一种半自动化 Web 服务合成的算法*)

张大陆　王炫召
(同济大学计算机科学与技术系　上海 200092)

摘　要　Web服务合成，即将现有的若干服务合成为一个新的服务以获得新的功能，对于 B2B、B2C 应用和企业应用集成等各个领域都是十分重要的。服务合成必须实现半自动化或自动化。半自动化是比较可行的，它在用户的参与下合成服务。本文提出了一个半自动化服务合成的原型。其要求对服务进行语义描述，合成算法根据服务的语义信息来完成服务的合成。本文提出了基于目标服务的合成算法。其关键问题是如何选择下一个服务，以产生合适的输出作为当前服务的输入参数；需要对待选服务进行语义匹配，提出了精确匹配、一般匹配和语义距离等概念来衡量匹配度，并且用户可以根据其它属性来过滤服务，选择最优的一个。最后开发了一个合成系统，进行实验测试。

关键词　Web服务，服务合成，合成服务，过程模型

An Algorithm for Semi-automatic Web Service Composition

ZHANG Da-Lu　WANG Xuan-Zhao
(Department of Computer Science and Engineering, Tongji University, Shanghai 200092)

Abstract　Composing existing Web services to obtain new functionality will be important for various areas such as business-to-business, business-to-consumer applications and enterprise application integration. Service composition should be completed semi-automatically or automatically. And semi-automatic composition, the process of which is controlled by users, is more practical. This paper proposes a prototype for semi-automatic composition. The services should be described semantically and composition algorithms can use the semantic information about services to create the new service. The paper proposes the object service based algorithm. How to choose the next service, which can produce a proper output to serve as the current input of the current service, is a key problem. The semantic match of each available service is needed. Generic match, exact match and semantic distance are defined to describe the match degree. And the users can filter the services on the other attributes to choose the best one. At last, a composition system is developed to test the algorithm.

Keywords　Web service, Service composition, Composite service, Process model

1　简介

Web 服务(Web service，简称服务)为自包含的、自描述的、模块化的和可以通过 Internet 被发布、定位和调用的应用程序。

服务可以分为两大类：简单服务(simple service)和合成服务(composite service)。简单服务是基于 Internet 的应用程序，它可以独立地提供用户所需要的功能；而合成服务需要依赖于一组其它服务，通过这一组服务相互协作来实现服务的功能。

原子服务(atomic service)：合成服务由一组相互协作的子服务构成，每一个子服务都为合成服务的一个原子服务。

过程模型(process model)：定义了一组原子服务实现合成服务功能的相互协作方式。过程模型由一组原子服务、它们相互协作的控制流和信息流组成。控制流定义了原子服务被调用执行的先后顺序，而信息流则定义了在原子服务间信息交换传递的方式。

服务合成(service composition)：根据一个合成的目标，选择一组已经存在的服务，定义一个合适可行的过程模型，使得这些服务可以相互协作，并最后实现所需要的功能。

一个合成服务的过程模型可以完全由用户来定义。而在一个自动化服务合成过程中，合成服务的过程模型完全由服务合成系统定义的。但是自动化服务合成技术还有许多的关键问题没有解决，在实际中并不可行。而半自动化服务合成是可行的。半自动化服务合成让人参与服务合成的过程，让人控制服务合成过程；它可以帮助用户来选择合适的服务，以渐增的方式来创建一个合成服务。

2　相关工作

当前解决服务合成的问题主要有两类方法：

1)基于工作流(workflow)的方法。如 Eflow[1]是一个描述、定制和管理合成服务的一个平台。它使用静态工作流生成方法。一个合成服务被建模成为一个图(graph)，这个图定义了过程模型中各个节点(node)的执行次序。

2)基于 AI Planning 的方法。如 PDDL 合成方法[2]。DAML-S 和 PDDL 具有很多相似性，可以直接地将一种语言映射到另一种语言。当进行服务合成时，首先将 DAML-S 描述映射为 PDDL 描述，然后就可以使用各种 PDDL 制定器

*)本课题的研究受到国家自然科学基金项目"电子商务服务的分布式模型的研究"(项目编号 90204010)的资助。张大陆　教授，博士生导师，研究方向为计算机网络和电子商务；王炫召　硕士研究生，研究方向为电子商务。

(planner)来完成服务合成。

Rule-based planning 合成方法。如文[3]提出了一种根据高层次的陈述性描述(declarative description)来生成合成服务的方法。它通过 CSSL(Composite Service Specification Language)来描述一个所需要的合成服务,并且它提出了可合成规则来决定两个 Web 服务是否是可以合成的,并根据可合成规则来产生合成方案。

文[4]提出了用 SHOP2 制定器完成服务合成的方法。SHOP2 是一种 HTN(Hierarchical Task Network)的制定器。Web 服务是通过 DAML-S 来描述的,所以首先得将 DAML-S 的描述转换为 SHOP2 的描述。

3 Web 服务的描述

对于 Web 服务的描述现在主要有两种语言:WSDL 和 DAML-S[5]。

DAML-S 可以对 Web 服务进行语义描述,在 Web 服务发现和合成等应用中可以提供语义信息。DAML-S 定义了一个类 Service 来对 Web 服务建模和描述。其有 3 个属性:presents、describedBy 和 supports。它们的值域(range)分别为类 ServiceProfile、ServiceModel 和 ServiceGrounding,所以 DAML-S 是从这 3 个方面对服务进行描述的。

ServiceProfile 是用来定义服务是干什么的。它提供了关于服务的功能和其它方面的信息,服务发现代理根据这些信息可以确定这个服务是否可以满足其要求,即它提供了代理用于发现服务的信息。

ServiceModel 则是用来定义服务是如何工作的。ServiceGrounding 则定义了如何访问这个服务,即 ServiceModel 和 ServiceGrounding 提供了代理使用这个服务的信息。

服务发现和合成需要使用关于服务的语义信息。这些语义信息主要包含在 ServiceProfile 中。ServiceProfile 的主要属性有:input、output、precondition、effect 和 serviceType 等。

服务通常可以看作为一个黑盒子(black box),其功能定义为从输入到输出的一个转换,即接受一组输入,产生一组输出。仅仅用输入和输出来描述一个服务的功能是不足够的,因为完全可能出现两个服务,它们具有完全相同的输入和输出;但是它们的功能是不同的。可以使用 serviceType(服务类型)这个属性来辅助地描述服务的功能和特征,所以通过服务的输入、输出和服务类型可以比较完备地描述其功能。

WSDL 则是一种基 XML 的 Web 服务的描述语言。它的缺点是:它只进行句法层次上的描述,不提供语义层次上的描述;它只提供操作层次上的信息,不能提供关于服务的语义信息,所以会给服务发现和合成等应用带来许多的困难。可以通过语义标注[6]的方法在 WSDL 文件中增加必要的语义信息。

半自动化服务合成要求服务的描述文件可以提供语义信息,即必须对于服务进行语义描述。

4 半自动化服务合成的原型系统

半自动化服务合成的原型系统由以下 5 个部分组成:

知识库(Knowledge Base)。主要有两个部分组成:一个部分是 Service Repository,功能是存放和管理所有已经被发现和被语义标注的服务。这些服务在合成中作为原子服务,其描述都是包含语义信息的,这些语义信息必然要引用领域本体(domain ontology)中的概念。所以,知识库需要包含所引用的领域本体,即 Domain Ontology。

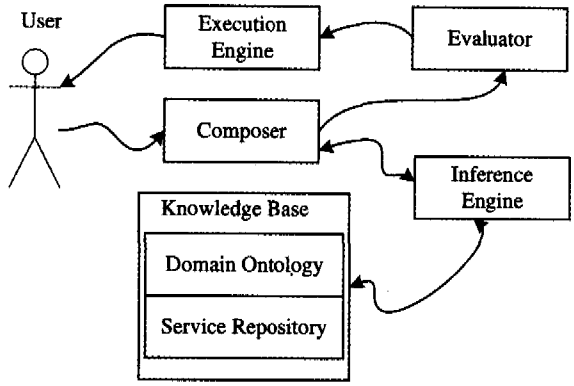

图 1 半自动化服务合成的原型系统图

合成器(Composer)。是一个用户接口,处理和用户的交互操作。使用推理机所提供的推理功能,在用户的参与下,完成整个服务合成的任务。

推理机(Inference Engine)。处理服务合成过程中的推理工作。最主要的工作是完成服务的语义匹配。在服务合成中的每一个步骤,都需要加入一个合适的服务。这需要向推理机发送一个查询(query),这个查询定义了所需要的服务必须满足的条件(如功能和 QoS 等方面的条件),由推理机处理这个查询,对服务进行语义匹配,将符合条件的服务返回给合成器。

执行引擎(Execution Engine)。执行合成服务并对执行过程进行监控,最后将执行结果返回给用户。

评估器(Evaluator)。当合成器产生多个合成方案时,对于各个合成方案进行评估,选出最优的一个,并提交给执行引擎执行。

5 基于目标服务的合成算法

该算法的思想如下:首先让用户选择一个目标服务 s,服务 s 所产生的输出可以满足用户的要求;服务 s 有 k 个输入参数,分别为 $I_1, I_2, \cdots, I_k$,对每一个输入参数 I_j,决定其是否由用户提供其值。如果不是,则选择另一个合适的服务来提供其值。如果对目标服务 s 引入了新的服务,则用相同的方式来处理每一个新服务的输入参数。重复这个过程,直至所有的服务的输入参数都得以满足,则服务合成的过程即得以完成。

引入服务 s 其实是引入服务 s 的一个实例,它接受特定的输入,产生特定的输出。如果同一个服务的多个实例接收相同输入,则必然会产生相同输出,那么它们是相同实例。相同实例只需存在一个,其它是多余的实例。因为没有必要以相同的输入执行同一个服务多次;只需要执行一次,将输出缓存起来,即可使用多次。

算法需要消除多余的服务实例,消除的方法为:在某一个步骤中,引入了服务 s,如果在前面的步骤中也引入了服务 s,这两个实例是接收相同输入产生相同输出的,即它们是相同实例,那么就应该对新引入的服务 s 的实例进行标记,标记它和前面引入的实例为同一实例,并且对于新引入的服务实例的输入参数不必进行处理。

在服务合成的过程中,对于各个服务的输入参数进行处理有两种方式:一种是深度优先匹配(Depth-First-Matching),另一种方式是广度优先匹配(Breadth-First-Matc-

hing)。

深度优先匹配(DFM)的算法如下：

```
1  s=ChooseDestinationService();
2  AddToCompositeProcess(s);
3  dfm(s)
```

而 dfm 的算法如下：

```
1  void dfm(s)
2  { if( SameInstExists(s))
3    { s1=SameInst(s);
4      MarkInstSameAs(s,s1);
5      return;
6    }
7  for(i∈s. IS )
8   if(! IsUserInput(i))
9    { s1=ChooseBestMatchedService(i);
10     AddToCompositeProcess(s1);
11     dfm(s1);
12   }
13 }
```

广度优先匹配(BFM)的算法如下：

```
1  s=ChooseDestinationService();
2  AddToCompositeProcess(s);
3  Queue q; InitQueue(q);
4  for(i∈s. IS)EnQueue(q, i);
5  while( ! QueueEmpty(q))
6  { DeQueue(q, i);
7    if( ! IsUserInput(i))
8    { s1=ChooseBestMatchedService(i);
9      AddToCompositeProcess(s1);
10     if( SameInstExists(s1))
11     { s2=SameInst(s1);
12       MarkInstSameAs(s1,s2);
13     }else
14     { for(j∈s1. IS)EnQueue(q, j);
15     }
16   }
17 }
```

$s.IS$、$s.OS$ 和 $s.T$ 分别表示服务 s 的输入参数集合、输出参数集合和服务类型。

ChooseDestinationService()的功能是选择满足用户要求的目标服务。

AddToCompositeProcess(s)的功能是将 s 加入到合成服务的过程模型中。

SameInstExists(s)用于判断新引入的服务 s 的实例是否存在相同实例；SameInst(s)返回 s 的相同实例；MarkInstSameAs($s1$,$s2$)将实例 $s1$ 标记为 $s2$ 的相同实例，这样，对于 $s1$ 的输入参数就不必进行处理了。

ChooseBestMatchedService(i)的功能是选择一个最匹配的服务 s，s 必须满足两个条件：1)服务 s 必须是可用的(匹配的)，即是服务 s 必须在功能上满足要求，如可以产生满足输入参数 i 的输出；2)服务 s 是最优的。

这个算法有两个要点：1)如何选择目标服务，使目标服务可以满足用户要求；2)对于某一个服务的某一个输入参数进行处理时，如何选择下一个服务，使其可以产生合适的输出来提供给该输入参数。

6 匹配服务的选择

无论是选择目标服务还是选择下一个服务，都得对已有的服务进行语义匹配。首先得进行功能上的语义匹配，使得所选择的服务在功能上满足要求；然后是其他方面的如性能上的语义匹配。

在选择目标服务时，对于功能方面的匹配考虑两个方面：输出集合和服务类型。假设要求目标服务产生的输出集合为 OS 和目标服务的服务类型为 T，那么所选择的目标服务 s 必须满足：

$Match(s.OS,OS) \wedge Match(s.T,T)$

在选择下一个服务时，对于功能方面的匹配也考虑两个方面：输出集合和服务类型。假设要求下一个服务满足输入参数 i 和下一个服务的服务类型为 T，那么所选择的下一个服务 s 必须满足：

$Match(s.OS,OS) \wedge Match(s.T,T)$

此时，$OS=\{i\}$。

而 $Match(s.T,T)$定义如下：

$Match(s.T,T) \Leftarrow subs(T,s.T)$

$Match(s.OS,OS)$定义如下：

$Match(s.OS,OS) \Leftarrow \forall i \in OS, \exists j \in s.OS, subs(i,j)$

这里，$subs(c1,c2)$表示 $c1$ 和 $c2$ 为同一概念或 $c1$ 为 $c2$ 的父概念。

为了进一步区分匹配的程度，引入了语义距离的概念。

有两个概念 $c1$ 和 $c2$，$c1$ 为 $c2$ 的父概念或同一概念，$c1$ 和 $c2$ 的语义距离定义为：在概念的层次体系中，从节点 $c1$ 到节点 $c2$ 的路径的长度；记作 $sd(c1,c2)$，$sd(c1,c2)=sd(c2,c1)$。

当 $c1$ 和 $c2$ 为同一概念时，$sd(c1,c2)=0$；而当 $c1$ 为 $c2$ 的直接父概念时，$sd(c1,c2)=1$；依次类推。

当 $c1$ 和 $c2$ 不存在继承关系时，$sd(c1,c2)=\infty$。

当 $Match(s.OS,OS)$时，可以定义 $sd(s.OS,OS)$来进一步区分它们的匹配程度。假设 $OS=\{b_1,b_2,\cdots,b_n\}$，$Match(s.OS,OS)$所建立的映射关系为 $a_1 \rightarrow b_1, a_2 \rightarrow b_2, \cdots, a_n \rightarrow b_n$，$a_1, a_2, \cdots, a_n \in s.OS$，那么 $sd(s.OS,OS)$定义为：

$$sd(s.OS,OS)=\sum_{i=1}^{n} sd(a_i,b_i)$$

可以定义两种类型的匹配：精确匹配(exact match)和一般匹配(generic match)。

精确匹配定义如下：

$ExactMatch(s.OS,OS) \Leftarrow sd(s.OS,OS)=0$

一般匹配定义如下：

$GenericMatch(s.OS,OS) \Leftarrow sd(s.OS,OS)>0$

选择目标服务或下一个服务进行功能方面语义匹配时，语义距离可以分别定义为：

$sd(s,OS)=sd(s.OS,OS)$

$sd(s,i)=sd(s.OS,\{i\})$

在选择目标服务或下一个服务时，首先进行语义匹配，找出所有满足条件的服务，然后再计算每一个满足条件的服务的语义距离，语义距离最小的那个服务是最匹配的。

在对于服务功能进行语义匹配后，可以根据服务的其它属性来进一步地过滤服务。合成器可以向推理机查询所选择的服务类型的所有属性，然后创建一个用户界面，列出这些属性，让用户输入各个属性的约束值，用户输入后，向推理机发送一个过滤请求，推理机将根据这些约束值对匹配的服务进行过滤，滤去所有不满足约束条件的服务。

7 实验测试

为了测试所提出的算法，实现了一个半自动化服务合成的系统。使用的语言为 java，使用了 prolog 库(JPL. jar)的 API。该系统所使用的推理机是建立在 prolog 基础之上的 DAML 的推理机。

建立了一个小型的服务库，其包含 21 种服务类型，103 个服务，服务合成过程中使用这个小型服务库中的服务作为原子服务。也可以将所创建的合成服务添加到服务库中，作

为以后合成的原子服务。

如果需要一种服务，它可以取得某一固定电话号码所在区域的地图，并且没有任何已存的服务可以满足这个要求，则可以通过服务合成的方法来动态创建。

第一步选择目标服务，$OS=\{Map\}$，$T=MapService$，对于服务进行功能的语义匹配，结果如表1。

表1 选择目标服务的语义匹配结果

服务	输出参数	服务类型	响应时间(ms)	语义距离
S11	Map	MapService	2526	0
S12	TrafficMap	MapService	2380	1
S13	FacilityMap	FacilityMapService	2465	1
S14	TrafficMap	TrafficMapService	5051	1
S15	Map	MapService	4610	0
S16	FacilityMap	MapService	2205	1

对于功能匹配，S11 和 S15 的语义距离都为 0，都是精确匹配。在性能方面主要考虑响应时间这个指标。因为 S11 的响应时间较短，所以选择 S11 作为目标服务。S11 有三个参数，Diameter 参数由用户提供，另外两个输入参数为 Latitude 和 Longitude，需要进行处理。

第二步选择下一个服务来提供 S11 的 Latitude 参数，$OS=\{Latitude\}$，$T=LocationService$，功能的语义匹配结果如表2。

表2 第二步语义匹配的结果

服务	输出参数	服务类型	响应时间(ms)	语义距离
S21	Latitude, Longitude	LocationService	2018	0
S22	Latitude, Longitude	LocationService	2550	0
S23	Latitude, Longitude	LocationService	1800	0

选择服务 S23，要处理其输入参数 GprsLocation。

依此处理，最后得到的合成服务为：

为 S11 提供 Latitude 参数和 Longitude 参数的 S23 的两个实例为相同实例，应消除一个。

合成服务的输入参数为：Diameter 和 FixedPhoneNumber；输出参数为 Map。用户只要输入一个固定电话号码和一个所需要地图的直径值，就可以取得其所在区域的地图。

服务合成是否可以成功，取决于服务库中是否有足够多的合适的服务。在服务合成过程中，如果需要某些特殊功能的服务不存在，那么服务合成就会失败。

总结 电子商务经历了发布式和动态交互式两个阶段。当前动态交互式电子商务已经暴露出了许多问题，不能适应电子商务进一步发展的需要。电子商务的进一步发展必然要引入新的技术，即 Web 服务技术。将 Web 服务技术应用于电子商务，服务合成是其中一个关键问题。本文提出了一个半自动化服务合成的原型系统和基于目标服务的合成算法，并实现了一个合成系统，对所提出的算法进行测试。

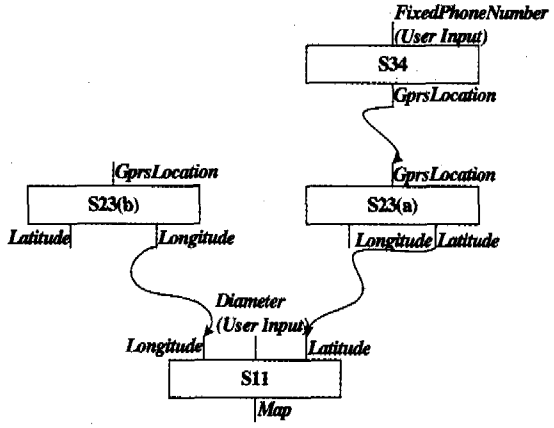

图2 一个简单的合成实例的过程模型

参考文献

1 Casati F, Ilnicki S, Jin L. Adaptive and dynamic service composition in EFlow. In: Proceedings of 12th International Conference on Advanced Information Systems Engineering (CAiSE), Stockholm, Sweden, June 2000

2 McDermott D. Estimated-regression planning for interactions with Web services. In: Proceedings of the 6th International Conference on AI Planning and Scheduling, Toulouse, France, 2002

3 Medjahed B, Bouguettaya A, Elmagarmid A K. Composing Web services on the Semantic Web. The VLDB Journal, November 2003, 12(4)

4 Wu D, Sirin E, Hendler J, et al. Automatic Web services composition using SHOP2. In: Workshop on Planning for Web Services, Trento, Italy, June 2003

5 Ankolenkar A, Burstein M, Hobbs J R, et al. DAML-S: Web Service Description for the Semantic Web. In: Proceedings of The First International Semantic Web Conference (ISWC), Sardinia (Italy), June 2002

6 Sivashanmugam K, Verma K, Sheth A, et al. Adding Semantics to Web Services Standards. In: Intl Conf. on Web Services, Las Vegas NV, June 2003

7 Paolucci M, Kawamura T, Payne T R, et al. Semantic matching of Web service capabilities. In: Proceedings of the 1st International Semantic Web Conference, Sardinia, Italy, June 2002. 318～332

(上接第 34 页)

9 Lui King-Shan, Nahrstedt K, Chen Shigang. Hierarchical QoS routing in delay-bandwidth sensitive networks. In: Proc of the 25th Annual IEEE Conference on Local Computer Networks (LCN 2000), Nov. 2000. 579～588

10 Lui King-Shan, Nahrstedt K, Chen Shigang. Routing with topology aggregation in delay-bandwidth sensitive networks. IEEE/ACM Transactions on Networking, 2004, 12(1): 17～29

11 闵应骅. 计算机网络路由研究综述. 计算机学报, 2003, 26(6): 641～649

12 Moh W M, Xiang L, Zhao X. Extending BGMP for QoS-based inter-domain multicasting over the Internet. In: Proc of IEEE International Conference on Communications (ICC 2000), Jun 2000, 2: 728～734

13 Yan Shuqian, Faloutsos M, Banerjea A. QoS-aware multicast routing for the Internet: The design and evaluation of QoSMIC. IEEE/ACM Transactions on Networking, 2002, 10(1): 54～66

14 Pradhan S, Yi Li, Maheswaran M. QoS-aware hierarchical multicast routing on next generation internetworks. In: Proc of the 20th IEEE International Conference on Performance, Computing and Communications, Arizona, USA, Apr 2001. 9～16

15 陆慧梅，向 勇，史美林. 支持 QoS 的层次组播路由算法框架 QHMR. 计算机学报, 2004, 27(6): 772～781

16 Awerbuch B, Du Y, Khan B, et al. Routing through networks with hierarchical topology aggregation. In: Proc of the Third IEEE Symposium on Computers and Communications (ISCC'98), 1998. 406～412

17 刘进. 大规模层次化网络中保证服务质量的路由算法: [学位论文]. 北京: 清华大学, 2000

18 Papadimitriou C H, Steiglitz K. Combinatorial optimization: Algorithms and complexity. New York: Dover Publications, Inc, 1998

19 Waxman B M. Routing of multiple connections. IEEE Journal on Selected Areas in Communications, 1998, 6(9): 1617～1622

20 Handley M, Perkins C, Whelan E. Session announcement protocol. RFC 2974, Oct 2000

21 Handley M, Jacobson V. SDP: Session description protocol. RFC 2327, Apr 1998

22 李腊元，李春林. 多 QoS 约束的多播路由协议. 软件学报, 2004, 5(2): 286～291