一种面向语义Web服务的语义程序变换方法

王权于[1,2] 应 时[1] 吕国斌[2] 赵 楷[1,3]
(武汉大学软件工程国家重点实验室 武汉 430072)[1] (中国地质大学(武汉)网络教育学院 武汉 430074)[2]
(新疆大学信息科学与工程学院 乌鲁木齐 830046)[3]

摘 要 语义程序变换是面向语义Web服务的软件设计方法的基础,语义程序只有通过程序变换后才能被运行环境执行和调用,然而目前还缺乏有效的语义程序变换方法。针对这一问题,基于语义编程语言SPL,提出了一种面向语义Web服务的语义程序变换方法。该方法通过对语义数据类型、语义规则、语义服务和语义流程等语义信息的有效变换,不仅提高了面向服务的程序设计的灵活性和健壮性,而且有助于提高业务流程的柔性和重用性。

关键词 语义Web服务,语义程序,程序变换
中图法分类号 TP301 **文献标识码** A

Semantic Web Service-oriented Semantic Program Transformation Approach

WANG Quan-yu[1,2] YING Shi[1] LU Guo-bin[2] ZHAO Kai[1,3]
(State Key Lab of Software Engineering, Wuhan University, Wuhan 430072, China)[1]
(Distance Education College, China University of Geoscience, Wuhan 430074, China)[2]
(College of Information&Engineering, Xinjiang University, Urumqi 830046, China)[3]

Abstract Semantic program transformation is the foundation of Semantic-Web-Service-oriented software design. Semantic program can't be executed and called by runtime environment until it passes the program transformation. There isn't, however, any effective semantic program transformation method. Aiming at this problem, this paper presented a Semantic-Web-Service-oriented semantic program transformation method which is based on semantic program language SPL. This method makes an effective transformation on semantic information such as semantic data type, semantic rule, semantic service, semantic flow and so on, which not only enhances the flexibility and robustness of service-oriented program design, but also increases the flexibility and reusability of the business process.

Keywords Semantic Web service, Semantic program, Program transformation

1 引言

面向语义Web服务[1]的程序设计语言和方法正日益成为面向服务计算(SOC)的研究热点和重点。而目前已有的面向服务的设计语言(如BPEL语言、WS-CDL语言等)一般只关注于语法层面,由于缺失足够的语义信息,无法支持面向服务的语义层面程序设计,影响了程序设计的灵活性和鲁棒性。通过对已有面向服务的程序设计语言的分析,提出并设计了一种以语义信息处理为基础、以语义Web服务为软件构成元素的语义编程语言SPL[2]。

语义程序变换方法是研究面向语义Web服务程序设计的关键任务之一。语义程序是基于语义程序设计语言的Web服务组合模型,它是自描述、非自运行的软件实体。语义程序不同于传统程序,它不仅包含现有面向服务程序设计语言的语法规则,而且还包含通过本体定义的语义元素。语义程序的运行既要保证程序执行的正确性,还要保证程序语义信息的保义性。这就要求语义程序必须通过语法解析,对构成语义程序的主体成分执行有效变换,将语义程序变换为程序运行环境能执行的内存对象。同时,为保证语义程序能够被其他的语义程序或应用系统调用,必须将经过执行变换后的可执行语义程序部署变换为Web服务或语义Web服务对外发布。程序变换既是面向语义Web服务的软件设计方法的基础,也是语义程序运行支撑环境的主要部件。因此,语义程序变换方法的研究,对于面向语义Web服务的软件开发方法的研究具有重要意义。

本文基于SPL语义编程语言,以SPL程序为语义程序研究对象,提出了一种面向语义Web服务的语义程序变换方法。本文第2节介绍了相关研究工作;第3节介绍了语义编程语言SPL;第4节说明了面向语义Web服务的语义程序变换方法;最后总结全文。

到稿日期:2009-10-02 返修日期:2009-12-22 本文受国家重点基础研究发展计划973项目(2007CB310800),国家高技术研究发展计划863项目(2006AA01Z168),国家自然科学基金项目(60773006)资助。

王权于(1971-),男,博士生,副教授,CCF会员,主要研究方向为SOA、语义Web服务;应 时 教授,博士生导师,主要研究方向为互联网上的软件工程、软件体系结构和模式;吕国斌 教授,主要研究方向为网络安全、数据库技术;赵 楷 博士生,主要研究方向为SOA、语义Web服务。

2 相关研究工作

目前，BPEL[3]和 WS-CDL[4]分别成为基于编制(Orchestration)和基于编排(Choreography)的最为重要的 Web 服务组合语言。BPEL 从局部定义了 Web 服务和业务流程之间的交互过程。BPEL 是可执行的流程编制语言，BPEL 程序无需变换就可被 BEPL 执行引擎执行。BPEL 的不足主要表现在：(1)BPEL 缺乏在流程中使用业务规则的能力；(2)BPEL 使用伙伴链接(PartnerLink)来定义目标 Web 服务，导致了流程实现和服务之间点对点的静态绑定。针对这些不足，学术界和工业界进行了深入研究，文献[5]提出了一种面向服务的业务规则和 BPEL 集成方法；文献[6]提出了一种流程和 Web 服务间运行时自适应绑定的方法，但由于缺乏服务发现的语义信息，这种自适应缺乏绑定异常处理机制。WS-CDL 则是从全局角度定义了参与协作的每个角色间的相互交互和协作，WS-CDL 程序需要专用的 WS-CDL 执行引擎执行[7]，文献[8]探讨了从 WS-CDL 到 BPEL 的转换方法。

BPEL 和 WS-CDL 等语言一般只在语法层面进行程序设计，通过组合 Web 服务来构造应用系统。由于分布在互联网上的 Web 服务的异质异构性，同样功能的 Web 服务可能具有不同的接口语法和数据格式，现有面向服务的软件设计语言编写的程序只能根据服务的具体语法进行绑定，无法支持运行时健壮的自适应性绑定。另外，这些语言都是面向 Web 服务的流程描述语言，其本身并不提供对业务规则的支持能力。所有这些造成了所编写的程序缺乏一定的灵活性。

通过对已有面向服务的软件设计语言的分析，我们发现造成上述问题的根本原因在于这些语言只关注语法层面，而并不包含语义层面的语言成分。文献[9]基于 OASIS 语义执行环境技术委员会所定义的语义 SOA 参考本体，给出了参考模型的语义执行环境描述。文献[10]提出了一种面向语义 Web 服务的语义编程语言 BPEL4SWS，BPEL4SWS 语言是对 BPEL4WS 语言的面向语义 Web 服务的一种扩展，通过增加的扩展活动(extensionActivity)来实现与语义 Web 服务的交互。BPEL4SWS 借助语义 Web 服务体系架构参考本体 RO4SSOA[11](Reference Ontology for Semantic Service Oriented Architectures)来定义 BPEL4SWS 需要使用的本体概念模型，为描述带来了极大的灵活性；文献[12]给出了支持 BPEL4SWS 程序的执行引擎程序说明。文献[13]从服务编制角度分析了面向语义 Web 服务的服务计算所面临的数据问题、控制问题和领域建模问题。文献[14]基于语义 Web 服务，提出了一个业务流程的设计和执行框架。所有这些工作为研究面向语义 Web 服务的语义程序变换方法提供了参考。

3 语义编程语言 SPL 概述

SPL(Semantic Programming Language，SPL)语言是一种以语义信息处理为基础、以语义 Web 服务为软件构成元素的语义编程语言。SPL 语言扩展了 BPEL 语言，涵盖了 BPEL 语言的所有语言成分，可以支持顺序、分支、循环、并发等各种控制结构和常规的数据操作；同时，SPL 语言还添加了一系列支持语义层程序设计的语言成分，增加了语义数据来描述 SPL 程序中使用的语义数据类型，增加了业务规则分离业务流程中可变的业务逻辑，增加了语义流程用于定义基于语义 Web 服务的业务流程，并扩展了 BPEL 中的活动以支持语义 Web 服务的访问。

由 SPL 语言编写而成的程序的总体语法结构如图 1 所示。一个 SPL 程序分为定义部分、程序体部分和程序的语义描述部分。(1)定义部分在 BPEL 已有语法成分的基础上，添加了描述语义数据类型、语义服务和语义规则的语法成分。(2)程序体部分由一个主流程和多个子流程构成。流程的语法成分中除了活动、变量、伙伴链接、相关集、事件处理、错误处理和补偿处理这些 BPEL 中已有的语法成分外，还可以定义语义变量，以及通过对 BPEL 中 invoke，receive，reply，assign 等活动进行扩展，支持调用语义服务、触发语义规则和操作语义变量。活动除了基本原子活动以外，还有用于将多个原子活动组织起来的结构化活动，包括顺序、选择、循环和并发活动等。(3)程序的语义描述部分主要描述了程序对外接口的语义信息，以便程序发布后，开发人员和智能代理能基于语义信息发现和访问 SPL 程序。

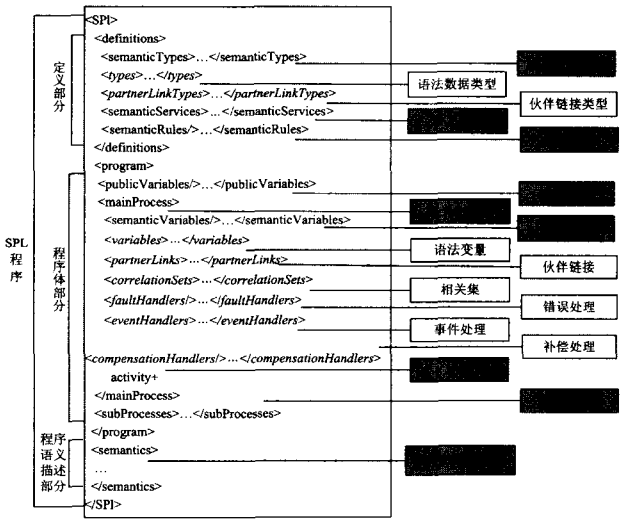

图 1 SPL 程序总体结构

4 面向语义 Web 服务的语义程序变换方法

SPL 程序是基于语义 Web 服务，以语义信息处理为基础的语义程序。SPL 程序需要经过程序变换后才能被执行和调用。程序变换是 SPL 程序设计方法的关键任务之一，用于将 SPL 程序变换为可执行的程序并部署到 SPL 运行环境中，同时将 SPL 程序以服务的形式对外进行发布。SPL 程序变换包括两个连续的变换过程：执行变换和部署变换。执行变换将 SPL 源程序变换成可执行的 SPL 程序；部署变换将已经部署到 SPL 运行环境的 SPL 可执行程序，变换为服务的形式对外发布。

4.1 语义程序的执行变换

SPL 源程序经过 XML 语法解析，提取出构成 SPL 程序的主要语法成分：语义数据、语义规则、语义服务和语义流程；然后分别对解析出的语义数据、语义规则、语义服务和语义流程执行变换。SPL 源程序经过执行变换后，语义数据与相应的通用语法结构相匹配，语义规则被封装为可在 SPL 规则引擎上部署和执行的规则服务，语义服务动态绑定到 Web 服务，语义流程被转换为流程对象；最后将这些经过变换处理的程序构成元素组合成可识别、可部署和可运行的可执行 SPL

程序。图 2 显示了 SPL 程序变换过程。

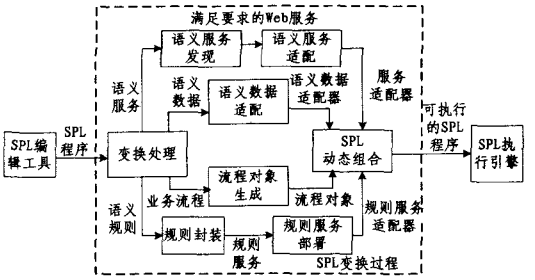

图 2　SPL 程序变换过程图

4.1.1　语义数据类型变换

语义数据类型定义了 SPL 程序用到的带有特定语义信息的数据类型，其中所包含的语义信息由指向本体的概念表示。语义数据类型分为简单语义数据类型和复杂语义数据类型，简单语义数据类型直接在本体中有对应的概念表示，复杂语义数据类型由多个简单语义数据类型组合而成，从而提供对复杂数据的表示。

语义数据类型的变换就是根据其本体的定义生成一个通用的语法数据类型，用于 SPL 程序运行时数据的存储。在语义数据的变换过程中，为每个语义数据类型生成一个语义数据类型适配器，用于实现语义数据和具体语法数据之间的绑定，以完成语义数据和 Web 服务的输入输出数据之间的适配。语义数据类型适配器的结构如图 3 所示。

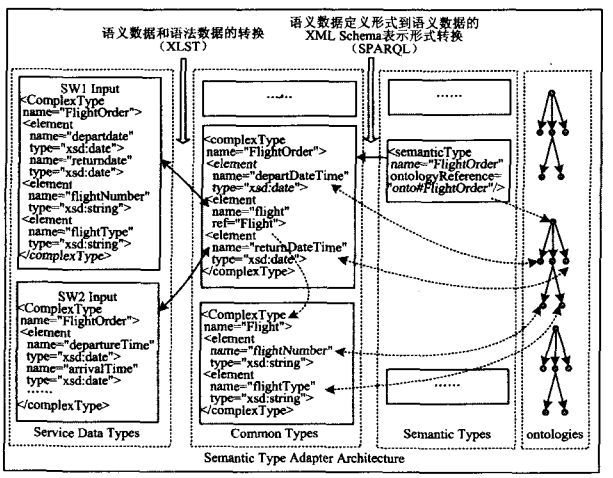

图 3　语义数据类型适配器结构

4.1.2　语义规则变换

业务规则是定义和约束企业业务结构的规范，用于断言业务结构或控制和影响业务的行为。规则分为两种：反应式规则和一致性约束规则。在 SPL 语言中，为了描述业务流程中容易变化的业务逻辑并将其和流程控制分离，采用反应式规则来说明 SPL 程序执行过程中某些事件发生的情况下需要采取的动作；为支持一致性约束规则，应使用反应式规则来模拟，SPL 引擎在每次监视数据发生变化时产生数据变化的事实，然后该事实触发对应的规则。

语义规则变换（如图 4 所示）将 SPL 程序中定义的语义规则转化为 Drools 规则引擎[15]的语法格式，并部署到规则执行引擎中，同时生成一个规则服务来接收规则的调用请求，规则服务是 SPL 流程访问规则的一个接口服务。每一个语义规则都以一个 Web 服务的形式展现给 SPL 语义流程，以便执行引擎执行流程时以标准的 Web 服务调用方式访问语义规则。语义规则中定义的规则部分将变换成具体的业务规则存储在 SPL 规则引擎的规则库中，当规则服务被 SPL 流程调用时，便触发相应的规则的执行。SPL 流程执行过程中，遇到与规则服务进行交互的情况时，中断其自身的执行并将执行控制权交给 SPL 规则引擎，由规则引擎执行相应的动作并将执行结果返回给 SPL 流程。Rule Provider 负责接收 SPL 流程发出的 SOAP 消息，并将规则执行结果响应传给 SPL 流程。

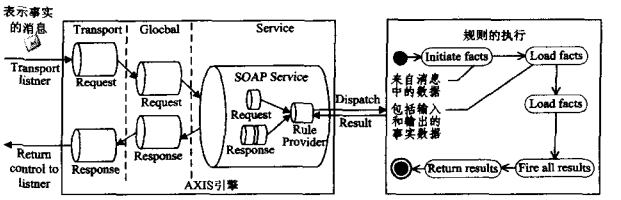

图 4　SPL 语义规则变换

4.1.3　语义服务发现和变换

语义服务发现将语义程序中解析和抽取出来的服务语义信息转换成为服务查询请求表达式，并通过语义 Web 服务搜索接口发现满足要求的服务。变换过程同时为每个语义服务生成一个语义服务适配器（Semantic Service Adapter），并以此来建立语义服务调用操作和 Web 服务调用操作之间的适配关系，将该适配器和返回的服务进行绑定。语义服务适配器将语义服务绑定到多个语义等价语法异构的 Web 服务上，并将语义服务的功能和 Web 服务响应的操作进行绑定，语义服务的输入输出消息和 Web 服务的输入输出消息进行绑定。语义服务适配器的结构如图 5 所示。

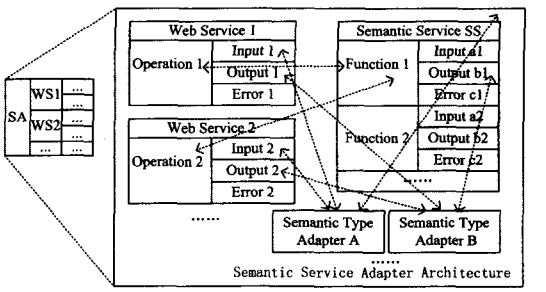

图 5　语义服务适配器结构

SPL 流程执行过程中调用语义服务时，语义服务适配器为其选择最优的 Web 服务，通过 SPL 执行引擎上的 Web 服务调用其相应的 Web 服务，并将执行结果返回给 SPL 流程。

4.1.4　语义流程变换

SPL 程序中的语义流程（Semantic Process）用于定义一组有序的活动来实现特定的业务目标。语义流程指定了流程自身的语义、一组活动和这些活动可能的执行顺序、活动之间共享的数据、参与该流程的语义服务和规则服务，以及一组共同的异常处理机制。语义流程分为主流程和子流程两种，主流程是整个执行逻辑的主程序。

SPL 语义流程变换（如图 6 所示）将 SPL 中定义的流程变换成由各种活动组成的一个流程模型。SPL 语义流程变换包括以下 3 个步骤：1）解析基于 XML 的 SPL 流程定义，并创建相应的 DOM 语法树结构；2）遍历 DOM 语法树结构来实例

（下转第 181 页）

tion project: back and forth between theory and practice[C]// Proceedings of the Twenty-Third ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (Paris, France, June 2004). PODS '04. ACM, New York, NY, 2004: 1-12

[4] Vassiliadis P, Simitsis A, Skiadopoulos S. Conceptual modeling for ETL processes[C]// Proceedings of the 5th ACM international Workshop on Data Warehousing and OLAP (McLean, Virginia, USA, November 2002). DOLAP '02. ACM, New York, NY, 2002: 14-21

[5] 陈有祺. 多根树[J]. 南开大学学报: 自然科学版, 1989(22): 26-28

[6] Furnas G W, Zacks J. Multitrees: enriching and reusing hierarchical structure[C]// Proceedings of the SIGCHI conference on Human factors in computing systems. 1994: 330-336

[7] Hristidis V, Papakonstantinou Y. Discover: Keyword Search in Relational Databases[C]// the 28th VLDB Conference. Hong Kong, China, 2002: 670-681

(上接第 177 页)

化 SPL 流程模型。实例化后的 SPL 流程模型将驻留在 SPL 执行引擎系统中，为具体流程实例的创建提供支持；3）根据 SPL 流程模型创建 SPL 流程实例。

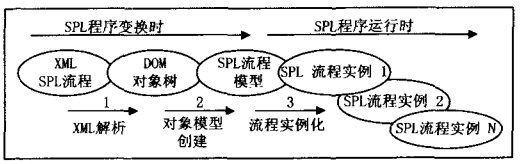

图 6 语义流程变换

流程模型通过语义服务适配器、规则服务适配器、数据适配器动态地将服务和规则组合起来，以支持流程执行。

4.2 语义程序的部署变换

SPL 程序执行变换后生成可执行程序，并被部署到 SPL 执行引擎中，SPL 程序只有通过部署变换后才能被其他的 SPL 程序或应用系统以服务的方式调用。

语义程序的部署变换包含 Web 服务变换和语义 Web 服务变换，部署变换过程就是实现 SPL 可执行程序的服务化，包括 Web 服务化和语义 Web 服务化。Web 服务变换将 SPL 可执行程序变换成对应的 Web 服务的 WSDL 文档，而后者则将 SPL 可执行程序变换成对应的语义 Web 服务，并将其服务描述文档发布到互联网上，SPL 程序的客户端可以采用标准的 Web 服务或语义 Web 服务方式调用，SPL 程序部署变换过程如图 7 所示。

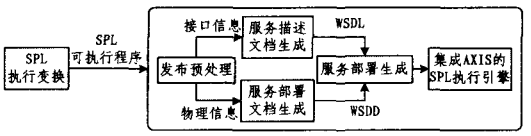

图 7 SPL 程序部署变换

SPL 程序部署变换包括 WSDL 服务描述文档生成、服务部署描述文档生成和服务部署 3 个主要部分。预处理部分获取经过变换后的 SPL 程序的所有对象，根据其中的 SPL 程序对外接口信息，通过服务描述文档生成 SPL 程序对应的 Web 服务的 WSDL 文档，并结合执行引擎物理地址、端口信息等物理信息，通过服务部署文档生成 WSDD 文档，从而将两个文档一并交给服务部署模块后部署在 AXIS 上，以便 SPL 程序运行时能调用 Web 服务访问程序中的规则。

结束语 在语义编程语言 SPL 基础上，提出了一种面向语义 Web 服务的语义程序变换方法。基于语义程序的语法结构和语义信息，该方法不仅给出了语义数据类型、语义规则、语义服务和语义流程等语义信息的执行变换，而且还给出了语义程序到 Web 服务的部署变换。该方法不但能有效地提高面向服务程序设计的灵活性和健壮性，还有助于提高业务流程的柔性和重用性。未来的研究工作是进一步完善本文中的语义程序变换方法，并研究其形式化方法和开发支撑工具，以形成一套完整的、更具有实用价值的语义程序变换体系。

参 考 文 献

[1] Burstein M, Bussler C, Zaremba M, et al. A Semantic Web Services Architecture[J]. IEEE Internet Computing, 2005, 9(5): 72281

[2] 曹虹华，应时，杜德慧，等. 一种面向语义 Web 服务的软件设计语言和设计方法[J]. 电子学报, 2007, 35(12A)

[3] Web Services Business Process Execution Language Version2. 0 [OL]. http://docs. oasis-open. org/wsbpel/2. 0/CS01/wsbpel-v2. 0-CS01. pdf, January 2007

[4] Web Services Choreography Description Language Version 1. 0 [OL]. http://www. w3. org/TR/2005/CR-ws-cdl-10-20051109/, November 2005

[5] Rosenberg F, Dustdar S. Business Rules Integration in BPEL-A Service-Oritented Approach[C]// 7th Internatinal IEEE Conference on E-Commerce Technology(CEC2005). Munich Germany

[6] Karastoyanova, Houspanossian D, Cilia A, et al. Extending BPEL for Run Time Adaptability[C]// Proc. of the 9th International Enterprise Distributed Object Computing Conference (EDOC 2005). Enschede, The Netherlands, September 2005

[7] Kang Zuling, Wang Hongbing, Hung P K. WS-CDL+: An Extended WS-CDL Execution Engine for Web Service Collaboration

[8] Mendling J, Hafner M. From WS-CDL Choreography to BPEL Process Orchestration[J]. Journal of Enterprise Information Management (JEIM), 2008, 21(5)

[9] Norton B, Pedrinaci C, Omingue J. Semantic Execution Environments for Semantics-Enabled SOA

[10] Nitzsche J, van Lessen T, Karastoyanova D, et al. BPEL for Semantic Web Services(BPEL4SWS)[C]// On the Move to Meaningful Internet Systems 2007: OTM 2007 workshops, ser. LNCS. vol. 4805/2007, Springer, 2007: 179-188

[11] RO4SSOA. Reference Ontology for Semantic Service Oriented architectures[OL]. http://docs. oasis-open. org/ex-semantics/sematicsoaro/0. 1, September 2007

[12] van Lessen, Nitzsche T, Dimitrov J, et al. An Execution engine for BPEL4SWS[C]// 2nd Workshop on Business Oriented Aspects concerning Semantics and Methodologies in Service-oriented Computing (SeMSoc) in conjunction with ICSOC. Vienna, Austria

[13] Margaria T. The Semantic Web Services Challenge: Tackling Complexity at the Orchestration Level[C]// Proc. ICECCS'08, 13th IEEE Int. Conf. on Engineering of Complex Computer Systems. Belfast (UK), April 2008

[14] Cao Honghua, Ying Shi, Cui Hua, et al. Towards a Framework for Designing and Executing Business Process Based on Semantic Web Services[C]// International Conference on Wireless Communications, Networking and Mobile Computing. 2008: 1-4

[15] Drools. Drools Boot Camp[EB/OL]. http://www. jboss. org/drools/, 2009