第 32 卷 第 14 期 计 算 机 工 程 2006 年 7 月
Vol.32 № 14 Computer Engineering July 2006
·软件技术与数据库· 文章编号：1000—3428(2006)14—0027—04 文献标识码：A 中图分类号：TP311

一种支持业务端编程的服务匹配算法

耿 晖，房 俊，韩燕波
（中国科学院计算技术研究所软件研究室，北京 100080）

摘 要：业务服务的动态绑定和替换是业务应用能够顺利执行的关键，而服务匹配算法是服务动态绑定和替换的核心依据。该文在分析现有服务匹配算法的不足的基础上，根据业务端编程架构的要求，提出了一种新的服务匹配算法，证明了该算法能更好地适应了动态业务需求。该方法在奥运公众信息服务平台原型系统 FLAME2008 中得到了应用，并总结了其实际效果。

关键词：服务匹配；业务服务；业务端编程

A New Service Matching Algorithm for Business End Programming

GENG Hui, FANG Jun, HAN Yanbo
(Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100080)

【Abstract】 Dynamic binding and substitution of services is the key to keep the applications run smoothly. On the basis of the requirements of service matching for business end programming and the analyses of several prevailing algorithms, this paper presents a new service matching approach to support dynamic and agile application execution. It is proved that the algorithms meets the frequently changed business requirements. This approach is used in a project which aims to mediate individual services for the general public in the Olympic Games 2008.

【Key words】 Service matching; Business service; Business end programming

服务作为一种大粒度的软件构件，逐渐成为构建松耦合网络应用的基石。服务动态、开放、自治的特点，对面向服务的应用的适应性提出了新的挑战。服务在自主演化的过程中，功能、参数或者属性的修改可能使得服务不能再满足应用的需求，使得应用的适应性降低。在服务不可用情况下，通过计算服务之间的匹配程度来对服务进行替换，是提高应用系统适应性的一种思路。相应地，服务匹配算法就成为实现服务替换依据的关键。

研究机构和科研单位从不同的角度研究了服务匹配的概念，并给出了衡量服务匹配程度的计算算法。本文以国家十五攻关计划项目“奥运信息平台 FLAME2008”为背景，针对业务端编程环境下对服务匹配的需求，对服务匹配概念和计算方法进行了探讨。

1 业务端编程环境中的服务匹配

1.1 业务端编程和 FLAME2008

作为一种新型的应用开发方法，业务端编程是在面向服务环境下，将最易变化的业务层面的服务资源的配置，留给业务人员(非计算机专业人员)去完成，而软件人员(计算机专业人员)则专注于打造相应的基础设施和使能环境。

基于业务端编程思想，建立了 FLAME2008 的实现平台。为实现服务提供者、服务使用者及服务组织者之间的语义共享，FLAME2008 平台构建在本体语义基础设施之上。如图 1 所示。

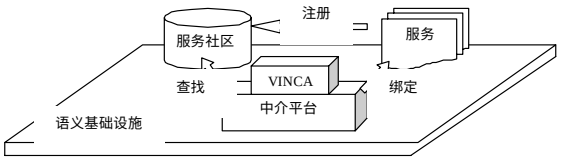

图 1 基于语义基础设施的 FLAME2008 平台架构

服务被注册到服务社区中，封装为业务服务；通过对业务领域的需求分类，业务服务被聚类成抽象业务服务，用户使用业务端建模语言VINCA[1]来组织服务描述应用逻辑，最后通过中介平台完成应用逻辑的解释执行。

1.2 服务描述

业务端编程对编程资源的组织和呈现提出了新的挑战。业务人员能够使用的编程资源应该是服务资源在业务层面的视图，是能够为业务人员所理解和使用的资源描述。在业务端编程模式下，业务用户能够“看到”2 种类型的服务：业务服务和抽象业务服务。

(1)业务服务的定义和表示

业务服务是业务用户从业务角度看到的服务资源的业务视图，是服务资源的业务层描述。业务服务表示为一个四元组。

bs = <FunSpe，BasicInfo,UsageSpe,Non-function>

1)FunctionSpe 是业务服务的功能描述。它包括 2 个部分：Service Category 和 Description。其中 Service Category 是服务按照业务功能分类的标签；Description 对应一个业务用户可以理解的功能描述字段。通过该功能描述业务，用户能获得服务的较为详细的功能信息。

2)BasicInfo 是记录业务服务基本信息，是业务用户用以识别不同业务服务的信息。它包括多项条目：业务服务名称，

基金项目：国家自然科学基金资助项目(60173018)；北京市科技计划基金资助项目“网格应用平台关键技术研究及应用”(H037530010130)；“十五”攻关计划奥运科技基金资助项目“‘服务中介平台及其在奥运应急联动中的应用”(2001BA904B07)

作者简介：耿 晖(1973—)，男，博士生，主研方向：Web 服务的流程集成，应用系统即时构造技术；房 俊，博士生；韩燕波，博导、研究员

收稿日期：2005-11-02 **E-mail：**genghui@software.ict.ac.cn

服务提供者名称，业务服务版本信息和服务颁布时间。

3)UseSpe 是业务服务的使用规范说明，包括 2 个部分内容：调用业务服务需要输入的信息(InputMessage)和服务调用返回的结果信息(OutputMessage)。其中每部分都包括多项条目，分别用 InputMessageItem 和 OutputMessageItem 来表示其中的每一项。

4)Non-functionCon 描述非功能属性信息。这些信息包括 Qos 服务质量，服务的地域范围，调用服务的前提条件(Precondition)和服务运行的效果(Effect)，服务访问的权限等。

(2)抽象业务服务的定义和表示。抽象业务服务是完成相同业务功能的一类业务服务的抽象描述。抽象业务服务用二元组来表示：

abs=<ServiceCategory,ConstrainSpe>

1)ServiceCategory 描述抽象业务服务的功能分类。

2)ConstrainSpe 是业务用户用以表达对服务约束要求的。根据对服务约束要求内容不同，ConstrainSpe 分为 3 个部分：服务输入信息约束，服务输出信息约束和非功能属性的约束。它们依次用 InCon，OutputCon 和 Non-functionCon 来表示。其具体含义如下：

InCon 是一个输入信息约束项的集合。集合中每个元素代表业务用户能够提供的一项输入。业务用户通过选择该集合中的元素，表达对候选服务的输入信息的约束要求。

OutputCon 是一个输出结果约束项的集合。集合中每个元素代表服务的一项输出。业务用户通过选择该集合中元素，表达对候选服务返回结果信息的要求。

Non-functionCon 表示非功能属性约束项的集合。业务用户通过设置对这些项目的取值，来表达对潜在业务服务的要求。与业务服务的 Non-function 不同，Non-functionCon 提供给业务用户的时候是没有值的，需要用户根据需求进行设定。

1.3 业务端编程对服务匹配算法的要求

在业务端编程中，由于服务是大量的，因此业务用户大都使用抽象业务服务来构建业务应用，以降低人工选择资源的复杂性。业务端编程对服务匹配算法的要求归纳为以下 2 个方面：

(1)使用场景要求：要求算法在业务应用运行的过程中，由业务应用中的一个抽象业务服务，选择满足用户要求的业务服务。

(2)算法能力要求：要求算法能够给出对服务匹配程度的定量计算公式，而不是定性分析。

2 相关研究

目前，关于服务匹配的研究还不成熟，不同的研究机构只是提出了解决问题的一些思路。Mecella从服务替换的角度对服务匹配概念进行了探讨，提出了服务兼容性(Service Compatibility)的概念[3]。Kawamura通过关系网络(Associative network)中概念之间的关系(包括泛化特殊化以及肯定)及关系权重，来比较参数之间的匹配程度[4]。在这些研究成果中，项目ONTOS[5]和VISPO[6]不但给出了服务匹配的计算公式，还定性地对公式进行了分析。本文将在分析其不足的基础上给出我们的匹配算法。

ONTOS 的目标是实现服务的自动的或半自动的服务组合。在 ONTOS 架构中，将服务表示为描述信息、功能和功能属性 3 个部分。

为了从候选服务中选择能够替换某服务的服务，ONTOS 依据“ pipe-and-filter ”的原则进行选择，即侧重在候选服务与被替换服务在输入、输出上的匹配程度。

ONTOS 的计算服务匹配程度的算法，综合考虑了服务功能的语义、个数和功能属性的语义、个数在服务匹配中作用。存在的缺陷如下：

(1)本算法没有考虑功能和功能属性在服务匹配中的不同权重，笼统的将功能和功能属性的匹配程度之和作为最终的服务匹配程度。

(2)针对功能属性的匹配，只考虑了功能属性的个数和语义，没有考虑具体的功能属性的取值对服务匹配的影响。

VISPO 中，服务的描述表示为三元组：<OP,IN,OUT>。其中，OP 是服务能够提供的操作的集合；IN 是输入信息项集合；OUT 是输出信息项集合。

为实现自动或者半自动的服务替换的目标，VISPO 引入了相容类(compatibility class)的概念。相容类是一组能够相互替换的服务。为了构造相容类，又引入了服务兼容性(service compatibility)的概念以计算 2 个服务 P_i 、 P_j 的兼容程度。

VISPO 引入下式进行计算：

$$GSim(P_i,P_j)=w_{ESim}\bullet NormESim(P_i,P_j)+w_{FSim}\bullet NormFSim(P_i,P_j)$$

(1) $Esim(P_i,P_j)$ 分别用来计算 2 个服务所有的输入、输出参数之间的兼容程度；

(2) $A(t,t')$ 计算参数 t ， t' 语义相似度，通过计算两个参数的语义在本体中的路径的权重值来表示；

(3) $FSim(P_i,P_j)$ 用来计算两个服务所有的操作之间的兼容性；

(4) w_{ESim} 和 w_{FSim} 分别是 $ESim(P_i,P_j)$ 和 $FSim(P_i,P_j)$ 的权重值。

可以看出，公式 $GSim(P_i,P_j)$ 计算 2 个服务兼容性的因素考虑了服务的所有输入、输出参数的个数和语义的兼容性，服务的所有操作的个数和语义的兼容性。同时，考虑了服务的所有参数(包括输入、输出)的兼容性和所有操作的兼容性在计算中的不同比重。VISPO 算法比 ONTOs 算法考虑的因素要全面，算法设计也更加合理。但存在如下的缺陷：

(1)VISPO 引入服务兼容性概念的目的是实现自动或者半自动的服务替换，但是该概念及其计算公式并不能达到上述目的。

假设有 3 个服务 P_x 、 P_y 和 P_z ，其中除了 P_y 的操作 op 的输出参数比 P_x 的操作 op 多一个参数外，服务 P_x 、 P_y 的描述相同；而服务 P_x 、 P_z 的描述完全相同。假设服务 P_x 不可用，需要分别计算 P_x 与 P_y 、 P_x 与 P_z 的兼容性，从而决定到底使用 P_y 还是 P_z 来替换服务 P_x 。根据公式 $GSim(P_i,P_j)$ 计算[6]， $GSim(P_x,P_z)$ 的值将比 $GSim(P_x,P_y)$ 的值大，反映了 P_z 比 P_y 更兼容。但从服务替换的角度来看，用 P_y 来替换 P_x 或者用 P_z 来替换 P_x 的效果是一致的，而公式 $GSim(P_i,P_j)$ 并没有从服务替换影响的角度来反映这种情况。另一方面，假设有 2 个服务 P_x 和 P_y ，其中除了 P_x 的操作 op 的输出参数比 P_y 的操作 op 多一个参数外，服务 P_x 、 P_y 的descriptor完全相同。假设服务 P_x 不可用，需要计算 P_x 和 P_y 的兼容性 $GSim(P_x,P_y)$ ；假设另一种情况， P_y 不可用，需要计算 P_y 和 P_x 的兼容性 $GSim(P_y,P_x)$ 。根据公式 $GSim(P_i,P_j)$ 定义， $GSim(P_x,P_y)=GSim(P_y,P_x)$ ，反映出用 P_y 替换 P_x ，或者用 P_x 替换 P_y 的效果相同。但从服务替换的角度来看，用 P_y 替换 P_x 比用 P_x 替换 P_y 的效果要好，而公式 $GSim(P_i,P_j)$ 也没有从服务替换的角度来反映这种情况。

(2)只计算替换的操作与其他服务的操作的兼容性。同时将服务的所有操作的兼容性作为衡量服务兼容性的因素，这没有意义。

(3)操作的参数间的匹配程度应该为参数之间的“ 一对一对应 ”的程度。因此操作的输入参数匹配程度，为计算一组分别来自 2 个操作的输入参数对的匹配程度之和，而一旦某输入参数对的匹配程

度不为零，这对参数则不能再与其他参数进行匹配计算。计算操作的输出参数间的匹配程度与此同理。而公式 $Esim(P_i, P_j)$ [6]计算 2 个服务操作的输入参数(输出参数)时，却依次计算输入参数对(输出参数对)之间的匹配程度，计算结果是参数间“一对多对应”的程度。

(4)没有考虑服务的非功能属性对服务匹配的影响。

3 业务端编程环境下服务匹配算法

业务端编程环境下，服务匹配算法通过 Select 函数来实现。Select 函数考虑了用户设定在抽象业务服务上的对服务输入信息项约束要求、输出结果信息项约束要求和非功能属性约束 3 方面的要求，分别通过函数 matchInput、matchOutput 和 matchNon-Function 来分别计算业务服务与之在 3 方面的匹配程度。

函数 matchParameter 通过参数在业务词汇库中关系，计算参数 advPara 匹配参数 reqPara 的程度。reqPara 表示要求的参数，advPara 表示提供的参数。思路为：对两个参数如果在词汇库中是同一个类(=)，那么为完全匹配，匹配程度设为 1；如果 reqPara 是 advPara 的父类(parentOf)，为部分匹配，匹配程度设为 m；如果 advPara 是 reqPara 的父类(parentOf)，也为部分匹配，匹配程度设为 n；其他情况下，为不匹配，匹配程度设为 0。需要说明的是：在本体库中，由于子类继承父类的所有属性，同时还拥有额外属性，因此认为用子类参数去匹配父类参数的要求比用父类参数去匹配子类参数的要求的效果更好，因此规定 m > n(m，n 的具体取值可以根据实现本算法系统的需求进行设置)。具体程序如下：

```
float matchParameter(reqPara,advPara){
 //计算 2 个参数间的匹配程度
 //m，n ∈ (0,1)，m > n，以表示参数匹配的程度
 If reqPara = advPara return 1；
 If advPara parentOf reqPara return n；
 If reqPara parentOf advPara return m；
 Otherwise
  return 0；}
```

函数 matchOutput 计算业务服务与抽象业务服务的输出参数的匹配程度。算法设计的原则：(1)如果抽象业务服务所要求输出参数集合是业务服务的输出参数集合的子集，即业务服务输出参数能够满足抽象业务服务对输出结果要求，认为输出参数完全匹配，匹配程度为 1；(2)否则，为部分匹配。匹配程度应该体现业务服务输出参数匹配抽象业务服务输出参数的个数和程度。具体程序如下：

```
float matchOutput (as,bs){
//计算业务服务匹配抽象业务服务的输出参数的程度
float weight ＝ 0；//表示业务服务所有输出参数匹配之和
Set ParaSet = bs.OutputMessage；
For each Pi in as.OutputCon {
If(|ParaSet|!=0)
{For each Pj in ParaSet {
        t = matchParameter(Pi,Pj)；
          If (t!=0)
         {
         weight= weight + t；
             ParaSet = ParaSet \{ Pj } ；
             return；
        }}
  }}
 return weight/| as.OutputCon |；}
```

与函数 matchOutput 计算业务服务匹配抽象业务服务的输出参数的程度不同，函数 matchInput 计算业务服务与抽象业务服务的输入参数的之间“差距”。在该算法设计中，由于抽象业务服务的输入参数是业务用户所能够提供的输入信息，而业务服务的输入参数是服务调用时要求业务用户提供的信息，因此输入参数越少、与抽象业务服务的输入参数的差别越小，应该认为其更加满足要求，差距程度越小。算法设计的原则：(1)如果业务服务输入参数集合是抽象业务服务的输入参数集合的子集，即抽象业务服务能够提供业务服务对输入参数要求，差距程度为 0；(2)否则，差距程度大于 0。差距程度应该体现在，业务服务输入参数与抽象业务服务输入参数“差别”的个数和语义程度上。具体程序如下：

```
float matchInput (as, bs){
float weight ＝ 0；bool flag = false；Set advSet＝as.InputCon；
Set reqSet＝bs.InputMessage
For each Pj in reqSet {
If (|advSet| ==0)
weight = weight + 1；
Else{
     For each Pj in advSet {
        t = matchParameter(Pi,Pj)；
          If (t!=0)
        {flag = true；
          weight = weight + (1- t)；//(1- t)为不匹配程度
           advSet = advSet \{ Pj } ；
          return；//跳出内层 for 循环
        }}
      If (flag==false)
        weight = weight + 1；
 }}
  return weight；}
```

函数 matchNon-Function 根据用户所设定在抽象业务服务上的非功能属性的约束要求(不包括对输入信息项个数的约束要求，也不包括对输出结果信息项个数的约束要求)，计算一个业务服务的非功能属性匹配抽象业务服务对非功能属性约束要求的程度。需要指出的是，不同的非功能属性，对匹配的衡量方式不同，因此不同的非功能属性都对应不同的计算方法。这些算法通过 SOM 模型中映射集合 F 中的与非功能属性相关的服务选取策略映射函数实现。

因为不同的非功能属性比较的方法不同，相应的计算公式不同，需要针对不同的非功能属性及系统要求来设计算法。在此用 f_a 表示非功能属性 a 的匹配算法，具体程序如下：

```
 float matchNon-Function (as，bs){
 //as 为抽象业务服务，bs 为业务服务
int n；//表示有效的非功能属性项约束
 float t；//表示单个非功能属性项匹配的程度
 float weight；//表示所有非功能属性项匹配的程度
n = 0；weight＝0；
For each a in as.Non-FunctionCon
{
     if a valid //用户在该信息项上设定了约束，则为有效
        {n = n + 1；
          t = fa(as,bs)； //fa 为非功能属性 a 的匹配算法
```

```
            weight＝weight + t；
        } }
    weight=weight / n；
    return weight；}
```

函数 Select 根据用户所设定的约束条件，包括对服务的输入、输出结果信息个数的约束要求和非功能属性 3 方面的约束要求，返回满足所设阈值条件的一组业务服务。具体程序如下：

```
Set Select (as){
 //Tin、Tout为设定的服务输入、输出匹配程度的阈值，t_nfp 为
//设定的非功能属性方面匹配程度的阈值
 Set  B_as^select ＝Null；
Set Bas=fc( as  )={bs|c(bs)=as}；
 for each bs in Bas do{
  Wat= matchOutput (as,bs)
  Win= matchInput (as,bs)
  Wnfp= matchNon-Function (as,bs)
  //匹配程度大于阈值，作为选取的业务服务
    if ((Wout>Tout) ∧ ( Win<Tin) ∧ ( Wnfp >Tnfp )
          B_as^select .add(bs)；}
 return  B_as^select ；}
```

计算步骤上，首先比较服务的输出的匹配程度；然后比较服务的输入的匹配程度；最后再比较非功能属性的匹配程度。

4 算法案例分析和总结

本节中将通过 2 个具体的服务匹配的例子说明 select 算法的计算结果，并对其算法的合理性进行分析。由于篇幅所限，没有给出非功能属性匹配的计算。

4.1 输出参数匹配

假设抽象业务服务 *as* 的输出参数集合为{ Out_a , Out_b , Out_c }，候选的 4 个业务服务 bs_1、bs_2、bs_3、bs_4 的输出参数集合分别是{ Out_a , Out_b , Out_c ， Out_d }、{ Out_a , Out_b , Out_c ， Out_d ， Out_e }、{ Out_a , Out_b , Out_d }、{ Out_a , Out_b ， Out_d }，如图 2 所示。

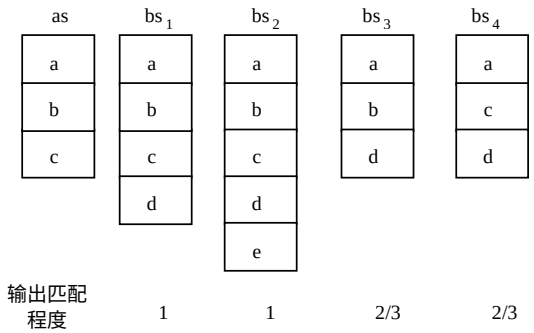

图 2 输出匹配程度举例

因为抽象业务服务输出参赛集合是业务服务bs_1、bs_2的输出参数集合的子集，都能够满足抽象业务服务对输出结果的要求，所以输出参数的匹配程度为 1。而业务服务bs_3、bs_4分别只有 2 个输出参数满足抽象业务对输出结果的要求，而抽象业务服务需要 3 个输出结果要求，因此匹配程度为 2/3。

按照函数 matchOutput 可以计算出上述取值。如图 3 所示。

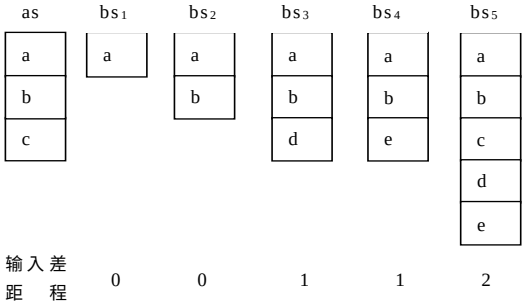

图 3 输入匹配程度举例

4.2 输入参数匹配

假设抽象业务服务 *as* 的输入参数集合为{ In_a , In_b , In_c }，候选的 4 个业务服务bs_1、bs_2、bs_3、bs_4、bs_5的输出参数集合分别是{ In_a }、{ In_a , In_b }、{ In_a , In_b , In_d }、{ In_a , In_b , In_e }、{ In_a , In_b , In_c , In_d , In_e }，如图 3 所示。因为业务服务bs_1、bs_2的输入参数集合是抽象业务服务输入参赛集合的子集，即抽象业务的输入参数能够满足业务服务bs_1、bs_2、对输入参数的要求，因此输入参数的差距程度值 0。而业务服务bs_3、bs_4、分别有 1 个输入参数不能由抽象业务的输入参数来满足，其差距程度为 1，业务服务bs_5有 2 个输入参数不能由抽象业务服务的输入参数来提供，因此差距程度为 2。按照函数 matchInput将计算出上述取值。

本文探讨了业务端编程中服务匹配算法的需求，在对当前主流服务匹配算法深入分析之后，提出一种新的服务匹配算法。该算法克服了这些算法的不足，并且被应用到奥运公众信息服务平台原型系统 FLAME2008 中，为实现抽象业务服务到业务的服务选取，提供了合理的支持。

参考文献

1 Han Y, Geng H, Li H, et al. VINCA——A Visual and Personalized Business-level Composition Language for Chaining Web-based Services[C]. Proc. of the First International Conference on Service Oriented Computing, Trento, Italy, 2003: 165-177.

2 Constantinescu I, Faltings B. Efficient Matchmaking and Directory Services[C]. Proc. of the 2003 IEEE/WIC International Conference on Web Intelligence, Halifax, Canada, 2003-10.

3 Mecella M, Pernici B, Craca P. Compatibility of E-services in a Cooperative Multi-platform Environment[C]. Proceedings of the 2nd VLDB Workshop on Technologies for E-services , 2001-09.

4 Paolucci M, Kawamura T, Terry R P, et al. Semantic Matching of Web Services Capabilities[C]. Proc. of the International Conference on Service Oriented Computing, Trento, Italy, 2002: 333-347.

5 Ruoyan Zhang, Jian Wang. Ontology-driven Web Service Compo-sition Platform[EB/OL]. http://Webster.cs.uga.edu/~ruoyan/ONTOS.htm, 2002.

6 Antonellis V D, Mechiori M. An Approach to Web Service Compatibility in Cooperative Processes[C]. Proc. of Symposium on Applications and Internet Workshops, Orlando, Florida, 2003-01: 27-31.