软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2013,24(12):2797−2813 [doi: 10.3724/SP.J.1001.2013.04396]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

面向数据流的网构软件服务动态演化分析*

宋 敏[1,2], 韦正现[3,1], 印桂生[1]

[1](哈尔滨工程大学 计算机科学与技术学院,黑龙江 哈尔滨 150001)

[2](北京外国语大学 信息技术中心,北京 100089)

[3](中国船舶工业系统工程研究院,北京 100036)

通讯作者: 韦正现, E-mail: weizhengxian@sina.com

摘 要: 网构软件需要组合多种异构服务并适应动态变化的网络环境,实现不间断服务和在线动态演化.为了将数据流显式地引入动态演化中,基于着色 Petri 网提出了面向数据流和控制流的网构软件服务模型.分析 5 种动态演化操作可能引发的数据流错误.为有效避免数据流错误的发生,首先提出面向数据流的服务实例可迁移性准则,然后提出了关于数据流/控制流交叉依赖关系的服务实例可迁移性准则,完整地刻画服务实例动态迁移约束特性.通过实验及结果分析,可以看出所提出的方法具有可行性和适用性.

关键词: 网构软件;数据流;动态演化;迁移;着色 Petri 网

中图法分类号: TP311 **文献标识码**: A

中文引用格式: 宋敏,韦正现,印桂生.面向数据流的网构软件服务动态演化分析.软件学报,2013,24(12):2797−2813. http://www.jos.org.cn/1000-9825/4396.htm

英文引用格式: Song M, Wei ZX, Yin GS. Evolution analysis of data flow oriented internetware service. Ruan Jian Xue Bao/Journal of Software, 2013,24(12):2797−2813 (in Chinese). http://www.jos.org.cn/1000-9825/4396.htm

Evolution Analysis of Data Flow Oriented Internetware Service

SONG Min[1,2], WEI Zheng-Xian[3,1], YIN Gui-Sheng[1]

[1](College of Computer Science and Technology, Harbin Engineering University, Harbin 150001, China)

[2](Information Technology Center, Beijing Foreign Studies University, Beijing 100089, China)

[3](Systems Engineering Research Institute, Beijing 100036, China)

Corresponding author: WEI Zheng-Xian, E-mail: weizhengxian@sina.com

Abstract: There is the need to combine internetware with various heterogeneous services and to adapt to the dynamic changing network environment to achieve uninterrupted service and online dynamic evolution. In order to explicitly draw the data flow into dynamic evolution, a data flow and control flow oriented internetware service model based on colored Petri nets (CPN) is put forward in this paper. Along with analyzing the data flow errors caused by five kinds of dynamic evolution operation, and in order to escape the data flow errors effectively, two data flow oriented service instance migratability criterions are given first. And then, a service instance migratability criterion about the cross dependencies between data flow and control flow is proposed to comprehensively describe the constraint attributes of service instance dynamic migration. The experiment results show that the methods provided in this paper are feasible and applicable to internetware service.

Key words: internetware; data flow; dynamic evolution; migration; colored Petri net

随着 Internet 应用的变化,要求计算机软件逐步适应开放、动态和多变的 Internet 环境,具备自主适应、动

* 基金项目: 国家自然科学基金(61272185, 61272186); 中央高校基本科研业务费专项资金(2012XJ020, HEUCFZ1219, HEUCF100608); 黑龙江省自然科学基金(F201110, F201238, F020510)

收稿时间: 2012-05-03; 修改时间: 2012-10-19, 2012-12-27; 定稿时间: 2013-03-27

态协同、在线演化和连续反应等形态特征,具有这些形态特征的软件称为网构软件(internetware)[1,2].网构软件的出现,使得 Internet 平台上存在大量可复用的构件或服务资源,这些资源以各种协同方式进行跨网络的互连、互通和相互协作形成联盟的软件实体,为用户提供 7×24、不断演化更新、智能化的服务,这要求网构软件能够感知环境的动态变化,并按照功能指标、性能指标和可靠性指标等进行在线动态演化[2−6].因此,在开放、动态网络环境下建立支持网构软件动态演化机制,是网构软件研究的一个重要课题.

从技术角度上,网构软件由提供具体服务的上层软件实体(网构软件服务)和底层支撑平台等组成,网构软件特征要求支撑平台暴露其内部状态和行为信息,还要实现上层软件实体内部状态和行为的实时展现,然后,通过监测、反射和切换等机制实现网构软件服务的自适应性和演化性[6].内部状态包括操作数据和控制状态,它们共同组成了数据流.因此,对数据流及其属性进行分析是实现网构软件自适应和动态演化的基础.

网构软件服务动态演化是将当前的服务实例(源模式)动态地迁移到新流程模式下(目标模式)继续执行,并且要求:(1) 源模式稳态的继承性,即已执行活动的相关数据、状态、关系和结果能够在目标模式下继承,并在后续的活动执行中得到应用;(2) 保证目标模式状态的有效性,不能引入动态演化错误,使目标模式产生死锁、活锁或流程异常终止等现象[3,4].网构软件是感知环境变化和自身状态的过程感知系统,需要从控制流(control flow)和数据流(data flow)两个方面对其动态演化开展研究[3].目前,控制流方面研究致力于保证服务过程模型实例在动态演化前后控制流的正确性、合理性、一致性[4,5,7−10].数据流方面研究主要分为两种方式:一是将变量作为过程模型的一阶实体[11−13],另一种是将活动变量作为标签附加到活动上[4,7,14−16].这两种方式将数据流依附在控制流上,既不能完全反映数据流对服务动态演化的影响特性,也不能满足网构软件的一个核心理论:软件协同分离化[2]的重要方面之一控制和数据分离的要求.同时,根据网构软件核心理论的形式化[2]要求,对网构软件进行动态演化分析首先需要建模,Petri 网是一种有效的方法.目前,基于 Petri 网的网构软件建模[17−19]缺乏对动态演化的直接与显式支持.因此,需要从控制流和控制流分离的角度对网构软件服务进行建模,并在模型的基础上对网构软件服务动态演化进程进行分析.

着色 Petri 网结合了 Petri 网和高级语言的优点,引入了数据概念,解决了 Petri 网无数据问题,具有强大的系统静态模型描述和动态行为分析的能力,能够自然表达系统并行、选择和循环等结构,适合异构软件联盟、柔性在线演化和数据/控制流交织等特点的网构软件建模与分析.因此,本文基于着色 Petri 网提出一种面向数据流和控制流的网构软件服务模型,目的是将数据流显式地引入网构软件服务建模过程中,从静态关系和动态运行两个方面研究网构软件服务实施过程中数据依赖关系及其特性,体现数据流和控制流在网构软件服务动态适应性和演化性中并重的特点.在此基础上,以面向数据流和控制流的网构软件服务模型为手段,分析数据流可能导致的动态演化错误,不但刻画出网构软件服务动态演化过程中数据流约束关系,而且描述了数据流和控制流在动态演化过程中表现出来的交叉依赖关系.为了避免数据流可能导致的动态演化错误,首先,提出面向数据流的服务实例动态可迁移性准则,适应以数据流为主要应用的网构软件服务动态适应性的要求;其次,提出服务实例关于数据流/控制流交叉依赖的动态可迁移性准则,保证网构软件服务动态演化实施的有效性;最后,通过实验验证本文的动态可迁移性准则的可行性和适用性.

本文的主要贡献表现在:(1) 给出了基于着色 Petri 网的网构软件服务模型,是对网构软件动态演化这一核心技术的形式化的深入分析;(2) 从数据流和控制流分离的角度,对 5 种动态演化操作可能引发的数据流错误、数据依赖关系的一致性、数据流和控制流的动态合理性这 3 个方面进行分析,是对网构软件协同的分离化[2]一个重要方面的研究;(3) 提出 3 个网构软件服务实例的可迁移性准则,是从数据流角度对网构软件动态演化机制的有益补充.

1 网构软件服务模型

1.1 网构软件服务模型描述

为了实现网构软件动态演化的形式化,满足动态演化过程中数据流和控制流分离的要求,本文参考已有基于着色 Petri 网(colored Petri nets,简称 CPN)的系统建模[20−22],给出网构软件服务模型,从控制流和数据流对网构

软件服务进行描述和分析.

定义 1(网构软件服务模型 IW_CPN):

$$IW\_CPN=(\Sigma,P_c,P_d,T,A_c,A_d,K,C,G,E_c,E_d,AT,IN,OUT,I),$$

其中,

- Σ为数据类型的集合(颜色集),控制托肯类型 $CONTROL\in\Sigma$,且 st 是控制托肯,其类型为 $CONTROL$,即 color $CONTROL$=with st;
- P_c 为非空有限控制库所集;
- P_d 为非空有限数据库所集;
- T 为非空有限变迁集,网构软件服务操作的集合;
- $A_c\subseteq P_c\times T\cup T\times P_c$ 是有限控制弧集;
- $A_d\subseteq P_d\times T\cup T\times P_d$ 是有限数据弧集,$(P_c\cup P_d)\cap T=\varnothing$;
- K 是控制库所容量函数,$\forall p_c\in P_c$ 有 $K(p_c)$=1st,即控制库所最多存放一个控制托肯;
- C 是数据类型函数,定义为 $C:(P_c\cup P_d)\to\Sigma$,即有:$P_c\to CONTROL$,$P_d\to\Sigma\setminus CONTROL$,指定库所 $p\in(P_c\cup P_d)$ 中的托肯类型为 $C(p)$;
- G 是防卫函数,定义为 $G:T\to G(t)$,满足$\forall t\in T:[Type(G(t))]=bool\wedge Type(Var(G(t)))\subseteq\Sigma$,指定变迁引发须满足的前提条件;
- E_c 是控制弧 A_c 上的函数,$\forall a_c\in A_c$ 有 $E_c(a_c)$=st,表示服务操作控制托肯的输入/输出;
- E_d 是数据弧 A_d 上的函数,定义为 $E_d:A_d\to E_d(a_d)$,满足:

  $$\forall a_d\in A_d:[Type(E_d(a_d))=C(p_d)_{MS}\wedge Type(Var(E_d(a_d)))\subseteq\Sigma\setminus CONTROL],$$

  p_d 是 $A_d(a_d)$中的数据库所,$C(p_d)_{MS}$ 是数据库所 p_d 的数据类型集上的多重集,E_d 用于表示服务操作数据托肯的输入/输出;
- AT 是变迁属性函数,定义为 $AT:T\to Attri(tType,opName,portType,inputSet,outputSet)$,指定了每一服务操作的变迁类型、服务操作名称、端口类型和输入/输出参数集,对于$\forall t\in T,(inv_1,inv_2)\in inputSet,outv_1\in outputSet$,则服务操作可记为 $f_d^t(inv_1,inv_2)\Rightarrow outv_1(inv_1,inv_2$ 为空时,可表示为 $f_d^t(0)$);
- IN 是变迁 T 中输入参数,$IN\in P_d$;
- OUT 是变迁 T 中输出参数,$OUT\in P_d$;
- I 是一初始化函数,满足$\forall p\in(P_c\cup P_d):[Type(I(p)=C(p)_{MS})$.

IW_CPN 模型入口为 $P_i=\{p_i|p_i\in P_c\}$,且 P_i 没有前集;模型出口为 $P_o=\{p_o|p_o\in P_c\}$,且 P_o 没有后集.模型入口点表示过程模型的开始,本文默认模型的入口库所为 1,对模型出口的数量不加限制.图 1 给出了 IW_CPN 的示意图.IW_CPN 的变迁表示网构软件服务操作(也称为活动),服务操作有两类托肯输入:控制托肯和数据托肯.控制托肯的流向指定了网构软件服务操作执行的顺序,数据托肯包括操作数据和业务逻辑控制数据,操作数据表示网构软件服务操作的输入/输出,业务逻辑控制数据和控制托肯一起决定流程的具体执行路径(关于类型(*Type*)、多重集(*MS*)、变量(*Var*)、托肯、前集($^\bullet t$)和后集($t^\bullet$)等概念详见文献[21]).

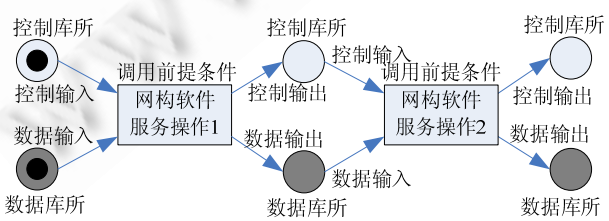


Fig.1 Internetware service model of IW_CPN

图 1 IW_CPN 的网构软件服务模型示意图

在 IW_CPN 模型中发生动态行为时,对所有的变迁 $t\in T$ 及所有的$(x_1,x_2)\in(P_c\cup P_d)\times T\cup T\times(P_c\cup P_d)$有:

1) $A(t)=\{a\in(A_c\cup A_d)|a\in(P_c\cup P_d)\times\{t\}\cup\{t\}\times(P_c\cup P_d)\}$:与 t 关联的弧集;
2) $Var(t)=\{v|v\in Var(G(t))\vee\exists a\in A(t):v\in Var(E(a))\}$:$t$ 的变量集;
3) $A(x_1,x_2)=\{a\in(A_c\cup A_d)|a\in(x_1,x_2)\}$:两端为 x_1,x_2 的弧集;
4) 当 $a\in(A_c(x_1,x_2)\cup A_d(x_1,x_2))$时,$E(x_1,x_2)=\sum E(a)$:两端点为 x_1,x_2 的弧函数集合.

在 IW_CPN 中,每次引发变迁都引起托肯在库所中动态的变化,从而推动服务流程向前运行.托肯元素是一(p,c)对,其中,$p\in(P_c\cup P_d)$,$c\in C(p)$,托肯元素集合用 TE 表示.在 IW_CPN 引发变迁时,需要先对变迁进行绑定(赋值),变迁绑定是定义在 $Var(t)$之上的函数 b,要求满足:1) $\forall v\in Var(t):b\langle v\rangle\in Type(v)$和 2) $G(t)\langle b\rangle$为“真”,绑定记为$\langle v_1=c_1,v_2=c_2,...,v_n=c_n\rangle$,$Var(t)=\{v_1,v_2,...,v_n\}$,绑定的集合记为 $B(t)$;绑定元素是(t,b)对,$t\in T,b\in B(t)$,绑定元素集合用 BE 表示.标识描述 IW_CPN 的状态,它是所有托肯元素集合 TE 之上的一个多重集,用 M 来表示.而步是描述引起 IW_CPN 状态变化事件的发生步骤,它是所有绑定元素集合 BE 之上的非空有限多重集,用 Y 来表示.

定义 2(步的使能). 当 IW_CPN 模型处于一个标识 M 时,步 Y 是使能的,当且仅当满足:

1) $\forall p_d\in P_d:\sum E_d(p_d,t)\langle b\rangle\leqslant M(p_d)$;
2) $\forall p_c\in P_c\wedge p_c\in{}^\bullet t:\sum E_c(p_c,t)\leqslant M(p_c)$;
3) $\forall p_c\in P_c\wedge p_c\in{}^\bullet t-t^\bullet:M(p_c)+E_c(t,p_c)\leqslant K(p_c)$;
4) $\forall p_c\in P_c\wedge p_c\in{}^\bullet t\cap t^\bullet:M(p_c)+E_c(t,p_c)-E_c(p_c,t)\leqslant K(p_c)$.

此时,称绑定(t,b)有效,同时称变迁 t 可引发.

其中,表达式 $\sum(p_d,t)\langle b\rangle$ 是当变迁 t 在绑定 b 引发时,需要从 t 的所有数据输入库所中删除的托肯.步 Y 的所有绑定元素$(b,t)\in Y$,当 Y 发生时,从每个数据输入库所删除的数据托肯数必须小于等于该数据输入库所中当前的数据托肯数.

定义 3(步的发生). 步 Y 的发生引起 IW_CPN 模型的状态从标识 M_1 改变为 M_2,并且称 M_2 从标识 M_1 直接可达的标识.

定义 4(步发生后的标识). 当在标识 M_1 时步 Y 使能,则其发生后标识变为 M_2:

$$\begin{cases}\forall p_d\in P_d,M_2(P_d)=(M_1(p_d)-\sum\limits_{(t,b)\in Y}E_d(p_d,t)\langle b\rangle)+\sum\limits_{(t,b)\in Y}E_d(p_d,t)\langle b\rangle\\ \exists p_c\in P_c,M_2(p_c)=\begin{cases}M_1(p_c)-E_c(p_c,t), & \text{if } p_c\in{}^\bullet t-t^\bullet\\ M_1(p_c)+E_c(t,p_c), & \text{if } p_c\in t^\bullet-{}^\bullet t\\ M_1(p_c)-E_c(p_c,t)+E_c(t,p_c), & \text{if } p_c\in{}^\bullet t\cap t^\bullet\\ M_1(p_c), & \text{if } p_c\notin{}^\bullet t\cup t^\bullet\end{cases}\end{cases}.$$

步 Y 的发生引起模型标识变化的过程,记为 $M_1[t\rangle M_2$.IW_CPN 模型定义了变迁之间的数据依赖关系,为了深入研究网构软件服务实例的动态迁移性,需要对 IW_CPN 模型的数据依赖关系及特性进行分析.

1.2 IW_CPN数据依赖关系分析

IW_CPN 的活动(变迁)中,当输入参数 v_i 在活动 t_i 中产生作用并输出参数 v_j 时,这两个参数就产生了依赖关系(称为参数依赖关系 $v_iR^Dv_j$),参数依赖关系可以分为直接参数依赖关系(记为 $v_iR^{DD}v_j$)和间接参数依赖关系(记为 $v_iR^{DI}v_j$).当一个活动 t_i 的输出参数成为另一个活动 t_j 的输入参数时,这两个活动就发生了数据依赖关系(称为活动数据依赖关系 $t_iR^Tt_j$),同样可以分为直接活动数据依赖关系(记为 $t_iR^{TD}t_j$)和间接活动数据依赖关系(记为 $t_iR^{TI}t_j$).在 IW_CPN 模型的活动序列$\delta=t_1t_2\ldots t_n$中,有活动 t_i,t_{i+1} 和 t_{i+2},它们的输入参数分别为 $IN_1=\{inv_{11},inv_{12},\ldots,inv_{1n}\}$,$IN_2=\{inv_{21},inv_{22},\ldots,inv_{2n}\}$ 和 $IN_3=\{inv_{31},inv_{32},\ldots,inv_{3n}\}$,输出参数分别为 $OUT_1=\{outv_{11},outv_{12},\ldots,outv_{1n}\}$,$OUT_2=\{outv_{21},outv_{22},\ldots,outv_{2n}\}$和 $OUT_3=\{outv_{31},outv_{32},\ldots,outv_{3n}\}$,经实施后的标识分别为 $M_{i1},M_{i2},M_{i3},M_{i0}$ 为 t_i 的初始标识,则有如下定义:

定义 5(IW_CPN 数据依赖关系).

(1) if $\begin{cases}\langle b_1\rangle=\langle inv_{11}=c_{in11},inv_{12}=c_{in12},...,inv_{1n}=c_{in1n}\rangle\\ G(t_i)\langle b_1\rangle=\text{true}\\ M_{i0}[t_i\rangle M_{i1}\\ f_d^{t_i}(inv_{11},inv_{12})\langle b_1\rangle\Rightarrow outv_{11}=c_{out11}\end{cases}$ then $inv_{11}R^{DD}outv_{11}$, $inv_{12}R^{DD}outv_{11}$;

(2) and if $\begin{cases}\langle b_2\rangle=\langle outv_{11}=c_{out11},inv_{21}=c_{in21},...,inv_{2n}=c_{in2n}\rangle\\ G(t_{i+1})\langle b_2\rangle=\text{true}\\ M_{i1}[t_{i+1}\rangle M_{i2}\\ f_d^{t_{i+1}}(outv_{11},inv_{21})\langle b_2\rangle\Rightarrow outv_{21}=c_{out21}\end{cases}$ then $inv_{11}R^{DI}outv_{21}$, $inv_{12}R^{DI}outv_{21}$, $t_iR^{TD}t_{i+1}$;

(3) and if $\begin{cases}\langle b_3\rangle=\langle outv_{21}=c_{out21},inv_{31}=c_{in31},...,inv_{3n}=c_{in3n}\rangle\\ G(t_{i+2})\langle b_3\rangle=\text{true}\\ M_{i2}[t_{i+2}\rangle M_{i3}\\ f_d^{t_{i+2}}(outv_{21},inv_{31})\langle b_2\rangle\Rightarrow outv_{31}=c_{out31}\end{cases}$ then $t_{i+1}R^{TD}t_{i+2}$, $t_iR^{TI}t_{i+2}$.

$f_d^{t_i}(inv_{11},inv_{12})\langle b_1\rangle\Rightarrow outv_{11}=c_{out11}$ 表示活动 t_i 在绑定 b_1 引发时,服务操作 $f_d^{t_i}(inv_{11},inv_{12})$ 实施时的具体动作. t_{i+1} 直接数据依赖于 t_i,t_{i+2} 直接数据依赖于 t_{i+1},但 t_{i+2} 不一定间接数据依赖于 t_i.如:

- $f_d^{t_i}(inv_{11},inv_{12})\langle b_1\rangle\Rightarrow outv_{11}=c_{out11}\wedge f_d^{t_{i+1}}(outv_{11},inv_{21})\langle b_2\rangle\Rightarrow outv_{21}=c_{out21}$;
- $f_d^{t_{i+1}}(inv_{22},inv_{23})\langle b_2\rangle\Rightarrow outv_{22}=c_{out22}\wedge f_d^{t_{i+2}}(outv_{22},inv_{31})\langle b_3\rangle\Rightarrow outv_{31}=c_{out31}$.

根据定义 5,在活动序列δ=$t_1,t_2,\ldots,t_n$ 中,对任意活动 t_i,t_{i+1} 和 t_{i+2} 有数据依赖关系 $t_iR^{TD}t_{i+1},t_{i+1}R^{TD}t_{i+2}$,并且从 t_i 到 t_{i+2} 存在传递参数依赖关系,则活动 t_{i+2} 数据依赖于活动 t_i,即 $t_iR^Tt_{i+2}$,因此,活动数据依赖关系的传递性通过参数依赖关系的传递性实现.根据关系传递闭包可得到定义 6 和定义 7.

定义 6(参数集关于活动的传递闭包). 设 R^T 为参数集 V 上的一组关于活动的数据依赖关系,$X\in V$, $R_D^+=\{A|X\rightarrow A$ 能由 R^T 根据参数依赖关系传递性导出$\}$, R_D^+ 称为参数集 X 关于活动数据依赖关系 R^T 的传递闭包,简称参数依赖传递闭包.

定义 7(活动集关于参数的传递闭包). 设 R^D 为活动集 T 上的一组关于参数的依赖关系,$X\in T$, $R_T^+=\{A|X\rightarrow A$ 能由 R^D 根据变迁数据依赖关系传递性导出$\}$, R_T^+ 称为活动集 X 关于参数的依赖关系 R^D 的传递闭包,简称活动数据依赖传递闭包.

依据定义 6,参数依赖传递闭包 $R_D^+=R^D\cup R^{D2}\cup R^{D3}\cup\ldots\cup R^{Dn}$.上面讨论了 IW_CPN 模型中的数据依赖关系及其性质,在网构软件服务实例动态迁移过程中,需要在源模式和目标模式之间保持这些数据依赖关系,避免动态演化中出现的数据流错误,因此需要分析在动态演化过程中可能产生的数据依赖关系错误.

2 网构软件服务数据流动态演化错误分析

网构软件服务实例从源模式动态迁移到目标模式下,数据流方面需要:

(1) 满足数据流的动态合理性,不能产生数据缺失(数据尚未建立)、数据冗余(数据未被使用)和数据丢失(多次写入覆盖)错误[14,16];

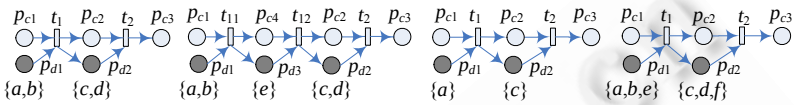
(2) 满足数据依赖关系的一致性;

(3) 满足数据流和控制流的动态合理性,不能引发由选择分支间存在数据依赖关系而产生死锁,不能引发由并行分支间存在循环的数据依赖关系而产生死锁,不能引发由并行分支间存在循环控制/数据依赖关系而产生死锁.

将网构软件服务实例从源模式动态迁移到目标模式,存在基本活动细化、判断条件细化、活动序列的增加或删除、并行选择循环分支的添加或删除、变量的增加或删除等演化操作方式[23].为了实施动态演化,需要从这 5 种演化操作角度,分析 3 类数据流方面存在的动态演化错误:

(1) 基本活动细化操作

将一个活动划分为几个活动,当该操作不改变该活动的数据输入/输出及其依赖关系,或者只是对细化内部的几个活动产生影响时,则该演化操作不会影响模型的整体依赖关系.当基本活动细化对该活动的输入/输出产生影响时,则有可能引发数据依赖关系错误,输入/输出的影响可以界定为变量的增加或删除.因此,这里同时讨论变量的增加或删除,如图 2 所示.



(a) 源模式 (b) 活动细化模式 (c) 输入输出删除模式 (d) 输入输出增加模式

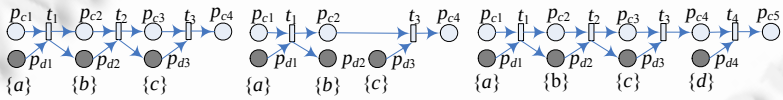
Fig.2 Data dependency of activity detailed

图 2 基本活动细化操作数据依赖关系

图 2(b)是将图 2(a)中的活动 t_1 细化为 t_{11} 和 t_{12},不涉及数据输入输出的改变,不会引发数据依赖关系错误;图 2(c)在图 2(a)的 p_{d1} 中删除了输入参数 b,并删除了 t_2 的输出参数 d,则有 $G(t_1)\langle a,b\rangle$=false 和 $G(t_2)\langle c,d\rangle$=false,因此产生参数 b 和 d 缺失;图 2(d)在图 2(a)中增加了 p_{d1} 的输入参数 e,在 t_1 中增加了输出参数 f,则 t_1 和 t_2 实施时,具体动作 $f_d^{t_1}(a,b,e)\langle b_1\rangle$ 和 $f_d^{t_2}(c,d,f)\langle b_2\rangle$ 未能执行,从而产生参数 e 和 f 冗余.

(2) 活动序列的增加或删除操作

在已有的活动序列中增加或删除一个活动或活动序列,如图 3 所示.



(a) 源模式 (b) 删除活动模式 (c) 增加活动模式

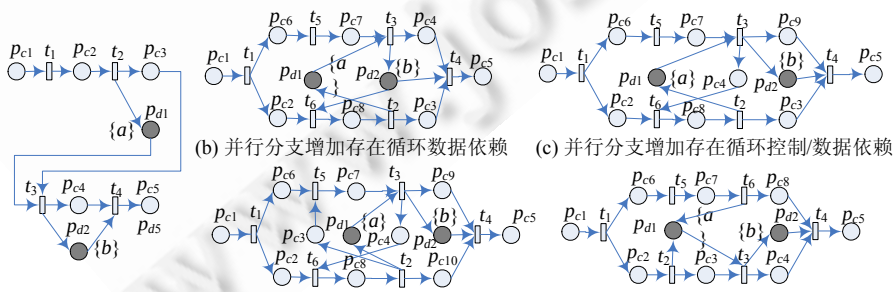
Fig.3 Data dependency of adding or deleting activity

图 3 活动序列增加或删除操作的数据依赖关系

图 3(b)在源模式中删除了活动 t_2,则有 $G(t_3)\langle c\rangle$=false,从而产生参数 c 缺失;同时具体动作 $f_d^t(b)\langle b_1\rangle$ 未能执行,从而产生参数 b 冗余;图 3(c)在源模式中增加活动 t_4,则有 $G(t_4)\langle d\rangle$=false,从而产生输入参数 d 缺失.

(3) 并行分支增加或删除操作

并行分支增加是在原来的模式上增加一个或多个并行分支,如图 4 所示.



(a) 源模式 (b) 并行分支增加存在循环数据依赖 (c) 并行分支增加存在循环控制/数据依赖

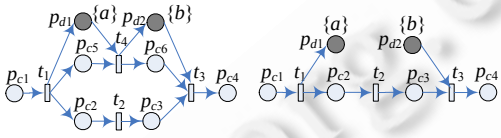
(d) 并行分支增加存在循环控制依赖 (e) 并行分支增加存在数据丢失

Fig.4 Data dependency of adding parallel branch

图 4 并行分支增加操作的数据依赖关系

从图 4(a)演化为图 4(b)的模式时有 aR^Db 和 bR^Da,导致 $f_d^{t_6}(b)\langle b_6\rangle$ 和 $f_d^{t_3}(a)\langle b_3\rangle$ 未能执行,使 t_3 和 t_6 形成循环数据依赖而引起死锁;演化为图 4(c)的模式时有 $t_2R^Tt_3$ 和 $M_3[t_6t_2\rangle M_2$,使 t_2 和 t_3 形成数据流/控制流交叉依赖关系而产生死锁;演化为图 4(d)时有 $M_5[t_3t_6\rangle M_6$ 和 $M_6[t_2t_5\rangle M_5$,使 t_5 和 t_6 形成控制流交叉依赖关系而产生死锁;演化为图 4(e)的模式时有 $f_d^{t_6}(0)\langle b_6\rangle \Rightarrow a$ 和 $f_d^{t_2}(0)\langle b_2\rangle \Rightarrow a$,使 t_2 和 t_6 都对 a 进行写入而产生数据丢失.

并行分支删除操作模式是在原来的模式上删除一个或多个并行分支.如从图 5(a)演化为图 5(b)模式时,具体动作 $f_d^t(a)\langle b_1\rangle$ 未能执行,从而产生参数 a 冗余;同时有 $G(t_3)\langle b\rangle$=false,从而产生 t_3 输入参数 b 缺失.



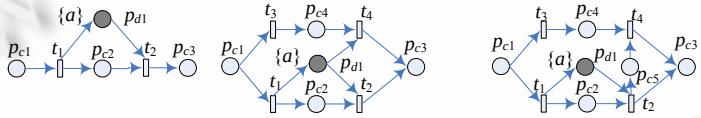
(a) 源模式 (b) 并行分支删除模式

Fig.5 Data dependency of deleting parallel branch

图 5 并行分支删除操作的数据依赖关系

(4) 选择分支的添加或删除操作

选择分支的添加是在原来的模式上增加一个或多个选择分支.如图 6(a)演化为图 6(b)模式后,$f_d^{t_1}(0)\langle b_1\rangle \Rightarrow a$, 执行 t_3t_4 分支时有 $G(t_4)\langle a\rangle$=false,使分支间存在数据依赖关系导致 t_4 因缺失输入参数 a 而产生死锁;由图 6(a)演化为图 6(c)模式后,执行 t_3t_4 分支时$\neg M_2[t_4\rangle M_4$,使分支间因存在控制依赖关系而形成死锁.

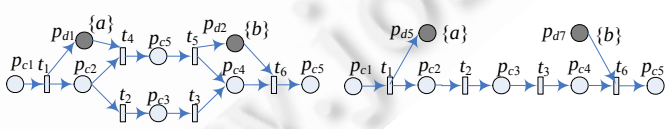


(a) 源模式 (b) 选择分支增加存在数据依赖 (c) 选择分支增加存在控制依赖

Fig.6 Data dependency of adding selection branch

图 6 选择分支添加操作的数据依赖关系

选择分支的删除模式是在原来模式上删除一个或多个选择分支,如图 7 所示.由图 7(a)演化为图 7(b)模式后,具体动作 $f_d^t(a)\langle b_1\rangle$ 未能执行,从而产生参数 a 冗余;同时有 $G(t_6)\langle b\rangle$=false,使 t_6 产生输入参数 k 缺失.



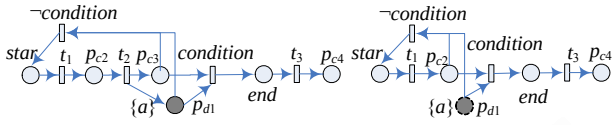
(a) 源模式 (b) 选择分支删除模式

Fig.7 Data dependency relations of adding selection branch

图 7 选择分支删除操作的数据依赖关系

(5) 循环体内活动添加或删除操作

如果删除循环体内改变循环条件判断参数的活动,如图 8(a)中删除活动 t_2 演化为图 8(b)模式后,则 $G(condition)\langle c\rangle$=false 和 $G(\neg condition)\langle c\rangle$=false,因此产生判断参数缺失而引起循环混乱.循环体内添加活动和删除不改变判断条件的活动,与活动序列的加入或删除操作情况相同.



(a) 源模式 (b) 循环体内删除活动模式

Fig.8 Data dependency of deleting loop activity

图 8 循环分支删除操作的数据依赖关系

不同的演化操作会引起不同的动态演化错误.从数据流角度,这些错误可以分为两类:数据依赖关系错误和数据流/控制流交叉依赖关系错误.网构软件服务实例迁移的正确性是一种动态正确性[3],因此需要确定服务实例的可动态迁移准则,确保在实施上述 5 种演化操作时,不会产生前述的 3 类数据流动态演化错误.

3 网构软件服务实例动态演化准则

当用户需求或环境改变时,当前的服务实例必须迁移到新流程下的一个有效的目标状态中恢复执行,以满足用户新需求或适应新环境.在网构软件服务实例动态迁移过程中,并不是所有实例都可以迁移,可能存在动态演化错误,为了避免这些错误,动态演化过程必须遵循一定的准则.

3.1 面向数据流的网构软件服务实例可迁移准则1

网构软件服务实例的动态迁移过程中,从数据流的角度,源模式中已执行活动之间的数据依赖关系在实例迁移到目标模式之后应保持这种数据依赖关系,因此得出可迁移性准则 1.

准则 1. 服务实例的源模式和目标模式分别为 N_s,N_t,在源模式 N_s 下,某一正在执行的服务实例的已执行活动序列 $\delta_s=t_1t_2\ldots t_n$、参数集合 $V_s=\{v_1,v_2,\ldots,v_n\}$、活动集合 $T_s=\{t_1,t_2,\ldots,t_n\}$、参数依赖关系 R^D 和活动数据依赖关系 R^T,动态迁移到目标模式 N_t 的活动序列 $\delta_t=t'_1,t'_2,...,t'_m$、参数集合 $V_t=\{v'_1,v'_2,...,v'_m\}$、活动集合 $T_t=\{t'_1,t'_2,...,t'_m\}$、参数依赖关系 R'^D 和活动数据依赖关系 R'^T,若 δ_t 的数据依赖关系满足:

(1) $\forall v_i,v_j\in(V_s\cap V_t),v_iR^Dv_j\Rightarrow v_iR'^Dv_j$;

(2) $\forall t_i,t_j\in(T_t\cap T_s),t_iR^Tt_j\Rightarrow t_iR'^Tt_j$,

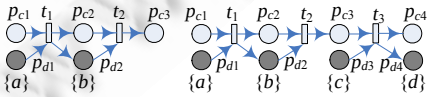
则该服务实例在数据依赖关系上是可迁移的.

准则 1 中:

(1) 如果源模式 N_s 有 $v_iR^Dv_j$,并且参数 v_i 和 v_j 在目标模式 N_t 中重现,则在目标模式 N_t 中同样有 $v_iR^Dv_j$;

(2) 如果源模式 N_s 存在 $t_iR^Tt_j$,并且活动 t_i 和 t_j 在目标模式 N_t 中重现,则在目标模式 N_t 中同样有 $t_iR^Tt_j$.

也就是说,如果源模式:首先,t_i 使用(生产)参数 v_i,t_j 再使用(生产)数据 v_j,其中,v_i 和 v_j 是相互依赖的参数,则在目标模式中,在 t'_i 使用(生产)数据 v_i 之前,t'_j 不能使用(生产)数据 v_j.如图 9(a)中有 aR^Db 和 $t_1R^Tt_2$,则图 9(b)中同样有 aR^Db 和 $t_1R^Tt_2$.准则 1 保证源模式和目标模式在数据依赖关系的一致性,在实施网构软件服务实例动态迁移时,不会产生数据缺失、数据丢失、数据冗余及相关依赖关系上的错误.



(a) 源模式 (b) 目标模式

Fig.9 Data dependency of migratability criterion 1

图 9 可迁移性准则 1 的数据依赖关系

可迁移性准则 1 要求保持服务实例源模式和目标模式相关的参数依赖关系以及活动数据依赖关系的一致性,通过 IW_CPN 模型,可以直接建立节点之间的直接依赖关系矩阵和直接参数依赖关系矩阵,而节点之间的依赖关系矩阵、参数依赖关系矩阵和活动数据依赖关系矩阵需要从这两个矩阵中求取.基于 Warshall 提出的求解传递关系闭包的算法,这里设计了相关依赖关系矩阵的生成算法.

设 IW_CPN 的节点集 $N=P_c\cup P_d\cup T$ 为 $\{n_1,...,n_{m+n+r}\}$,活动集合 $\{t_1,t_2,\ldots,t_n\}$,参数集合 $V=\{v_1,v_2,\ldots,v_n\}$.

- 节点之间的直接依赖关系矩阵为 *DirNM*[][],当 n_j 直接依赖于 $n_i(n_iR^{DD}n_j$ 或 $n_iR^{TD}n_j)$时,*DirNM*[*i*][*j*]为 1,否则为 0;
- 节点之间的依赖关系矩阵为 *NM*[][],当 n_j 依赖于 n_i 时,*NM*[*i*][*j*]为 1,否则为 0;
- 直接参数依赖关系矩阵为 *DirAPM*[][],当 $v_iR^{DD}v_j$ 时,*DirAPM*[*i*][*j*]为 1,否则为 0;
- 参数依赖关系矩阵为 *APM*[][],当 $v_iR^{D}v_j$ 时,*APM*[*i*][*j*]为 1,否则为 0;
- 活动间数据依赖关系矩阵为 *ADM*[][],当 $t_iR^{T}t_j$ 时,*ADM*[*i*][*j*]为 1,否则为 0.

算法 1. 参数依赖关系矩阵生成算法.

Input:直接参数依赖关系矩阵 *DirAPM*;

Output:参数依赖关系矩阵 *APM*.

```
Initialization: APM=DirAPM;
  If (对于 APM[i][j]=1) then {
    for (k=0; k<n; k++){
      APM[j][k]=APM[j][k]∪APM[i][k];}}
```

算法 2. 节点之间的依赖关系矩阵和活动数据依赖关系矩阵生成算法.

Input:节点之间的直接依赖关系矩阵 *DirNM*;

Output:节点之间的依赖关系矩阵 *NM* 和活动数据依赖关系矩阵 *ADM*.

```
Initialization: NM=DirNM; int u=0; int h=0;
  If (对于 NM[i][j]=1) then {
    for (int k=0; k<(m+n+r); k++){
      if (当 i 所对应节点的输入参数与输出参数相同时) then {
        NM[j][k]=NM[j][k]∪NM[i][k];}}}
  If (对于 NM[i][j]中每个元素对应的节点类型为活动(变迁)) then {
    repeat:{
      ADM[u][h]=NM[i][j]; u++; h++;}
    Until:遍历 NM}
```

算法 2 的主要思想是,遍历 IW_CPN 模型中的每个数据库所 p_{di} 中具有相同参数的两类数据弧集 $A_{d1}=T_{in}\times p_{di}$ 和 $A_{d2}=p_{di}\times T_{out}$,找到与之相关联的活动.即如果 $t_1\in T_{in},t_2\in T_{out},a_{d1}\in A_{d1},a_{d2}\in A_{d2},a_{d1}=t_1\times p_{di},a_{d2}=p_{di}\times t_2$,并且数据弧 a_{d1} 和 a_{d2} 具有相同的参数,则 t_1 和 t_2 数据相关,从而由节点之间的直接依赖关系矩阵得出节点之间的依赖关系矩阵,然后将节点类型为活动(变迁)的节点提取出来,形成活动数据依赖关系矩阵.

3.2 面向数据流的网构软件服务实例可迁移准则2

准则 1 中并没有考虑到源模式和目标模式参数的非重复性复现,即参数在目标模式中所属的活动,并不是源模式中该参数所属的活动,如图 10(a)的源模式 N_s 有 $\delta_s=t_1t_2t_3t_4$,其中,aR^Db 和 $t_2R^Tt_3$,参数 *a* 和 *b* 分别是 t_2 的输入/输出参数;图 10(b)的目标模式 N_t 中 $\delta_t=t_1t_5t_6t_4$,有 aR^Dc 和 bR^Dd,*a* 和 *b* 分别是 t_5 和 t_6 的输入参数.准则 1 下存在这种情况时,服务实例不可迁移.为了使这种情况下服务实例可迁移,这里给出可迁移性准则 2.

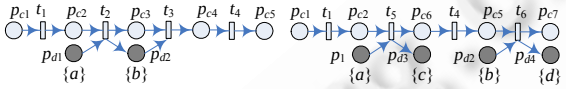
准则 2. 服务实例的源模式和目标模式分别为 N_s,N_t,在源模式 N_s 下,某一正在执行的服务实例的已执行活动序列 $\delta_s=t_1t_2\ldots t_n$、参数集合 $V_s=\{v_1,v_2,\ldots,v_n\}$、活动集合 $T_s=\{t_1,t_2,\ldots,t_n\}$、参数依赖关系 R^D 的传递闭包 $\delta_s(R_D^+)$ 和活动数据依赖关系 R^T 的传递闭包 $\delta_s(R_T^+)$,其动态迁移到目标模式 N_t 的活动序列 $\delta_t=t_1',t_2',...,t_m'$、参数集合 $V_t=\{v_1',v_2',...,v_m'\}$、活动集合 $T_t=\{t_1',t_2',...,t_m'\}$、参数依赖关系 R'^D 的传递闭包 $\delta_t(R_D'^+)$ 和活动数据依赖关系 R'^T 的传递闭包 $\delta_t(R_T'^+)$ (其中,$\delta_t=\delta_s\uparrow T_t$,表示 δ_s 到 T_t 的投影,即去掉 δ_s 中不属于集合 T_t 中的活动,而 δ_s 中剩余活动的相对顺序保持不变[4]),若 δ_t 的数据依赖关系满足:

(1) $(\forall v_i' \in V_t) \subseteq V_s, \delta_t(R_D'^{+}) \subseteq \delta_s(R_D^{+})$;

(2) $\delta_t(R_T'^{+}) \subseteq \delta_s(R_T^{+})$,

则该服务实例在数据依赖关系上是可迁移的.

准则 2 中,当源模式到目标模式的参数依赖关系和活动数据依赖关系的传递性得到保证时,则服务实例可迁移.依据准则 2,图 10(b)有 $\delta_t(R_T'^{+}) = \delta_s(R_T^{+})$ 和 $\delta_t(R_D'^{+}) = \delta_s(R_D^{+})$,所以可迁移,从而保证了源模式到目标模式参数的非重复性复现.这里需要计算参数依赖传递闭包和活动数据依赖传递闭包.



(a) 源模式 (b) 目标模式

Fig.10 Data dependency of migratability criterion 2

图 10 可迁移性准则 2 的数据依赖关系

算法 3. 参数依赖传递闭包生成算法.

Input:参数依赖关系矩阵 *APM*;

Output:APM 中参数的依赖传递闭包 *Param_Str*[].

Initialization: int *num*=0; int *pos*=0; String *t_str*=0, *Param_Str*[*n*]=0;

If (对于 *APM*[*i*][*j*]=1) then {

t_str←*APM*[*i*][*j*]对应行节点的参数 v_i+*APM*[*i*][*j*]对应列节点的参数 v_j(即串 *t_str*=v_iv_j);

for (int *u*=0; *u*<*n*; *u*++){

if (参数 v_u 的依赖传递闭包 *Param_Str*[*u*]中包含没有 *t_str*) then {

Param_Str[*u*]←*Param_Str*[*u*]+*t_str*;}}

for (int *k*=0; *k*<*n*; *k*++){

Param_Str[*u*]←*Param_Str*[*u*]+*APM*[*i*][*j*]对应行节点的参数 v_i+*k* 在 *APM* 所对应的参数 v_k+*APM*[*i*][*j*]对应列节点的参数 v_j;}}

算法 3 的主要思想是:遍历参数依赖关系矩阵,将每个参数的依赖传递闭包以串的形式表现出来.活动数据依赖传递闭包与算法 3 相似,这里不介绍.

网构软件服务实例迁移过程中,要求不产生数据缺失、数据冗余和数据丢失等错误,并保持数据依赖关系的一致性,同时还需要避免由数据流/控制流交叉依赖关系而导致的动态演化错误.

3.3 面向数据流/控制流的服务实例可迁移准则3

从 IW_CPN 模型的动态运行规则上看,引起控制流/数据流的冲突而产生死锁的两个原因是:数据流之间形成的循环依赖关系;数据流和控制流之间形成的循环依赖关系.由于动态演化的正确性是一种动态正确性,因此,要消除因为数据依赖关系的改变而引起的控制流/数据流交叉依赖关系错误,服务实例的源模式和目标模式必须遵循 IW_CPN 模型的约束及其动态运行规则,因此得到可迁移性准则 3.

准则 3. 服务实例的源模式和目标模式分别为 N_s,N_t,在源模式 N_s 下,某一正在执行的服务实例的已执行活动序列 $\delta_s=t_1t_2\ldots t_n$、参数集合 $V_s=\{v_1,v_2,\ldots,v_n\}$、活动集合 $T_s=\{t_1,t_2,\ldots,t_n\}$,其动态迁移到目标模式 N_t 的活动序列 $\delta_t=t_1',t_2',\ldots,t_m'$、参数集合 $V_t=\{v_1',v_2',\ldots,v_m'\}$、活动集合 $T_t=\{t_1',t_2',\ldots,t_m'\}$、参数依赖关系 R'^{D} 的传递闭包 $\delta_t(R_D'^{+})$ 和活动数据依赖关系 R'^{T} 的传递闭包 $\delta_t(R_T'^{+})$ (其中,$\delta_t=\delta_s\uparrow T_t$),如果 $\exists\lambda=t_1''t_2''\ldots t_m''$ 和 $\exists\rho=t_1''t_1't_2''t_2'\ldots t_m''t_m'$ (其中,t_i'' 为空活动或者为目标模式下可发生的活动)、参数依赖关系传递闭包 $\lambda(R_D^{+})$ 和活动数据依赖关系传递闭包 $\lambda(R_T^{+})$,若 δ_t 的数据依赖和控制依赖关系满足:

(1) $t_i'' \cap \delta_t = \varnothing \wedge \lambda(R_D^{+}) \not\subseteq \delta_t(R_D'^{+}) \wedge \lambda(R_T^{+}) \not\subseteq \delta_t(R_T'^{+})$;

(2) $M_0[t_0t_1t_2\ldots t_m\rangle M_t$,

则该服务实例是可迁移的,M_t 即为目标状态.

要证明准则 3 保证在服务实例的动态迁移过程中不会因数据流/控制流的冲突而产生死锁,并且迁移后的目标状态是有效的,则需证明:

(1) 准则 3 不会引起数据依赖关系错误;

(2) 准则 3 确保控制流/数据流依赖关系不会产生冲突,从而导致服务流程产生死锁.

证明:

(1) 不会引起数据依赖关系错误.

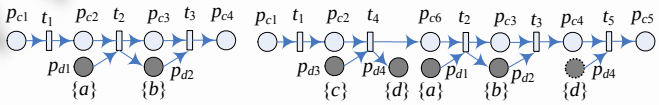
由于 $t''\cap\delta_t=\varnothing$,因此不存在重复的活动;同时,由于 $\lambda(R_D^+)\underline{\subseteq}\delta_t(R_D'^+)\wedge\lambda(R_T^+)\underline{\subseteq}\delta_t(R_T'^+)$,表明服务实例从源模式迁移到目标模式下已执行活动的参数依赖关系、活动数据依赖关系及其传递闭包,与目标模式下其他活动的参数依赖关系、活动的参数依赖关系及其传递闭包是相互独立的,因此不会引起数据依赖关系错误的发生.

(2) 确保控制流/数据流依赖关系不会产生冲突,从而导致服务流程产生死锁.

控制流/数据流依赖关系产生冲突主要发生在并行分支增加模式和选择分支增加模式.

由于 $M_0[t_0t_1t_2\ldots t_m\rangle M_t$,则从初始状态开始经过 m 步,对于每一步 Y_i 是使能的,并且其绑定是有效的;同时,该步对应的活动 t_i 可引发,在所有的 m 步发生后,其标识变为 M_t,根据 IW_CPN 网中的动态运行规则,M_t 是有效的.□

准则 3 保证服务实例在发生迁移时数据流/控制流的一致性,如图 11(a)的源模式中有 $aR^Db\wedge t_2R^Tt_3$,演化为图 11(b)模式后,有 $aR^Db\wedge t_2R^Tt_3, cR^Dd\wedge t_4R^Tt_5, \lambda=t_4t_5, \delta_t=t_1t_2t_3$ 和 $\lambda(R_D^+)\underline{\subseteq}\delta_t(R_D^+)\wedge\lambda(R_T^+)\underline{\subseteq}\delta_t(R_T^+)$;同时,$G(t_4)\langle b\rangle$和 $G(t_5)\langle b\rangle$ 为真,所以 $M_s[t_4t_5\rangle M_t$.因此,该服务实例可迁移.



(a) 源模式 (b) 目标模式

Fig.11 Data and control dependency of migratability criterion 3

图 11 可迁移性准则 3 的数据依赖和控制依赖关系

准则 1 保证服务实例在源模式和目标模式中,不会产生数据缺失、数据丢失、数据冗余及其依赖关系的错误;准则 2 则进一步保证了数据流在源模式和目标模式上,参数的非重复性复现的可迁移性;准则 3 则从数据流和控制流相结合的角度,保证服务实例从源模式到目标模式可迁移的一致性、合理性和正确性.这 3 个准则确保当实施本文论述的 5 种演化操作时,不会产生相应的 3 类数据流动态演化错误.

4 实验与分析

为验证本文所提出的动态演化准则的正确性和一致性,我们构建了动态演化管理引擎原型,并进行物流服务系统网构软件实例动态迁移的模拟实验.管理引擎主要管理、判断和实现物流服务系统网构软件原型实例的动态迁移,其过程为:

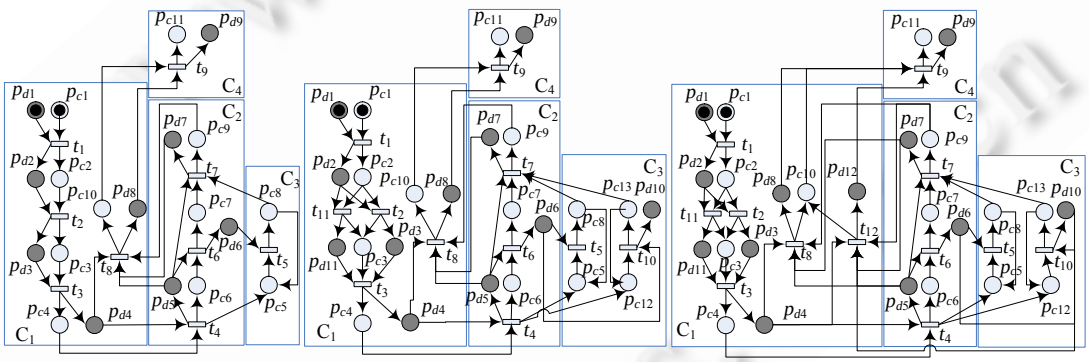
(1) 构建物流服务系统网构软件源模式,借鉴文献[11−13,24]的方法与规则,构建与源模式流程相容的新流程,新流程的活动用 IW_CPN 中的变迁来表示,活动的输入变量和输出变量转换为 IW_CPN 中的相应元素,从而获得目标模式及其活动系列,将源模式和目标模式转换为 IW_CPN,建立各模式的节点之间的直接依赖关系矩阵和直接参数依赖关系矩阵;

(2) 按照源模式的 IW_CPN,模拟多个物流服务系统应用实例运行;

(3) 暂停当前正在执行的实例,判定实例的可迁移性以及确定目标状态.从日志文件中获取各个实例的历史执行信息(即已执行活动序列以及相关参数信息),根据实例的已执行活动序列和数据依赖关系,判断在源模式下正在执行的实例是否可以迁移到目标模式;

(4) 恢复已暂停实例的执行.若实例可迁移到目标模式,则可根据已执行的活动信息以及目标状态下能够发生的活动,在目标模式下恢复执行.若实例不可迁移,则在原流程下恢复该实例的执行.

物流服务系统网构软件由物流业务服务软件(C_1)、运输服务模拟软件(C_2)、网上跟踪监控服务软件(C_3)和银行支付服务软件(C_4)组成的软件联盟.第 1 个版本(其 IW_CPN 简化模型如图 12(a)所示)的服务流程描述为:当客户要求货运时,首先选择货运的目的地及相关要求,然后确定运输货物并签订合同;在运输过程中,客户可以通过互联网对货物的运输情况进行跟踪,当货物送达目的地后,客户验货、计算实际运输费用并付款,在计算实际运输费用时,根据货物交付运输和货物到达时的日期、状态与合同规定的误差,分为 3 个等级,如果没有误差,则按照合同规定(f)的金额付款(p),即 $p=f$;如果误差在一定范围内,即$\Delta t=[0,t_1]$,$\Delta s=[0,s_1]$,则 $p=f(\alpha_1\Delta t+\beta_1\Delta s)$;如果误差超过了一定范围,即$\Delta t>t_1$,$\Delta s>s_1$,则 $p=f(\alpha_2\Delta t+\beta_2\Delta s)$.在第 1 个版本的基础上改进了业务流程(如图 12(b)所示),新增对货物运输环境实施监控的功能(t_{10}),即采用二维码对货物进行标识,并采用温度、湿度和空气等传感器获取运输途中的环境状态,然后,通过有线网络或无线传感器网络实现运输途中货物状态、运输环境和物品位置等与互联网连接,客户可以在互联网对运输环境实施监控,该功能在试运行期间对客户免费;相应地,C_1 增加了一种货物运输业务,将运输方式分为两种:一种是普通运输(t_2);另一种是状态监控运输,即在互联网上为客户提供对运输过程的环境实施监控(t_{11}),实现从版本 1 演化到版本 2.新增功能试运行一段时间稳定后,决定针对状态监控运输方式新增一种监控费用结算方法(如图 12(c)中的 t_{12} 所示),当客户选择普通运输方式(t_2)时,则采用原来的计费方法(t_8);当客户选择监控运输方式(t_{11})时,则采用新增的监控计费方法(t_{12}),实现版本 2 演化到版本 3.当 t_{12} 将运输过程中的环境状态作为计费的依据,其 3 种计费方式分别为 $p=f$,$p=f(\varepsilon_1\Delta t+\delta_1\Delta s+\gamma_1\Delta e)$和 $p=f(\varepsilon_2\Delta t+\delta_2\Delta s+\gamma_2\Delta e)$.该系统 IW_CPN 模型的相关参数见表 1.



(a) 物流服务网构模式1 (b) 物流服务网构模式2 (c) 物流服务网构模式3

Fig.12 Internetware evolution mode of logistics service

图 12 物流服务网构系统动态演化模式

Table 1 Name and implication about IW_CPN

表 1 IW_CPN 相关术语及含义

T	含义	P_d	V	含义
t_1	选择运输的货物及目的地	P_{d1}	–	–
t_2	确定运输(普通)要求	P_{d2}	v_1	运输的货物及目的地
t_3	签订合同	P_{d3}	v_2	运输过程相关(普通)要求
t_4	货物交付开始运输	P_{d4}	v_3	计划交付运输及抵达日期及运输金额
t_5	货物运输位置跟踪	P_{d5}	v_4	交付货物数量、状态和日期
t_6	货物运输	P_{d6}	v_5	货物当前所在位置
t_7	货物到达客户验货	P_{d7}	v_6	货物到达数量、状态和状态
t_8	(普通)费用结算	P_{d8}	v_7	实际运输(普通)应付金额
t_9	付款	P_{d9}	v_8	付款金额
t_{10}	货物运输状态监控	P_{d10}	v_9	运输环境温度、湿度和空气状态
t_{11}	确定运输(监控)要求	P_{d11}	v_{10}	运输过程相关(监控)要求
t_{12}	(监控)费用结算	P_{d12}	v_{11}	实际运输(监控)应付金额

在实验过程中,设定两个演化案例,即从模式 1 到模式 2、模式 2 到模式 3 的动态演化(实例动态迁移场景见表 2).在每个演化案例中设计了 10 个实例在随机运行,实验结果如下:

服务迁移实验案例 1:从模式 1 到模式 2 的动态演化,所有的实例都是可迁移的.如某一服务实例(实例 1)在模式 1 下已执行活动系列为$\delta_1=t_1t_2t_3t_4t_6(t_5)^3t_7((t_5)^3$ 表示 t_5 执行 3 次),将其动态迁移到模式 2 下继续执行,则该实例迁移到新模式下的活动系列为$\delta_2=t_1t_2t_3t_4t_6(t_5)^3t_7t_8t_9$.依据可迁移准则 2 及其相关算法判定该实例的可迁移性:

(1) $V_2\subseteq V_1,\delta_2(R_D^+)\subseteq\delta_1(R_D^+)$;

(2) $M_1[t_8t_9\rangle M_2$.

具体地,在该状态下通过执行有效的绑定(t_8,b_8)和(t_9,b_9),并且当步 Y_8 和 Y_9 发生后,得到有效的目标状态为 M_2,因此,该服务实例是可动态迁移的.

Table 2 Instances migration scenario

表 2 实例动态迁移场景说明

实验案例	源模式	目标模式	演化场景切换	迁移准则
实例 1	$t_1t_2t_3t_4t_6(t_5)^3t_7$	$t_1t_2t_3t_4t_6(t_5)^3t_7t_8t_9$	源模式(版本 1)在执行到顾客验货后(t_7),迁移到增加过程监控(t_{10})功能和状态监控运输方式(t_{11})的新模式下(版本 2)执行解算和付款	准则 2
实例 2	$t_1t_{11}t_4t_6t_5(t_{10})^2t_5t_7$	$t_1t_{11}t_4t_6t_5(t_{10})^2t_5t_7t_{12}t_9$	源模式(版本 2)中客户选择状态监控运输(t_{11}),在执行到顾客验货后(t_7),迁移到执行两种结算方式(t_8 和 t_{12})的新模式下(版本 3)执行解算和付款	准则 3

服务迁移实验案例 2:从模式 2 到模式 3 的动态演化.如果已执行活动系列包含 t_1t_{11},而未执行 t_8 的实例,并且运输过程存在误差时,则不能动态迁移到目标模式下,所以并不是所有的实例都是可迁移.如有一服务实例(实例 2)在模式 2 下已执行活动系列为$\delta_3=t_1t_{11}t_4t_6t_5(t_{10})^2t_5t_7$,要将其动态迁移到服务模式 3 下继续执行,则该活动系列迁移到新模式下的活动系列为$\delta_4=t_1t_{11}t_4t_6t_5(t_{10})^2t_5t_7t_{12}t_9$.根据算法 3,在$\delta_3$ 中参数 v_7 的传递闭包为 $v_1v_2v_3v_4v_6v_7$,而在δ_4 中参数 v_{11} 的传递闭包为 $v_1v_2v_3v_4v_6v_9v_{11}$,即 $V_4\subseteq V_3,\delta_4(R_D^+)\underline{\subset}\delta_3(R_D^+)$.所以,该服务实例不可立即迁移到目标模式下,需要采用延时迁移的方法来解决.

实验案例同时具有数据流和控制流的特点.为了更深入地说明本文提出方法的有效性,在实验中,我们还利用一些已有的可用于判定实例可迁移的方法,与本文提供的方法进行了对比实验,实验结果表明:

(1) 对于从模式 1 到模式 2,文献[7]、文献[11−13]和文献[14]提供的方法对其实施动态演化,其结果显示所有的实例都是可迁移的,与本文方法的结果相同;

(2) 对于从模式 2 到模式 3 的实例 2 的可迁移性判定,其结果见表 3(√表示从对应的数据流或控制流方面,实例在相应的可迁移性判定方法下是可迁移的,×表示不可迁移,--表示未单独从数据流或控制流进行可迁移性判定).

Table 3 Comparison between relevant experiences about migratability

表 3 实例 2 在相关工作中可迁移性比较

	文献[7]	文献[11−13]	文献[14]	文献[25]	本文方法
数据流	–	√	√	×	×
控制流	×	√	–	–	√

具体分析如下:

(1) 文献[7]的方法将数据绑定到活动中,通过判定活动执行系列的向前兼容和向后兼容对服务实例的可迁移性进行判定,向前兼容要求实例在源模式已执行活动的每一条后续活动的执行序列都能在目标模式下重现,向后兼容要求该实例活动序列可以在目标模式下按照既定序列重现.实例 2 的源模式已执行的活动序列为 $t_1t_{11}t_4t_6t_5(t_{10})^2t_5t_7$,则它完全执行时既定的顺序为 $t_1t_{11}t_4t_6t_5(t_{10})^2t_5t_7t_8t_9$,而目标模式为 $t_1t_{11}t_4t_6t_5(t_{10})^2t_5t_7t_{12}t_9$,因此,实例 2 已执行活动序列不满足向后兼容,从控制流上判定是不可迁移的;

(2) 文献[11−13]的方法从读写依赖上进行可迁移性判定,如果实例的已执行活动序列对数据的读写依赖能够严格按照既定顺序在目标模式下重现,则判定可迁移.实例 2 的源模式已执行 $t_1t_{11}t_4t_6t_5(t_{10})^2t_5t_7$,在目标模式 $t_1t_{11}t_4t_6t_5(t_{10})^2t_5t_7t_{12}t_9$ 下,已执行的活动序列对数据的读写依赖能够按照既定顺序重现,因此判定可迁移;

(3) 文献[14]的方法将活动的输入/输出变量作为活动的标签,通过判定活动使用数据时是否存在数据尚未建立就被使用、数据创建后从未被使用和对同一数据进行多次写入覆盖等数据流错误,来验证工作流的流程.该方法可用于服务实例的可迁移性判定,实例 2 中并没有出现该文规定的数据流错误,因此判定可迁移;

(4) 文献[25]的方法采用数据在活动之间的继承关系进行判定,如果在源模式和目标模式之间符合该文规定的继承关系,则判定可迁移.在实例 2 的目标模式中,t_{12} 的参数 v_{11} 不继承 t_{10} 的参数 v_9,判定不可迁移.与本文的方法一致,但它并没有从数据流和控制流相结合的角度综合考虑实例的可迁移性.

从表 2 上看,与现有的可迁移性判定方法相比较,一方面,本文提出的服务实例可迁移性准则首先考虑了服务过程模型的数据流,避免了由于数据依赖关系,一些原本不可迁移的实例迁移到新流程下而引发错误;另一方面,本文提出的服务实例可迁移性标准从数据流和控制流两个方面保证可迁移性判定结果的正确性和有效性,从而确保可迁移的服务实例能够实施动态演化.

5 相关工作

目前,网构软件服务演化技术是将已有演化方法应用于网构软件,在动态演化方面的研究主要集中于工作流和面向服务的动态演化两个方面.Papazoglou 教授在文献[26]中较为全面地总结了几种常见的面向服务演化类型,并提出了一个面向演化的服务生命周期(lifecycle)方法学以应对演化带来的特殊要求.文献[27]引入了服务演化管理的理念,并对面向服务的演化过程中所产生的各个不同版本的一致性进行了研究.文献[3]对动态演化面临的挑战、相关方法和技术框架研究进行了较为全面的论述,将动态演化分为两类情形:特设式(ad-hoc)演化和进化式(evolutionary)演化;将动态演化依据发生情形分为过程编制和过程编排的演化;同时,强调从控制流和数据流两方面相结合对动态演化技术进行研究.上述文献并没有对服务动态演化具体机制、实现方法和技术进行深入的探讨.从控制流和数据流分离的角度提供一种面向网构软件服务特性的动态演化实现机制和方法,是本文的研究动机.

目前,关于工作流动态演化研究中考虑数据依赖关系的主要有德国乌尔姆大学的 ADEPT[11−13]项目,它将变量作为一阶实体(first-class entities)引入到一种有向图模型(称为 WSM Nets)中.WSM Nets 不仅描述了活动、活动之间的控制依赖,还显式地刻画了变量(数据)以及活动对变量的读写操作.WSM Nets 的数据流正确性需要满足以下两条规则:

(1) 任何活动的输入变量必须在使用之前被定义过;

(2) 并发执行的两个活动不能写同一个变量.

这种方法可以避免动态演化数据读写的错误,然而该方法并没有考虑在活动之间的数据依赖关系,因此当动态演化发生时,已执行活动和未执行活动存在数据依赖关系时显得力不从心.

在服务及服务组合实例的可迁移性研究上也逐步关注数据流,文献[8]研究在服务编排层面上,提出当前的 Web 服务组合实例迁移到新模式下的关键,在于求出一个合适的目标状态,并且可以充分利用已执行部分的结果;同时,该文基于控制流提出了目标状态的判定标准.文献[4]关注服务编制层面的 Web 服务组合实例动态演化,将数据依赖关系以活动所定义的变量形式纳入控制流,定义了相应的 Web 服务组合实例可迁移性标准.文献[7]将数据绑定到活动中,通过活动执行系列的向前兼容和向后兼容对服务实例的可迁移性进行判定.文献[11−13]的方法从读写依赖上进行可迁移性判定,如果实例的已执行活动序列对数据的读写依赖能够严格按照既定顺序在目标模式下重现,则判定可迁移.文献[14]在验证工作流等过程感知系统的流程时,将活动的输入/输出变量作为活动的标签加入到活动中,判定活动使用数据时是否存在相应的数据流错误.该方法可用于从数据

流角度判定网构软件服务实例的可迁移性.以上方法初步考虑了活动之间的数据依赖关系,将这种依赖关系作为考察服务组合实例可迁移性的条件之一.然而,这种方法将数据(参数)与活动捆绑得过于紧密,忽略了数据流的特点.文献[25]方法采用数据在活动之间的继承关系,对服务实例的可迁移性进行判定,但该方法并没有从控制流以及控制流/数据流的相互关系角度对服务实例的可迁移性加以判断.

与上述研究工作相比,本文结合网构软件服务特点,在网构软件服务的过程模型上增加了对数据流的刻画,强调了数据依赖关系对动态演化实施的影响.为此,首先将数据流单独提取出来,而不是依附在控制流之上,从过程模型的参数依赖关系和活动间数据依赖关系两个层面对数据流在服务实例迁移中的特性进行研究;然后,综合考虑网构软件服务实例动态迁移过程中控制流和数据流的约束关系,保证可迁移性判定结果的正确性和有效性,从而确保该方法对网构软件服务动态演化的实施更具可行性和适用性.

6 结束语

如何确保网构软件服务实例迁移到新模式下继续运行,并保证其目标状态是有效的,是网构软件服务动态演化研究的难点之一.本文根据网构软件服务特点,基于着色 Petri 网,提出了面向数据流和控制流的 IW_CPN 模型,为研究和分析网构软件服务动态演化提供了模型支持;然后,分析了网构软件服务数据依赖关系,将其细分为参数依赖关系和活动数据依赖关系,全面刻画了网构软件服务数据流的约束关系,将数据流显式地引入网构软件服务动态演化过程中.在此基础上,描述不同演化操作可能出现的数据依赖关系错误和数据流/控制流交叉依赖产生的动态演化错误.为了避免这些错误,本文首先提出面向数据流的网构软件服务实例需要遵循的可迁移性准则,同时设计关于数据依赖关系的生成算法,保证发生动态迁移的实例在数据依赖关系上的一致性;再者,根据 IW_CPN 模型对数据流/控制流的约束关系及其动态运行规则,提出网构软件服务实例关于数据流/控制流依赖关系的动态可迁移性准则,并证明该迁移性准则确保发生演化的服务实例的目标状态是有效的.最后,实验与分析表明,相对于已有的方法,本文的方法对网构软件服务,尤其以数据流为主的网构软件服务动态演化的实施更具可行性和适用性.

我们下一步的工作包括:

1) 探讨面向数据流和控制流的服务实例可迁移性的可操作性和自动化实现;
2) 研究如何高效地检测服务实例的可迁移性,并高效地实现迁移.

致谢 感谢东南大学计算机科学与工程学院汪芸教授和南京大学计算机科学与技术系郑国梁教授富有建设性的建议.感谢审稿专家提出的宝贵意见.

References:

[1] Yang FQ. Thinking on the development of software engineering technology. Ruan Jian Xue Bao/Journal of Software, 2005,16(1): 1−7 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/16/1.htm

[2] 吕建,马晓星,陶先平,徐锋,胡昊.网构软件的研究与进展.中国科学(E 辑),2006,36(10):1037−1080.

[3] Song W, Ma XX, Hu H, Lü J. Dynamic evolution of processes in process-aware information systems. Ruan Jian Xue Bao/Journal of Software, 2011,22(3):417−438 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/3962.htm [doi: 10.3724/SP.J.1001.2011.03962]

[4] Song W, MA XX, Lü J. Instance migration in dynamic evolution of Web service compositions. Chinese Journal of Computers, 2009,32(9):1816−1831 (in Chinese with English abstract).

[5] Zeng J, Sun HL, Liu XD, Deng T, Huai JP. Dynamic evolution mechanism for trustworthy software based on service composition. Ruan Jian Xue Bao/Journal of Software, 2010,21(2):261−276 (in Chinese with English abstract). http://www.jos.org.cn/1000- 9825/3735.htm [doi: 10.3724/SP.J.1001.2010.03735]

[6] 梅宏,黄罡,赵海燕,焦文品.一种以软件体系结构为中心的网构软件开发方法.中国科学(E 辑),2006,36(10):1100−1126.

[7] Ryu SH, Casati F, Skogsrud H, Benatallah B, Saint-Paul R. Supporting the dynamic evolution of Web service protocols in service-oriented architectures. ACM Trans. on the Web, 2008,2(2):1−45. [doi: 10.1145/1346237.1346241]

[8] Song W, Ma XX, Dou WC, Lü J. Toward a model-based approach to dynamic adaptation of composite services. In: Proc. of the IEEE Int'l Conf. on Web Services. 2008. 561−568. [doi: 10.1109/ICWS.2008.65]

[9] Van der Aalst WMP, Basten T. Inheritance of workflows: An approach to tackling problems related to change. Theoretical Computer Science, 2002,270(1-2):125−203. [doi: 10.1016/S0304-3975(00)00321-2]

[10] Rinderle S, Reichert M, Weber B. Relaxed compliance notions in adaptive process management systems. In: Proc. of the Int'l Conf. on Conceptual Modeling. 2008. 232−247. [doi: 10.1007/978-3-540-87877-3_18]

[11] Reichert M, Rinderle-Ma S, Dadam P. Flexibility in process-aware information systems. Trans. on Petri Nets and Other Models of Concurrency, LNCS 5460, 2009,2:115−135. [doi: 10.1007/978-3-642-00899-3_7]

[12] Reichert M, Dadam P. $ADEPT_{flex}$-Supporting dynamic changes of workflows without losing control. Journal of Intelligent Information Systems, 1998,10(2):93−129. [doi: 10.1023/A:1008604709862]

[13] Rinderle S, Reichert M, Dadam P. Flexible support of team processes by adaptive workflow systems. Distributed and Parallel Databases, 2004,16(1):91−116. [doi: 10.1023/B:DAPD.0000026270.78463.77]

[14] Trčka N, van der Aalst WMP, Sidorova N. Data-Flow anti-patterns: Discovering data-flow errors in workflows. In: Proc. of the 21st Int'l Conf. on Advanced Information Systems Engineering. 2009. 425−439. [doi: 10.1007/978-3-642-02144-2_34]

[15] Van der Aalst WMP. The application of Petri nets to workflow management. The Journal of Circuits, Systems and Computers, 1998, 8(1):21−66. [doi: 10.1142/S0218126698000043]

[16] Song W, Ma XX, Cheung SC, Hu H, Lü J. Preserving data flow correctness in process adaptation. In: Proc. of the 7th IEEE Int'l Conf. on Services Computing. 2010. 9−16. [doi: 10.1109/SCC.2010.24]

[17] Zhou H, Huang ZQ, Zhang GQ, Zhu Y, Hu J. Modeling and analysis of internetware based on PTCPN. Ruan Jian Xue Bao/Journal of Software, 2010,21(6):1254−1266 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/3578.htm [doi: 10.3724/SP.J.1001.2010.03578]

[18] Baldan P, Corradini A, Ehrig H, Heckel R. Compositional semantics for open Petri nets based on deterministic processes. Mathematical Structures in Computer Science, 2005,15(1):1−35. [doi: 10.1017/S0960129504004311]

[19] Guo YB, Du YY, Xi JQ. A CP-net model and operation properties for Web service composition. Chinese Journal of Computers, 2006,29(7):1067−1075 (in Chinese with English abstract).

[20] Li JX, Yan CG. Verification mechanism for Web service composition based on extended colored Petri net. Computer Science, 2009, 36(10):146−149 (in Chinese with English abstract).

[21] Yuan CY. Petri Nets Theory and Application. Beijing: Publishing House of Electronics Industry, 2005 (in Chinese).

[22] Jensen K. Colored Petri Nets: Basic Concepts, Analysis Methods and Practical Use: Vol.1, Basic Concepts. 2nd ed. Monographs in Theoretical Computer Science, Springer-Verlag, 1997.

[23] Lam L, Tang Q, Zou ZL, Fong L, Frank D. Identifying data constrained activities for migration planning. In: Proc. of the 7th IEEE Int'l Conf. on Services Computing. Washington: IEEE Computer Society Press, 2009. 364−371. [doi: 10.1109/SCC.2009.33]

[24] König D, Lohmann N, Moser S, Stahl C, Wolf K. Extending the compatibility notion for abstract WS-BPEL processes. In: Proc. of the 17th Int'l Conf. on World Wide Web. 2008. 785−794. [doi: 10.1145/1367497.1367603]

[25] Moser S, Martens A, Gorlach K, Amme W, Godlinski A. Advanced verification of distributed WS-BPEL business processes incorporating CSSA-based data flow analysis. In: Proc. of the IEEE Int'l Conf. on Services Computing. 2007: 98−105. [doi: 10.1109/SCC.2007.22]

[26] Papazoglou MP. The challenges of service evolution. In: Proc. of the Int'l Conf. on Advanced Information Systems Engineering. 2008. 1−15. [doi: 10.1007/978-3-540-69534-9_1]

[27] Andrikopoulos V, Benbernou S, Papazoglou1 MP. Managing the evolution of service specifications. In: Proc. of the Int'l Conf. on Advanced Information Systems Engineering. 2008. 359−374. [doi: 10.1007/978-3-540-69534-9_28]

附中文参考文献:

[1] 杨芙清.软件工程技术发展思索.软件学报,2005,16(1):1−7. http://www.jos.org.cn/1000-9825/16/1.htm

[3] 宋巍,马晓星,胡昊,吕建.过程感知信息系统中过程的动态演化.软件学报,2011,22(3):417−438. http://www.jos.org.cn/1000-9825/3962.htm [doi: 10.3724/SP.J.1001.2011.03962]

[4] 宋巍,马晓星,吕建.Web 服务组合动态演化的实例可迁移性.计算机学报,2009,32(9):1816−1831.

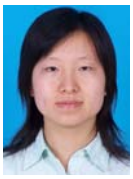
[5] 曾晋,孙海龙,刘旭东,邓婷,怀进鹏.基于服务组合的可信软件动态演化机制.软件学报,2010,21(2):261−276. http://www.jos.org.cn/1000-9825/3735.htm

[17] 周航,黄志球,张广泉,祝义,胡军.基于 PTCPN 的网构软件建模与分析.软件学报,2010,21(6):1254−1266. http://www.jos.org.cn/1000-9825/3578.htm [doi: 10.3724/SP.J.1001.2010.03578]

[19] 郭玉彬,杜玉越,奚建清.Web 服务组合的有色网模型及运算性质.计算机学报,2006,29(7):1067−1075.

[20] 李景霞,闫春钢.一种基于扩展颜色 Petri 网的 Web 服务组合验证机制.计算机科学,2009,36(10):146−149.

[21] 袁崇义.Petri 网原理与应用.北京:电子工业出版,2005.



宋敏(1977－),女,黑龙江哈尔滨人,博士生,主要研究领域为软件体系结构,软件构件.

E-mail: soongmin@163.com

韦正现(1977－),男,高级工程师,主要研究领域为软件体系结构,软件构件.

E-mail: weizhengxian@sina.com

印桂生(1964－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为网构软件,数据库,软件体系结构,虚拟现实,信息安全.

E-mail: yinguisheng@hrbeu.edu.cn