业务逻辑层和数据层的三层结构构成. 接口集成层主要是针对业务逻辑层而来的，允许应用系统间的业务逻辑共享. 从另一个角度来看，目前分布应用系统更多地采用组件结构，因此这一层也称为组件集成层，组件外部化其接口作为集成点. 接口集成的核心是使用分布组件封装应用系统的业务逻辑，通过远程方法调用业务逻辑.

EJB，CORBA 和 COM＋是目前支持接口集成的三种技术. EJB 是 SUN 公司的服务器端的组件模型，它允许开发、部署分布对象，EJB 运行在应用服务器的 EJB 容器中，并有丰富的支持服务，如事务管理、队列组件. CORBA 是 OMG 的分布对象说明，提供 Object Bus 允许组件动态方法调用. COM＋组件只限于 Windows 平台，这一点也限制了在分布应用集成中的应用. 在互操作方面，目前 IIOP 协议能够支持 EJB 和 CORBA 组件的互操作.

3.4 过程集成层

过程集成层是集成的较高层次，其实现面向过程的集成. 过程集成的对象不是物理实体(如数据和组件)而是过程实体，通常表示为逻辑实体. 更具体地说，过程集成的是由活动驱动的业务过程，而不是由应用数据传输和转换来驱动的，被连接的逻辑实体不是依据信息定义，而是依据活动或工作流. 过程集成层的逻辑表示需要映射到物理实体. BPM 是过程集成管理工具，由多个组件构成，每个组件执行过程集成的不同方面，通称包括过程建模、过程管理和过程流引擎. 其中，过程建模是建立图形化的业务过程表示；过程管理监控过程流、过程事件及业务规则；过程流引擎管理过程的运行. 过程集成是以服务集成、接口集成、数据集成为基础的，综合运用这些低层的集成技术，如支持 Web Service 相关的协议、组件客户端的 Stub 及数据集成适配器等.

4 分布应用集成的基本问题

分布应用集成是解决地域上分布的、同构的或异构的应用程序间交互的问题. 造成分布应用集成的困难主要是由于应用系统间的不匹配，而这种不匹配问题分布在应用系统的多个方面. 文献[2]使用“重用一个 C 语言的模块，该模块使用事件声明作为交互方法，而不是使用过程调用”这么一段话来说明系统不匹配的普遍性. 应用系统的这种不匹配性在某种程度上来说是不可避免的，减少组件不匹配的一个策略是遵循特定的组件打包标准，从而达到“即插即用”的适配，但还是有多个标准并存，如 CORBA，EJB，COM＋，Java class library 等等. 遵循一种标准的组件很难实现与遵循另一标准的系统相集成. 同样，在软件体系结构研究中，认为好的软件系统设计应该选择最适合系统的体系风格，而每种风格暗示着至少采用不同交互机制，因此也很难实现互操作.

层次化的分布应用集成方案能够为分布应用集成问题提供一个层次框架，并试图在不同层次解决这些基本问题，而每一层中都隐含了共性的基本问题. 本节是从另一个角度，归纳总结这些基本问题及相应的解决方法.

4.1 数据多样性问题

分布应用集成是集成多个异构的应用系统，最基本的问题就是要达到数据共享，这也是数据集成层的主要任务. 达到数据共享，从微观上来讲意味着分布应用系统共享相同的数据类型，系统并能够在数据项的类型上达成一致，或者说分布应用系统能

够在数据项的位层次(bit-level)上达成一致，如 IEEE 浮点数据类型标准. 从宏观上来讲，数据共享意味着能够在数据的格式、句法及语义三方面达到一致.

分布应用系统所采用的数据存储格式通常采用数据库和文件格式. 采用数据库的方法能够实现应用系统的业务逻辑与数据相分离，使系统具有更好的层次性、可扩展性、可集成性，但缺点是绕过应用系统的业务逻辑，通过数据库直接访问应用系统的数据，容易破坏数据的完整性，造成脏数据. 目前，很多数据库管理系统已采用 API 方式或对外输出中间文档的方式提供对数据库的访问. 仅仅这样也是不够的，为了避免破坏数据的完整性，通常的办法是在应用程序端开发一个组件(或者说是中间件)来包含涉及访问数据的业务逻辑，保证数据的完整性. 例如，修改客户关系数据库中的某个表格中的字段，可能涉及该数据库中的多个表格中数据的修改.

基于数据库的分布应用集成，需要能够访问、提取数据，并且在此基础上对数据进行转换. 对数据库中的数据进行转换需要知道该数据的元数据，转换内容包括数据格式、数据句法及语义三方面. 数据库中的数据通常是关系型的，对象数据库也是实现对象与关系型数据的映射，然后存放在关系型数据库中. 对于基于数据库的应用系统，集成的应用程序可以直接使用数据库中的数据，而对于基于文件的、基于对象的应用系统，在使用数据之前需要将其转化为数据文件或数据对象.

基于文件的分布应用集成是以文件作为数据交换的基础. 文件可以是结构化的和非结构化的. 集成非结构化的文件需要描述该文件的元数据，并通常是通过硬代码写入程序中，很难达到集成的动态性. XML 的出现可以说给分布应用集成领域带来巨大的改变，它在一定程度上屏蔽了分布应用集成的异构性. XML 是一种封装了数据和元数据的结构化文档，这种封装能力允许不同的应用系统和数据库在彼此不知道数据格式的情况下交换信息，可以说 XML 为分布应用系统的信息交换提供了公共机制. XML 在分布应用集成中优势是明显的，但对于解决复杂的分布应用集成问题也有其局限性. XML 的语义说明是有限的，XML 能够使用特殊的标记描述特定的实体或行为，而 XML 文档本身并不包含对标记语义的说明，对于分布集成的双方需要明确 XML 标记和数据的意义. XML 同样需要数据转换和映射，不同应用系统所规定的 XML 格式不同. 在分布应用集成中，XML 只是应用系统信息格式之一，对数据转化的需求仍然是挑战之一. 此外，分布应用集成需要更复杂地分裂合并多个数据源. XML 是基于文本的文档，因此使用 XML 作为传输数据具有低效率的特点. 此外，XML 在本质上不具备基于内容的路由能力，而基于内容的路由和规则是动态地、自动地管理集成业务过程的核心.

除在数据层的主要问题是解决数据多样性问题，其它层次，如界面层、通信层、过程层同样需要解决数据多样性的问题. 正是由于这种特性，数据多样性问题的解决通常不作为一个独立的中间件存在，而是隐含在各种集成代理或者适配器中.

分布应用集成对数据多样性处理的理想状态是能够达到动态性，即能够动态地对数据及元数据加载和解析，这也是实现动态集成的基础. 在文献[3]中，提出了类型对象模型(Typed Object Model, TOM)描述数据格式的抽象结构、具体表示及关系，并试图包含已有的数据格式、复杂数据类型，作为数据转换的中介，利用分布的调节器代理(mediator agents)解决 Web 应用中的数据多样性. OEM 也采用中间数据格式的方法解决数据多种性的问题，但与 TOM 模型相比依赖于单一的中间格式，更缺少柔性[8]. 基于规则的系统是解决句法和语义多样性的基本技术. 此外，基于本体的方法为解决数据句法和语义的多样性提供了新的机遇，同义词、概念层次和映射函数等本体论的方法目前也尝试着应用于数据集成，其明显的不足是要维护一个全局的本体，相应的成本较高.

4.2　分布应用集成中的通信问题

分布应用集成是集成在地域上分布的多个应用系统，实现有效的集成需要在应用系统间传递数据流和控制流，而数据流和控制流的传输需要有效的载体. 对于非分布应用系统来说，传统的方法包括共享变量、共享内存、环境变量、数据库、过程调用、对象消息等，提供应用程序内部的数据和控制的交换. 分布应用集成所需要的载体与应用程序内部的载体有所不同，它是一个相对独立的实体，需要满足系统分布特性的需求，如系统对耦合性、实时性、多对多、点对点等空间、时间、拓扑特性的需求[4]. 对于分布式应用集成来说，通常采用消息中间件、事件中间件、元组空间、RPC、RMI、SOAP 等方法作为分布应用集成的载体，实现异地的数据流和控制流的传输.

消息中间件(或者称为消息系统)能够支持不同的编程语言，支持异构平台上的、异地系统间的通信，系统间的信息以离散的消息方式传递，每个消息

都是自包含的,能够独立地路由和处理.消息中间件采用队列作为消息的存储机制,包括暂时性队列和永久性队列两种类型,通过队列机制实现应用系统间的异步通信,提供可靠传输能力,保证消息单次可达,同时支持系统间的低耦合,即对于整个集成系统来说,增加新的应用程序或删除旧的应用程序不会对整个系统造成大的影响,在一定程度上实现了集成的柔性.此外,消息中间件中还可以包含消息代理,解决不同应用程序间的消息差异,并且有的消息代理支持定义消息处理规则,使其具有一定的处理业务逻辑的功能,例如根据业务逻辑定义消息的路由.对于消息的订阅和发布,通常采用主题和渠道两种方式,基于主题的方式,可以实现消息与应用的一对一和一对多的路由,基于渠道的方式,可以实现消息与应用的多对一的路由[5].

事件中间件与消息中间件相类似,也是一种典型的单向(one-way)通信方法.与消息中间件所不同的是事件中间件的事件产生机制,事件是由状态的改变、特定时间的到达、过程的调用所触发的.事件中间件一般不提供事件的暂存机制,它本身是一种实时中间件.事件中间件对事件的路由采用基于事件与事件的订阅者之间匹配的多点广播(multi-cast)方式[6,7].事件中间件,也称为隐含唤起(implicit invocation),是一种较消息中间件更能很好地支持低耦合特性的集成中间件,但不具备消息中间件的可靠特性[8].目前,也有很多消息中间件包含了事件中间件的部分功能,或者将两者捆绑在一起,但这样会导致失去事件中间件的实时特性.

元组空间(tuple space)通信,也称为 Generative Communication,最早是由 Gelernter 在 Linda 语言中引入的作为一种基础的通信模型,目前,被引入到分布应用集成中作为移动计算的一种通信方式[9].在分布应用集成中,独立的系统可以通过元组空间作为彼此之间进行通信的中介.发送的请求信息中的变量列表形成了一个元组,发送应用程序将元组插入元组空间,每个元组在元组空间中都是唯一的、独立的.接受者同样在它的元组获取请求中给出它自己的变量列表.在元组空间中,有与请求元组相匹配的元组时,获取发生,否则,请求者将等待直到元组提供.在分布应用集成中,分布的应用系统可以通过这种机制达到系统间的数据与控制的交互[10].

远程过程调用(或者说远程方法调用)主要是传递控制流的方法,同时使用参数来传递数据.客户端和服务器端分别使用 stub 和 Skeleton 作为代理,完成参数序列化及对象定位功能.这些远程对象的调用方法基本上是一种实时的集成通信方式,目前在这种强耦合的调用方式的基础上,扩展了低耦合的单向(one-way)方法及回叫(callback)机制[11].SOAP 是一种面向 Web 服务的简单对象访问协议,能够附加 XML 文档作为处理对象,并可打包在 Http 协议中,支持请求/应答模式.

在分布应用集成中,消息中间件一般在消息代理中使用别名机制来路由输入的消息;事件中间件使用主题或渠道来分类事件,事件发布者给每个事件标记特定渠道或主题名称,事件订阅者接受渠道中的所有事件或特定主题的事件.此外,基于内容的事件中间件使用谓词匹配事件内容,它本身既保持了集成的低耦合性,又在柔性和表达能力方面具有更小的粒度,这种方式更能满足企业应用集成的需求.例如,在订单处理流程中,可以依据订单中任何信息项(如 $AmountMoney>1000$)选择不同的处理流程.

4.3 应用系统的访问问题

应用系统通常采用最适合其问题域的软件体系风格,而每一种软件体系风格意味着系统采用不同的对外交互机制,即提供访问系统的机制不同.对于分布应用集成来说,应用系统的这种交互的多样性是导致应用系统相互集成的困难之一.

屏蔽交互机制多样性的典型技术包括数据网关(data gateway)、包(wrapper)及适配器(adapter)技术.数据网关是一种集成多种数据库的中间件,它可以屏蔽多种 SQL 语言的差异性及不同数据库访问方式的差异.包技术是对采用不同打包方式的组件进行二次打包,目前比较流行的组件打包方式包括 Corba,COM/ActiveX,JavaBean,Java class libraries,Microsoft Visual Basic controls 等.文献[12,13]从软件体系描述语言(Archtecture Description Language,ADL)的角度,将系统分为组件和连接器,组件具有不同的端口,连接器具有不同的角色,此基础上采用模板的方法对二次打包技术进行了分类,包括在线桥、离线桥、包裹器、中间表示器、调节器、单边/双边协商器及组件扩展器共八种二次打包方式,屏蔽不同打包技术间的差异.

包可以说是适配器的原型,适配器是在包的基础上的功能扩展.适配器,或者称为应用集成适配器,定义为一个软件服务的集合,它通过特定的 API、数据库访问或其它集成点连接和进入应用程序数据,从而达到集成应用程序的目的.依据应用集

成适配器主要功能的不同可以将适配器分为 9 种基本类型，分别为文件适配器、查询适配器、事件适配器、消息适配器、远程调用适配器、直连适配器、星型适配器、会话适配器、服务适配器.

其中，文件适配器限定于传输的只是文件，而不包含控制流，文件适配器的特点是一般不需要特定的连接器支持，只是简单地使用系统或应用本身的 Socket 或 Stream. 查询适配器是与数据库建立连接，返回查询结果，其本质上是一种“拉”的方式，也可以利用数据库中的触发器构造一种“推”的方式. 事件适配器、消息适配器和远程调用适配器是针对不同的交互机制构建的适配器，它们可以为事件、消息及远程方法调用系统实现一个基于事件、消息及远程调用的接口机制，包装为一个事件、消息及远程调用系统. 此外，对适配器的研究还包括对适配器集成模式的研究，适配器的集成模式是指面向特定的集成场景，能够反复多次解决该场景下出现的集成适配问题. 目前，人们已经提出了 7 种适配器集成模式，分别为数据库访问模式、文档交换模式、远程方法调用模式、消息模式、事件模式、事务代理模式、用户界面协议模式. 对集成适配器的类型和集成模式的研究有助于对不同集成适配器进行辨别，明确其特性，对不同的集成场景，选择使用适配器集成模式来解决特定的应用系统访问问题. 并且，对适配器类型和集成模式的研究也在随着应用场景的复杂化及新的需求的引入而不断地扩展新的类型和模式.

4.4　分布应用集成的协同管理

系统间的集成是基础，其更高的层次是在集成的基础上实现协同管理. 图 4 给出了一个分布应用集成协同和管理的框架. 其中，协同层包含必要的功能来协同多个系统实现协同的问题解决和提供整体的服务. 协同功能控制各个集成系统的执行，管理各个集成系统间的数据流；监控功能订阅各个集成系统产生的事件和信息，并产生更高层次的事件和信息；QoS 包括成本、性能、安全、可靠性、可扩展性等信息. 管理层提供管理服务，要求集成系统提供方管理系统运行平台、应用系统的布署及商业契约等服务[14,15].

就分布应用集成层来讲，低层的数据层和功能层的集成方法较缺乏协同管理机制，通常只是在集成代理中包含低级的协调调度逻辑. 过程层较好，特别是工作流系统也已扩展为一个协同模型，其具有良好的协同性能. 另外，网格技术及 SOA 技术是与分布应用集成协同管理密切相关的两个不同的研究方向. 网格技术的目标是协同多种资源解决复杂的科学问题，而 SOA 作为一种新的集成模型，在协同方面更注重协同服务组合.

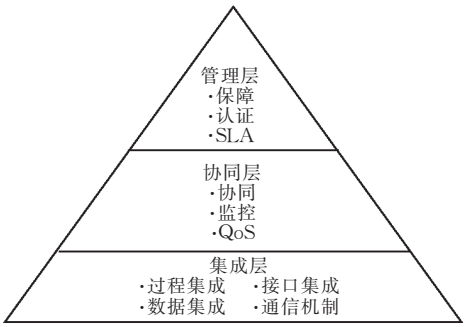

图 4　分布应用集成的协同管理

5　分布应用集成与网格技术

网格产生的目的与分布应用集成有所不同，它最初用于先进科学和工程的分布计算基础架构. “网格问题”定义为在动态的个体、机构和资源的集合中柔性地、安全地协调资源共享，集中反映为虚拟组织的概念[16,17].

网格概念下的问题是动态的、跨多组织的虚拟组织内的协调资源共享和问题解决. 网格中的这种共享是基于高度控制的，资源的提供者和消费者决定共享什么资源、允许谁共享及在什么条件下产生共享，这些具有特定共享规则的个体和机构的集合就形成了虚拟组织. 对于这种高柔性的共享关系，网格覆盖从 client/server 到 peer-to-peer 的关系；对如何使用共享资源的控制程度，网格提供小粒度的、多方访问控制机制、授权机制、本地和全局政策；对于使用模式，包括从单用户到多用户，从性能敏感到成本敏感，因此包括关于 QoS、调度、协同分配、审计问题. 在建立这种可扩展的虚拟组织过程中，网格的典型协同技术包括资源管理协议和服务，支持远程访问计算资源和数据资源并协调多资源的分配；信息查询协议和服务提供关于资源、组织及服务的配置和状态信息[18~20].

在网格中，主要采用通过定义标准协议的方法作为互操作和协同的基础，这些协议定义了用于资源共享的协商、建立及调度的标准信息格式和服务接口. 而这些协议本身也由于网格集成资源的多样性，需要不同的载体，如通过消息方式、RPC 方式等等. 因此，可以说分布应用集成是实现网格的关键技术之一，特别是实现网格互操作的关键技术，网格在分布应用集成的基础上通过多种标准协议来实现跨

组织的协调资源共享.

从某种角度来讲,网格重在解决分布应用集成的上层问题,协同调度跨组织的资源共享和问题解决.而在网格的体系框架中,这样动态的、柔性的跨组织的协调资源共享,需要底层的 Fabric 层资源及 connect 层的支持,如允许查询低层资源的状态信息才能够实现顶层的多资源的协同调度,因此它所达到的动态、柔性的协调资源共享,是以各个独立资源遵循共同的标准协议为前提的,对集成的资源要求较高,也是以丧失一定的可扩展性为代价的.从某种程度上来讲,网格是分布应用集成技术上的协同扩展,它重在解决分布资源的协调工作问题,这一点是分布应用集成所缺乏的.

6 分布应用集成与 SOA

面向服务的体系(Service-Oriented Architecture, SOA)是一种新的分布应用集成示范.在体系结构上,它的基本框架由三个参与者和三个基本操作构成,三个参与者分别为服务提供者、服务请求者和服务代理,三个基本操作分别为发布、查找和绑定.服务提供者将其服务发布到服务代理的一个目录上,当服务请求者需要调用该服务时,它首先利用服务代理提供的目录去搜索该服务,得到如何调用该服务的信息,然后根据这些信息去调用服务提供者发布的服务.当服务请求者从服务代理得到调用所需服务的信息之后,通信是在服务请求者和提供者之间直接进行,而无须经过服务代理.其中,Web 服务是面向服务体系的一个实例,Web 服务体系使用一系列标准和协议实现相关的功能,如使用 WSDL 描述服务、使用 UDDI 来发布和查找服务,使用 SOAP 协议来执行服务调用[21,22].

Web 服务较其它分布应用集成方法最大的特点是它是完全低耦合的,服务请求者只有在需要服务时才动态地绑定服务提供者,其次,采用 SOAP 交互协议及 XML 作为消息格式具有跨平台特性,对集成的应用系统也要求较低,只要能够支持 SOAP 协议及处理 XML 文档就可以了.

目前,SOA 集中在对服务组合、服务协同和服务管理方面的研究.其中,服务组合以探讨了包括基于类型、QoS、有向图、Petri-net 等多种服务组合方法;在服务协同方面,也出现了相应的标准,如 BPEL,BPEL4WS,WS-Coordination 等;服务管理包括通过包装遗留系统为 Web 服务的方法来协调管理集成系统和管理 Web 服务两方面[23~25].其中,管理 Web 服务还包括对基础层管理、应用层管理和业务层管理的研究.在对 SOA 集成方法的研究中,目前,服务协同主要是依据业务流程的方法而建立,具有单调性,还需要扩张多种协同方法,并且服务组合和服务协同的方法常常交叉覆盖.在管理方法方面,还没有相应的标准出现.在 SOA 方面,还有广泛的研究空间和待解决的问题,如在服务建立协同之后,如何保障服务的可靠性,换句话说,采用哪些机制能够保障服务的可靠性;随外界环境的变化,组合服务又如何进化.

7 分布应用集成与 B2B 集成

B2B(Business-to-Business)集成是面向商业流程的集成方法,多采用 Web 服务作为交互机制.B2B 中的 Web 服务可以分为两大职能:集成 Web 服务和协作 Web 服务.集成 Web 服务目标是扩展应用程序接口或业务服务到远程系统,提供 RPC 风格的异步、同步应用程序访问,它也提供简单的路由和基于规则的转换,甚至实现某些商业逻辑.在 B2B 中,伙伴企业间的交互活动是基于角色和顺序规则及特定消息格式的,顺序规则和消息格式是由行业标准定义的.协作 Web 服务的目标是管理商业伙伴间的共享业务过程,这需要协作 Web 服务能够管理和验证交互信息,维护某些状态信息来同步和异步应答及期限检查.

B2B 集成与分布应用集成的技术框架存在一定的差异,这些差异导致了 B2B 与分布应用集成的侧重点有所不同.分布应用集成是集成应用服务,而 B2B 提供的是商业伙伴管理服务;B2B 的数据交换是基于 XML 的文档文本间的转换,使用 XSLT(eXtensible Style Language for Translation)可以实现 B2B 的数据转换,而对分布应用集成来说是不适合的.此外,分布应用集成与 B2B 的问题假设的前提条件不同,在分布应用集成中,要求多平台支撑,集成运行在多平台的应用系统要求集成组件也能够运行在多平台上;而在 B2B 集成中,没有多平台的前提条件,每个 B2B 参与者可以配置不同的集成组件,而采用统一的文档(如 XML),并定义执行序列.在 B2B 集成中,集成的实体不是应用程序,而是商业组织,因此需求不同的服务,如伙伴简介管理、交易通知和协调服务等.正是由于伙伴关系的复杂性,B2B 集成要求主动地管理 B2B 的交互过程,

从技术和权利范围的角度，采用模式的方法对 B2B 的交互方式进行了分类，包括主仆模式、对等模式、开放过程模式、封闭过程模式等[26~28].

8　分布应用集成的可重配与自适应能力

可重配与自适应是指能够根据外部环境的变化调节自身行为和结构以适应变化的能力. 在分布应用集成中，这种可重配和自适应能力可以显著地提高集成系统的柔性和可扩展性，允许集成系统动态地改变或重配自身的行为和结构来屏蔽应用系统间在交互方式、数据结构及接口等多方面的差异，在一定程度上实现异构系统间集成的即插即用(plug and play).

最早的自适应机制体现在编程语言和算法中，如高级语言中的异常处理，运行时的断言检查，算法中的网络协议、自稳定算法、负载平衡算法等等. 这种基于编程语言和算法的自适应机制固定在程序中，具有不宜更改和柔性差的特点. 在集成代理和适配器中，通常使用扩展器(extension)来达到一定的自适应功能，扩展器是一种基于模块化的方法，在集成应用程序时，将集成提交的请求委派给模块集合，再由模块集合中的相应模块处理集成提交的请求，该模块集合在初始化时就集成在系统中，这样就能够达到柔性地支持多个应用程序. 反射和元模型技术是另一种新的应用范式，即便它是以牺牲部分性能为代价的，但是它所带来的适应能力也是显著的. 反射系统通常包括两部分：一部分是元模型部分，它获取系统本身信息，执行计算，返回结果给系统本身；另一部分是系统本身，它获取外部信息，执行计算，返回结果给外部. 通过具体化(reification)过程，元模型部分能够获得系统本身的信息，能够对系统本身进行推理计算；并通过有效化(effectuation)机制改变系统本身. 构造反射模型的重点是构造元模型，该元模型能够反映系统需要自适应的部分，能够依据该元模型执行推理并采用适当的自适应策略[29,30].

在分布应用集成中，对反射中间件的研究是非常活跃的方向之一，如 OpenORB，OpenCOM 等，这些反射中间件通常采用结构反射模型和行为反射两种反射模型[31~33]. 另外，文献[34]提出了一种渠道反射(channel reflection)，如在面向对象中，渠道反射将消息作为反射实体(reflected entities)，通过对消息采用不同的行为来进行自我调节活动. 渠道反射很容易扩展到分布应用集成环境中，将分布系统中的连接器(Connector)作为渠道具体化到元模型中，可以实现根据不同的集成外部环境调整连接器的行为. 此外，文献[35,36]提出一种基于外部模型的闭环控制系统，它认为以往的方法都是面向特定应用的，与程序代码紧密绑定，因此它采用基于软件体系结构的外部模型，不同的体系风格针对不同的应用系统提供不同的反射实体、不同的推理技术和策略，具有不同的适配能力.

在分布应用集成中，由于集成环境的多样性导致各个方面存在差异性，集成的目的就是屏蔽这些差异性，而自适应、可重配技术为解决这些差异性提供了更大的柔性，特别是反射技术可以应用到分布应用集成的各个组件中，而需要解决的问题是如何构造支持集成组件推理的元模型. 例如，如何应用反射技术到适配器中来构造一种智能适配器，该适配器能够理解连接到它的源系统和目标系统，由此而具有自我适配的能力，可以动态集成多个应用系统，这也是实现动态互操作的方法，而有别于采用标准协议的方法.

9　分布应用集成的建模与形式化方法

在建模和形式化方法描述方面，始终缺乏对分布应用集成的有力支持. 从分布应用集成的角度来说，形式化模型需要能够解决描述和分析两方面的功能. 首先需要能够描述分布系统的基本组成，并刻画系统间的复杂的动态交互；其次，建模方法的支撑理论应允许分析集成的系统，包括不依赖于特定环境的系统行为分析，能够对应用系统和集成中间件的组合进行推理，能够检测不同级别的不匹配问题.

最早的与分布应用集成相关的建模和形式化方法应该是由 DeRemer 和 Kron 创造的模块互连语言(Module Interconnection Language，MIL). 在 MIL 中，模块输出输入资源，该资源是程序语言的命名元素，如类型定义、约束、变量和函数等，MIL 的编译器通过模块间的类型检查确保系统的一致性. 接口描述语言(Interface Description Languages，IDL)可以说是 MIL 在组件层次上的扩展，主要是保障组件的互操作性和可替换性，事实上，两个具有相同接口的组件可能具有完全不同的含义. 在 IDL 的基础上，一些方法扩展了接口模型包含了静态的行为语义，主要是利用组件状态中的不变量及与组件操作相关的前条件(Pre-condition)和后条件

(Post-condition),这种建模方法能够很好地描述特定时间点的组件状态,但是不能表示组件是如何到达特定状态的.

体系结构描述语言(Architecture Description Language,ADL)可以说是一个突破,它能够描述软件体系结构概念框架,提供相应的语法来刻画该软件体系结构[37~39].在 ADL 中,主要包括组件、连接器、系统、属性及风格.组件表示系统的计算元素和数据存储;连接器表示组件间的交互;系统是由组件和连接器所组成的体系结构框架图;属性用来描述体系结构部件的附加信息,例如,一个接口的属性可以定义其交互协议;风格典型地定义了词汇的集合及组合这些元素的规则[40~43].ADL 从其基础来看也是一种静态的结构建模方法,而 Wright 等人在 ADL 的基础上使用 CSP 及 Object-Z 来对组件和连接器的动态行为进行建模,能够对系统的行为进行一定程度的分析[44,45].

在更高的抽象层次上,UML 及工作流管理系统采用图形化方法描述系统模型,例如,UML 采用状态图来描述类的动态行为模型,工作流管理系统通常采用基于活动的过程建模语言[46~48].文献[49]采用一种基于通信状态机的面向消息的过程建模语言(Business Modeling Language,BML)对企业应用集成中的消息通信模型进行了描述,包括请求/应答(request/reply)、单向(one-way)、同步轮询(synchronous polling)、发布/订阅(publish/subscribe)、广播(broadcast),但它还是图形化的建模方法,本身不支持系统行为分析.

10 结束语

随着软件开发应用示范及应用场景的转变,分布应用集成已逐步成为构造信息系统的基本保障,它能无约束地连接分布的应用程序,并实现应用程序间的数据和功能的共享,并且这种共享是以不对应用程序本身作大的修改为前提条件的.从分布应用集成的发展来看,工业界对分布应用集成技术的研究要比学术界更加深入,但对方法的使用不规范,各种集成技术缺乏一个统一的标准或框架,这就导致了各个集成方案及产品在功能和性能方面的巨大差异,因此,这些集成的经验和方法需要归纳并提升为理论.本文首先阐述了分布应用集成的基本概念及特征,在此基础上对分布应用集成的集成层次进行了划分,界定了各个技术的逻辑边界,针对分布应用集成的关键问题、数据多样性、传输载体及应用程序访问、分布应用集成协同管理、分布应用集成的可重配和自适应、建模及形式化、分布应用集成与网格技术、SOA 及 B2B 的关系等核心技术作了全面的概述.既分析了存在的问题,又概述和比较了已有的技术,同时展望了未来技术发展的方向.

从分布应用集成的核心技术来看,还有很多问题有待进一步解决,特别是随着新需求的产生,还有广阔的研究空间.例如,如何在分布应用集成的基础上协同管理集成的各种应用资源,包括对各种资源可管理能力的扩充、不同场景下协同机制的研究及对 QoS 的需求等;如何解决分布应用集成的柔性和可扩展性等问题;分布应用集成如何与 Web 服务、B2B 及电子商务有效地集合;分布应用集成其本身如何引进和利用新技术,如自适应技术、面向服务的软件体系及 CEP(Complex Event Processing)等等.

参 考 文 献

1 Shaw M., Garlan D.. Software Architecture: Perspectives on An Emerging Discipline. New Jersey: Prentice Hall, 1996

2 Dashofy E. M., Medvidovic N., Taylor R. N.. Using off-the-shelf middleware to implement connectors in distributed software architectures. In: Proceedings of the 21st International Conference on Software Engineering, Los Angeles, California, 1999, 3~12

3 Ockerhloom John. Mediating among diverse data formats. Carnegie Mellon Computer Science, Pittsburgh, PA: Technical Report CMU-CS-98-102, 1998

4 Gregory R. Andrews. Paradigm for process interaction in distributed programs. ACM Computing Surveys, 1991, 23(1): 49~90

5 Antonio Carzaniga, Alexander l Wolf. A benchmark suite for distributed publish/subscribe systems. University of Colorado, Department of Computer Science, Colorado: Technical Report Colorado 80309-0430, 2002

6 Bradbury S. J., Juergen Dingel. Evaluating and improving the automatic analysis of implicit invocation systems. In: Proceedings of the 9th European Software Engineering Conference, Helsinki, Finland, 2003, 78~87

7 Barrett D. J., Clarke L. A., Tarr P. L., Wise A. E.. A framework for event-based software integration. ACM Transactions on Software Engineering and Methodology, 1996, 5(4): 378~421

8 Dingel J., Garlan D., Jha S., Notkin D.. Towards a formal treatment of implicit invocation. Formal Aspects of Computing, 1998, (10): 193~213

9 Satashi Matsuoka, Satoru Kawai. Using tuple space communication in distributed object-oriented languages. In: Proceedings

of the Object-Oriented Programming Systems, Languages and Applications, San Diego, California, 1988, 276～284

10 Andrea Omicini, Franco Zambonelli. Tuple centres for the coordination of internet agents. In: Proceedings of the 1999 ACM Symposium on Applied Computing, San Antonio, Texas, 1999, 183～190

11 Birrel A., Nelson B.. Implementing remote procedure calls. ACM Transactions on Computer Systems, 1984, 2(1): 356～372

12 Deline R.. Resolving packaging mismatch [Ph. D. dissertation]. Carnegie Mellon, School of Computer Science, Pittsburgh, PA, 1999

13 Deline R.. A catalog of techniques for resolving packaging mismatch. In: Mili A., Mittermeir R. eds. Proceedings of the Symposium on Software Reusability. New York: ACM Press, 1999, 44～53

14 Papazoglou M. P., Georgakopoulos D.. Service-oriented computing. Communication of ACM, 2003, 46(10): 25～28

15 Casati Faio, Shan Eric, Dayal Umeshwar, Shan Ming-Chien. Business-oriented management of Web services. Communication of ACM, 2003, 46(10): 55～60

16 Foster I., Kesselman C., Tuecke S.. The anatomy of the Grid: Enabling scalable virtual organizations. International Journal of High Performance Computing Applications, 2001, 15(3): 200～222

17 Foster I., Kesselman C.. The Grid: Blueprint for A New Computing Infrastructure. San Fransisco, CA: Morgan Kaufmann, 1999

18 Foster I., Kesselman C., Nick J. M., Tuecke S.. Grid services for distributed systems integration. IEEE Computer, 2002, 35(6): 37～46

19 Buyya R., Abramson D., Giddy J.. Nimrod/G: An architecture for a resource management and scheduling system in a global computational Grid. In: Proceedings of the 4th International Conference on High Performance Computing in Asia-Pacific Region, Beijing, 2000, 283～289

20 Czajkowski K., Fitzgerald S., Foster I., Kesselman C.. Grid information services for distributed resource sharing. In: Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing, San Francisco, California, 2001, 181～184

21 Benatallah B., Dumas M., Fauvet M. C.. Overview of some patterns for architecting and managing composite Web services. ACM SIGecom Exchange, 2002, 3(3): 9～16

22 Zou Ying, Kontogiannis Kostas. Web-based specification and integration of legacy services. In: Proceedings of Advanced Study on Collaborative Research, Mississauga, Ontario, Canada, 2000, 17～25

23 Hammdi Rachid, Benatallah Boualem. A petri net-based model for Web service composition. In: Zhou Xiao-Fang, Klaus Dieter Schewe eds. Proceedings of the 14th Australasian Database Conference in Research and Practice in Information Technology. Adelaide Autralia: Australian Computer Society, 2003, 191～200

24 limthanmapphon Benchaphon, Zhang Yan-Chun. Web service composition with case-based reasoning. In: Zhou Xiao-Fang, Klaus Dieter Schewe eds. Proceedings of the 14th Australasian Database Conference in Research and Practice in Information Technology. Adelaide Autralia: Australian Computer Society, 2003, 201～208

25 Tsalgatidou Aphrodite, Pilioura Thomi. An overview of standards and related technology in Web services. Distributed and Parallel Database, 2002, 12(2): 135～162

26 Medjahed Brahim, Benatallah Boualem, Bouguettaya Athman, Ngu Anne H. H., Elmagarmid Ahmed K.. Business-to-business interactions: Issues and enabling technologies. The VLDB Journal, 2003, 12(6): 59～85

27 Aaron R., Decina M., Skillen R.. Electronic commerce: Enablers and implications. IEEE Communications Magazine, 1999, 37(9): 47～52

28 Dayal U., Hsu M., Ladin R.. Business process coordination: State of art, trends, and open issues. In: Proceedings of the Conference of Very Large Data Bases, San Francisco, CA, 2001, 3～13

29 Poladian V., Sousa J. P., Garlan D., Shaw M.. Dynamic configuration of resource-aware services. In: Proceedings of the 26th International Conference on Software Engineering, Edinburgh, Scotland, United Kingdom, 2004, 604～613

30 Maes Pattie. Concepts and experiments in computational reflection. In: Norman K Meyrowitz eds. Proceedings of the 2nd Conference on Object-oriented Programming Systems, Languages, and Applications, Orlando, Florida: ACM Press, 1987, 147～156

31 Clarke M., Blair G., Coulson G., Parlavantzas N.. An efficient component model for the construction of adaptive middleware. In: Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms, London, UK, 2001, 160～178

32 Blair G. S., Coulson G., Andersen A., Blair L., Clarke M.. The design and implementation of OpenORB. IEEE Distributed System, 2001, 2(6): 34～56

33 Kon Fabio, Costa Fabio, Blair Gordon, Campbell Roy H.. The case for reflective middleware. Communications of the ACM, 2002, 45(6): 33～38

34 Cazzola Walter. Evaluation of object-oriented reflective models. In: Proceedings of ECOOP Workshop on Reflective Object-Oriented Programming and Systems, Brussels, Belgium, 1998, 386～387

35 Cheng Shang-Wen. Software architecture-based adaptation for Grid computing. In: Proceedings of the 11th IEEE Conference on High Performance Distributed Computing, Edinburgh, Scotland, United Kingdom, 2002, 389～396

36 Oreizy P., Gorlick M. M., Taylor N. R.. An architecture-based approach to self-adaptive software. IEEE Intelligent Sys-

tem, 1999, 14(3): 54～62

37 Aguiire N., Maibaum T. S. E.. A temporal logic approach to component based system specification and reasoning. In: Crnkovic I., Schmidt H., Stafford J., Wallnau K. eds. Proceedings of the 5th ICSE Workshop on Component-Based Software Engineering. Orlando: Spinger-Verlag, 2002, 456～468

38 Allen R., Garlan D.. A formal basis for architectural connection. ACM Transactions on Software Engineering and Methodology, 1997, 6(3): 231～249

39 Spitznagel B., Garlan D.. A compositional approach for constructing connectors. In: Proceedings of the Working IEEE/IFIP Conference on Software Architecture, Amsterdam, Netherlands, 2001, 148～157

40 Mehta N. R., Medvidovic N., phadke S.. Towards a taxonomy of software connectors. In: Proceedings of the 22nd International Conference on Software Engineering, Limerick, Ireland, 2000, 178～187

41 Hirsch D., Uchitel S., Yankelevich D.. Towards a periodic table of connectors. In: Proceedings of the 3rd International Conference on Coordination Languages and Models, Amsterdam, Netherlands, 1999, 418～427

42 Garlan D., Allen R., Ockerbloom J.. Architecture mismatch or why it's hard to build systems out of existing parts. In: Proceedings of the 17th International Conference on Software Engineering, Seattle, Washington, 1995, 179～185

43 Gio Wiederhold. Mediators in the architecture of future information systems. IEEE Computer, 1992, 25(3): 38～49

44 Plasil F., Visnovsky S.. Behavior protocols for software components. IEEE Transactions on Software Engineering, 2002, 28(11): 1056～1076

45 Hoare C.. Communicating Sequential Processes. New Jersey: Prentice Hall, 1985

46 Hnatkowska B.. Consistency checking in UML models. In: Proceedings of the 4th International Conference on Information System Modeling, Hradec nad Moravici, Czech, 2001, 173～180

47 Casati F., Ceri S., Pernici B., Pozzi G.. Conceptual modeling of workflow. In: Mike P Papazoglou eds. Proceedings of the 14th International Object-oriented and Entity-Relationship Modeling Conference. Gold Coast: Springer-Verlag, 1995, 341～354

48 Dumas M., Ter Hofsted A.. UML activity diagrams as a workflow specification language. In: Martin Gogolla, Cris Kobryn eds. Proceedings of the 4th International. Conference on the Unified Modeling Language. Toronto: Springer, 2001, 76～90

49 Petia Wohed, Erik Perjons. Pattern based analysis of EAI languages-the case of the business modeling language. In: Camp O. ed. Proceedings of the 5th International Conference on Enterprise Information Systems. Dordrecht: Kluwer Academic Publishers, 2003, 174～182

XU Gang, born in 1973, Ph. D.. His research interests include software engineering, distribution computation, enterprise application integration and service-oriented architecture.

HUANG Tao, born in 1965, Ph. D., professor, Ph. D. supervisor. His research interests include software engineering, distribution computation.

LIU Shao-Hua, born in 1976, Ph. D. candidate, assistant professor. His research interests include distribution computation, workflow and service collaboration.

YE Dan, born in 1971, Ph. D., associate professor. Her research interests include software engineering, distribution computation and virtual enterprise.

Background

This research belongs to the project of "The Research of Net-Component Software Middleware Platform Model and Framework" as the fifth sub-project under the National Basic Research Program of China (973 Program) project-"The Research of Agent-Based Software Middleware Theory and Method under the Internet Environment", and is also supported by the National High Technology Research and Development Program (863 Program) of China. The fifth sub-project mainly focus on the middleware platform supporting the running of net-component under Internet, and researches the common and basic services needed by net-component, such as platform model, framework and prototype. In the past years, in the domain of net-component and distribution computation, the authors have researched and developed Web Application Server, Data Integration Middleware, Workflow, Transaction Monitor, Information Portal, and so on. As the an important part of the fifth sub-project, the research of the paper mainly helps to resolve the interacting, integrating and collaborating between net-components and ultimately builds a mi1ddleware-based integration and collaboration mechanism and model.