一种用户为中心的即时服务组合方法

张峰[1]，韩燕波[2]，魏永山[1]，陈欣[1]
（1. 山东科技大学信息科学与工程学院，266590，山东青岛；2. 北方工业大学云计算研究中心，100144，北京）

摘要：针对现实中存在的业务需求在问题求解过程中动态变化、逐渐明确的情况，提出了一种以用户为中心的即时服务组合方法. 基于动、静两类业务规则定义服务组合模型，在组合过程中，用户自主选择服务，通过否定性规则提高组合的灵活性，通过肯定性规则降低解空间；提出了基于规则的服务推荐方法和即时服务流程构造方法，为用户推荐无冲突的候选服务，帮助用户快速找到所需服务；定义了服务流程状态，为用户呈现服务流程的状态，实时为用户提示所构造流程的正确性. 通过案例和相关工作比较说明：用户在服务组合过程中处于主导地位，流程实例边构造、边执行；系统通过规则状态判定服务流程状态，为正确构造服务流程提供支持；基于业务规则推荐的服务更准确，更能适应实际业务的需要.

关键词：用户为中心；即时服务组合；业务规则
中图分类号：TP311 **文献标志码**：A **文章编号**：0253-987X(2012)12-0123-08

User-Centric Just-in-Time Service Composition

ZHANG Feng[1], HAN Yanbo[2], WEI Yongshan[1], CHEN Xin[1]
(1. College of Information Science and Engineering, Shandong University of Science and Technology, Qingdao, Shandong 266590, China; 2. Research Center for Cloud Computing, North China University of Technology, Beijing 100144, China)

Abstract: For the situation where business requirements change and become clear step by step during problem-solving process in reality, a user-centric just-in-time service composition strategy is presented. The service composition model is brought forward following the dynamic and static business rules. During the process of composition, users can choose needed services freely. Flexibility of the service composition is enhanced by the negative rules, and the solution space is narrowed by the positive rules. A strategy for services recommendation based on business rules and a just-in-time service process construction are proposed to recommend services without conflict and to help users to find needed services quickly. The definition of service process state is introduced, and the correctness of current service process is clearly shown for users. The cases, compared with related works, indicate that users are of dominance during the service composition process, and the composition can be executed while being constructed. And the system is able to judge the state of the service process by state rules, and to support users to construct correct service process. In addition, the recommended services based on business rules get more accurate to meet real business requirements.

Keywords: user-centric; just-in-time service composition; business rules

随着互联网技术的不断发展和服务计算技术的深入应用，采用服务计算技术实现跨地域、跨机构、

收稿日期：2012-07-03. 作者简介：张峰(1981—)，男，博士生，讲师. 基金项目：山东省自然科学基金资助项目(ZR2011FQ028).
网络出版时间：2012-10-10 网络出版地址：http://www.cnki.net/kcms/detail/61.1069.T.20121010.1708.005.html

跨平台的业务协作成为可能.在面向服务计算范型下,系统的构造和问题求解以服务为基本单位,通过服务组合完成.现行的服务组合技术通常需要事先定义完备的服务组合流程,适用于业务需求明确的问题求解,用户的参与程度有限.在当前诸多领域,如卫生医疗、应急救援等,业务需求可变,问题求解的服务流程无法事先完备定义,需要根据所处环境和需求边构造、边执行,是一个即时构造的过程.

上述类似问题的求解有3个特点:第一,问题求解开始之前无法确定整个过程,需要根据问题求解过程中前面的执行情况、所处环境的即时信息确定下一步的工作;第二,问题求解过程中用户应处于主导地位,用户可以根据具体情况选择下一步调用的服务,服务流程边构造、边执行,而计算机系统处于从属地位,提供服务主动推荐等辅助智能;第三,用户虽然处于主导地位,但还应该遵循领域已有的业务规则,保证服务流程即时构造的正确性.

针对问题求解的服务组合流程无法事先完备定义、需要提高流程模型柔性的问题,一些研究工作使用声明式方法对流程建模,典型代表是Declare[1].Declare通过组合不同的约束模板定义不同的建模语言,如ConDec[2]和基于约束的服务组合语言DecSerFlow[3],采用声明式的建模方法,规定了执行时"不能做什么",所有不违反这些约束的行为都是允许的,从而提高了过程构建的灵活性.但其约束建模只考虑活动之间的时序约束关系,没有考虑实际业务所依赖的其他方面约束,如数据、资源、时间等[2].在诸如应急救援的情景中,数据对活动、服务的约束对服务流程的即时构造十分重要.

在用户主导、边构造、边执行的服务流程构造方面,文献[4-5]提出一种支持最终用户探索式服务组合的方法,以用户可理解的服务形式向最终用户提供可用的服务资源,实现了边执行、边构造的应用构造方式,同时系统为用户提供了辅助的服务推荐功能[6].一些研究在服务组合中引入各类业务规则以保证服务组合的正确性和合理性.文献[7]给出了一种混合式的服务组合方法,将组合逻辑分为核心部分和业务规则部分,在核心的组合逻辑中通过嵌入业务规则实现服务的组合.由于业务规则作为独立的模块管理,因此规则可以被灵活的管理.文献[8-9]定义了多种类型的业务规则,如结构规则、数据规则、约束规则、资源规则等,在服务组合生命周期的各个阶段基于业务规则逐步生成可执行的服务组合流程.

针对上述问题的特点,本文给出了一种以用户为中心的即时服务组合方法.首先,提出了一种基于业务规则的服务组合模型,作为即时服务组合的依据,通过否定性规则提高服务组合的灵活性,通过肯定性规则降低解空间,提高即时服务组合的正确性和合理性.其次,提出了基于业务规则的服务推荐方法和即时服务流程构造方法.根据服务组合模型中的两类规则,系统为用户提供主动的服务推荐,用户自主的选择、执行服务,系统为用户呈现当前服务流程的状态,以提示用户当前的服务流程是否满足业务规则的约束.

1 基于业务规则的服务组合模型

1.1 业务规则

以用户为中心的即时服务组合在为用户提供构造自主性的同时,还应保证构造结果的正确性.业务规则利用若干约束或限制条件规定构成实体的各元素之间的关系,定义、约束了业务过程中的若干方面[10-11].因此,将业务规则整合到服务流程的构建,基于业务规则控制和管理服务组合,可以提高服务组合的正确性.业务规则有多种类别和形式,本文引入两类业务规则.一类是描述服务在应用中使用模式的行为约束规则[12-13],描述服务流程构造过程中禁止的行为,即"不能做什么",是一种否定性规则;第二类规则描述了业务数据与服务之间的约束关系,规定了服务流程构造过程中在特定条件下"需要做什么",是一种肯定性规则[14].

本文否定性规则的定义参考了ConDec[2]中约束模板的定义方式,这些约束可以转化为线性时态逻辑(LTL)表达式去执行,具体内容见文献[2].

本文肯定性规则规定了业务数据与服务执行之间的约束关系.相关的业务数据能够反应业务过程整体业务指标[15],不一定都是流程中活动的输入、输出数据,也可以是流程运行环境中用户所关心的数据,如火灾救援中,用户决策所需要的火势大小、风力等相关数据.肯定性规则包括两类:一类由业务数据的状态描述服务执行的前提和结果;另一类描述业务数据处于特定状态时要执行哪些服务,采用"If Then"的描述形式,表示在"If"指定的条件成立时,需要执行的服务.

1.2 服务组合模型

即时服务组合需要为用户提供自主构造服务流程的能力,同时需要将业务规则整合到服务流程构造中.本文将业务数据、服务、业务规则作为即时服

务组合的基础,形成了基于业务规则的服务组合模型(BRBSCM),记为 M.

定义 1　基于规则的服务组合模型 $M=(S,D,C_d,C_a)$,其中:$S=\{s_1,s_2,\cdots,s_n\}$ 是服务的集合;$D=\{d_1,d_2,\cdots,d_n\}$ 是业务数据集合;$C_d=\{C_{d\text{-}pre},C_{d\text{-}post},C_{d\text{-}if}\}$ 是数据服务规则集合,描述业务数据与服务之间的约束关系,$C_{d\text{-}pre}$ 是描述服务执行前提的规则,对任意 $c_{d\text{-}pre}\in C_{d\text{-}pre}$,$c_{d\text{-}pre}=(s_i,c_i)$,$s_i\in S$,$c_i$ 是服务 s_i 执行的前提表达式,$C_{d\text{-}post}$ 是描述服务执行结果的规则,对任意 $c_{d\text{-}post}\in C_{d\text{-}post}$,$c_{d\text{-}post}=(s_j,c_j)$,$s_j\in S$,$c_j$ 是服务 s_j 执行的结果表达式,$C_{d\text{-}if}$ 是描述数据满足特定条件时需要执行哪些服务的规则,对任意 $c_{d\text{-}if}\in C_{d\text{-}if}$,$c_{d\text{-}if}=(c_m,S_m)$,$c_m$ 是业务数据状态构成的条件表达式,$S_m\subseteq S$ 为条件 c_m 成立时要执行的服务的集合;$C_a=(C_m,C_o)$ 是服务行为约束规则集合,C_m 为强制约束,C_o 是可选约束[2].

在定义 1 中,任意一个业务数据 $d_i\in D$ 可以有自己的子属性. 例如,在高层建筑火灾救援中,数据"着火建筑"可以有其子属性"建筑高度"、"材料结构"等子属性. C_m 为服务流程构造过程中必须满足的行为约束,C_o 为非强制约束,系统可以根据这些约束给出提示[2]. 从图 1 模型可看到,业务数据和服务之间的约束构成了数据-服务规则,服务之间的约束关系构成服务行为约束规则.

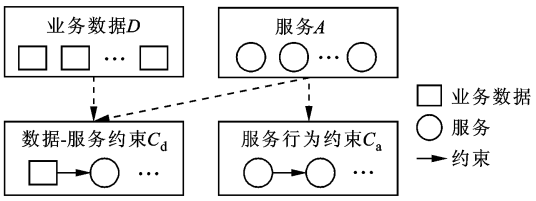

图 1　基于业务规则的服务组合模型

2　用户为中心的即时服务组合方法

即时服务组合方法是在 BRBSCM 模型的业务规则约束下,由用户自主构造服务流程,实现问题的求解. 在服务流程的构造过程中,系统为用户提供了两方面的支持:①系统根据已有规则和业务数据的即时信息,主动为用户推荐可执行的服务;②在任意时刻,根据服务流程执行情况,服务组合模型中的业务规则呈现不同的状态(满足或不满足规则)[2],从而可以判断已有服务流程是否满足规则约束,确定服务流程的状态. 系统可以为用户提示服务流程的状态,使用户了解整个服务流程构造的正确性.

2.1　即时服务组合中的服务推荐

在用户构造服务流程的过程中,根据业务数据的即时信息、服务流程执行情况,结合模型中的业务规则,系统可以为用户主动推荐下一步可调用的服务. 服务推荐需要解决两个关键问题.

第一,根据业务数据的即时信息,可能有多个肯定性规则的条件成立,因此可以推荐多个服务. 但是,由于肯定性规则中包括了描述服务调用前提和结果的规则,这些被推荐的服务中,有些服务的执行前提不成立,从而导致冲突;有些服务的执行可能导致其他服务的执行前提不成立,导致了服务之间的冲突;有些服务的执行会使其他服务的前提成立,产生了服务之间的使能关系. 因此,推荐的要求是产生一个服务推荐列表,按列表中的顺序无冲突地调用服务.

第二,上述根据肯定性规则推荐的服务,如果被执行,可能还会违反否定性规则,因此还需要结合否定性规则过滤服务推荐列表.

需要说明的是,本文提出的服务推荐方法仅考虑上述两个问题中的冲突解决问题,没有讨论肯定性规则集合本身的冲突验证[16]等问题. 否定性规则的冲突、合理性等问题,可以基于线性时态逻辑(LTL)转化为自动机来验证[2].

定义 2　两个服务 a、b,如果服务 a 的执行导致服务 b 的执行前提不成立,则称服务 a 冲突于服务 b,记作 conflict(a,b).

定义 3　两个服务 a、b,如果服务 a 的执行结果使得服务 b 的执行前提成立,则称服务 a 使能服务 b,记作 enable(a,b).

服务推荐算法需要为用户推荐一个服务的有序列表,在业务数据状态仅通过服务调用发生改变的前提下,如果用户按照列表中的顺序依次调用服务,这些服务可以无冲突地被执行. 因为涉及到肯定性和否定性两种业务规则,推荐算法实现的基本思想是:根据肯定性规则,确定可推荐服务;在将可推荐服务加入到推荐列表之前,首先根据否定性规则判断该服务是否可加入到推荐列表,如果可以,则将该服务加入到推荐列表,否则不能加入推荐列表. 算法中调用了方法 boolean ifAddedToList(serviceProcess,serviceList,service),用于判断当前的服务流程 serviceProcess 在已有服务推荐列表为 serviceList 的前提下,服务 service 能够加入到推荐列表中,如果可以,方法返回 true,否则返回 false.

下面给出算法的实现步骤.

算法 1:服务推荐列表生成算法 recommend-

Services.

输入:M模型,业务数据即时信息,当前服务流程 serviceProcess.

输出:当前可推荐的服务列表.

第1步:根据业务数据的状态和M模型中肯定性规则$C_{\text{d-pre}}$和$C_{\text{d-if}}$,得到所有条件成立的$C_{\text{d-if}}$规则所需执行且前提成立的服务,放入集合S_1,条件成立的$C_{\text{d-if}}$规则所需执行但前提不成立的服务放入集合S_{dis}.

第2步:由S_1得到S_1中服务对应的服务冲突有向图G_{c}.

根据M模型中的$C_{\text{d-pre}}$和$C_{\text{d-post}}$,检测S_1中服务之间的冲突关系,产生一个描述S_1中服务之间冲突关系的有向图$G_{\text{c}}\langle V,E\rangle$.$G_{\text{c}}$中的顶点与$S_1$中的服务一一对应.如果服务$a,b\in S_1$,且conflict$(a,b)$,则在$G_{\text{c}}$中$a$、$b$对应的顶点$V_a$、$V_b$之间添加有向边$e\langle a,b\rangle$.检测$S_1$中所有服务的执行前提和执行结果,形成对应$S_1$的服务冲突有向图$G_{\text{c}}$.

第3步:根据G_{c}过滤S_1中的服务,得到可推荐服务有序列表S_2和冲突服务集合S_{conf},其中S_2是可推荐服务有序列表,S_{conf}是与S_2中服务有冲突的服务集合.

由于服务之间存在冲突关系,S_1中的服务未必都可以推荐.选择可放入S_2中的服务的基本思想是:优先选择与其它服务冲突小的服务,这些服务放到S_2中;与S_2中服务有冲突关系的服务放入到S_{conf}中,以进行下一步的处理.该步骤的算法如下.

输入:服务冲突有向图$G_{\text{c}}\langle V,E\rangle$.

输出:可推荐服务列表S_2和冲突服务集合S_{conf}.

算法伪代码:

1)$S_2=\{\}$

2)While($|V|>0$){

3)从G_{c}中选择出度最小的顶点V_i;//若最小出度顶点存在多个,则随机选一个

4)if(ifAddedToList(serviceProcess,S_2,s_i)){

add(S_2,s_i)//将顶点V_i对应的服务s_i加入S_2

}

5)从G_{c}中删除顶点V_i,删除所有的边$e\langle V_i,V_j\rangle$,删除顶点V_j,add(S_{conf},s_j),s_j是V_j对应的服务;

}

6)return S_2,S_{conf}.

第4步:根据S_2、S_{conf}和M中的规则$C_{\text{d-pre}}$、$C_{\text{d-post}}$,将S_{conf}内可以被S_2中服务使能的服务加入到S_2中.

根据上一步得到的列表S_2中的服务是没有任何冲突的,而由于服务之间的使能关系,S_{conf}中的部分服务也可以加入到S_2中,从而扩大可推荐服务的范围.将S_{conf}服务加入到S_2中的基本思想是:假设$S_2=\langle s_1,s_2\cdots s_n\rangle$,$s\in S_{\text{conf}}$,conflict$(s_i,s)$且$s_i\in S_2$.如果存在$s_x\in S_2$,$i<x\leqslant n$且enable$(s_x,s)$,则将$s$添加到$S_2$的尾部,并从$S_{\text{conf}}$中删除$s$.即在$S_2$中,$s_x$在$s_i$之后,而$s_x$的执行导致$s$的前提成立,从而$s$也可以执行,$s$可以加入到$S_2$中.该步骤算法如下.

输入:S_2、S_{conf}、$G_{\text{c}}\langle V,E\rangle$.

输出:更新后的推荐服务列表S_2.

算法伪代码:

for($i=1$; $i<=|S_{\text{conf}}|$; $i++$){

s_i为S_{conf}中第i的服务;

$S_{ci}=\text{conf}(s_i,G_{\text{c}})$;//从$G_{\text{c}}$中得到与$S_i$冲突的所有的服务构成的集合$S_{ci}$

$s_x=\text{last}(S_2,S_{ci})$;//在$S_2$中遍历,找到最后一个出现在$S_{ci}$中的服务$s_x$

在S_2中,从s_x开始往后遍历,

if($\exists s\ \wedge$ enable(s,s_i)){

if(ifAddedToList(serviceProcess,S_2,s_i))

{

add(S_2,s_i);//将s_i添加到S_2的末尾

}

delete(S_{conf},s_i);//将s_i从S_{conf}中删除

$i=1$;//S_{conf}中服务变化后,从头开始遍历S_{conf}中的服务

}

}

return S_2

第5步:根据S_2、S_{dis}和肯定性规则$C_{\text{d-pre}}$、$C_{\text{d-post}}$,将S_{dis}中可以被S_2中服务使能的服务加入到S_2中.该步骤的实现思想是:对S_{dis}中的服务s以及S_2中服务按照顺序执行后,如果使得s的执行前提成立,则s加入到S_2中,该算法步骤如下.

输入:S_2、S_{dis}.

输出:更新后的推荐服务列表S_2.

算法伪代码:

while($\exists s\in S_2\ \wedge$ s使能S_{dis}中的服务){

s_i为S_{dis}中可被S_2中服务使能的一个服务;

$s_x=\text{last}(S_2,s_i)$;//在S_2中遍历,找到最后一个与s_i冲突的服务s_x

在 S_2 中,从 s_x(s_x 为空则从 S_2 第一个服务)开始往后遍历,

```
if(∃s ∧ enable(s,s_i)){
if(if AddedToList(serviceProcess,S_2,s_i))
{
add(S_2,s_i);//将 s_i 添加到 S_2 中
}
delete(S_dis,s_i)//将 s_i 从 S_dis中删除
}
}
return S_2.
```

服务列表 S_2 为当前时刻的服务推荐列表.

2.2　业务规则状态和服务流程状态

在服务流程构造过程中,根据用户构造的服务流程和业务数据的即时信息,可以判断业务规则的状态.服务组合模型中的服务行为约束采用了文献[2]中约束的定义:满足、暂时违反、违反.

在服务流程构造过程中,业务数据的状态一般会随着环境的变化而变化.例如,火灾救援中,风向是一个业务数据,风向的变化会导致由风向而得到的业务规则的状态的变化.因此,数据-服务约束的状态随着数据的状态和服务调用的变化而变化.在服务流程构造过程中,数据-服务规则的状态包括:满足、暂时违反.其中,"满足"表示当前服务执行满足了业务规则;"暂时违反"表示当前服务执行不满足业务规则,但随着业务数据的状态变化以及服务的执行,业务规则可能变为"满足".

定义 4　数据-服务规则状态.假设 $r_i \in C_d$ 为数据-服务规则,p_i 为一服务流程,$S(p_i,r_i)$ 表示 p_i 中 r_i 的状态,$S(p_i,r_i)$ 定义为

$$S(p_i,r_i)=$$

满足　若[$(r_i \in C_{d\text{-if}}) \wedge (r_i.c_m=\text{true} \wedge r_i.S_m$ 启动$) \vee r_i.c_m=\text{false}$]
　　若$(r_i \in C_{d\text{-pre}}) \wedge$ [$(r_i.s_i$ 启动$\wedge r_i.c_i=\text{true}) \vee r_i.s_i$ 未启动]
　　若$(r_i \in C_{d\text{-post}}) \wedge$ [$(r_i.s_j$ 完成$\wedge r_i.c_j=\text{true}) \vee r_j.s_j$ 未完成]

暂时违反　若$(r_i \in C_{d\text{-if}}) \wedge$[$r_i.c_m=\text{true} \wedge (s \in r_i.S_m, s$ 未启动)]
　　若$(r_i \in C_{d\text{-pre}}) \wedge (r_i.s_i$ 启动$\wedge r_i.c_i=\text{false})$

违反　若$(r_i \in C_{d\text{-post}}) \wedge (r_i.s_j$ 完成$\wedge r_i.c_j=\text{false})$

服务流程的状态随着业务数据和服务执行情况的变化而变化.服务流程的状态包括:满足、暂时违反、违反,其定义与文献[2]中的流程实例状态一致.

定义 5　服务流程 p_i 对应的服务组合模型中的数据-服务规则集合为 C_d,服务之间的行为约束规则集合为 C_a,$S(p_i,c)$ 表示 p_i 中约束 c 的状态,则 p_i 的状态 $S(p_i)$ 为

$$S(p_i)=\begin{cases}\text{满足} & (\forall r_i \in C_d)(S(p_i,r_i)=\text{满足}) \wedge (\forall a_j \in C_a)(S(p_i,a_j)=\text{满足}) \\ \text{违反} & (\exists a_i \in C_a)(S(p_i,a_i)=\text{违反}) \\ \text{暂时违反} & \text{其他}\end{cases}$$

根据定义 5 可以看到,服务是否处于"违反"状态只是根据服务之间的行为约束规则来判定,原因是数据-服务规则中业务数据可能随着环境、时间的变化而变化,规则不会永远是"违反"状态.同时,只有在所有的服务行为约束规则和数据-服务规则状态都是"满足"时,服务流程的状态才为"满足".

采用即时服务组合方法,只有在所构造的服务流程状态为"满足"时才能够成功的结束[2],此时系统可以根据业务规则的状态提示用户,便于用户在构造过程中及时发现问题,提高构造的正确性.

2.3　服务流程即时构造过程

服务流程即时构造的过程如图 2 所示.在初始阶段,服务流程实例为空,在构造阶段,用户可以选择服务执行.根据业务数据的即时信息和服务组合模型中的业务规则,系统形成服务推荐列表;用户可以自主选择服务,每次服务的执行会影响当前服务流程的状态,服务推荐列表会即时更新.同时,系统根据服务组合模型中业务规则的状态判断服务流程的状态,为用户提示相应信息.最后,在服务流程状态为"满足"的前提下,用户可以自主结束服务流程的构造,完成问题的求解.

3　案例与评价

3.1　场景及服务组合模型

某日,位于郊区的一化工厂由于一个乙烯球罐底部液相管发生泄漏,遇静电突发粉碎性爆炸而引发火灾.指挥部门立刻启动《XX 化工厂火灾应急预案》,预案中主要包括了爆炸事件的相关数据、救援相关部门能够提供的服务、救援相关的业务规则.为

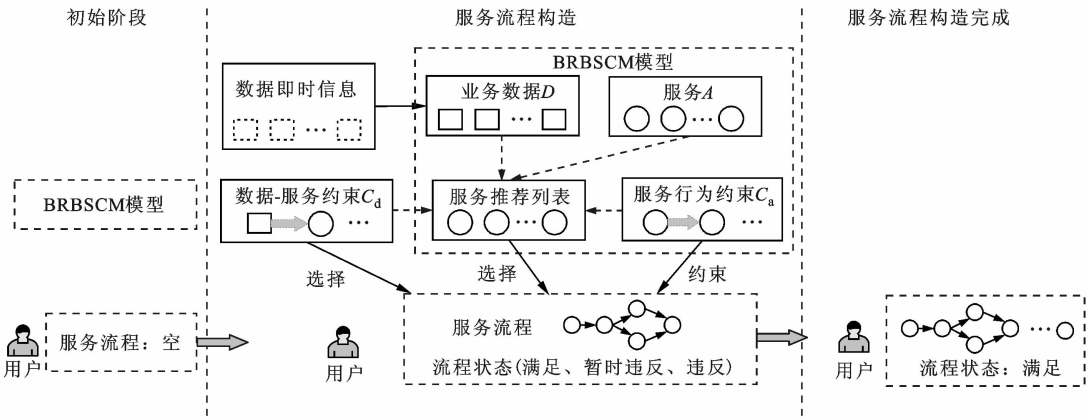

图2 服务流程即时构造过程

了简单起见，本文引用部分业务规则，对应的服务组合模型如下.

(1)服务集合 A. 预案中涉及到的计算机系统及人员提供的服务有：①关闭油罐阀门；②调集洒水车拉水；③节水方式灭火；④用水泵抽水；⑤大水量方式灭火；⑥抢修供电系统；⑦确定可能发生爆炸的油罐；⑧下达撤离命令；⑨检查管道阀门；⑩阻隔油罐阀门.

需要说明的是，本案例中的服务包括计算机系统提供的软件类服务和由人提供的人工服务. 这些服务经过虚拟化后，形成可以由业务用户可理解、可直接使用的业务级别的服务[8].

(2)业务数据集合 D. 主要的业务数据包括：①现场水泵状态；②现场水供应量；③油罐状态，子属性包括是否颤抖、安全阀是否鸣响、火焰颜色、阀门状态、油罐温度、与其他油罐是否连通.

(3)服务之间的行为约束 C_a. 服务之间的行为约束主要包括：①确定可能发生爆炸的油罐是起始活动，即 init[2](确定可能发生爆炸的油罐)；②关闭油罐阀门和阻隔油罐阀门不能同时发生，即 not co-existence[2](关闭油罐阀门，阻隔油罐阀门).

(4)数据-服务规则 C_d. 数据-服务规则的 $C_{d\text{-}pre}$、$C_{d\text{-}post}$、$C_{d\text{-}if}$ 三类规则，分别如表1～表3所示.

表1 $C_{d\text{-}pre}$ 规则

序号	服务	执行前提
1	大水量方式灭火	“现场水供应量”=充足
2	关闭油罐阀门	“油罐温度”=较低
3	用水泵抽水	“现场水泵状态”=可用

表2 $C_{d\text{-}post}$ 规则

序号	服务	执行结果
1	抢修供电系统	“现场水泵状态”=可用
2	用水泵抽水	“现场水供应量”=充足
3	大水量灭火	“油罐温度”=较低
4	节水方式灭火	“油罐温度”=过高
5	关闭油罐阀门	“油罐状态：与其他油罐是否连通”=否

表3 $C_{d\text{-}if}$ 规则

序号	业务数据状态	服务
1	“油罐状态：与其他油罐是否连通”=是	关闭油罐阀门
2	“油罐状态：是否颤动”=true OR “油罐状态：安全阀是否鸣响”=true OR “油罐状态：火焰颜色”=白色	下达撤离命令
3	“现场水供应量”=不充足	调集洒水车拉水
4	“现场水供应量”=不充足	节水方式灭火
5	“现场水供应量”=不充足	用水泵抽水
6	“油罐温度”=过高	大水量方式灭火
7	“现场水泵状态”=不可用	抢修供电系统

3.2 服务流程构造过程实例

本节以一个服务流程 pi_1 的构造过程为例，说明本文的即时服务组合方法的构造过程. 表4描述 pi_1 按照时间顺序(t_1、t_2、t_3、t_4)在不同时刻的状态，包括每个时刻的业务数据状态、服务推荐列表、当前时刻用户执行的服务、服务流程状态信息.

初始时刻 t_1，由于服务行为约束 C_a 中的规则1，推荐服务7，用户执行了服务7. 根据 $C_{d\text{-}if}$ 规则中的3、4、5，由于未执行对应的服务，此时服务流程的状态为“暂时违反”.

以 t_2 时刻为例说明服务推荐算法 recommend-

Services 的工作过程. 根据业务数据、模型中的 $C_{\text{d-if}}$ 规则,此时 $C_{\text{d-if}}$ 中的规则 1、3、4、5、6、7 的条件成立,但同时由于该时刻不满足 $C_{\text{d-pre}}$ 中的规则 1、2、3,服务 1、4、5 此时无法执行. 所以,$S_1=\{2,3,6\}$,$S_{\text{dis}}=\{1,4,5\}$;由 S_1 及其冲突关系,$S_2=\langle 2,3,6\rangle$,$S_{\text{conf}}=\{\}$;然后根据 S_2、S_{dis}、肯定性规则 $C_{\text{d-pre}}$ 和 $C_{\text{d-post}}$,得到最终的服务推荐列表 $S_2=\langle 2,3,6,4,5,1\rangle$. 此时用户按照 S_2 中推荐的服务执行,可以保证服务流程的正确性. 假设 t_2 时刻用户此时依次执行了服务 2、3、6. 由于 1、4、5 没有执行,所以 $C_{\text{d-if}}$ 中的规则未被完全满足,此时服务流程处于"暂时违反"状态.

在 t_3 时刻,根据业务数据的状态,由 recommendServices 算法系统推荐的服务列表 $S_2=\langle 2,3,4,5,1\rangle$. 用户可以不选择服务 2,直接选择依次执行服务 3、4、5、1,每执行一个服务,系统会判断规则的状态,更新服务推荐列表和服务流程的状态,并给出相应的提示信息. 此处由于篇幅原因,没有给出执行每个服务后的服务推荐列表.

最后,在 t_4 时刻,根据业务数据的状态,此时服务推荐列表为空,用户可以自主地选择服务去执行,如执行服务 5,系统自动判断规则和服务流程的状态,此时服务流程为"满足"状态. 服务流程处于该状态时,用户可以结束流程的构造.

表 4　服务流程构造过程实例

pi_1	业务数据状态	服务推荐列表	执行服务	服务流程状态
t_1	(1)"现场水供应量"=不充足	〈7〉	7	暂时违反
t_2	(1)"现场水供应量"=不充足 (2)"油罐状态:与其他油罐是否连通"=是 (3)"油罐温度"=过高	〈2,3,6,4,5,1〉	2,3,6	暂时违反
t_3	(1)"现场水供应量"=不充足 (2)"油罐状态:与其他油罐是否连通"=是 (3)"油罐温度"=过高 (4)"现场水泵状态"=可用	〈2,3,4,5,1〉	3,4,5,1	暂时违反
t_4	(1)"现场水供应量"=充足 (2)"油罐温度"=较低 (3)"油罐状态:与其他油罐是否连通"=否 (4)"现场水泵状态"=可用	〈〉	5	满足

3.3　分析与比较

第一,在构造开始之前,M 模型没有事先完整定义的服务流程. 在 pi_1 的构造过程中,用户可以自主地选择服务. 例如,在 t_1 时刻,用户选择执行服务 7,t_4 时刻选择执行服务 5,在 t_3 时刻系统推荐的服务为〈2,3,4,5,1〉,用户也可以跳过服务 2 直接执行其他服务. 因此,服务流程实例是边构造、边执行的,用户在构造过程中处于主导地位.

第二,由于引入了业务规则,根据业务数据的状态,系统可即时为用户提供服务推荐功能,同时系统即时提示服务流程当前的状态. 从 pi_1 的构造过程可以看到,只要否定性规则不被违反,服务流程就不会处于"违反"状态. 肯定性规则由于与业务数据相关,而业务数据的信息是即时变化的,因此肯定性规则如果处于"违反"状态,会导致服务流程处于"暂时违反"状态,不会处于"违反"状态,例如,t_1、t_2、t_3 时刻服务流程的状态均为"暂时违反". 本文方法通过规则状态判定服务流程状态,为用户正确构造服务流程提供了支持.

与基于约束的服务组合语言 DecSerFlow[3] 相比,本文提出的服务组合模型中不仅包括了服务之间的行为约束,还添加了与数据关联的肯定性规则,在执行过程的每一步,由于肯定性规则的引入,解空间(所有可执行服务构成的集合)可以降低,推荐的服务也更加准确. 表 5 给出了在流程实例 pi_1 的构造过程中,基于 M 模型的即时服务组合方法与基于约束的 DecSerFlow 模型在每个时刻可执行(推荐)的服务的对比.

表 5　本文方法和文献[3]方法的服务推荐比较

pi_1	本文方法	文献[3]方法	执行服务
t_1	〈7〉	〈7〉	7
t_2	〈2,3,6,4,5,1〉	〈1,2,3,4,5,6,8,9〉	2,3,6
t_3	〈2,3,4,5,1〉	〈1,2,3,4,5,6,8,9〉	3,4,5,1
t_4	〈〉	〈1,2,3,4,5,6,8,9〉	5

由表 5 可以看到,在流程构造过程的 4 个时刻,由于 M 模型中引入了与数据相关的肯定性规则,根据本文提出的服务推荐算法,其推荐的服务个数都少于或等于文献[3]推荐的服务个数. 因此,推荐的服务更加准确,更能适应实际业务的需要.

4　总　结

针对面向服务计算环境中用户需求不明确的问

题,本文提出了一种以用户为中心的即时服务组合方法,目的是找到服务组合中"静"与"动"的平衡,其中,"静"表现为服务要在领域固有的业务规则的约束下组合,而"动"表现为需求在构造过程中逐渐明确、用户在服务组合过程中应有主动性.本文的主要贡献有:①提出了一种基于业务规则的服务组合模型,将数据、服务及其之间的约束规则作为服务组合的依据,在即时构造过程中,既保证服务流程构造的灵活性和用户的主动性,也保证了构造的正确性和灵活性;②从系统支撑层面,给出了一种基于业务数据和规则的服务推荐算法,为用户推荐无冲突的候选服务,帮助用户快速有效地找到所需服务,同时给出了服务流程状态定义,在即时构造过程中可以实时为用户提示所构造流程的正确性和合理性,辅助用户的流程构造.

参考文献:

[1] PESIC M, SCHONENBERG H, VAN DER AALST W M P. Declare: full support for loosely-structured processes [C]// Proceedings of 11th IEEE International Enterprise Distributed Object Computing Conference. Annapolis, Maryland, USA: IEEE, 2007: 287-298.

[2] PESIC M. Constraint-based workflow management systems: shifting control to users [D]. Eindhoven, Holland: Technische University Eindhoven, 2008.

[3] VAN DER AALST W M P, PESIC M. DecSerFlow: towards a truly declarative service flow language [C]// International Conference on Web Services and Formal Methods. Berlin, Germany: Springer-Verlag, 2006: 1-23.

[4] 韩燕波,王洪翠,王建武,等. 一种支持最终用户探索式组合服务的方法 [J]. 计算机研究与发展,2006,43(11):1895-1903.
HAN Yanbo, WANG Hongcui, WANG Jianwu, et al. An end-user-oriented approach to exploratory service composition [J]. Journal of Computer Research and Development, 2006,43(11): 1895-1903.

[5] 闫淑英. 最终用户参与的探索式服务编排关键技术研究 [D]. 北京:中国科学院研究生院, 2009.

[6] YAN Shuying, HAN Yanbo, WANG Jing, et al. Service hyperlink for exploratory service composition [C]// Proceedings of the 3rd IEEE International Workshop on Service-Oriented System Engineering. Piscataway, NJ, USA: IEEE, 2007: 581-588.

[7] MASUHARA H, KICZALES G. Modeling crosscutting in aspect oriented mechanisms [C]// Proceedings of European Conference on Object-Oriented Programming. Darmstadt, Germany: Springer-Verlag, 2003: 2-28.

[8] ORRIES B, YANG J, PAPAZOGLOU M P. A framework for business rule driven service composition [C]// Proceedings of the Fourth International Workshop on Conceptual Modeling Approaches for e-Business Dealing with Business Volatility. Berlin, Germany: Springer-Verlag, 2003: 14-27.

[9] ORRIES B, YANG J, PAPAZOGLOU M. A framework for business rule driven web service composition [M]// Lecture Notes in Computer Science: Vol 2814. Berlin, Germany: Springer-Verlag, 2003: 52-64.

[10] The Business Rules Group. Defining business rules, what are they really? [EB/OL]. [2012-06-22]. http://www.businessrulesgroup.org.

[11] VON HALLE B. Business rules applied: building better systems using the business rules approach [M]. New York, USA: John Wiley & Sons, Inc, 2001: 1-30.

[12] 丁维龙,王菁,赵栓. 一种用户为中心、基于多视图合成的服务组合方法 [J]. 计算机学报,2011,34(1):131-142.
DING Weilong, WANG Jing, ZHAO Shuan. A user-centric service composition method synthesizing multiple views [J]. Chinese Journal of Computers, 2011, 34(1): 131-142.

[13] 韩燕波,王桂玲,刘晨,等. 互联网计算的原理与实践:探索网格、云和 Web X.0 背后的本质问题和关键技术 [M]. 北京:科学出版社,2010:221-226.

[14] WAINER J, DE LIMA BEZERRA F. Constraint-based flexible workflow [C]// Proceedings of the 9th International Workshop on Groupware: Design, Implementation, and Use. Berlin, Germany: Springer-Verlag, 2003: 151-158.

[15] VAN DER AALST W M P, DESEL J, OBERWEIS A. Business process management: models, techniques, and empirical studies [M]. Berlin, Germany: Springer-Verlag, 2000: 23-54.

[16] 姜跃平,汪卫,施伯乐,等. ECA 规则的模型和行为特定理论 [J]. 软件学报,1997,8(3):191-196.
JIANG Yueping, WANG Wei, SHI Bole, et al. Model and behavioral determinism theory for ECA rules [J]. Journal of Software, 1997,8(3): 191-196.

(编辑 杜秀杰 赵大良)