一种基于匹配策略的 Web 服务组合方法

郑永清　梁　伟
（山东大学计算机科学与技术学院　济南 250061）

摘　要　现有的 Web 服务只能被单独调用，不能提供复杂的组合服务。文中提出了一种基于匹配策略的 Web 服务组合方法，该方法利用匹配关系，通过自顶向下的服务分解和自底向上的服务组合，自动将原子服务组合成满足用户要求的组合服务。通过实例说明该方法的执行过程，应用结果表明该方法减少了用户的交互操作，降低了服务成本，提高了服务效率。

关键词　Web 服务，服务匹配，服务组合，业务流程

A Matching Strategy-Based Web Service Composition Method

ZHENG Yong-Qing　LIANG Wei
(College of Computer Science and Technology, Shandong University, Jinan 250061)

Abstract　Existing Web service can be independently invoked and is incapable of provide complex composite service. A matching strategy-based Web service composition method is put forward in this paper, using matching relation, some atomic services been automatically composed into a meet user's requirements composite service though Top-down service decomposition and Bottom-up service composition in this method. The executive process of the method is illustrated by an instance, results indicate this method decreases user mutual operation, decreases cost of service and improves efficiency of service.

Keywords　Web service, Service matching, Service composition, Business process

1　引言

随着信息技术的发展，人们希望能从网络上得到更多更复杂的服务，而不仅仅是简单的独立的服务，从而需要将各种服务进行组合生成满足用户要求的组合服务。但是对于一般用户来说，他们不知道服务的分布和机制，缺少服务组合的知识，并且应用请求大多是临时的、不确定的，因而能够自动地进行服务查找、服务组合和服务执行的系统成为当前的迫切需求。

业务流程[1]是一组将输入转化为输出的相互关联或相互作用的活动，具有目标性、内在性、整体性、动态性、层次性、结构性等特点。Web 服务[2]是一种部署在 Web 上的对象/组件，它具备完好的封装性、松散耦合、使用协约的规范性、使用标准协议规范、高度可集成能力等特点；Web 服务的努力目标是通过使用 Web 标准实现应用程序间的通用的互操作性。Web 服务使用松散耦合的集成模型以支持各种领域（包括企业到消费者、企业到企业和企业应用程序集成）中的各种系统的灵活集成。

业务流程技术和 Web 服务结合进行服务的自动组合是一种备受关注的方法。Manish Agarwal[3]、Lerina Aversano[4]等提出了利用已有 Web 服务，按照预先定义的目标和限制等条件进行服务组合的方法，它注重某一服务域内的 Web 服务之间的协作，组成一个功能更强的服务；但是它对不同服务域之间的服务缺少协调，并且单一基于参数的组合难以控制组合的趋势，效率较低。Chintan Patel[5]、Junwei Cao[6]、Srinivas Padmanabhuni, Jai[7]等提出了已有业务流程中的任务在执行时动态绑定具体服务来满足用户需求的方法，它降低了因服务的动态变化而导致系统执行失败的可能性，并能高效地利用已有服务；但是这种方法需要业务流程的预先设计，难以满足即时的用户需求，降低了系统的灵活性。

本文提出了一种基于匹配策略的 Web 服务组合方法，它综合已有方法的优点，利用匹配的思想，在已有 Web 服务资源的基础上，通过自上向下的服务分解和自下向上的服务组合，自动地将原子服务组合成满足要求的组合服务，提高了服务组合的动态性和即时性，最大程度地满足用户的需求。业务流程技术的运用进一步确保了服务组合过程的合理性，提高了系统的灵活性和重用性。

本文在第 2 部分介绍了基于匹配策略的服务组合方法中的主要概念和关键技术，第 3 部分给出该方法的执行模型，第 4 部分通过一个实例说明基于匹配策略的服务组合过程，最后对本文进行总结并对需要进一步研究的方向进行探讨。

2　基于匹配策略的 Web 服务组合方法中的关键技术

2.1　服务匹配

服务由不同的提供者创建和发布，用户可以通过一定的机制来调用执行某个服务。由于服务和网络环境的动态性，指定的某个服务可能不再适用，为了提高系统的健壮性和灵活性，需要寻找一个可以替代该服务的服务来继续系统的运行，因此便引出服务匹配的问题。本文所采用的服务匹配是基于服务的输入、输出参数的名称和数据类型，以及服务的类型和角色，这些都是描述一个服务最基本的信息，为了提高服务查找的效率，在服务发布时将这些信息保存在 UDDI 中。

定义 1（匹配关系）　如果 B 满足 A 的所有功能要求，则称 B 匹配于 A，记为 $B \propto A$；其中 $\propto$ 称为匹配关系，其满足自反性、非对称性、传递性。

本文认为两个服务之间的匹配应当满足三个层次上的要求：服务类型等同、角色的匹配和参数的匹配。本文分别用 C、R、P 表示服务类型、角色、参数的集合。

第一个层级是服务类型等同。本文中将 Web 服务分为

三种类型:原子服务(A)、组合服务(C)和流程服务(P)。

定义 2(原子服务) 具有一个或多个角色,提供不需要再分解的服务,可被调用执行并完成一个任务的 Web 服务。

定义 3(组合服务) 由原子服务动态组合而成,在系统中不独立存在,可以被调用执行。

定义 4(流程服务) 一种特殊的 Web 服务,它的表现形式同原子服务相同,但是不能被调用执行。其中的一个输出参数以服务过程描述文本的形式存在,描述其中各个子服务的输入参数、输出参数、角色及服务执行的逻辑关系,用来完成某一个任务。

对于 $c_1 \in C, c_2 \in C$,如果 c_1、c_2 表示同一个服务类型,则称 c_1、c_2 服务类型等同,记为 $c_1 = c_2$。

第二个层次是角色的匹配。

定义 5(角色) 服务的某一类功能行为称为一个角色,它表示服务的性质和能力。

角色是一个层级结构,子角色具有父角色的所有功能并对其进行相应的扩充,因此角色是向上兼容的,本文采用分类法表示角色。对于角色 $r_1 \in R, r_2 \in R$,如果 r_2 是 r_1 的本身或子角色,则称 r_2 匹配于 r_1,记为 $r_2 \propto r_1$。

第三个层次是参数的匹配。参数的匹配包括参数名称的匹配、参数数据类型的匹配和参数数目的匹配三个方面,本文分别用 N、T 表示名称的集合、数据类型的集合。

对于 $n_1 \in N, n_2 \in N$,如果 n_2 in synonymy(n_1),其中 synonymy(n_1)表示 n_1 的同义词集合,则称 n_2 匹配于 n_1,记为 $n_2 \propto n_1$。

同角色的层次关系相似,数据类型也是向上兼容的,对于 $t_1 \in T, t_2 \in T$,如果 t_2 是 t_1 的本身或子类型,则称 t_2 匹配于 t_1,记为 $t_2 \propto t_1$。

对于 $p_1 \in P, p_2 \in P$,如果 $np_2 \propto np_1 \wedge tp_2 \propto tp_1$,则称 p_2 匹配于 p_1,记为 $p_2 \propto p_1$,其中 np_i、tp_i 表示参数 p_i 的名称、数据类型。

对于参数集,如果 $Sp_1 \subset P, Sp_2 \subset P$,如果 $\forall p_i \exists p_j (p_j \propto p_i \wedge p_i \in Sp_1 \wedge p_j \in Sp_2)$,则称 Sp_2 匹配于 Sp_1,记为 $Sp_2 \propto Sp_1$。

本文用 S 标识服务的集合,对于 $s_1 \in S, s_2 \in S$,如果 $c_1 = c_2 \wedge r\_s_2 \propto r\_s_1 \wedge Spi\_s_1 \propto Spi\_s_2 \wedge Spo\_s_2 \propto Spo\_s_1$,则称 s_2 匹配于 s_1,表示为 $s_2 \propto s_1$,其中 c_1、c_2、r_s_1、r_s_2、Spi_s_1、Spi_s_2、Spo_s_1、Spo_s_2 分别表示 S_1、S_2 的角色、输入参数集合、输出参数集合。

对于一个服务 $s_i \in S$,可能存在许多与它匹配的服务,因此需要进行相应的评估以决定那个服务是最佳的。本文利用服务执行时间 t 和服务执行成本 c 的加权值 value 来判断其优劣,认为加权值 value 最小的服务最佳,value = weight(t,c) $= t * wt + c * wc$,其中 wt,wc 分别为服务执行时间 t、服务执行成本 c 的权值,且 $wt + wc = 1$。

对于每个服务 sr 的匹配请求,生成如下类结构化的查询语言用于在 UDDI 中查询与其匹配的服务。

查询语句的格式为:

```
select id,address,weight(t,c)
from UDDI
where r∝r_sr1 and Spi_sr ∝Spi and Spo∝Spo_sr and
      category=category_sr
and rownum=1
order by weight(t,c);
```

其中 *id*、*address*、*r*、*Spi*、*Spo*、*category* 分别表示服务的标识、地址、角色、输入参数集合、输出参数集合、类别。这个查询语句表示从某个 UDDI 中查找一个最优的匹配于服务 sr 的服务。

2.2 服务分解

流程服务不能被调用执行,它通过一定的形式描述了其中各个子服务的输入参数、输出参数、角色及子服务间的逻辑关系。为了使一个流程服务完成其对应的角色功能,必须将其中的每一个子服务对应一个或一组原子服务,由这些匹配得到的原子服务代替该流程服务,从而完成一个服务的分解。本文采用业务流程来表示服务间的逻辑关系,将每个子服务看为业务流程中的一个活动。

对于一个流程服务 sp,可以从其输出参数 process 中获取相应的服务过程描述文档,然后将其转换为规定的流程描述格式,得到一个流程图 G_0。将 G_0 中的每一个活动看作一个新的任务进行处理,直到每一个活动都可以由一个或一组原子服务完成,此时 G_0 变为 AOV 图 G_1,其中每一个活动节点对应一个原子服务。

本文利用流程 G_1 的加权值 value 来评价 G_1 所表示的一组服务的性能,通过 BPWV 算法可以计算出 G_1 的加权值 value。

BPWV 算法:

输入:G_1 中各个活动节点的前驱后继关系及对应服务的执行时间 t 和执行成本 c,权值 wc、wt。

输出:流程 G_1 的加权值 value。

步骤 1:利用图的深度遍历算法删除掉 G_1 中的循环路径得到一个有向无环的 AOV 图 G_2;

步骤 2:将 G_2 转换为等价的 AOE 图 G_3;

步骤 3:利用关键路径算法得到 G_3 的关键路径和关键活动;

步骤 4:G_3 中所有关键活动对应的服务的执行时间之和为流程 G_1 的执行时间 t;

步骤 5:G_1 所有活动节点对应的服务的执行成本之和为流程 G_1 的执行成本 c;

步骤 6:流程 G_1 的加权值 value = weight(t,c) $= t^* wt + c^* wc$,其中 wt,wc 分别为执行时间 t、执行成本 c 的权值,且 $wt + wc = 1$。

BPWV 算法不考虑循环路径的情况,并且认为每个活动都被执行一次,因此所得的加权值 value 仅是一种估测。

设流程服务 sp 对应原流程图中一个活动节点 V_i,建立 V_i 的每一个前驱活动节点与 G_1 中开始活动节点 V_s 的有向连接弧;建立 G_1 中结束活动节点 V_e 与 V_i 的每一个后续活动节点的有向连接弧;删除 V_i 及与其相关的连接弧。这样就把所得到的流程 G_1 作为一个局部子流程纳入整个流程之中,完成一个服务的分解。

2.3 服务组合

当对一个服务 sr 查找不到与其匹配的服务时,便需要通过服务组合将若干个小的原子服务或组合服务组合成满足匹配要求的一个新的组合服务。服务组合可以利用的信息与服务匹配时相同,但是在本文采用的方法仅是基于输入、输出参数的匹配。对于所得到的组合服务 sc,应当满足以下条件:

$Spi\_sr \propto Spi\_sc \wedge Spo\_sc \propto Spo\_sr$,其中 Spi_sr、Spi_sc、Spo_sr、Spo_sc 分别为 sr、sc 的输入、输出参数集。

本文采用反向链算法(BWC)由 sr 的输出参数开始逐步得到一组合适的服务使其满足上述的要求。

BWC 算法:

输入:Spi_sr,Spo_sr。

输出:有向无环图 G_0 及 G_0 中每个节点对应的服务。

步骤 1：将 Spi_sr 作为一个源节点 V_0，Spo_sr 作为一个目标节点 Vg_0；

步骤 2：建了目标节点集合 Svg，初始值 Svg={Vg_0}；

步骤 3：如果系统超时，服务组合失败，返回；

步骤 4：如果 Spi_sr$\propto\bigcup$Spi_vgj，(其中，$\bigcup$(Spi_vgj)表示所有属于 Svg_0 的节点所对应的服务的输入参数的集合，$Svg_0\subset$Svg)，则对 $\forall Vg_j\in Svg_0$，Svg=Svg−{Vg_j}，并建立 V_0 到 Vg_j 的有向连接弧；

步骤 5：如果 Svg=Φ，服务组合成功，返回；

步骤 6：从 UDDI 中查找满足以下条件的服务集合 Ss：$\bigcup$(Spo_si)$\propto\bigcup$(Spi_vgj)，(其中$\bigcup$(Spo_si)表示所有属于 Ss 的服务的输出参数的集合，$\bigcup$(Spi_vgj)表示所有属于 Sv 的节点所对应的服务的输入参数的集合，Sv$\subset$Svg)；

如果没有得到满足条件的服务，服务组合失败，返回；

否则，将 $\forall s_i\in$ Ss 作为一个新的目标节点 Vg_i，Svg=Svg$\bigcup${Vg_i}，对于 $\forall Vg_j\in$ Sv，Svg=Svg−{Vg_j}，并建立 Vgi 到 Vgj 的有向连接弧，跳转到步骤 3 继续执行。

在步骤 6 中，如果 $|Ss|=1\wedge|Sv|=1$，则得到一个顺序结构；如果 $|Ss|>1\wedge|Sv|=1$，则得到一个汇聚结构；如果 $|Ss|=1\wedge|Sv|>1$，则得到一个分支结构；本文暂不考虑 $|Ss|>1\wedge|Sv|>1$ 的情况。

如果服务组合成功返回则得到一个以 V_0 为源节点，Vg_0 为汇点的有向无环图 G_0，并且 G_0 是拓扑有序的。将 G_0 中 V_0、Vg_0 看作权值 value 为 0 的活动，其余节点看作权值为其对应的服务的加权值 value 的活动，于是 G_0 变为一个活动图(AOV)G_1。

同样可以利用 2.2 节中的 BPWV 算法得到组合服务 *sc* 的加权值 value。

设组合服务 *sc* 对应原流程图中一个活动节点 V_i，建立 V_i 的每一个前驱活动节点与 V_0 的每一个后续活动节点的有向连接弧；建立 Vg_0 的每一个前驱活动节点与 V_i 的每一个后续活动节点的有向连接弧；删除 V_i、V_0、Vg_0 及与其相关的连接弧。这样就把组合服务 *sr* 作为一个局部子流程纳入整个流程之中，完成了一个服务的组合。

2.4 服务抽象

当为实现用户请求而创建的业务流程中的每一个活动节点都对应于一个原子服务时，便得到一组可以完成用户请求的原子服务，并且通过业务流程来描述它们之间的逻辑关系。为了提高服务的重用性和对动态环境的适应性，本文将得到的一组服务抽象为一个流程服务 sp，并将 sp 在 UDDI 中注册，当下一个相同的用户请求到来时便可以使用该服务。

本文将所得到业务流程和具体的原子服务分离形成抽象业务流程，其中的活动不再依赖于某个具体的服务；同时，为了利用已有的匹配服务，提高程序的执行效率，在抽象流程中的每一个活动的描述中增加一项用于表示与该活动匹配的原子服务。本文采用 BPEL4WS(业务流程执行语言)[8] 描述抽象流程的格式。

对于服务分解时由流程服务所得到的活动节点，在抽象流程中保留其对应的输入、输出参数和角色信息；对于服务组合时由原子服务所得到的活动节点，在抽象流程中保留该服务的输入参数和角色信息，但是仅采用其后续活动节点用到的输出参数作为抽象活动节点的输出参数，这样避免了服务匹配时参数的单向增长性。

流程服务 sp 的输入参数、输出参数、角色是从用户请求信息中得到的，通过信息解析可将其转换符合 UDDI 的格式，它的服务过程描述文档便是所得到的抽象业务流程。

3 基于匹配策略的 Web 服务组合方法的执行模型

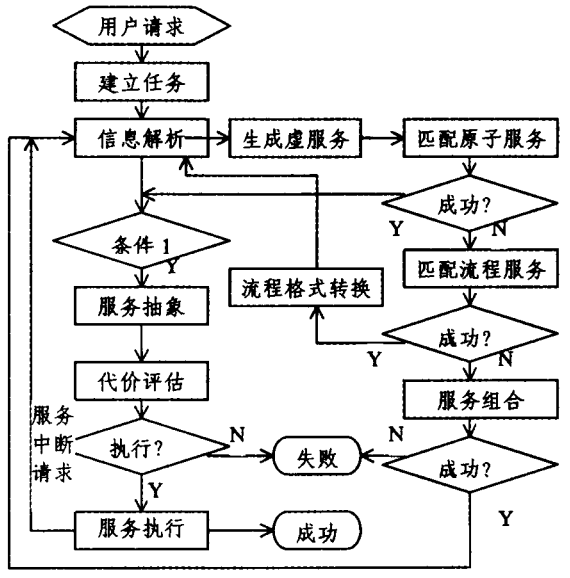
为了便于服务匹配的处理，本文引入虚服务的概念。

定义 6(虚服务) 根据 Web 服务的描述规范，利用一定的输入、输出参数信息和角色信息形成的临时服务，不能被调用和执行。

系统接收到用户的请求后建立一个任务，生成由三个活动节点组成的初始业务流程，其中活动节点 V_0 的输入、输出参数对应于用户请求的输入、输出参数信息，角色为用户请求的服务对应的角色；活动节点 Vs 为 V_0 的前驱活动节点，代表整个流程的开始活动节点，输入、输出参数同为 V_0 的输入参数；活动节点 Ve 为 V_0 的后继活动节点，代表整个流程的结束活动节点，输入、输出参数同为 V_0 的输出参数；并且 Vs、Ve 绑定一个系统内置的原子服务。然后对 V_0 进行信息解析生成一个虚服务，查找与其匹配的原子服务；在得不到匹配的原子服务的情况下则查找与其匹配的流程服务，并将得到的子流程中的每一个活动作为新的虚服务进行处理；在得不到匹配的流程服务的情况下则进行服务组合，生成一个与虚服务匹配的组合服务。当生成的流程中的每一个活动节点都对应一个具体的原子服务时，将所得到的流程进一步处理为抽象流程，生成一个流程服务。

利用 2.2 节中的 BPWV 算法得到业务流程所绑定的一组原子服务的加权值 value，如果用户接受加权值 value，则将业务流程交付给执行引擎，按照业务流程的描述逐次调用、执行绑定的原子服务。

由于网络资源的动态性，某个原子服务可能因为不再存在、功能改动、服务超载等原因不能执行，此时执行引擎发出服务中断请求，系统再次进行服务匹配、服务组合等处理，并将处理结果返还给执行引擎继续执行。基于匹配策略的服务组合方法的执行模型如图 1 所示。



注：条件 1=每个节点是否对应一个原子服务

图 1 基于匹配策略的 Web 服务组合方法的执行模型

4 实例分析

本节用一个求年度个人医疗补贴的例子来简单说明基于匹配策略的服务组合过程。

假设 UDDI 中存在以下服务的注册信息，如表 1 所示，此

处仅简单列出服务匹配中需用到的信息。

表 1 服务注册信息表

标识	类型	角色	输入参数	输出参数
01	P	10.01	id:string,year:date	allowance: deciaml, process:document
02	A	10.03.02	id:string,year:date	length-of-service: decimal
03	A	10.03.03	id:string	:string
04	A	10.03.04	length-of-service: decimal,level:string	allowance_rate: deciaml
05	A	10.03.05	id:string,year:date	fee:decimal
06	A	10.03.01	fee: deciaml, allowance_rate:decimal	allowance:deciaml

步骤 1:系统根据用户的请求建立一个新任务,将用户的请求作为一个活动节点 V_1 建立初始业务流程,如图 2 所示。

步骤 2:对 V_1 进行信息解析,得到一个输入参数为(id:string,year:date),输出参数为(allowance:deciaml),角色为 10.01 的虚服务 S_1,对 S_1 进行服务匹配得到服务 01。

步骤 3:对由服务 01 得到的业务流程进行格式转化并替换节点 V_1,如图 3 所示。

步骤 4:对 V_2 进行信息解析得到输入参数为(id:string,year:date),输出参数为(fee:deciaml, allowance_rate:decimal),角色为 10.02 的虚服务 S_2,对 S_2 进行服务匹配,不能得到匹配的原子服务。

步骤 5:通过服务组合得到由服务 02、03、04、05 得到的组合服务 C_1 与虚服务 S_2 匹配。

步骤 6:用组合服务 C1 替换节点 V_2。

步骤 7:对 V_3 进行信息解析得到输入参数为(fee:deciaml,allowance_rate:decimal),输出参数为(allowance:deciaml),角色为 10.03.01 的虚服务 S_3,对 S_3 进行服务匹配,得到原子服务 06。

步骤 8:此时每一个活动节点都对应一个原子服务,进行服务抽象得到如图 4 所示的抽象业务流程 Bpn,和输入参数为(id:string,year:date),输出参数为(allowance:deciaml,process:document),角色为 10.01 的流程服务 Sp,其中参数 process 是抽象业务流程 Bpn 的描述文档。

步骤 9:按照业务流程 Bpn 的描述逐次调用、执行绑定的原子服务完成用户的需求。

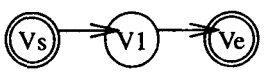


图 2 初始状态流程图

图 3 中间状态流程图

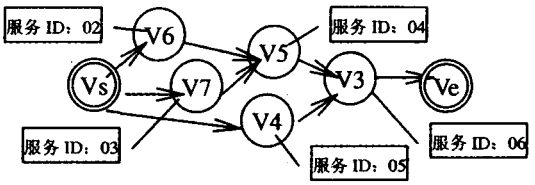


图 4 最终状态流程图

结束语 本文利用已有的业务流程技术 Web 和服务技术,提出了一种基于匹配策略的服务组合方法,该方法充分利用已有的 Web 服务资源,通过匹配关系将若干个原子服务组合为一个满足用户需求的组合服务,并且在服务执行时可以根据 Web 服务的状态进行动态调整,减少了用户的交互操作和预先的业务流程设计工作,降低了服务成本,提高了服务效率。整个过程中所有的服务匹配和服务组合都是基于匹配关系的,实施快捷、简单。

本文采用的匹配关系是基于 Web 服务的一些注册信息进行的,下一步的工作是利用语义技术提高匹配的性能。此外在服务组合时采用引导和推理机制,减少组合趋势的不确定性,提高组合速度和效率。

参 考 文 献

1 范玉顺. 工作流管理技术基础. 北京:清华大学出版社,2001

3 http://www.w3.org/2002/ws/

2 Agarwal M, Parashar M. Enabling Autonomic Compositions in Grid Environments. In: Proc. of the Fourth Intl. Workshop on Grid Computing (GRID'03)

4 Aversano L, Canfora G, Ciampi A. An algorithm for Web service discovery through their composition. In: Proc. of the IEEE Intl. Conf. on Web service (ICWS'04)

5 Patel C, Supekar K, Lee Y. Provisioning Resilient, Adaptive Web Services-based Workflow: A Semantic Modeling Approach. In: Proc. of the IEEE Intl. Conf. on Web Services (ICWS'04)

6 Cao J, Jarvis S A, Saini S, Nudd G R. GridFlow: Workflow Management for Grid Computing. In: Proc. of 3rd IEEE/ACM Intl. Symposium on Cluster Computing and the Grid (CCGrid 2003), Tokyo, Japan, May 2003. 198~205

7 Padmanabhuni S, Ganesh J, Moitra D. Web Services, Grid Computing, and Business Process Management: Exploiting Complementarities for Business Agility. In: Proc. of the IEEE Intl. Conf. on Web Services (ICWS'04)

8 http://www-128.ibm.com/developerworks/library/ws-bpel/