基于 TBAC 的 BPEL 访问控制技术研究*)

刘晓玲 汤 庸 冀高峰 易 盛
(中山大学计算机科学系 广州 510275)

摘 要 业务流程执行语言(Business Process Execution Language for Web Services)是一种可以定义抽象的和可执行的业务流程的语言。然而,BPEL 缺乏访问控制方面的安全性考虑。本文主要关注基于 BPEL 的流程的访问控制的实现方法。我们对现有的基于任务的访问控制模型(Task-Based Access Control)进行扩展,修改了授权结构体的定义并重新定义了授权结构体的类型,并实现了从 BPEL 定义的流程到 TBAC 模型的映射,提出了一种 TBAC 应用到 BPEL 中的方法。本文最后讨论了下一步的研究工作。

关键词 业务流程执行语言,基于任务的访问控制,访问控制

The Research on TBAC-based Access Control of BPEL

LIU Xiao-Ling TANG Yong JI Gao-Feng YI Sheng
(Department of Computer Science, Zhongshan University, Guangzhou 510275)

Abstract Business Process Execution Language for Web Services is a language, which can be used to define abstract and executable processes. However, the security aspect of access control is explicitly mentioned to be outside the scope of BPEL. This paper focuses on the implement of access control in the BPEL-based processes. We extend the existing Task-Based Access Control model, modify the definition of authorization unit and redefine the types of authorization units. Moreover we implement the mapping from the process defined by BPEL to TBAC model, put forwards an approach of using TBAC in BPEL. The future work is mentioned at the end of the paper.

Keywords BPEL, TBAC, Access control

1 引言

BPEL[1]是一种流程组合的语言,可以将松散的服务捆绑在一起,组合成符合业务需要的完整、功能强大的业务流程。在一个业务流程中,构成流程各个环节的 Web 服务之间存在各种依赖关系,Web 服务调用比一个单独的 Web 服务调用需要更多的访问控制。而 BPEL 本身没有提供访问控制的机制。BPEL 定义的流程在任何条件下都可以调用流程中的服务。因而,考虑引入安全模型,为 BPEL 提供访问控制的功能。

TBAC[2]是一种新的安全模型,它采用"面向任务"的观点,从任务(活动)的角度来建立安全模型和实现安全机制,在任务处理的过程中提供动态实时的安全管理。TBAC 作为一种主动安全模型,可以充分地描述和实现 BPEL 流程所需的访问控制。

本文第 2 部分,对相关工作做简要的介绍。第 3 部分,简要介绍 BPEL。第 4 部分,对 TBAC 模型的基本概念进行介绍,并对现有的 TBAC 模型进行扩展,修改了授权结构体的定义,并重新定义了授权结构体的类型。在第 5 部分,我们指出了从 BPEL 定义的流程到 TBAC 模型的映射关系。在第 6 部分,我们提出了一种通过 TBAC 引擎在 BPEL 中应用 TBAC 模型的方法,并给出了 TBAC 引擎的基本结构及各个主要组成部分的功能。同时,我们进一步阐述在 BPEL 中引入 TBAC 模型的优点。最后,对我们的工作进行总结并说明下一步工作的方向。

2 相关工作

本文提出的在 BPEL 中应用 TBAC 的方法是基于工作流系统中访问控制的重要性考虑的。近年来,人们在访问控制方面的研究已取得很大的成果,提出了多种模型。其中,包括最初的 HRU 模型、Take-Grand 模型、自主访问控制模型 DAC 和强制访问模型 MAC 等。1996 年提出的 RBAC 模型[6]的主要思想,是访问权限的许可的授予对象是角色而不是作为个体的用户。在此基础上,又提出了多种对 RBAC 进行扩展的新模型,例如 OASIS 模型[7]和 Temporal-RBAC 模型[8]等。2001 年提出的基于上下文的访问控制模型 Context-Aware Access control[9]则基于 environment role,角色的授予是取决于在被请求时的环境的条件。

然而,工作流系统这一特定的应用环境,对访问控制模型提出了新的要求。Yu 和 Schmid 提出一个为工作流建模的框架,这个框架包含了可被授予给角色的访问许可[10]。文[11]也对工作流系统的访问控制进行研究。文[12]则在 PerDiS groupware platform 的访问控制中将角色与任务进行结合。基于任务的访问控制模型的最早提出是在 Thomas 和 Sandhu 的文[13,14]中。他们提出了 a family of models,支持主动的安全模型。其中,许可是根据当时的任务或流程的状态动态地授予。本文采用的 TBAC 模型是基于邓集波等人在文[2]中描述的基于任务的访问控制模型。

BPEL 是一种典型的组合服务的语言,其定义的流程也是一种工作流。BPEL 没有访问控制[3],从而,Jan Mendlin 等人提出一个 XSLT converter,提供从 BPEL 中抽取 RBAC 模型的功能。抽取的结果可以作为组件 xoRBAC[15,16]的输入,从而能够对 BPEL 的各种角色及其层次结构进行分析。

然而如前所述,RBAC 不是主动的安全模型,不能根据工作流程中的执行状态动态管理权限,不能很好地满足 BPEL

*)基金项目:国家自然科学基金项目(60373081)、广东省自然科学基金项目(04105503,5003348)、教育部"新世纪优秀人才支持计划"资助项目。刘晓玲 硕士研究生,主要研究方向为 CSCW,时态数据库;汤 庸 博士,教授,博士生导师,主要研究方向为数据库与知识库,CSCW 等;冀高峰 博士研究生,主要研究方向为 CSCW 和协同软件;易 盛 硕士研究生,主要研究方向为 CSCW,时态数据库。

定义的流程的访问控制要求。因而，我们将 TBAC 模型引入到 BPEL 中，为 BPEL 提供比 RBAC 模型更灵活的访问控制机制。

3 业务流程执行语言

面向 Web 服务的业务流程执行语言(Business Process Execution Language for Web Services，BPEL 或 BPEL4WS)是一种使用 Web 服务定义和执行业务流程的语言，可将多个 Web 服务组合到一个新的复合服务(称作业务流程)中。

一个 BPEL 流程通过使用基元活动和结构活动来定义 Web 服务执行的顺序。一个基元活动，是一个 Web 服务间的消息交换，或是 BPEL 引擎的本地操作。基元活动包括：invoke，receive，reply，assign，throw，wait，terminate 等。各种基元活动的功能如表 1 所示。其中，invoke，receive，reply 与 Web 服务有关，是本文的访问控制关注的重点。

表 1 BPEL 的基元活动功能

基元活动名称	功能
invoke	调用一个远程 Web 服务
receive	阻塞流程直到一个匹配的消息到达
reply	一个对正在进行的 receive 活动的同步的回答
assign	用于变量的操作
throw	异常和故障处理
Wait	等待一段时间
terminate	结束流程

结构活动是用于组合基元活动的。BPEL 支持几个结构活动，包括 sequence，flow，switch，while，pick 等，功能见表 2。

表 2 BPEL 的结构活动功能

结构活动名称	功能
flow	并发执行
sequence	顺序执行
switch	分支执行
while	循环
pick	类似 receive 的活动

结构活动定义了流程的步骤，流程涉及的各个服务在流程中的先后、同步等依赖关系。因此，它们对 BPEL 的相关访问控制方式的影响很大。

另外，每个 BPEL 业务还将使用〈partnerLink〉定义合作伙伴链接，使用〈variable〉声明变量。

在文[3]中，作者指出在 BPEL 将安全性方面的问题由 WS-Security[4] 处理。WS-Security 提供数据安全方面的安全性，没有涉及到访问控制方面的问题。从本质上来说，一个 BPEL 流程是对所有人公开的，只要这些人可以支持相应的 Partner Link type。然而，在很多基于 Web 的业务环境下，采用更严格的访问控制策略是必要的。同时，在业务流程的执行过程中，一个访问控制的许可被授予，可能取决于执行时的上下文环境。所以，在 BPEL 流程的执行中采用一种主动的访问控制策略是必要的。

4 基于任务的访问控制

基于任务的访问控制(Task-Based Access Control)是一种主动安全模型。它从工作流中的人物角度建模，可以根据任务和任务状态的不同，对权限进行动态管理。TBAC 非常适合于工作流、分布式处理和事务管理系统中的决策制定。

在 TBAC 中，对象的访问权限控制不是静止不变的，而是随着执行任务的上下文环境发生变化的。TBAC 的访问控制的最小单元是授权步(Authorization Step)。授权步表示一个原始授权处理步，是指在一个工作流程中对处理对象的一次处理过程。

在 TBAC 模型中，授权是关联到授权步的。这使得主体获得的权限是有时限的，主体并不是在整个流程的执行过程中都具有某种权限，主体仅在必需的特定授权步才具有权限。每个授权步对应一个受托人集和一个许可集。受托人集是可被授予执行授权步的用户的集合，许可集则是受托人集的成员被授予授权步时拥有的访问许可的集合。

授权结构体(Authorization Unit)是由一个或多个授权步组成的结构体，它们在逻辑上是联接在一起的。任务是工作流程中一个逻辑单元，授权结构体是任务在计算机中进行控制的一个实例。

在原有的 TBAC 模型中，授权结构体有两种类型：一般授权结构体和原子授权结构体。一般授权结构体内部的授权步是顺序执行的。一个原子授权体内部的各个授权步，任何一个授权步失败都会导致整个结构体的失败。

然而，必须指出授权结构体是由授权步组成的这种说法有必要进行修改。这是因为：授权结构体是任务在计算机中进行控制的一个实例。任务是工作流中的逻辑单元，任务一般是以功能来进行划分。任务之中可以包含子任务，任务的概念是可嵌套的。相应地，授权结构体的概念也应该是可嵌套的。授权结构体内部的组成部分可以是授权步，也可以是授权结构体。因而，如下的授权结构体的概念更为合适：授权结构体是由一个或多个授权步或授权结构体组成，这些授权步或授权结构体之间在逻辑上相互联系。

在原有模型中，授权结构体的两种类型，对应着两种不同的结构。然而，实际上仅用这两种结构是不足以表达工作流的各个子活动的组合方式的。在使用 Petri 网描述工作流的时候，Wil van der Aalst 指出路由有四种基本结构：分别是顺序路由、并行路由、选择路由、循环路由[5]。

根据上述四种路由结构，我们定义了四种授权结构体类型(表 3)：顺序结构体、并发结构体、选择结构体、循环结构体。顺序结构体内部的授权结构体或授权步是顺序执行的。并发结构体内部的各个授权体或授权步，任何一个失败都会导致整个结构体的失败。选择结构体是内部的授权结构体或授权步，只要有一个完成，整个结构体就完成。循环结构体内部的授权结构体或授权步是需要重复执行的。

授权步或授权结构体之间的相互关系称为依赖。考虑四种依赖关系：顺序依赖、失败依赖、分权依赖和代理依赖。授权结构体 AS1 顺序依赖授权结构体 AS2 是指：只有当 AS1 完成，AS2 才能被激活。失败依赖是指只有当 AS1 失败，AS2 才能被激活。前两种依赖关系与结构体在流程中的位置有关，反映的是在同一个流程中的不同授权结构体之间的执行顺序的关系。需要说明的是，并不是任何两个结构体都存在以上的两种关系之一，例如，并发的两个授权结构体之间就不存在顺序依赖或失败依赖的关系。

分权依赖是 AS1 和 AS2 必须由不同用户执行，而代理依赖则是 AS1 和 AS2 必须由具有不同权限的不同用户执行。分权依赖和代理依赖是从用户的角度去衡量授权结构体之间的关系，反映与用户相关的访问控制策略。

表 3 授权结构体类型

授权结构体名称	结构
顺序结构体	内部的各个授权结构体或授权步是顺序执行
并发结构体	内部的各个授权体或授权步，任何一个失败都会导致整个结构体的失败
选择结构体	内部的授权结构体或授权步，只要有一个完成，整个结构体就完成
循环结构体	内部的授权结构体或授权步，是需要重复执行的

TBAC 模型的授权采用五元组(S,O,P,L,AS)来表示。其中，S、O、P 分别表示主体、客体、许可，AS 是指授权步，L 则是该授权步的生命期。在授权步 AS 被触发之前，授权五元组的状态是无效的。当授权步 AS 被触发，生命期开始倒计时，五元组有效，委托者开始拥有执行者许可集中的权限。当生命期终止，授权步 AS 被定为无效时，五元组无效，委托执行者所拥有的权限被回收。授权步的状态变化一般自我管理，依据执行的条件而自动变迁状态，但有时也可以由管理员进行调配。授权步的生命期、许可的次数限制和授权步的自我动态管理，三者形成 TBAC 的动态授权。

由上面所述，TBAC 模型由工作流 Wf、授权结构体 Au、受托人集 T、许可集 P 四部分组成。

(1)Wf 由一系列的 Au 组成，Au 之间的关系为上述的四种依赖关系之一。

(2)Au 与 T 是 1:n 的关系，$Au\rightarrow R$，是一个从 $T=\{R_1,R_2,\cdots,R_n\}$选择一个执行委托者的函数。

(3)Au 与 P 是 1:n 的关系，

$F(Au,R)\rightarrow P$，R 属于 T，$P=\{p_1,p_2,\cdots,p_n\}$为许可集，F 为初始化执行者许可集函数；

$G(Au,P_1)\rightarrow P_2$，P_1 包含于 P，$P_2=P-P_1$，G 为权限回收函数。

TBAC 的访问控制策略包含在上述的三类关系中：Au-Au，Au-T，Au-P。Au-Au 决定一个工作流的执行流程，Au-T 和 Au-P 组合决定一个授权结构体的运行。这些关系一般由系统管理员直接配置。

在实际的应用中，使用 TBAC 引擎来管理工作流中的权限问题。TBAC 引擎(或称 TBAC 管理器)可以根据系统管理员的配置，动态地管理工作流中的访问权限，支持多种安全控制原则，例如：

· 最小特权原则：在执行任务时只给用户分配所需的权限，未执行任务或该任务终止后用户不再拥有所分配的权限；而且在执行任务过程中，当某一个权限不再使用时，授权步自动将该权限收回。

· 职责分离原则：有时一些敏感的任务需要不同的用户执行，这可通过授权步的分权依赖实现。

5 从 BPEL 到 TBAC 模型的映射

从前面的介绍中，可以看到 TBAC 模型，可以提供充分灵活的访问控制机制。为一个 BPEL 管理系统建立一个完整的 TBAC 模型，至少需要如下三个步骤的工作：

1)抽取 TBAC 的基本元素；

2)抽取授权结构体；

3)确定各种依赖关系。

在下文中，将进一步详细介绍。

5.1 从 BPEL 中抽取 TBAC 的基本元素

本文主要关注远程服务的访问控制问题，所以只考虑与 Web 服务相关的内容，基元活动仅考虑 invoke，receive，pick。我们把上述的每一个基元活动映射为一个授权步，作为访问控制的最小单元。

Web 服务是流程中的资源，是访问控制保护的对象，即访问的客体 O。将每一个授权步涉及的 Web 服务的操作 operation 的集合映射成许可集，表示主体获得授权时可以执行的操作。受托人集由可以执行该授权步的逻辑用户的集合得到。虽然在 BPEL 中对 Web 服务的调用本身是没有用户概念，但是 Web 服务的服务对象事实上是存在的。因此，我们把 Web 服务的服务对象看作逻辑用户。例如，在分配设备流程中，申请人触发申请设备的服务。对于流程来说，只是一个服务被触发，没有用户概念。而实际上，可以触发申请设备的这些用户，就构成了这一个步骤的受托人集。在一个实例中，真正触发了这个服务的用户，则是获得访问该服务的许可的主体。受托人集由系统管理员提供。

基于上面的映射，我们可以定义 BPEL 流程中的授权五元组(S,O,P,L,AS)。

(1)$User\rightarrow S$ 将用户对应到主体 S。

(2)$Porttype\rightarrow O$ 将 *Web* 服务(使用 *porttype* 表示)对应到客体 O。

(3)$Operation\rightarrow P$ 将 *porttype* 包含的操作 *operation* 对应到许可 P。

(4)基元活动 *invoke*，*receive*，*pick*$\rightarrow$AS 授权步。

(5)AS 对应的基本元的存活期限$\rightarrow L$ 生存期。

5.2 授权结构体的映射

在本文的第 4 部分，我们对授权结构体的类型进行扩展，定义四种类型的授权结构体：顺序结构体、并发结构体、选择结构体和循环结构体，对应顺序、并发、选择、循环四种结构。

我们将 BPEL 中的结构活动作如下的映射：

(1)将 BPEL 中一个 sequence 结构，或顺序执行的若干个 sequence 按需要构成顺序结构体。

(2)把 BPEL 中的每一个 flow 结构活动映射为 TBAC 中的一个并发结构体。在 TBAC 中，对于一个并发授权体的内部的各个授权结构体，任何一个授权结构体失败都会导致整个结构体的失败。这正好体现 BPEL 中的 flow 结构活动内部的各个活动之间关系。

(3)将 BPEL 中的一个 switch 结构活动包含的结构对应一个选择授权结构体。

(4)将 BPEL 中的一个 while 结构活动包含的结构对应一个循环授权结构体。

由于授权结构体是可嵌套的，因此上述的每个结构体的内部可以是授权结构体，内部的授权结构体又可以包含更小的授权结构体。授权结构体的划分，是由授权结构体的实现的功能确定，也即对任务及子任务的划分确定。任务的划分可以由系统管理员设定，也可以选定每个授权步为一个最小的授权结构体，再由不同的组合形式，构成不同的授权结构体。

BPEL 的结构活动不只是上面四种，其它的结构活动可以根据实际的意义转换为上述的四种结构。

根据这种处理方法，我们可以将一个 BPEL 定义的流程抽象为一系列可嵌套的相互联系的授权结构体。

5.3 依赖关系的确定

经过上面两个步骤，我们为 BPEL 流程提取一个 TBAC

模型所需的基本元素：授权步，受托人集，许可集，授权结构体。TBAC的访问控制策略包含在三类关系中：*Au-Au*，*Au-T*，*Au-P*。*Au-Au*的前两种依赖关系（成功依赖和顺序依赖）是由流程的结构决定的，所以授权结构体间的这两种依赖关系与BPEL中的活动的组合方式是相互对应的。授权结构体的分权依赖和代理依赖反映访问控制策略，*Au-T*和*Au-P*组合决定一个授权结构体的运行，这些关系一般由系统管理员直接配置，也可以由系统管理员提供一些策略，再将这些策略转化为模型可以识别的依赖关系加入到模型中来。

6 TBAC引擎的结构

通过抽取模型的基本元素、完成授权结构体的映射、确定依赖关系这三个步骤的工作，一个TBAC模型就可以确立了。建立模型之后，可以将其应用到BPEL的执行过程中，对流程的访问控制进行动态管理。我们提出一个TBAC引擎，来完成相应的功能。

一个TBAC引擎至少应该包含如下方面的功能：

1 能够从BPEL定义的流程中抽取基本元素。

2 可以对模型的基本元素进行解析。

3 提供系统管理员的管理访问控制策略的接口。

4 支持和处理系统管理员输入的访问控制策略。

5 根据系统管理员的数据以及BPEL文件，可以自动检查和动态修改用户的访问权限，即对授权五元组的动态管理。

由此，我们提出一个TBAC引擎的体系结构，如图1所示。

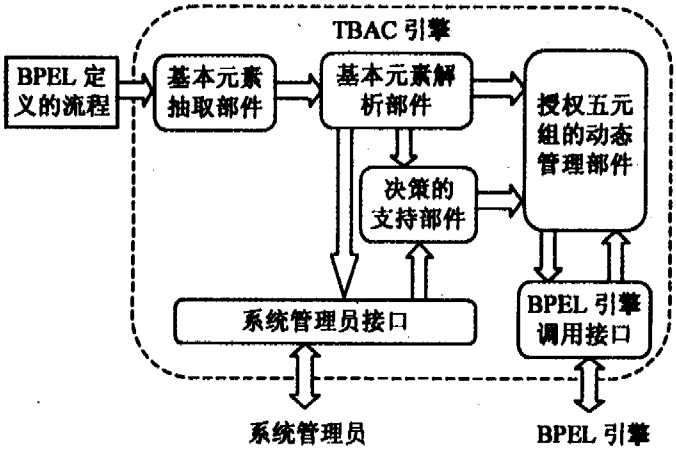

图1 TBAC引擎的结构

基本元素抽取部件的功能是以BPEL文件为输入，得到TBAC模型的基本元素，以XML文件保存。在TBAC中，可以由系统管理员提供访问控制的策略及各种依赖关系，因此，必须提供系统管理员接口。决策支持部件提供将系统管理员的输入转化为TBAC可识别的知识的功能。授权五元组的动态管理部件是TBAC引擎的核心部件，主要功能是根据模型基本元素和访问控制策略，动态地管理授权五元组，从而动态地管理流程中的访问控制。BPEL引擎通过BPEL引擎调用接口调用TBAC引擎，实现在BPEL引擎执行流程的过程中动态管理访问控制权限。

通过将TBAC模型引入到BPEL中，并采用TBAC引擎作为在BPEL引擎中的TBAC模型处理部件，可以为BPEL的流程执行提供基于任务的主动安全模型，管理流程执行过程的访问控制权限问题，其优点体现在以下三个方面：

1）打破BPEL流程作为一个超级用户在任何时候都拥有权限这个特权。引入模型之后，权限的授予与拥有是与授权步相联系的。用户仅在特定的授权步具有许可。这使得权限的获得具有时限，符合最小特权原则。

2）对权限的管理是基于上下文的。在具体的任务实例的执行过程中，可以根据流程的执行情况，动态地修改权限。

3）支持多种系统管理员自定义的访问策略。通过系统管理员按实际需要给出各个授权结构体之间的依赖关系，支持不同的访问控制策略。也可以预先设计好一些常用的访问策略，以供系统管理员选择。这样可以根据具体的应用环境和不同的流程，提供不同的访问策略以满足不同的访问控制需要。

结论和展望 在本文中，我们提出了一种BPEL与TBAC模型结合的方法。我们的工作主要是基于下面的两个事实：第一，BPEL没有提供访问控制机制，然而访问控制是一个业务流程中重要的不可缺少的部分。第二，TBAC是一种基于上下文的主动的访问控制模型，能够为工作流管理系统提供灵活的安全机制。我们提供了BPEL与TBAC模型的映射关系。在这个映射关系的基础上，我们提出了一种通过TBAC引擎在BPEL中应用TBAC模型的方法，并给出了这种TBAC引擎的基本结构及各个主要组成部分的功能。

我们的方法旨在提高工作流管理系统的安全性。通过在BPEL中应用TBAC模型，可以实现对BPEL的访问控制权限的动态管理，支持系统管理员自定义的访问策略，可以根据具体的应用环境选择不同的访问策略，满足不同的访问控制需要。在我们的下一步工作中，我们将开始本文讨论的TBAC引擎的进一步详细设计与实现。另一个我们感兴趣的方向是如何将系统管理员需要的策略自动地转化为模型中的依赖关系的具体表示。

参考文献

1 Andrews T, Curbera F, Dholakia H, et al. Business Process Execution Language for Web Services, Version 1.1. Specification, BEA Systems, IBM Corp., Microsoft Corp., SAP AG, Siebel Systems, 2003

2 邓集波，洪帆. 基于任务的访问控制模型. 软件学报，2003，14(1)

3 Mendling J, Strembeck M, Stermsek G, et al. An Approach to Extract RBAC Models from BPEL4WS Processes. In: Proceedings of the 13th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE'04), 2004

4 Atkinson B, Della-Libera G, Hada S, et al. Web Services Security. Specification, IBM Corp., Mircosoft Corp., VeriSign, Inc., 2002

5 van der Aalst W, van Hee K. 工作流管理——模型、方法和系统. 王建民，文立杰，等译. 清华大学出版社，2004

6 Sandhu R S Coyne E J, Feinstein H L, et al. Role-Based Access Control Models. IEEE Conputer, 1996, 29(2): 38～47

7 Yao W, Moody K, Bacon J. A Model of OASIS Role-Based Access Control and its Support for Active Security. In: ACM Symposium on Access Control Model and Technology. ACM, Chantilly, VA,

2001
8 Bertino E, Ferrari E, Bonatti P A. TRBAC: A Temporal Role-Based Access Control Model. In: Proceedings of 5th ACM Workshop on Role-Based Access Control. Berlin, Germany, 2000
9 Convington M, Long W, Stinivasan S, et al. Securing Context-aware Applications Using Environment Roles. In: ACM Symposium on Access Control Model an Technology. Chantilly, VA, 2001
10 Zhang L, Ahn G, Chu B. A Rule-Based Framework for Role-Based Delgation and Revocation. ACM Transactions on Information and System Security(TISSEC), 2003, 6(3)
11 Atluri B, Huang WK. An Authorization Model for Workflows. In: Proceedings of the 5th European Symposium on Research in Computer Security, Lecture Notes in Computer Science Vol. 1146, Springer-Verlag, 1996. 44～64
12 Coulouris G, Dollimore J, Roberts M. Role and Task-Based Access Control in the PerDiS Groupware pPlatform. In: Proceedings of the ACM Workshop on Role-Based Access Control. George Mason University, 1998. 115～121
13 Thomas R, Sandhu R. Task-Based Authorization Controls (TBAC): A Family of Models for active and enterprise-oriented authorization management. In: Proc. of the IFIP WB11. 3 Conference on Database Security, August 1997
14 Thomas R, Sandhu R. Task-Based Authorization Controls (TBAC): Models for Active and Enterprise-oriented Authorization Management. In: Database Security XI: Status and Prospects, T. Y. Lin and X. Qian, eds. North Holland, 1997
15 Neumann G, Strembeck M. Design and Implementation of a Flexible RBAC-Service in an Object-Oriented Scripting Language. In: Proc. of the 8th ACM Conference on Computer and Communications Security (CCS), 2001
16 Neumann G Strembeck M. An Approach to Engineer and Enforce Context Constraints in an RBAC Environment. In: Proc. of 8th ACM Symposium on Access Control Models and Technologies (SACMAT), 2003

(上接第 131 页)

之间的三种关系：等价关系(⇔)、蕴涵关系(⇒)、没有关系(null)，操作动态行为匹配度以如下方法计算：

(1)对于替换组合：当源操作的模式是 *In/Out* 时，如果 $\forall PreC_j^n \in R_j$，$\exists PreC_i^n \in R \mid (PreC_i^n \Leftrightarrow PreC_j^m)$ 且 $\forall PostC_i^n \in R_i$，$\exists PostC_j^m \in R_j \mid (PostC_j^m \Leftrightarrow PostC_i^n)$；或者 $\forall PreC_j^m \in R_j$，$\exists PreC_i^n \in R_i \mid (PreC_i^n \Rightarrow PreC_j^m)$，且 $\forall PostC_i^n \in R_i$，$\exists PostC_j^m \in R_j \mid (PostC_j^m \Rightarrow PostC_i^n)$，我们称 op_i 和 op_j 是行为精确匹配的，匹配度设为 1；否则动态行为匹配度的计算公式如下：

$$C_{op}^{Behaviour} = \frac{\sum_{i=1}^{n} C_{op}^{Behaviour}(PreC_{ik}, PreC_{jl}) + \sum_{k=1}^{m} C_{op}^{Behaviour}(PostC_{jl}, PostC_{ik})}{N+M} \tag{8}$$

其中：N 表示操作 op_j 中前置条件个数；$\sum_{j=1}^{n} C_{op}^{Behavior}(PreC_{ik}, PreC_{jl})$ 表示操作 op_j 中前置条件匹配度的总和；M 表示操作 op_i 中后置条件个数；$\sum_{k=1}^{m} C_{op}^{Behavior}(PostC_{jl}, PostC_{ik})$ 表示操作 op_i 中后置条件匹配度的总和。

(2)对于替换组合：当源操作的模式为 *Out/In* 时，精确匹配的计算与(1)相似，匹配度设为 1；否则动态行为匹配度的计算公式如下：

$$C_{op}^{Behavior} = \frac{\sum_{i=1}^{n} C_{op}^{Behavior}(PostC_{ik}, PreC_{jl}) + \sum_{k=1}^{m} C_{op}^{Behavior}(PostC_{il}, \Pr eC_{ik})}{N+M} \tag{9}$$

(3)对于顺序组合：操作模式是“*In/Out*”或者“*Out/In*”，如果 $\forall \Pr eC_j^n \in R_j$，$\exists r \in (\Pr eC_i \cup PostC_i) \mid (r \Leftrightarrow \Pr eC_j^n)$；或者 $\forall \Pr eC_j^n \in R_j$，$\exists r \in (\Pr eC_i \cup PostC_i) \mid (r \Rightarrow \Pr eC_j^n)$，我们称 op_i 和 op_j 是行为精确匹配的，匹配度设为 1；否则动态行为匹配度的计算公式如下：

$$C_{op}^{Behavior} = \frac{\sum_{i=1}^{n} C_{op}^{Behavior}((\Pr e_{ik} \cup Post_{ik}), \Pr eC_{jl})}{N} \tag{10}$$

式(6)说明了语法匹配度计算方法，式(7)说明了静态语义匹配度计算方法，式(8)、(9)、(10)说明了操作动态语义匹配度的计算方法。综合这三个方面匹配度的计算方法，操作层总的匹配度的计算方法如式(11)所示。其中，ω_{op}^{Syntax} 表示语法匹配的权重，ω_{op}^{Static} 表示静态语义匹配的权重，$\omega_{op}^{Behaviour}$ 表示操作动态语义匹配的权重，其取值范围 $\omega_{op}^{Syntax} \in [0,1]$，$\omega_{op}^{Static} \in [0,1]$，$\omega_{op}^{Behaviour} \in [0,1]$，且 $\omega_{op}^{Syntax} + \omega_{op}^{Static} + \omega_{op}^{Behaviour} = 1$。

$$C_{op} = \omega_{op}^{Syntax} \times C_{op}^{Syntax} + \omega_{op}^{Behaviour} \times C_{op}^{Behaviour} + W_{op}^{Behaviar} \times C_{op}^{Behaviar} \tag{11}$$

4.4 多层次服务匹配度计算

第 4.1～4.3 节分别从消息参数、消息、操作三个层次，进行 Web 服务接口的匹配。表 1 综合了各匹配的层次以及相关的权重，用户可根据具体应用自行定义这些权重。消息层匹配度的计算，依赖于消息参数层匹配度计算的结果，因此我们从消息层和操作层两方面，计算一个服务操作的匹配度，如式(12)所示。其中，C_M 为 4.2 节消息层匹配度计算结果，ω_M 为消息层匹配的权重；C_{op} 为 4.3 节操作层匹配度计算结果，ω_{op} 为操作层匹配的权重，$\omega_M + \omega_{op} = 1$。

$$C = \omega_M \times C_M + \omega_{op} \times C_{op} \tag{12}$$

多层次接口语义匹配模型能够量化地计算服务接口之间的相似程度，对候选 Web 服务实例按匹配相似程度排序，从而能够动态地从当前系统中，选择最优的服务参与组合。作者将在另一篇文章中详细介绍基于该匹配模型的 Web 服务自动组合过程。

总结 WSDL 是当前 Web 服务接口描述语言的标准，但是，它对接口的描述只限于语法层次，不能表达语义信息。本文通过对 WSDL 接口描述语言进行语义扩展，提出了一种基于扩展接口语义的 Web 服务描述模型。该模型从服务接口的消息参数层、消息层和操作层分别扩展语义描述，从而能够全方位地描述 Web 服务的功能以及服务质量等属性，并且这种语义描述模型，使得语义信息可以直接应用到当前的工业标准中。在此基础上，提出了多层次 Web 服务接口语义匹配模型。该模型能够对候选 Web 服务实例按匹配相似程度排序，从而能够动态地从当前系统中，选择最优的服务参与组合，这也是我们下一步的研究工作。

表 1 Web 服务接口语义匹配的层次和相应权重

消息参数层		消息层			操作层	
参数类型	语义概念	消息参数	消息类型	语法	静态语义	动态语义
ω_P^{DT}	ω_P^{Smcep}	ω_M^{P}	ω_M^{MT}	ω_{op}^{Syntax}	ω_{op}^{Static}	$\omega_{op}^{Behavion}$
ω_M					ω_{op}	

参 考 文 献

1 Casati F, Shan M C. Definition, Execution, Analysis, and Optimization of Composite E-Services. IEEE Data Engineering Bulletin, 2001, 24(1): 29～34
2 Akkiraju R, Goodwin R, Doshi P, et al. A Method for Semantically Enhancing the Service Discovery Capabilities of UDDI. In: Proceedings of IJCAI-03 Workshop on Information Integration on the Web (IIWeb-03), Acapulco, Mexico, August 2003. 87～92
3 Lara R, Roman D, Polleres A, et al. A Conceptual Comparison of WSMO and OWL-S. In: Proceedings of European Conference on Web Services (ECOWS 2004), Erfurt, Germany, September 2004. 254～269
4 Christensen E, Curbera F, Meredith G, et al. Web Services Description Language (WSDL)1. 1. http://www.w3.org/TR/wsdl, March 2001
5 Bechhofer S, Harmelen F, Hendler J, et al. OWL Web Ontology Language Reference. World Wide Web Consortium. http://www.w3.org/tr/owl-ref, February 2004