面向服务领域软件的参考模型*)

蒋哲远　韩江洪
(合肥工业大学计算机与信息学院　合肥 230009)

摘　要　基于构件和面向服务体系结构(SOA)的软件工程被认为是提高大规模的分布式软件开发效率和质量的有效途径。但是,SOA 目前还停留在抽象的高层概念模型层面上,还没有一个实用、具体的应用参考模型。介绍了一种基于 Web 服务软件体系结构的领域系统构造过程。在此基础上,提出了一种面向 Web 服务的领域软件体系结构参考模型(WS-DSARD),对其主要元元素角色、操作、服务构件和服务构件类等进行了较为详细的描述,并从服务构件交互与集成的角度分析了服务构件的组合语义。该研究对于面向服务领域软件开发活动的成功实施具有一定的指导作用。

关键词　Web 服务,服务构件,领域软件,参考模型

A Reference Model for Service-Oriented Domain Software

JIANG Zhe-Yuan　HAN Jiang-Hong
(School of Computer and Information, Hefei University of Technology, Hefei 230009)

Abstract　Component-based and Service Oriented Architecture (SOA) domain software engineering is considered as an effective and efficient approach to improve the productivity and quality of large-scale distributed software development. But, current SOA are concentrating mostly on high-level conceptual model, lacking concrete reference model to guide the practical development process. The construction process of domain system based on Web services software architecture is introduced. On the basis of the process, a Reference Model for Web Services-oriented Domain Software Architecture, named WS-DSARD, is proposed, and its some primary meta-elements, for example role, operation, service component and service component class, is depicted in detail. From the perspective of service component interaction and integration, compositional semantics of service components is described. This approach will be beneficial to successful service-oriented domain software practices.

Keywords　Web services, Service component, Domain software, Reference model

1　引言

可重用性(reusability)和可维护性(maintainability)是软件开发极为关键的两个方面[1]。从代码级封装到功能级、模块化和面向对象封装,进而发展到面向构件技术,人们一直在增强可重用性和可维护性方面进行努力。在 Internet 技术迅速发展和普遍应用的今天,软件开发面临更加复杂的问题,如应用的分布性、平台的异构性等。面向服务体系结构(Service Oriented Architecture,SOA)[2]是解决这些问题的可行方案之一。

SOA 是由一系列相互交互的服务组成,每种服务带有定义明确的可调用接口。SOA 整体上设计和实现为一系列相互交互的服务,这种将业务功能实现为服务的方法可以增强系统的灵活性,系统通过增加新的服务来实现演化。SOA 定义了系统由哪些服务组成,描述了服务之间的交互,并将服务映射到一个或多个具体技术的实现。总体来讲,SOA 的优势在于它的高可重用性(如业务关系和功能的重用)、灵活性(agility),以及更好的扩展性和可用性,从而有利于减少软件开发的时间、费用和风险。一个典型 SOA 风格的应用概念模型(如图 1 所示)主要是由服务提供者(provider)、请求者(requester)和注册中心(registry)三部分组成,其中:提供者在注册中心发布它们的服务,请求者发现服务并调用它们。不过,SOA 目前还停留在抽象的高层概念模型层面上,还没有一个实用、具体的应用参考模型[3]。

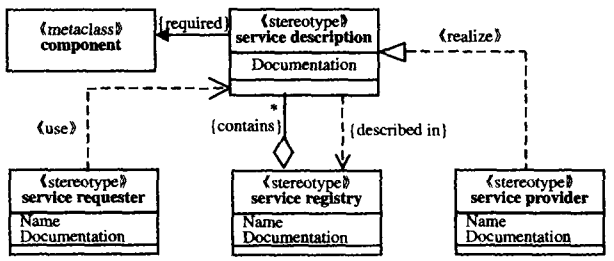

图 1　SOA 风格的应用概念模型

基于 XML 技术的 Web 服务作为一种新的跨 Internet 上的分布式计算范式,现已成为使用 SOA 构建大规模分布式应用实现的一种服务概念抽象标准,并且正在扮演越来越重要的角色。SOA 和 Web 服务的体系结构是两个不同层面的问题,前者是概念模式,面向商业应用;后者则是实现模式,面向技术框架。SOA 作为一个概念上的模型,它将网络、传输协

*)基金项目:国家"八六三"高技术研究发展计划基金项目(2002AA415280);教育部博士点基金项目(20050359004);教育部新世纪优秀人才计划项目(NCET-04-050562);合肥工业大学科学研究发展基金项目(040501F)。**蒋哲远**　博士生,助理研究员,CCF 高级会员,主要研究领域为面向服务和软件工程和分布式系统;**韩江洪**　教授,博士生导师,主要研究领域为分布式系统和智能控制技术。

议以及安全等具体的细节都遗留给特定的实现，而 Web 服务体系结构实际上是面向服务的体系结构的一个特定的实现。但正是这种实现使得面向服务的体系结构与传统的体系结构（如客户端/服务器结构）相比有了更好的特性，突出体现在其整体结构的分布性，松耦合（loosely coupled）的可集成性，标准的开放性以及应用过程的可管理性。

为此，以面向服务体系结构驱动的领域应用软件的系统化建模与开发实现为背景，提出一种基于 Web 服务的领域系统构造过程，着重对面向 Web 服务的领域软件体系结构参考模型（Reference Model for Web Services-oriented Domain Software Architecture，WS-DSARD）的基本元元素角色、操作、服务构件和组合服务构件进行定义和详细描述，并从服务构件交互与集成的角度分析了服务构件的组合语义，讨论了服务构件类和服务构件实例的关系。

2 Web 服务的领域系统构造过程

基于 Web 服务的领域软件开发是在 SOA 规范下开发的，注意力集中在它的体系结构和服务构件（service component）[4] 的交互操作处理上，目的是为开发者理解软件和特定事务领域的大系统开发的集成问题提供一个一般框架。事务领域是指真实世界中人类活动的自我包含空间，在这个空间中，定义好了功能和服务。图 2 描述的是基于 Web 服务的领域系统构造过程。

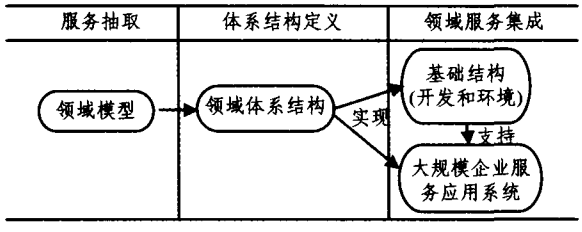

图 2 Web 服务的领域系统实现过程

该构造过程主要包括三个阶段：(1)定义感兴趣的服务集，称之为构建服务库；(2)映射服务到构件配置，可以得到系统的体系结构描述；(3)服务构件配置的组装和集成。

第一个阶段是识别相关用例和用例之间的关系。这些用例图和活动图中的用例、角色和它们的交互将被作为定义服务元素的来源之一。接下来是生成包含角色的领域模型（domain model）。领域模型是一个半形式化的领域描述，它创建一个综合知识库并影响领域中的所有开发和集成的结果。下面是领域模型的定义。

定义 1（领域模型） 领域模型是一个五元组 $DM=\langle D, O, R, F, C\rangle$，其中：

(1)D 是领域有关的主题文档（subject document）集，该文档集中的数据具有相对的内聚性和稳定性；

(2)O 是基于主题文档的操作集；

(3)R 是主题文档的角色（role）驱动关系集；

(4) F 是操作的顺序流程集；

(5)C 是操作和关系的约束规则。

在第二阶段中，角色领域模型将被细化为一个构件配置，在该配置基础上，系统包含的服务将被映射成为一个体系结构配置和描述，其中定义了服务构件的规约、构件之间的交互（即连接器类型）、构件与连接器的拓扑结构和配置组成。这些体系结构描述和配置能被容易地实现，且经过考查和评估可作为目标系统的体系结构，即领域体系结构（domain architecture）。

领域体系结构是一个统一的、相关的、多级软件体系结构规范，这个规范被用作开发活动的指南，它将设计限制在低层，并支持互操作性和重用性[5]。基础结构（infrastructure）是一个一致的开发和操作支持环境，它为构件提供一般服务并实现由体系结构定义的通用功能。

领域体系结构是面向 Web 服务软件构造的核心，它对整个领域开发过程有强烈的影响。领域体系结构促进技术的标准化并担当一个事实上的应用标准。在保护完整性和语义一致性时，它支持分布式开放系统的基本原理。从广域的角度看，这样一个体系结构作为一个参考和遵照的构架，确保了目标系统和它的组成部分能满足它们的功能的和非功能的需求。WS-DSARD 包括服务构件的结构模型、框架模型、动态模型和过程模型。

(1)结构模型定义了服务构件的角色和操作，这是一个最直观、最普遍的建模方法。这种方法以体系结构的构件、角色和其他概念来刻画结构，并力图通过结构来反映系统的重要语义内容，包括系统的配置、约束、隐含的假设条件、风格、性质。研究结构模型的核心是面向 Web 服务的领域体系结构描述语言（Architecture Description Language，ADL）。

(2)框架模型与结构模型类似，但它不太侧重描述结构的细节而更侧重于整体的结构。框架模型主要以一些特殊的问题为目标建立只针对和适应该问题的结构，例如 QoS 驱动的 Web 服务有效发现中间件框架、特定领域软件协同集成框架等。

(3)动态模型是对结构或框架模型的补充。研究 Web 服务系统的“大颗粒”的动态行为性质。例如，描述系统的重新配置或演化。而动态行为面向系统的预定义演化逻辑，使系统能够自适应演化，以体系结构元素为处理对象，如增删构件、建立新的角色等。

(4)过程模型研究构造面向 Web 服务系统的开发步骤和过程，因而结构是遵循某些过程脚本的结果。

将这四种模型有机统一在一起，构成了对面向 Web 服务的领域体系结构的一个完整刻画模型。

第三阶段是根据领域体系结构配置和描述，组装和集成现有的构件，形成面向服务的应用系统。上述构造过程实际上是一个三阶段的反复迭代过程，且跨越从角色和服务的抽取，到构件选取、包装和集成并反馈到用例图的定义；领域体系结构能被细化和重构而产生新的领域体系结构配置，进而可形成用例的一个全新视角。

3 角色和操作

3.1 角色和角色类

定义 2（角色） 角色是定义了接口以及调用规约（specification）而没有完全实现的构件，它是一个五元组 $\langle ID, Action, Event, States, LConstrains\rangle$。其中：$ID$ 是角色的标识；$Action$ 是角色活动的集合，每个活动由事件的连接（谓词）组成；$Event$ 是角色产生的事件集合；$States$ 是角色参与交互时的各种可能状态集合；$LConstrains$ 是角色的约束集合。

角色的概念提出较早，在信息安全领域基于角色的访问控制研究中，通过将用户和访问控制基于角色相关联，实现访问策略的精确定义[6]。而在本文的角色定义中，角色反映了一个服务构件（service component）在协作中所承担的责任，

它起到了占位符(placeholder)的作用。在 WS-DSARD 的领域框架实现以及重用中,角色通常还实现以下 3 个方面的功能:

(1)定位和查找服务构件;

(2)检查服务构件接口是否符合框架定义的约束;

(3)在框架构件对服务构件的调用中负责传送消息,并进行接口以及数据的转换。

WS-DSARD 中的角色构件可以分为两类,即具有部分实现特性的抽象构件和模板构件,前者仅进行了接口定义,而没有相应功能的具体实现;而模板(template)构件包含实现算法,但是在使用时需要设置具体参数(包括数值参数和类型参数)。角色构件需要进一步实例化,包括编写具体实现和指定运行时参数,这体现了框架的部分实现特性。

角色类型(type)是一种由构件(或对象)扮演的类型。在任意给定时刻,构件都可以扮演几个不同的角色类型。这就使得不同的客户可以对同一个构件对象拥有不同的视图。同样,不同的构件也可以提供由同一个角色类型所说明的行为。集成角色类型集合由那些作为接口,用于与其他构件进行交互的角色类型所组成。当然,这种扮演行为不是由框架本身决定的,而是通过定义连接器(connector)的某一角色类型与框架的某一角色类型之间的扮演关系来确定。这种对应关系体现为配置定义中的扮演声明。从广义上讲,框架既可以仅由一个类构成,也可以包括上百个类。无论在哪一种情况下,都必须把集成角色类型集合的角色类型映射成某种独立的对象行为。

3.2 角色模型

角色类型和它们之间的约束及其关系就构成了一个角色模型(role model)。也就是说,角色模型是由角色类型及其约束和关系所创建的一组运行时对象(构件)合作的约束规约,它是描述对象协作的最有效的方法之一[7]。从软件体系结构角度出发,角色模型被进一步界定为由角色等价约束和角色协同关系构成的角色交互(collaboration)及其角色行为(behavior)[8]构成。

定义 3(角色等价,role-equivalent) 假设构件 a 扮演由角色类型 A 所定义的角色,构件 b 扮演由角色类型 B 所定义的角色。若存在 a 所完成的行为 b 也需要完成,反之亦然,则称 *role-equivalent*(A,B)成立。

角色等价关系主要用于连接器(connector)的组装方面,即由一些重用库中的简单连接器定制较为复杂的连接器。

定义 4(协同关系,cooperation) 假设构件 a 扮演由角色类型 A 所定义的角色,构件 b 扮演由角色类型 B 所定义的角色,如果构件 a 与构件 b 之间有相互要求的合作服务序列,则有 *cooperation*(A,B)成立。

至于构件之间的具体协同行为则由脚本序列加以定义。

WS-DSARD 的连接器的规约由角色和处理这两个基本要素构成。连接器的角色定义如前述角色类型的定义。例如:一个管道连接器有两个角色:数据源(source)和数据池(sink)。一个事件广播连接器的角色可以有一个播音员和一个或多个倾听者。连接器的处理描述由连接器所连接的参与者如何在一起工作而产生一种交互。而且,处理也可以拥有自己的控制线程和数据存储。这使得连接器成为一种可执行的实体。

WS-DSARD 从体系结构的连接器角度出发对角色模型所表示的一种约束和一种关系加以改进并作为连接器加以使用,并作为 ADL 语言的基本元素进行规约。

3.3 操作

操作(operation)是指可以调用对象执行的转换或查询的规格说明,通常可表示为由操作的特性组成的字符串。

定义 5(操作) 操作的缺省语法是:

操作 ::=[《stereotype》][visiblity] name (parameter-list)[:return-type][{property-string}]

构造类型、可见性、返回类型表达式、特性字符串(以及它们的定界符)可省略。参数列表可为空。

标准的 Web 服务体系结构的服务构件角色之间有 3 种操作:发布(publish)、查找(find)和绑定(bind),其基本操作原语分别描述如下:

(1) publish (Service, Operation, In, Out, Restriction, Category, Function, URI);

(2) find (Operation, In, Out, Restriction, Category, Function);

(3)bind (Service, Operation, In ,URI)。

上式中,Service 是服务名;Operation 是操作名;In 是输入变量的集合;Out 是输出变量的集合;Restriction 是输入变量的定义域,它由一组对偶(x, range)构成的集合,其中 x 是 In 中的枚举变量,而 range 是 x 的所有可能值的集合;Category 描述兴趣域;Function 描述业务功能;URI 是提供该服务的统一资源标识符。

4 Web 服务构件和构件类

4.1 Web 服务构件

提出 Web 服务构件模型(component model)的目的是使用户在构建企业应用时有一个不再直接面对具体的技术细节的层次,而是从业务模型(主要是业务过程模型)出发,通过大粒度的 Web 服务构件的方式来构建应用。构件是 Web 服务的实现体,Web 服务是构件的调用接口,但并不是所有的构件都可直接提供 Web 服务,可以作为 Web 服务构件的是自包含的、相对独立的构件,这种构件称之为 Web 服务构件。从调用者角度看,一个 Web 服务是一个无状态的单一实例的构件,管理着其内部的资源分布。Web 服务构件是组合构件,包含成员构件及内部协作关系。图 3 给出了 Web 服务构件的参考模型。

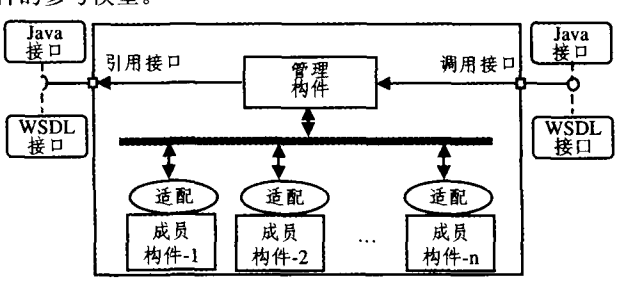

图 3 Web 服务构件模型

定义 6(Web 服务构件模型) Web 服务构件模型的语法规范的扩展巴科斯范式(Extended Backus-Naur Form, EBNF)定义如下:

服务构件::=〈管理构件,成员构件集合,适配器集合,服务〉

管理构件::=〈构件接口,引用的成员构件类型,实例声明,实例适配连接,服务映射〉

成员构件::=〈构件接口,构件实现〉

构件接口::=〈接口方法集合〉

接口方法::=〈接口方法标识,参数列表,返回值〉

适配器::=〈适配器标识,构件接口,适配成员构件能力参数列表,适配协作成员构件能力参数列表,消息映射〉

服务::=〈消息类型定义,端口,端口绑定〉

端口::=〈端口操作集合〉

端口操作::=〈服务操作标识,输入参数列表,输出参数列表〉

Web服务构件通过其内部管理子构件创建内部其它成员构件实例、维持内部状态转化和持久化。基于消息调度机制的适配器协作其它成员构件的动态交互和组合关系,并使得这些行为对外界不可见。提供一个统一的接口并映射为服务供使用者调用。Web服务构件把提供给别的服务调用的入口叫调用(invocation)接口,而Web服务构件本身可能也需要调用别的Web服务,这个调用出口叫引用(reference)接口。无论是调用接口还是引用接口,其调用规范都是WSDL或Java接口。

Web服务构件是一种全新的、跟语言无关的编程模型,它提供了一种统一的调用方式,从而使得客户可以把不同的组件类型。例如,EJB、流程组件、以及人工交互组件等都可以通过一种标准的接口来封装和调用。

Web服务构件与传统构件的主要区别在于:

(1)Web服务构件往往是粗粒度的,而传统构件以细粒度居多。

(2)Web服务构件的接口是标准的,主要是WSDL接口,而传统构件常以具体API形式出现。

(3)Web服务构件的实现与语言是无关的,而传统构件常绑定某种特定的语言。

(4)Web服务构件可以通过构件容器提供服务质量(Quality of Service,QoS)的服务,而传统构件完全由程序代码直接控制。

4.2 Web服务构件类

为了方便发现特定领域Web服务构件,还需要引入Web服务构件类的概念。

定义7(Web服务构件类) 一个Web服务构件类是由一组具有相同的功能或数据访问为特征的Web服务构件所组成的集合。Web服务构件是其Web服务构件类的实例。

一个Web服务构件类的期望功能定义是由一个Web服务构件类描述(Web Service Class Description,WSCD)来规定。WSCD定义了组成Web服务构件类的相关元素,却不说明实例的特定细节。WSCD清楚地规定了一个抽象接口,并提供了确定一个特定Web服务到一个给定的Web服务构件类的关系引用。WSCD一开始通常由用户或服务开发者来提供,并能在服务搜索和匹配过程中被所嵌入的自动学习算法进一步修改完善。

一个Web服务构件类描述(WSCD)是一个Web服务构件类的抽象描述,该描述规定了一个Web服务要想成为某个Web服务构件类成员所必须输出(export)的最小功能。

定义8(Web服务构件类描述) 一个$WSCD$可表示为如下的三元组(triple):$WSCD = \langle T, C, P \rangle$,其中,$T$代表一组构件类型定义,$C$代表一个业务过程控制流图,$P$代表领域搜索模式(或模板)。

一个WSCD把Web服务构件类中所有成员的公共部分封装定义为构件,并为Web服务构件间的细小差别提供一个隐藏机制,其中包括对Web服务构件功能没有影响的不一致接口。另外,WSCD提供了足够的信息来区分一组等待仲裁的Web服务构件集。

5 组合Web服务构件

在WS-DSARD中,Web服务构件之间通过接口之间的连接组合成为新的Web服务构件,本文称这种构件为组合Web服务构件。

5.1 Web服务构件组合的相关定义

为了能够对组合Web服务构件的接口语义进行精确的描述,一个Web服务构件可抽象建模成由输入和输出参数组成的实体。

定义9 一个Web服务构件i可以描述成$WS_i(I_i, O_i)$形式。其中:WS_i是Web服务构件的名字;I_i是该服务构件的输入参数集合;O_i是其输出参数集合。

定义10 一个Web服务构件k的请求可以用表达式$WSR_k(I_k, O_k)$来描述。其中:WSR_k是Web服务构件请求的名字;I_k是该服务构件请求的输入参数集合;O_k是该服务构件请求的输出参数集合。

定义11 对于两个服务构件$WS_i(I_i, O_i)$和$WS_j(I_j, O_j)$,如果有$O_i \supseteq I_j$则称从服务构件WS_i到WS_j语义关联,记为$WS_i \succ\!- WS_j$。其中WS_i称为WS_j的前驱服务构件,WS_j称为WS_i的后继服务构件。

定义12 对于服务构件请求$WSR_k(I_k, O_k)$和服务$WS_i(I_i, O_i)$,如果$I_i \supseteq I_k$,则称从服务构件请求WSR_k到服务构件WS_i语义关联,记为$WSR_k \succ\!- WS_i$,其中,WS_i称为WSR_k的后继服务构件;如果$O_i \supseteq O_k$则称从服务构件WS_i到服务构件请求WSR_k求语义关联,记为$WS_i \succ\!- WSR_k$,其中,WS_i称为WSR_k的前驱服务构件。

Web服务构件组合过程的服务匹配与传统的模式匹配(schema matching)[9]不同,其匹配是有方向的,一个活动的后置条件必须符合其后继活动的前置条件。因此,Web服务构件组合对参与组合的Web服务构件之间的协作要满足方向的约束要求。

定义13(Web服务构件组合) 一个Web服务构件组合是指能够满足某个服务请求的一个Web服务构件序列$WS_1, WS_2, \cdots, WS_n$,该序列满足以下3个条件:

(1)$WSR_k \succ\!- WS_1$;

(2)该序列中任意两个相邻的服务构件WS_i和WS_{i+1}都满足$WS_i \succ\!- WS_{i+1}$;

(3)$WS_n \succ\!- WSR_k$。

5.2 Web服务构件的交互语义

从接口调用角度来看,基于消息的Web服务构件交互有两种不同的语义:同步调用和异步调用。常见的方法调用都是同步调用,这种调用方式是一种阻塞式的调用方式,即客户端(主调用方)代码一直阻塞等待直到被服务端(被调用方)返回为止。这种调用方式相对比较直观,也是大部分编程语言直接支持的一种调用方式。但是,如果面对是基于粗粒度的Web服务构件,面对的是一些需要比较长时间才能有响应的应用场景,那么就需要一种非阻塞式调用方式,即异步调用方式。

WS-DSARD支持单向调用方式、延迟响应方式和请求回调方式三种方式的异步调用。单向调用方式是最为简单的异

步调用方式，在这种调用方式中，客户端发出请求之后就不再关心服务端的情况，包括是否执行成功，返回值是什么等等；延迟响应方式是指客户端在发出调用请求之后继续执行，但是经过一段时间之后，客户端再调用相应的方法去检索返回结果，并通过参数指定如何根据调用的结果而执行进一步动作；请求回调方式与延迟响应方式类似，也能得到服务端的响应，但是不同的是这个响应是由服务端通过回调方式来触发的，而不像延迟响应方式由客户端来主动检索的。

结束语 在传统的软件开发领域，目前构件技术如 EJB、CORBA 和 COM 等已经获得巨大的成功，成为软件开发的主流技术。因此，要想使得 Web 服务体系结构的研究成果得到广泛的领域应用，则必须和构件技术相结合。在现有的构件模型中，构件接口描述仅仅是接口的基调，如何扩充构件接口使其包含有关 Web 服务方面的信息是 Web 服务体系结构和构件技术相结合的有效途径。

本文基于 SOA 特点探讨了一种 Web 服务的领域系统构造过程，对其 WS-DSARD 的主要元元素角色、操作、服务构件和服务构件类等进行了定义和较为详细的描述，并从服务构件交互与集成的角度分析了服务构件的组合语义。WS-DSARD 的概念模型具有以下特点：(1)使用结构模型、框架模型、动态模型和过程模型集成刻化面向服务的领域软件体系结构；(2)支持捕获服务作为一组角色间的交互模式，并把每一个角色模拟一个具体构件在参与某个服务实现时所必须展示的行为；(3)把服务视为首要的类建模元素，支持大粒度的 Web 服务构件重用；(4)支持基于 Web 服务构件软件的增量、迭代式开发。

目前，我们已在 WS-DSARD 的概念模型基础上，提出了一种新的基于 XML 的用来描述服务软件体系结构的体系结构描述语言 WS-XADL，并已初步实现了 QoS 驱动的 Web 服务有效发现中间件框架、以及特定领域软件协同集成框架等支撑工具。至于如何更加全面地形式化的描述 WS-DSARD 的各种模型，并基于该模型实现其集成支持环境，还需要在进一步的工作中加以研究。

参 考 文 献

1 Selby R W. Enabling reuse-based software development of large-scale systems. IEEE Transactions on Software Engineering, 2005, 31(6):495～510

2 Papazoglou M P. Service-oriented computing: concepts, characteristics and directions. In: Proceedings of Fourth International Conference on Web Information Systems Engineering (WISE 2003). Los Alamitos, CA, USA: IEEE Computer Society Press, 2003. 3～12

3 Nadhan E G. Service-oriented architecture: implementation challenges. The Architecture Journal, 2004, 2(4):24～32

4 Yang J. Web service componentization. Communications of the ACM, 2003, 46(10):35～40

5 Kang K C. Feature-oriented development of applications for a domain. In: Proceedings of the Fifth International Conference on Software Reuse. New York, NY, USA: IEEE Press, 1998. 354～355

6 Feng X, Guoyuan L, Hao H, et al. Role-based access control system for web services. In: Proceedings of the 4th International Conference on Computer and Information Technology. Los Alamitos, CA, USA: IEEE Computer Society Press, 2004. 357～362

7 Zhao L P, Kendall E. Role modelling for component design. In: Proceedings of the Proceedings of the 33rd International Conference on Technology of Object-Oriented Languages. New York, NY, USA: IEEE Press, 2000. 312～323

8 Rumbaugh J, Jacobson I, Booch G. The Unified Modeling Language Reference Manual, Second Edition. Boston, MA, USA: Addison-Wesley, 2004

9 Rahm E, Bernstein P A. A survey of approaches to automatic schema matching. The VLDB Journal, 2001, 10(4):334～350

(上接第 253 页)

何借助形式语义对系统进行检查与分析。动态角色的概念源于 Darwin 的两种动态机制，借助 pi 演算目前已获得了 Darwin 的形式语义[16]。而在我们过去的研究中，也曾给出了组合连接器的 CSP 语义。因此，将 CSP 与 pi 演算结合起来研究动态角色的语义可能是一种选择。

另一个值得研究的重要问题是关于无限动态结构的描述。所谓无限动态结构，指的是在动态体系结构的结构演化中，系统的拓扑结构能够按某种模式无限地扩展，比如 Darwin 就曾给出了一个基于管道-过滤器体系结构风格的例子[7]。显然为了描述这种结构必须采用递归机制，但是否还应该扩充已有的方法，这需要作更深入的探讨。

参 考 文 献

1 Garlan D, Perry D E. Introduction to the special issue on software architecture. IEEE Trans on Software Engineering, 1995,21(4): 269～274

2 Medvidovic N, Taylor R N. A classification and comparison framework for software architecture description languages. IEEE Trans on Software Engineering, 2000,26(1):70～93

3 Allen R J, Garlan D. A Formal Basis for Architectural Connection. ACM Transactions on Software Engineering and Methodology, 1997,6(3):213～249

4 Allen R J. A formal approach to software architecture: [Ph. D. Thesis]. Pittsburgh: Carnegie Mellon University, 1997

5 Allen R, Douence R, Garlan D. Specifying and analyzing dynamic software architectures. In: Astesiano E, ed. Proc of the 1st Int'l Conf on Fundamental Approaches to Software Engineering. Berlin: Springer-Verlag, 1998. 21～37

6 Magee J, Dulay N, Eisenbach S, et al. Specifying distributed software architectures. In: Schafer W, Botella P, eds. Proc of the 5th European Software Engineering Conf. Berlin: Springer-Verlag, 1995. 137～153

7 Magee J, Kramer J. Dynamic structure in software architectures. In: Kaiser G E, ed. Proc. of the 4th ACM SIGSOFT Symp. on Foundations of Software Engineering. New York: ACM Press, 1996. 3～14

8 Luckham D C, Augustin L M, Kenney J J, et al. Specification and analysis of system architecture using rapide. IEEE Trans on Software Engineering, 1995,21(4):336～355

9 熊惠民，应时，虞莉娟，等. 基于反射的连接器组合重用方法. 软件学报，2006,17(6):1298～1306

10 Hoare C A R. Communicating Sequential Processes. Englewood Cliffs, New Jersey: Prentice Hall, 1985

11 Cazzola W, Savigni A, Sosio A, et al. Architectural reflection: Concepts, design, and evaluation: [Technical Report]. Milano: University of Milano Bicocca, 1999

12 Cuesta C E, de la Fuente P, Barrio-Solorzano M. Dynamic coordination architecture through the use of reflection. In: Lamont G B, ed. Proc. of the 2001 ACM Symp. on Applied Computing. New York: ACM Press, 2001. 134～140

13 Morrison R, Kirby G, Balasubramaniam D, et al. Support for evolving software architectures in the ArchWare ADL. In: Proc. of the 4th Working IEEEE/IFIP Conf on Software Architecture. Washington D C: IEEE Computer Society Press, 2004. 69～78

14 Inverardi P, Wolf A. Formal specification and analysis of software architectures using the chemical abstract machine model. IEEE Trans on Software Engineering, 1995,21(4): 373～386

15 Métayer D L. Describing software architecture styles using graph grammars. IEEE Trans on Software Engineering, 1998,24(7): 521～533

16 Eisenbach S, Paterson R. π-Calculus semantics for the concurrent configuration language Darwin. In: Proc. of the 26th Annal Hawaii Int'l Conf on System Sciences, Volume 2. Washington D C: IEEE Computer Society Press, 1993. 456～462