基于全局依赖网的 Web 服务组合自动演化方法研究

张元鸣[1,2],倪 宽[1],陆佳炜[1],徐 俊[1],肖 刚[1,2]
(1. 浙江工业大学计算机科学与技术学院,浙江杭州 310023;2. 浙江工业大学机械工程博士后流动站,浙江杭州 310023)

摘 要: 为适应互联网环境下动态的运行环境以及多变的用户需求,快速构建面向服务架构的软件系统,本文提出一种基于全局依赖网的 Web 服务组合自动演化方法. 该方法能够根据用户演化需求对服务组合执行演化操作,在服务全局依赖网的基础上从各演化点出发执行正向演化推理与反向演化推理,确保执行演化操作之后的正确性和有效性,自动生成服务组合演化结果. 应用实例表明本方法能够有效实现服务组合的自动化演化,有助于降低基于服务组合的软件开发成本,缩短软件开发周期.

关键词: 服务组合; 自动演化; 演化操作; 演化推理; 全局依赖网
中图分类号: TP311 **文献标识码:** A **文章编号:** 0372-2112 (2017)02-0267-11
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2017.02.002

Automatic Web Service Composition Evolution Based on Global Dependence Network

ZHANG Yuan-ming[1,2], NI Kuan[1], LU Jia-wei[1], XU Jun[1], XIAO Gang[1,2]
(1. *College of Computer Science & Technology, Zhejiang University of Technology, Hangzhou, Zhejiang* 310023, *China*;
2. *Post-doctoral Research Center of Mechanical Engineering, Zhejiang University of Technology, Hangzhou, Zhejiang* 310023, *China*)

Abstract: In order to rapidly construct software based on service-oriented architecture under dynamic running environments of Internet and to satisfy various requirements of users, an automatic service-composition evolution approach based on global dependence network (GDN) is proposed. It can automatically execute defined evolution operations on existed composited service according to evolution requirements. After that, it continues to execute reverse evolution reasoning and positive evolution reasoning based on the GDN from each evolution point. Then, it will automatically generate evolution results of composited service. An actual application is given to illustrate the entire evolution procedure. This application shows the correctness and effectiveness of the approach, which helps to reduce the cost and the cycle of software development.

Key words: service composition; automatic evolution; evolution operations; evolution inference; global dependence network

1 引言

随着 Internet 的迅速发展和广泛应用,以 Web 服务为基础的面向服务架构作为新型的分布式计算模式已成为互联网环境下主流的软件形态[1,2]. 由于单个 Web 服务功能有限,难以满足复杂的业务需求,服务组合成为构建软件系统的关键[3~5]. 然而,运行环境的动态性以及用户需求的多样化,使得基于服务组合的软件系统不得不经常重新组合以适应新的变化,导致较高的组合成本和较长的开发周期.

软件演化是软件进行渐变并达到所希望的形态的过程[6],随着用户需求的变化以及软件运行环境的变化,软件系统需要不断的演化,以适应这种新的需求和变化[7]. 例如,余萍等[8]提出了一种面向动态软件体系结构的在线演化方法,设计并实现了一种运行时刻的软件体系结构元模型,保证演化前后系统的一致性、完整性和演化的可追溯性.

服务组合演化则是在已经组合好的软件系统基础

收稿日期:2015-12-30;修回日期:2016-05-05;责任编辑:孙瑶
基金项目:国家自然科学基金(No. 61379017);浙江省钱江人才计划 D 类(No. QJD1302009);浙江省重大科技专项(No. 2014C01408)

上根据外部运行环境和用户需求进行的演化,这有助于降低重新组合带来的成本与周期. 与传统软件演化相比,由于服务具有动态、异构以及自治的特点,且系统所集成的服务往往来源于不同的组织,这使得服务组合演化面临更多的挑战[9].

国内外学者已在服务演化影响、服务演化操作、服务演化框架等方面进行了初步研究,取得了一定成果.

在服务的功能与非功能属性对演化影响方面:WANG 等[10]使用有向图根据接口与控制流程建立服务依赖模型,依据其得到相应服务依赖度量集,为服务组合演化提供分析基础;彭焕峰等[11]利用 BPEL 控制流图对服务组合的 BPEL 进行建模,通过模型对服务组合的数据流进行分析,提出一个面向成员服务的数据依赖度量集,从而分析成员服务对 BPEL 中数据流的影响程度与范围,为服务组合演化策略的制定及实施提供依据;龙军等[12]提出了逐步逼近评价实体的演化方法以改变信任推理中的信任缺失与信任泛化的不足.

在演化操作方面:黄光奇等[13]在基于服务集合不变的情况下,利用工作流网对服务组合进行形式化建模分析,提出了一种组合服务的合理演化行为集 SEBS,保持演化后组合服务的合理性;SONG 等[14]提出一套演化操作准则来指导开发人员正确地修改服务组合,只要遵循这些准则,服务组合的数据流在演化过程中的正确性就可以得到保持.

在演化框架方面:CHEN LV 等[15]提出了一种基于事件驱动的动态服务组合演化方法,通过有向无环图建立服务依赖图,通过事件驱动方法主动监控并处理动态服务,根据监控结果更新受到动态服务影响需要发生结构变化的服务依赖图;刘涛等[16]通过增加归属操作符和条件控制符对 PI 演算进行扩展,使之与事件条件动作模式更好地结合起来,从而提出一种描述组合服务流程方法;王晓璇等[17]以机器辅助处理为目标,进一步提出了一种语义 Web 环境下组合服务演化方法——EM4CS,将组合服务演化分为六个步骤,但该框架从演化需求到演化结束并不是自动实施的;另外,还有一些学者通过面向服务编排[18-19]建立相应的服务组合演化技术框架或模型.

深入分析可以发现,已有服务组合演化研究存在以下不足:(1)服务组合演化模型不够完善,不能够表达较为复杂的服务依赖关系,如文献[15]提出的有向无环图演化模型的表达能力不足,文献[16]提出的 PI 演算演化模型的数学理论不成熟;(2)缺乏完整的演化需求形式化描述方法,演化过程不能完全满足实际的演化需求,如文献[13]给出的演化行为集没有考虑反向演化操作,文献[14]给出的演化操作准则只涉及了服务层面的演化操作,没有涉及服务接口的演化操作;(3)演化自动化程度不高,演化过程中需要较多的人工校验,演化效率较低,如文献[17]提出的 EM4CS 方法的每个演化步骤都需要人工校验.

针对这些问题,本文提出一种基于全局依赖网的服务组合自动演化方法,对服务组合的演化模型、演化需求、演化过程、演化推理等进行了系统的研究,其主要贡献包括以下 4 个方面:

(1)提出了一种基于 Petri 网的服务组合演化模型,为服务组合自动演化提供形式化描述工具.

(2)提出了服务全局依赖网的概念,为演化操作及演化推理提供理论依据.

(3)提出了完整的演化需求形式化表达方法,将用户不确定性需求转换为计算机能够理解的演化操作.

(4)提出了正向演化推理与反向演化推理,确保服务组合在执行演化操作之后是正确的和有效的.

2 服务组合演化基础

2.1 基本概念

先给出服务组合演化的几个基础性概念:

定义 1 原子服务(Atomic Service,AS)是指可被独立调用且功能不可再分的 Web 服务.

定义 2 复合服务(Compound Service,CS)是指由若干原子服务组合且可被独立调用的 Web 服务.

一般地,服务通过自身对外发布的接口进行调用,接口是服务间传递数据信息与控制信息的主要机制,包括输入接口和输出接口. 为此,基于接口匹配的方式能够较好地描述服务之间的组合关系,从而生成一个功能更加复杂的复合服务.

定义 3 对于两个原子服务 $AS_i(I_i,O_i)$ 与 $AS_j(I_j,O_j)$,若 $O_i \supseteq I_j$,则 $AS_i \rightarrow AS_j$,则 AS_i 与 AS_j 之间是顺序接口依赖,AS_i 为 AS_j 的前驱服务,AS_j 为 AS_i 的后继服务[20].

根据定义 3,可以推出以下服务接口依赖关系:

推论 1 对于原子服务 $AS_1,AS_2,\cdots,AS_m$ 以及 AS_j,若 $O_1 \cup O_2 \cup \cdots \cup O_m \supseteq I_j$,则 $AS_1 \wedge AS_2 \wedge \cdots \wedge AS_m \rightarrow AS_j$,称为同步接口依赖.

推论 2 对于原子服务 $AS_1,AS_2,\cdots,AS_m$ 以及 AS_j,若 $(O_1 \supseteq I_j) \vee (O_2 \supseteq I_j) \vee \cdots \vee (O_m \supseteq I_j)$,则 $AS_1 \vee AS_2 \vee \cdots \vee AS_m \rightarrow AS_j$,称为合并接口依赖.

推论 3 对于原子服务 AS_i 以及 $AS_1,AS_2,\cdots,AS_m$,若 $(O_i \supseteq I_1) \wedge (O_i \supseteq I_2) \wedge \cdots \wedge (O_i \supseteq I_m)$,则 $AS_i \rightarrow AS_1 \wedge AS_2 \wedge \cdots \wedge AS_m$,称为并发接口依赖.

推论 4 对于原子服务 AS_i 以及 $AS_1,AS_2,\cdots,AS_m$,若 $(O_i \supseteq I_1) \vee (O_i \supseteq I_2) \vee \cdots \vee (O_i \supseteq I_m)$,则 $AS_i \rightarrow AS_1 \vee AS_2 \vee \cdots \vee AS_m$,称为选择接口依赖.

2.2 服务组合演化模型

Petri 网是一种高效的建模与分析工具,适合描述

异步、并发的计算机系统模型,被用于服务组合建模. 将服务、服务接口映射为 Petri 网的各种要素,构建基于 Petri 网的服务组合演化模型:

定义 4　服务组合演化模型(Service Composition Evolution,SCE)可以表示为一个八元组:

$SCE = <P,T,I,O,\mu,\Gamma,C,M_c>$,其中

(1)$P = \{p_1,p_2,\cdots,p_m\}$:表示一个有限库所集,每个库所代表一个原子服务;

(2)$T = \{t_1,t_2,\cdots,t_n\}$:表示一个有限变迁集,表示服务依赖关系的实现;

(3)I:表示输入连接弧权系数 $0 < \sigma \leqslant 1$;

(4)O:表示输出连接弧权系数 $0 < \tau \leqslant 1$;

(5)μ:表示变迁的阈值;

(6)Γ:表示服务依赖关系的可信度;

(7)C:表示 Token 的集合;

(8)M_c:表示一个 Token 的值,即服务的真实度,取值范围[0,1].

本文聚焦于服务组合的结构演化特性,将定义中的连接弧权系数 I 与 O、变迁阈值 μ、可信度 Γ、M_c 设为 1,从而简化服务的选择.

根据上述定义以及服务接口依赖的定义,可以得到包括顺序控制结构、同步控制结构、合并控制结构、并发控制结构、选择控制结构在内的五种控制结构 Petri 网基本模型,如图 1 所示.

①顺序控制结构:对应定义 1,指执行 AS_i 之后才能执行 AS_j.

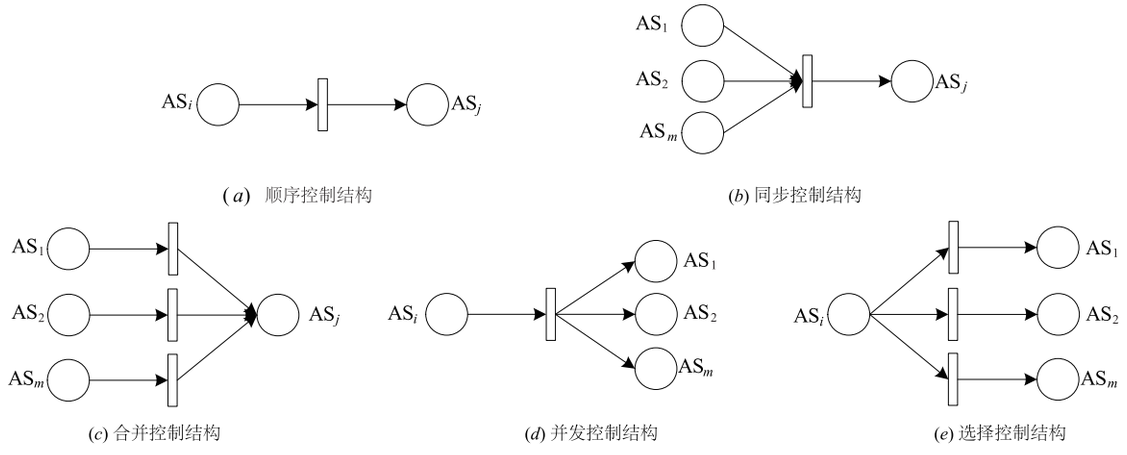
②同步控制结构:对应推论 1,指执行全部的 AS_1,AS_2,…,AS_m 之后才能执行 AS_j.

③合并控制结构:对应推论 2,指 AS_1,AS_2,…,AS_m 中至少执行一个才能执行 AS_j.

④并发控制结构:对应推论 3,指执行 AS_i 后才能执行 AS_1,AS_2,…,AS_m 的所有服务.

⑤选择控制结构:对应推论 4,指执行了 AS_i 只能执行 AS_1,AS_2,…,AS_m 中其中一个服务.

对于任意一个服务组合,基于以上五种控制结构建立与其对应的服务组合控制结构 Petri 网,简称控制结构网(Control Structure Network,CSN),将服务组合演化定义为一组 CSN 的渐变过程.



(a) 顺序控制结构　(b) 同步控制结构　(c) 合并控制结构　(d) 并发控制结构　(e) 选择控制结构

图 1　服务组合控制结构Petri网基本模型

2.3　服务全局依赖网

为了保证服务组合演化是可计算的,给出服务组合演化域的概念,并在此基础上构建全局依赖网.

定义 5　演化域(Evolution Domain)是指服务组合演化涉及的所有原子服务集合.

定义 6　全局依赖网(Global Dependence Network,GDN)是指在演化域范围内基于接口依赖关系构建的 Petri 网.

GDN 限定了服务组合的演化范围,是演化操作以及演化推理的基础和参照物. 根据 CSN 与 GDN 的定义,有:

$$CSN \subseteq GDN$$

因此,任意服务组合的 CSN 是 GDN 的一个子网,而服务组合演化是在 GDN 的基础上根据演化需求求解的一个过程.

GDN 可以通过遍历演化域中的所有原子服务及其接口依赖进行构建,可以采用接口依赖反向匹配策略[21],基本思想是通过对演化域中任意一个原子服务执行服务反向匹配,生成其前驱服务结构,并循环这个过程,直到演化域中的所有原子服务及其接口依赖被遍历完成,从而得到 GDN.

当演化域变化之后,可以按照以上策略重新生成 GDN,但这种方式在原子服务较多的情况下将导致较高的空间复杂度. 为此,可以根据演化域的变化情况在

已有 GDN 的基础上做更新操作. 一般地,更新操作包括新增原子服务、删除原子服务和替换原子服务三个操作:新增原子服务被定义为在 GDN 中加入新的库所及其连接弧;删除原子服务被定义为在 GDN 中删除库所及其连接弧;替换原子服务可以被定义为删除和新增的复合操作.

对 GDN 的更新操作的基本方法如下所述. 以新增的原子服务或删除的原子服务对应的库所为起点,从该库所向输入端方向进行更新. 对库所执行服务反向匹配策略,逐步得到其服务前驱结构;若其中所有枝干与 GDN 已有的网络相接,或者无法执行服务反相匹配策略,则从库所向输出端进行更新. 根据该原子服务输出端找出与之匹配的服务集合,然后对集合中的所有元素执行服务反向匹配策略. 若原子服务在某元素的服务前驱结构中,则该元素为库所的后继服务. 依此类推,逐步得到其服务后继结构,直到后继结构的所有枝干与 GDN 已有的网络相接,此时更新结束. 更新过程如算法 1 所示.

算法 1 全局依赖网更新算法

```
Input:GDN, evolution domain D, new atomic services W
Output:Updated GDN
1.  for(i=1; i≤W.size; i++){
2.      Func(W[i]);
3.  }
4.  Func(WS){
5.      W=W-{WS};
6.      D=D ∪{WS};
7.      if(∃ WS_pre∈Pre, WS_pre∉D){
8.          Func(WS_pre); //∃ WS_pre∈Pre
9.      }
10.     List S;
11.     for(j=1; j≤(W ∪ D).size; j++){
12.         if(O⊇I_j){
13.             S.add((W ∪ D)[j]);
14.         }
15.     }
16.     if(S.size!=0){
17.         for(k=1; k≤S.size; k++){
18.             Func(S[k]);
19.         }
20.     }
21. }
```

定义 7 GDN 对应的关系矩阵可表示为 $\boldsymbol{A}=\{a_{ij}\}$, $1\leqslant i\leqslant m, 1\leqslant j\leqslant n$,其中

$$a_{ij}\begin{cases}1, & \text{库所 } P_i \text{为变迁 } T_j \text{的输入}\\ -1, & \text{库所 } P_i \text{为变迁 } T_j \text{的输出}\\ 0, & \text{其他}\end{cases} \quad (1)$$

CSN 对应的关系矩阵也可按照上述定义得到. 此外,在根据服务组合构建 CSN 及其对应的关系矩阵时,如果服务组合中包括复合服务,需要将复合服务分解为原子服务.

3 演化需求表达

3.1 演化需求

演化需求是服务组合演化的基本依据,将驱动服务组合演化过程. 为提供能够使计算机理解的演化需求,需要对其进行形式化描述.

定义 8 演化点(Evolution Point,EP)是指服务组合中根据需要进行变更的原子服务.

定义 9 演化操作(Evolution Operation,EO)可以表示为一个四元组 EO = < EP,OP,I,O >,其中 EP 是指演化点;OP = {Add,Del} 表示演化基本操作集合,其中 Add 操作指的是使 EP 接口数量增加的操作,Del 操作指的是使 EP 接口数量减少的操作;I 是指 EP 需要变化的输入接口集合;O 是指 EP 需要变化的输出接口集合.

定义 10 演化需求(Evolution Requirements,ER)可以表示为一个二元组 ER = < PEO,NEO >,其中正向演化操作集合(Positive Evolution Operations,PEO) = $\{EO_i|1\leqslant i\leqslant n\}$ 表示对 EP 需要执行的演化操作集合;反向演化操作集合(Negative Evolution Operations,NEO) = $\{EO_j|1\leqslant j\leqslant m\}$ 表示对 EP 需要避免的演化操作集合.

由于 PEO 中的演化操作是需要执行的,而 NEO 与之相反,说明 NEO 是 PEO 的约束,若 NEO 中存在任意一个属于 PEO 的演化操作,即 PEO 不满足 NEO 的约束,则该演化需求 ER 不能正确执行.

3.2 演化需求冲突检测与消解

一般地,演化需求包括一组演化点及其演化操作,这些演化操作之间可能会存在冲突. 为此,在执行演化之前,需要对这些演化操作进行冲突检测,消解可能的冲突.

从演化操作对演化点接口的影响效果来看,演化操作可能导致互逆、相似、覆盖和触发[22]四种冲突. 先设置四种冲突的优先级级别如下:

互逆冲突 > 覆盖冲突 > 触发冲突 > 相似冲突.

给出两个演化操作冲突的检测与消解方法,实现过程如算法 2 所示.

算法 2 演化操作冲突检测与消解算法

```
Input:Two evolution operations EO_x = <WS_i, OP_x, I_x, O_x> and EO_y = <WS_j, OP_y, I_y, O_y>
Output:A set of evolution operation after digesting
1.  CS=φ;// digestion set
2.  POS= CS;
3.  hasConflict=0;
```

4. if(WS_i equals WS_j && OP_x is additive operation && OP_y is subtractive operation && ($I_x \cap I_y \neq \varphi$ || $I_x \cap I_y \neq \varphi$)) {
5. the conflict of OP_x and OP_y is inverse conflict;
6. hasConflict = 1;
7. $I_x = I_x - I_x \cap I_y, I_y = I_y - I_x \cap I_y, O_x = O_x - O_x \cap O_y, O_y = O_y - O_x \cap O_y, CS = CS \cup \{EO_x, EO_y\}$;
8. }
9. if(WS_i equals WS_j && OP_x is subtractive operation && I_x equals input of WS_i && equals output of WS_i) {
10. the conflict of OP_x and OP_y is advanced conflict;
11. hasConflict = 1;
12. CS = CS ∪ { EO_x };
13. }
14. if(WS_i equals WS_j && OP_x and OP_y is the same operation && ($I_x \subseteq I_y$ && $O_x \subseteq O_y$)) {
15. the conflict of OP_x and OP_y is triggered conflict;
16. hasConflict = 1;
17. CS = CS ∪ { EO_y };
18. }
19. if(WS_i equals WS_j && OP_x and OP_y is the same operation && ($I_x \cap I_y = \varphi$ || $O_x \cap O_y = \varphi$)) {
20. the conflict of OP_x and OP_y is similar conflict;
21. hasConflict = 1;
22. $EO_{xy} = \{WS_i, OP_x, I_x \cup I_y, O_x \cup O_y\}$;
23. $CS = CS \cup \{EO_{xy}\}$;
24. }
25. CS is the set of evolution operation after digesting.

4　服务组合演化过程

4.1　服务组合演化执行

根据定义的演化操作集合,对服务组合执行演化操作,对于演化操作的两种基本类型,首先判定 PEO 的演化操作是否满足 NEO 中的约束,如果不满足,则演化需求 ER 无法执行,否则,开始执行 PEO.

对于 PEO,先执行 Del 操作,后执行 Add 操作. 步骤包括:首先,执行 Del 操作,删除指定的演化点的输入输出接口,若演化点成为一个孤立点,根据关系矩阵表示方法,当控制结构网表示为关系矩阵时孤立点将不会在其中表示,即被自动消去;其次,执行 Add 操作,增加演化点,在控制结构网中加入新的库所;最后,演化点的替换可以由 Del 操作和 Add 操作复合而成,可以综合上面两个步骤执行.

演化操作的执行可能会导致控制结构网产生一部分失效连接以及孤立的演化点,这些演化点将会在后续的演化推理中被补全到控制结构网中.

4.2　全局依赖网伪演化

演化操作执行之后服务组合将在演化点及其接口上发生变更. 除此之外,还应考虑演化操作对服务组合中其他服务所产生的影响. 因为演化操作除了直接影响演化点所对应的服务外,还可能还会级联影响到其他的服务,这些服务是被迫进行的变更. 为此,将直接受影响的演化点可以被理解为“策动源”,而被迫调整的服务可以被理解为“激活点”,这是在演化操作之后需要考虑的重要内容,称为演化推理.

为了给演化推理提供基础和参照系,需要对 GDN 进行“伪演化”. 即对 GDN 执行演化操作中的“Del 操作”,屏蔽掉 GDN 有关的服务及其接口,使这些服务及其接口失效. 由于这一过程仅仅是屏蔽操作,不会真正删除相关的服务及其接口,因此称为 GDN 伪演化.

5　服务组合演化推理

服务组合演化操作之后可能会导致组合控制结构网中产生无效的连接弧. 为此,需要对服务组合进行演化推理以补全断开的连接弧和剔除无效的连接弧,演化推理包括反向演化推理、正向演化推理两个阶段. 下面定义了反向演化推理与正向演化推理需要使用的参数与运算规则[23,24].

对于一个拥有 m 个库所与 n 个变迁的全局依赖网:

(1)输入关联矩阵:

$$\boldsymbol{D}^- = [d_{ij}^-]_{m\times n}, \text{其中 } 1 \leqslant i \leqslant m, 1 \leqslant j \leqslant n, \quad d_{ij}^- = \begin{cases} 1, & p_i \in I(t_j), p_i \in P, t_j \in T \\ 0, & \text{其他} \end{cases} \tag{2}$$

(2)输出关联矩阵:

$$\boldsymbol{D}^+ = [d_{ij}^+]_{m\times n}, \text{其中 } 1 \leqslant i \leqslant m, 1 \leqslant j \leqslant n, \quad d_{mn}^+ = \begin{cases} 1, & p_i \in O(t_j), p_i \in P, t_j \in T \\ 0, & \text{其他} \end{cases} \tag{3}$$

(3)⊕加运算:

$$\boldsymbol{X} \oplus \boldsymbol{Y} = \boldsymbol{Z}, z_{ij} = \text{Max}\ (x_{ij}, y_{ij}) \tag{4}$$

其中 $\boldsymbol{X}, \boldsymbol{Y}$ 均为 $s \times q$ 维矩阵;

(4)⊗乘运算:

$$\boldsymbol{X} \otimes \boldsymbol{Y} = \boldsymbol{Z}, z_{1i} = \underset{1\leqslant k\leqslant s}{\text{Max}}(x_{1k} \times y_{k1}), 1 \leqslant i \leqslant r \tag{5}$$

其中 $\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}$ 分别为 $1 \times s$、$s \times q$、$1 \times q$ 维矩阵;

(5)Θ 减运算:

$$\boldsymbol{X}\Theta\boldsymbol{Y} = \boldsymbol{Z}, z_{1i} = \underset{1\leqslant k\leqslant s}{\text{Min}}(x_{1k} - y_{k1}), 1 \leqslant i \leqslant q \tag{6}$$

其中 $\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}$ 分别为 $1 \times s$、$s \times q$、$1 \times q$ 维矩阵.

5.1　反向演化推理

反向演化推理是为了补全服务组合初始库所到演化点之间断开的通路并剔除无效的支路,其推理过程包括反向推理通路补全和剔除无效支路. 反向推理通路补全是以伪演化后的 GDN 为基础,初始补全部分为所有演化点,之后每执行一步,补全部分沿着 GDN 在演化点前驱结构方向上扩展一个库所,直到扩展到经演化操作后控制结构网的所有起始库所上,补全过程结

束之后得到的 Petri 网是 GDN 的子网,记为 $\boldsymbol{D}_p$.

将演化推理得到的 $\boldsymbol{D}_p$ 与控制结构网 $\boldsymbol{D}$ 执行加运算得到一个新的 Petri 网 $\boldsymbol{D}_{cp}$:

$$\boldsymbol{D}_{cp} = \boldsymbol{D}_p \oplus \boldsymbol{D} \tag{7}$$

$\boldsymbol{D}_{cp}$ 中存在一些支路,其端点并不是控制结构网的输入输出接口或演化点,因此这些支路并不在所需的服务组合中,需要将其剔除.

对于 Petri 网转换得到的关系矩阵,不考虑全为 0 的行和列,给出以下判定无效支路的规则:

① 无效支路端点:除起止库所与演化点的行外存在若干行,其中的值除 0 外皆为 1 或 -1,表示该行所代表的库所并非期望开始或结束的端点;

② 无效变迁:作为一个有效变迁,必须同时具有输入和输出连接弧,否则就成为一个无效变迁,其在关系矩阵中表示为列的值除 0 以外,其余值皆为 1 或 -1.

剔除无效支路的方法是:首先根据无效支路端点的判定规则找出所对应的行并将其删除;然后根据无效变迁的判定规则找出所对应的列并将其删除;最后删除全为 0 的行和列. 多次执行以上过程,直到无任何行和列满足无效支路端点和无效变迁的判定规则为止,如算法 3 所示.

算法 3 服务组合反向演化推理算法

Input: initial place vector $\boldsymbol{A}(0) = (a_1, a_2, \cdots, a_n)$, initial transition vector $\boldsymbol{B}(0) = (0,0,\cdots,0)$, relational matrix of global dependency relationship Petri net $\boldsymbol{D}$, matrix of Web service composition after evolution operation $\boldsymbol{W}$

Output: route from beginning place of Web service composition to evolution points

1. reverse Inference cycle value i, initial value of i is 1;
2. while(value of beginning place of Web service composition in $\boldsymbol{A}(i)$ equals 0){
3. $\boldsymbol{B}(i) = \boldsymbol{A}(i-1) \otimes (-\boldsymbol{D})$, $\boldsymbol{A}(i) = \boldsymbol{B}(i-1) \otimes \boldsymbol{D} \oplus \boldsymbol{A}(i-1)$;
4. $i++$;
5. }
6. $\boldsymbol{W} = \boldsymbol{W} \oplus \boldsymbol{D}_p$;
7. while(!(∃row and column of which all value is 1 or-1 but 0 except rows of input, output and evolution point)){
8. delete row of which all value is 1 or-1 but 0;
9. delete column of which all value is 1 or-1 but 0;
10. delete row and colomn of which all value is 0;
11. }

5.2 正向演化推理

正向演化推理则是为了补全演化点到服务组合终止库所之间断开的通路并剔除无效的支路,其推理过程分为正向推理通路补全和剔除无效支路. 正向推理通路补全是以伪演化后的 GDN 为基础,初始补全部分为所有演化点,之后每执行一步,补全部分沿着 GDN 在演化点后继结构方向上扩展一个库所,直到扩展到经演化操作后控制结构 Petri 网的所有终止库所上,补全过程结束,得到的一个 GDN 的子网,表示为 $\boldsymbol{D}_r$.

将演化推理得到的 $\boldsymbol{D}_r$ 与控制结构网 $\boldsymbol{D}$ 执行加运算得到一个新的 Petri 网 $\boldsymbol{D}_{cr}$:

$$\boldsymbol{D}_{cr} = \boldsymbol{D}_r \oplus \boldsymbol{D} \tag{8}$$

进行与反向演化推理中同样的剔除无效操作,如算法 4 所示.

算法 4 服务组合正向演化推理算法

Input: initial place vector $\boldsymbol{A}(0) = (a_1, a_2, \cdots, a_n)$, initial transition vector $\boldsymbol{B}(0) = (0,0,\cdots,0)$, relational matrix of global dependency network $\boldsymbol{D}$, output relational matrix $\boldsymbol{D}^+$, input relational matrix $\boldsymbol{D}^-$, matrix of Web service composition after evolution operation $\boldsymbol{W}$

Output: route from evolution points to ending place of Web service composition

1. positive Inference cycle value i, initial value of i is 1;
2. while(value of ending place of Web service composition in $\boldsymbol{A}(i)$ equals 0){
3. $\boldsymbol{B}(i) = \boldsymbol{A}(i-1)\boldsymbol{\Theta}(\boldsymbol{D}^-) + (1,1,\ldots,1)$, $\boldsymbol{A}(i) = \boldsymbol{D}^- \otimes \boldsymbol{B}(i)^{\mathrm{T}} \oplus \boldsymbol{A}(i-1)$;
4. $i++$;
5. }
6. $\boldsymbol{D}_r$ is the projection of $\boldsymbol{D}$ on $\boldsymbol{A}(i)$;
7. $\boldsymbol{W} = \boldsymbol{W} \oplus \boldsymbol{D}_r$;
8. ***while***(!(∃row and column of which all value is 1 or-1 but 0 except rows of input, output and evolution point)){
9. delete row and colomn of which all value is 0;
10. delete row of which all value is 1 or-1 but 0;
11. delete column of which all value is 1 or-1 but 0;
12. }

将以上服务组合反向演化推理得到的 $\boldsymbol{D}_{cp}$ 和正向演化推理得到的 $\boldsymbol{D}_{cr}$ 执行加运算:

$$\boldsymbol{D}_w = \boldsymbol{D}_{cr} \oplus \boldsymbol{D}_{cp} \tag{9}$$

$\boldsymbol{D}_w$ 即是演化推理得到的最终服务组合演化结果.

6 应用实例

以文献[25]建立的政务云平台中的 Web 服务库为例说明服务组合的演化过程. 该云平台面向专业技术职称(职务)评审领域,用户涵盖各系列的申报对象、审批部门、评审部门和评审专家等,为网上申报、网上审批、网上评审和电子投票提供了一个公共的服务平台,其中已注册了与应用领域相关的 Web 服务. 与申报和审批相关的主要原子服务如表 1 所示,这些原子服务构成了服务组合的演化域,基于演化域构建的全局依赖网如图 2 所示,Petri 网对应的关系矩阵可以根据定义 7 转换得到.

设已存在一个服务组合实例 W_0,它通过上述原子服务 A、B、D、E、F、G、H、I、M、O、R、T、V、W 组合而成,实现了新建申报材料、填写申报材料、导出/导入申报材料、审核申报材料、录入中评委意见、送审申报材料等主要功能. 由这些服务组合成的 W_0 的控制结构网如图3所示,由于原子服务 A、B、D、E、F、R 是申报环节和审批环节中必须使用的原子服务,在服务组合演化过程中不允许变更.

表1　原子服务列表

ID	服务名称	服务描述	输入接口参数	输出接口参数
A	Login	用户登录	UserName,Password	UserInfo,Role
B	CheckMaterial	检查材料状态	UserInfo,Role	MaterialFlag
C	GetDeptByPerson	查询报送单位	UserInfo,Role	DeptList
D	CreateMaterial	新建申报材料	MaterialFlag	ApplyNo
E	ShowMaterial	显示申报材料	MaterialFlag	ApplyNo
F	AddBaseInfo	添加基本信息	ApplyNo	BaseList
G	AddPaperInfo	添加论文信息	ApplyNo	PaperList
H	AddProjectInfo	添加项目信息	ApplyNo	ProjectList
I	AddTeachInfo	添加教学信息	ApplyNo	TeachList
J	UploadFilesInfo	上传佐证材料	ApplyNo	FileList
K	AddWaterMask	添加水印	FileList	FileListWM
L	SelectSubmitDept	选择报送单位	DeptList	DeptId
M	ExportMaterial	导出申报材料	BaseList,PaperList, ProjectList,TeachList	MaterialZip
N	SubmitMaterial	提交申报材料	BaseList,PaperList,ProjectList, TeachList,FileListWM,DeptId	MaterialSubmitting
O	ImportMaterial	导入申报材料	MaterialZip,Role	MaterialView
P	AcceptMaterial	接收申报材料	MaterialSubmitting,Role	MaterialView
Q	ReturnMaterial	退还申报材料	MaterialSubmitting,Role	MaterialReturned
R	CheckMaterial	审核申报材料	MaterialView	Checked,ApplyNo
S	SelectNextDept	选择上级单位	Checked	NextDept
T	CheckIntermediate	录入中评委意见	Checked,ApplyNo	Opinion,Checked
U	SendNextDept	报送材料给上级单位	NextDept	NextDeptSending
V	SelectCommit	选择评委会	Opinion,Checked	Commit
W	SubmitCommit	送审材料给评委会	Commit	CommitSubmitting

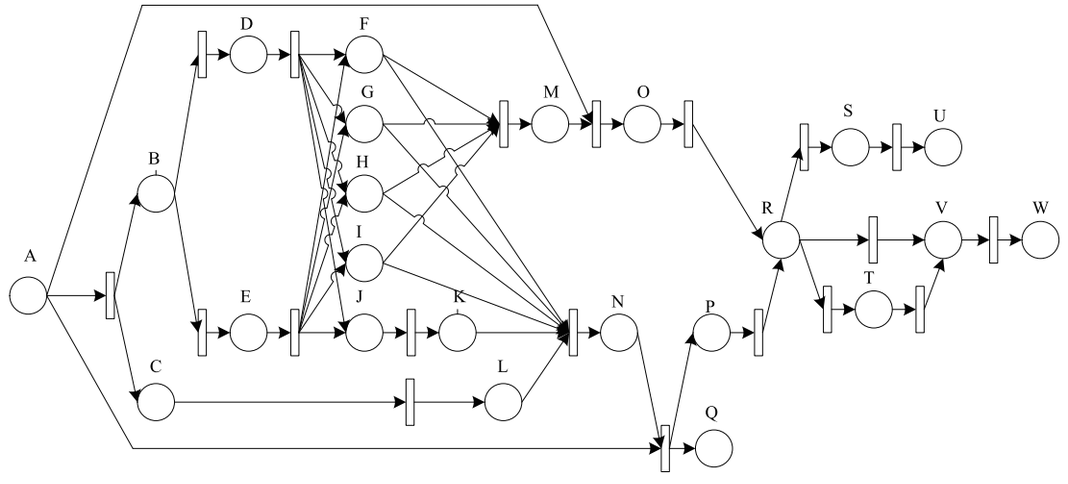


图2　基于演化域的服务全局依赖网

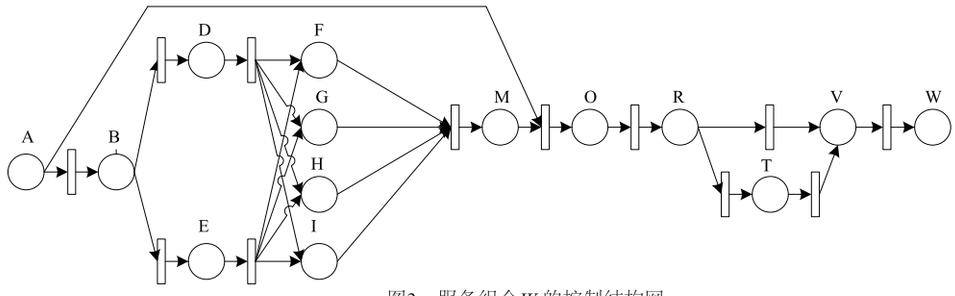


图3 服务组合W_0的控制结构网

根据以上描述,在服务组合 W_0 中,申报对象通过“导入/导出”向审批部门报送申报材料,且提供的申报材料中不含佐证材料. 由于新的需求,需要将报送材料的方式变更为“网上报送”,且申报材料中需要包含佐证材料并添加水印. 根据以上需求,找出演化点,明确演化操作,建立演化需求. 演化点为 J、M、N、O 和 P,删除服务 M 和 O 中所有输入输出接口,新增服务 J 的输入接口 ApplyNo 和输出接口 FileList,新增服务 N 的输入接口 BaseList、PaperList、ProjectList、TeachList、FileListWM 和 DeptId 以及输出接口 MaterialSubmitting,新增服务 P 的输入接口 MaterialSubmitting 和 Role 以及输出接口 MaterialView. 表 2 与表 3 给出了根据需求得到的正向与反向演化操作集合,由此得到初步的演化需求 ER 为:

ER = < { op_1, op_2, op_3, op_4, op_5, op_6 }, { op_1', op_2', op_3', op_4', op_5', op_6' } >.

表 2 正向演化操作集合 PEO

ID	正向演化操作四元组
op_1	< M, del, { BaseList, PaperList, ProjectList, TeachList }, { MaterialZip } >
op_2	< O, del, { MaterialZip Role }, { MaterialView } >
op_3	< J, add, { ApplyNo }, { FileList } >
op_4	< N, add, { BaseList, PaperList, ProjectList, TeachList, FileListWM }, ∅ >
op_5	< N, add, { DeptId }, { MaterialSubmitting } >
op_6	< P, add, { MaterialSubmitting, Role }, { MaterialView } >

表 3 反向演化操作集合 NEO

ID	反向演化操作四元组
op_1'	< A, del, { UserName, Password }, { UserInfo, Role } >
op_2'	< B, del, { UserInfo, Role }, { MaterialFlag } >
op_3'	< D, del, { MaterialFlag }, { ApplyNo } >
op_4'	< E, del, { MaterialFlag }, { ApplyNo } >
op_5'	< F, del, { ApplyNo }, { BaseList } >
op_6'	< R, del, { MaterialView }, { Checked, ApplyNo } >

对演化需求 ER 中的演化操作进行冲突检测与消解. ER 中的 PEO 操作集合通过算法 2 可以检测到演化操作 op_4 与 op_5 中存在相似冲突,因此将这两个演化操作进行合并得到一个新的演化操作:

op_7 = < N, add, { BaseList, PaperList, ProjectList, TeachList, FileListWM, DeptId }, { MaterialSubmitting } >.

由于其他演化操作之间不存在冲突,由此得到最终的演化需求 ER 为:

ER = < { op_1, op_2, op_3, op_6, op_7 }, { op_1', op_2', op_3', op_4', op_5', op_6' } >.

根据演化需求 ER 对服务组合 W_0 执行演化操作:

由于 PEO 完全满足 NEO 的约束,可以执行 PEO 中的演化操作. 首先执行 Del 演化操作,对于演化点 M 和 O,由于 op_1、op_2 对其接口进行了 Del 操作,因此需要将接口对应的连接弧消除;其次执行 Add 演化操作,将代表相应服务的库所添加到控制结构网中;演化操作执行完成后得到的控制结构网如图 4 所示,其中灰色库所代表执行演化操作后的演化点,该孤立的演化点 J、N、P 将会在后续的服务组合演化推理中被补全到控制结构网中.

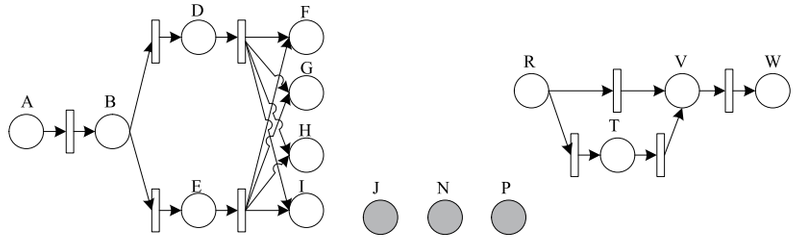


图4 对W_0执行演化操作后的控制结构网

在执行演化推理之前,需先对 GDN 执行伪演化操作,得到新的 GDN 如图 5 所示.

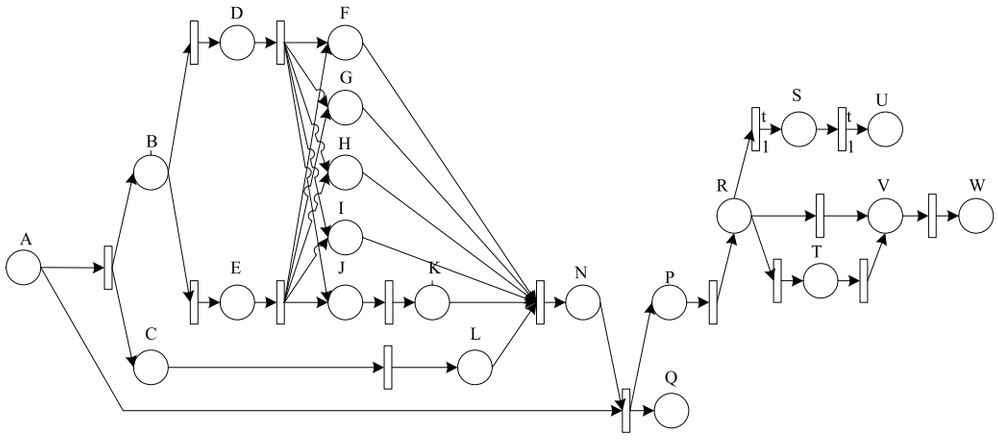


图5　伪演化后的全局依赖网

基于伪演化后的 GDN,对图 4 所示的控制结构网进行反向演化推理和正向演化推理.

(1)反向演化推理. 根据算法 3,执行反向推理通路补全过程,初始补全部分为 J、N、P,补全部分沿着 GDN 在演化点前驱结构方向上扩展到库所 A 后结束,得到包含库所 C、L、K 的 GDN 的子网 $\boldsymbol{D}_p$,将 $\boldsymbol{D}_p$ 与 $\boldsymbol{D}$ 执行加运算得到 $\boldsymbol{D}_{cp}$,剔除其中无效支路端点 O、R、T、V、W 及与其相关的无效变迁,其控制结构网如图 6 所示.

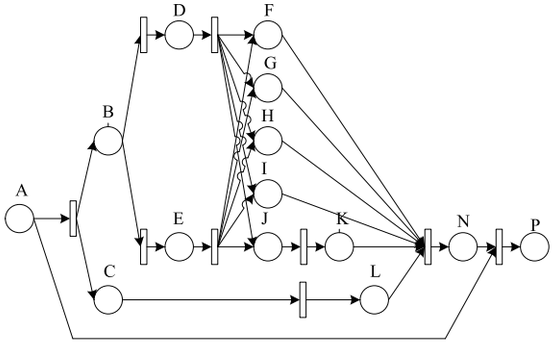


图6　反向演化推理后的控制结构网

(2)正向演化推理. 根据算法 4,执行正向推理通路补全过程,初始补全部分为 J、N、P,补全部分沿着 GDN 在演化点前驱结构方向上扩展到库所 W 后结束,得到包含库所 K 的 GDN 的子网 $\boldsymbol{D}_r$,将 $\boldsymbol{D}_r$ 与 $\boldsymbol{D}$ 执行加运算得到 $\boldsymbol{D}_{cr}$,剔除其中无效支路端点 A、B、D、E、F、G、H、I、O 及与其相关的无效变迁,其控制结构网如图 7 所示.

将反向演化推理得到的 $\boldsymbol{D}_{cp}$ 与正向演化推理得到的 $\boldsymbol{D}_{cr}$ 执行加运算,即执行 $\boldsymbol{D}_{cp}\oplus\boldsymbol{D}_{cr}$,得到服务组合演化的最终结果 $\boldsymbol{D}_w$,其控制结构网如图 8 所示.

根据图 8 演化得到的服务组合,用户首先登录到系统,如果是申报用户,则通过服务 B 判断是否需要新

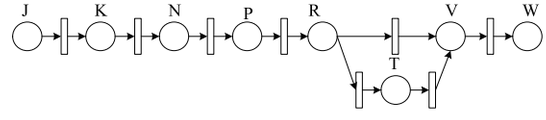


图7　正向演化推理后的控制结构网

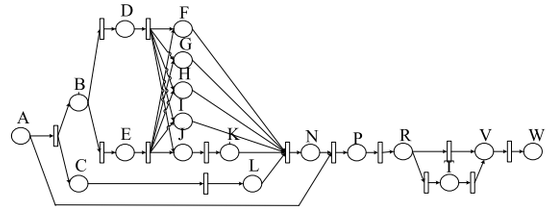


图8　最终演化得到的控制结构网

建申报材料,再通过服务 F、G、H、I、J、K 填写申报材料并加水印,同时,通过服务 C 和服务 L 查询并选择报送单位,最后通过服务 N 进行网上报送;如果是审批部门,则通过服务 P 接收申报材料,再通过服务 R 进行审核,然后判断是否通过服务 T 填写中评委意见,最后通过服务 V 和服务 W 选择评委会并将申报材料送审到指定的评委会,到此,申报与审核结束. 因此,演化得到的服务组合完全符合用户需求.

7　结束语

运行环境的动态性以及用户需求的多样化使得基于服务组合的软件系统不得不经常重新组合以适应新的变化. 为此,本文给出了一种基于全局依赖网的服务组合自动演化方法,将用户不确定性演化需求转换为能够被计算机理解的演化操作,以驱动服务组合演化过程,给出了详细的服务组合演化过程与执行方法,实现了服务组合的正向演化推理和反向演化推理,以补全断开的连接弧和剔除无效的连接弧. 下一步将从服

务组合演化引擎、服务演化可信性等方面继续开展研究.

参考文献

[1] 喻坚,韩燕波. 面向服务的计算-原理和应用[M]. 北京:清华大学出版社,2006.

[2] NEWCOMER Eric, LOMOW Greg. Understanding SOA with Web Services[M]. USA:Addison Wesley,2005.

[3] 张广泉,戎玫,朱雪阳,等. 基于 XYZ/ADL 的 Web 服务组合描述与验证[J]. 电子学报,2011,39(3A):86 -93.
ZHANG Guang-quan, RONG Mei, ZHU Xue-yang, et al. Specification and verification of web service composition based on XYZ/ADL[J]. Acta Electronica Sinica,2011,39(3A):86 -93. (in Chinese)

[4] 黄龙涛,邓水光,戴康,等. 基于 MapReduce 的并行 Web 服务自动组合[J]. 电子学报,2012,40(7):1397 -1403.
HUANG Long-tao, DENG Shui-guang, DAI Kang, et al. Automatic service composition in parallel with mapreduce [J]. Acta Electronica Sinica,2012,40(7):1397 -1403. (in Chinese)

[5] 倪悦,范玉顺. 基于着色 Petri 网的语义 Web 服务组合形式化验证[J]. 清华大学学报(自然科学版),2010,50(5):714 -717.
NI Yue, FAN Yu-shun. Formal verification for semantic Web services composition based on colored Petri nets[J]. Journal of Tsinghua University (Science and Technology), 2010,50(5):714 -717. (in Chinese)

[6] 王映辉,王立福. 软件体系结构演化模型[J]. 电子学报,2005,33(8):1381 -1386.
WANG Ying-hui, WANG Li-fu. research about model and ripple effect analysis of software architecture evolution[J]. Acta Electronica Sinica, 2005, 33(8): 1381 -1386. (in Chinese)

[7] 何成万,张立军,张慧. 基于元数据和反射的面向方面软件演化方法[J]. 电子学报,2011,39(8):1771 -1777.
HE Cheng-wan, ZHANG Li-jun, ZHANG Hui. An approach to aspect-oriented software evolution based on metadata and reflection[J]. Acta Electronica Sinica, 2011, 39(8):1771 -1777. (in Chinese)

[8] 余萍,马晓星,吕建,等. 一种面向动态软件体系结构的在线演化方法[J]. 软件学报,2006,17(6):1360 -1371.
YU Ping, MA Xiao-xing, LÜ Jian, et al. A dynamic software architecture oriented approach to online evolution[J]. Journal of Software, 2006, 17(6): 1360 -1371. (in Chinese)

[9] 彭焕峰,黄纬,范大娟,等. Web 服务演化综述[J]. 科学技术与工程,2015,(30):63 -70.
PENG Huan-feng, HUANG Wei, FAN Da-juan, et al. A survey of web service evolution[J]. Science Technology & Engineering,2015,(30):63 -70. (in Chinese)

[10] Wang S, CAPRETZ M A M. A dependency impact analysis model for web services evolution[A]. Proceedings of the IEEE International Conference on Web Services (ICWS)[C]. Los Angeles: IEEE Computer Society, 2009: 359 -365.

[11] 彭焕峰,黄纬,范大娟,等. 面向数据流的服务组合演化影响性分析方法[J]. 科学技术与工程,2015,15(1):257 -262.
PENG Huan-feng, HUANG Wei, FAN Da-juan, et al. Method for evolution impact analysis of service composition based on data flow[J]. Science Technology and Engineering,2015,15(1):257 -262. (in Chinese)

[12] 龙军,刘昕民,袁鑫攀,等. 一种基于信任推理与演化的 Web 服务组合策略[J]. 计算机学报,2012,35(2):298 -314.
LONG Jun, LIU Xin-min, YUAN Xin-pan, et al. A web services composition strategy based on trust reasoning and evolution[J]. Chinese Journal of Computers, 2012, 35(2):298 -314. (in Chinese)

[13] 黄光奇,蔡传青,沈陈平,等. 集不变条件下的服务组合演化行为研究[J]. 计算机科学,2012,39(6A):360 -364.
HUANG Guang-qi, CAI Chuan-qing, SHEN Chen-ping, et al. study on evolution behavior of service combination under unchanged set[J]. Computer Science, 2012, 39(6A): 360 -364. (in Chinese)

[14] SONG W, MA X, CHEUNG S C, et al. Preserving data flow correctness in process adaptation[A]. Proceedings of the IEEE International Conference on Services Computing [C]. Miami: IEEE Computer Society, 2010: 9 -16.

[15] LV C, JIANG W, HU S, et al. Efficient dynamic evolution of service composition[J]. IEEE Transactions on Services Computing, 2015, PP(99): 1 -1.

[16] 刘涛,曾国荪. 一种基于条件 Pi 演算的组合服务柔性演化模型[J]. 计算机科学,2011,38(S1):230 -238.
LIU Tao, ZENG Guo-sun. Model for flexible evolution of composite services based on conditional Pi calculus[J]. Computer Science, 2011, 38(S1): 230 -238. (in Chinese)

[17] 王晓璇,鲍爱华,缪嘉嘉,等. 一种面向语义 Web 的组合服务演化方法研究[J]. 计算机科学,2011,38(2):138 -143.
WANG Xiao-xuan, BAO Ai-hua, MIAO Jia-jia, et al. Research on the semantic web oriented method for the evolution of composite service[J]. Computer Science, 2011, 38(2):138 -143. (in Chinese)

[18] 尤殿龙,申利民,杨永涛. 大粒度 Web 服务组合业务流程演化类型判定方法[J]. 计算机集成制造系统,2014,

20(3):710 – 722.
YOU Dian-long, SHEN Li-min, YANG Yong-tao. Decision method for business process evolution type in larger-granularity Web service composition[J]. Computer Integrated Manufacturing Systems, 2014, 20(3):710 – 722. (in Chinese)

[19] 宋巍,马晓星,吕建. Web 服务组合动态演化的实例可迁移性[J]. 计算机学报,2009,32(9):1816 – 1831.
SONG Wei, MA Xiao-xing, LU Jian. Instance migration in dynamic evolution of web service compositions[J]. Chinese Journal of Computers, 2009, 32(9):1816 – 1831. (in Chinese)

[20] 张艳梅. 基于 Petri 网的 web 服务动态组合方法[A]. 北京地区高校研究生学术交流会--通信与信息技术会议论文集[C]. 北京:北京邮电大学出版社,2006. 1317 – 1321.
ZHANG Yan-mei. Dynamic composition of services based on perti net[A]. Proceedings of Beijing Academic Conference for Postgraduate on Communication and Information[C]. Beijing: Beijing University Posts and Telecommunications Press, 2016. 1317 – 1321. (in Chinese)

[21] 何丰. 语义 Web 服务组合若干关键技术研究[M]. 科学出版社,2013.

[22] 鲍爱华. 语义 Web 环境下组合服务演化方法及其关键技术研究[D]. 长沙:国防科学技术大学,2009.
BAO Ai-hua. Research on the Method and Key Technologies of Composite Service Evolution in Semantic Web [D]. Changsha: National University of Defense Technology, 2009. (in Chinese)

[23] 鲍培明. 模糊 Petri 网模型的反向推理算法[J]. 南京师范大学学报(工程技术版),2003,3(3):21 – 25.
BAO Pei-ming. Reverse reasoning algorithm based on the fuzzy petri net model[J]. Journal of Nanjing Normal University (Engineering and Technology Edition), 2003, 3(3):21 – 25. (in Chinese)

[24] 陈星,刘杰,余童兰. Petri 网的正向推理算法[J]. 微计算机信息,2006,22(12X):154 – 156.
CHEN Xing, LIU Jie, YU Tong-lan. Forward reasoning algorithm based on petri net[J]. Microcomputer Information, 2006, 22(12X):154 – 156. (in Chinese)

[25] ZHANG Yuan-min, NI Kuan, LU Jia-wei, et al. DOGCP: a domain-oriented government cloud platform based on PaaS[A]. Proceedings of the 2nd IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)[C]. New York: IEEE Computer Society, 2015: 115 – 120.

作者简介

张元鸣　男,1977 年 10 月出生于河南省濮阳市. 现为浙江工业大学计算机学院副教授、硕士生导师,研究方向为云计算、服务计算、并行计算等.
E-mail:zym@ zjut. edu. cn

倪　宽　男,1991 年 3 月出生于浙江省金华市. 现为浙江工业大学计算机学院硕士研究生,研究方向为服务计算.
E-mail:675657433@ qq. com

肖　刚(通信作者)　男,1965 年 4 月出生于浙江省绍兴市. 现为浙江工业大学计算机学院教授、博士生导师,研究方向为云计算、智能信息系统等.
E-mail:xg@ zjut. edu. cn