WS-CDL的形式化模型和执行过程研究

梁智远 张为群 黄 娟

(西南大学计算机与信息科学学院 重庆 400715)

摘 要 WS-CDL以全局的视角描述了基于交互的各个服务之间的组合方式,定义了公共和互补的可观察行为,以及具有一致见解的顺序规则。但是WS-CDL没有形式化的模型,也没有定义执行过程。提出了一种为WS-CDL建立全局模型的方法,定义了语法和语义,然后将全局模型映射到局部模型,并且定义了全局执行过程,最后讨论了一致性问题。

关键词 Web服务编排描述语言,Web服务组合,一致性

中图法分类号 TP311 **文献标识码** A

Research on Formal Model and Implementation for WS-CDL

LIANG Zhi-yuan ZHANG Wei-qun HUANG Juan

(College of Computer and Information Science, Southwest University, Chongqing 400715, China)

Abstract WS-CDL describes the composition between interacting services from a global viewpoint, and specifies the common and complementary observable behavior, and the jointly agreed ordering rules. But WS-CDL lacks for formal model and implementation. This paper proposed a global model of WS-CDL, and defined its syntax and semantic. Then we map the global model to local one, and defined the global implementation. At last we discussed the conformance problem.

Keywords Web service choreography description language, Web service composition, Conformance

1 引言

随着因特网的不断发展,功能单一的Web服务已经不能胜任企业的应用。于是,Web服务组合成为了大家所关注的焦点。关于Web服务组合的方法有两种不同的观点,分别叫编制(Orchestration)和编排(Choreography)。前者站在局部的视角,以单个参与方定义服务交互;后者则是以全局的视角描述参与方之间的点对点协作。现在,关于编制的规范是业务流程执行语言(Business Process Execution Language, BPEL),关于编排的规范是Web服务编排描述语言(Web Service Choreography Description Language, WS-CDL)。

随着企业应用越来越庞大,在建立Web服务组合时仅以局部视角进行系统分析和设计难免出现纰漏。WS-CDL所描述的组合方式站在更高的视角,统筹全局,更适应复杂系统。但WS-CDL仅仅是一种声明性的语言,没有可理解的形式化基础,没有将元模型和语法分离[1]。另外,WS-CDL没有具体可执行的过程,要执行编排还须把模型从全局视角转换成局部视角。因此,将WS-CDL模型化,在其上建立推导规则,探讨编排的执行,保证执行过程能符合WS-CDL描述等一系列工作成为了现在研究的热点。

Decker等人[2]建立了一个图形化的小语言Let's Dance作为编排的元模型,用π演算描述语义。Foster等人[3]用基于模型的方法对组合进行验证,并且将WS-CDL描述转化为FSP模型。Busi等人[4]建立了模型CL,着重讨论了请求与应答结构。Qiu等人[5-7]建立了全局和局部模型,通过手工推导或SPIN自动推导得出一些性质,并通过引入轨迹语义来探讨一致性。本文对简化了的WS-CDL规范进行建模,展示了编排的核心思想,建立了全局和局部的语法规则和映射过程,用带条件的轨迹表示语义,最后根据语义来讨论执行过程。

2 WS-CDL概述

WS-CDL是一种基于XML的语言,从全局的角度描述各个参与方的可见的公共或互补行为,通过具有特定顺序的消息交换完成某一公共的业务目标,其核心内容是消息和活动。

WS-CDL规范的主要元素可以分为三类(本文不考虑异常情况和终结处理)。1)基于参与方的元素:其中角色类型〈roleType〉代表了一组可观察行为的逻辑集合;关系类型〈relationshipType〉代表为达到交互,两个角色类型之间形成的关系;参与方类型〈participantType〉代表了一组角色类型的集合,比如一个参与方可以充当多个角色;通道类型〈channelType〉定义了交互发生的场所和方式。2)基于信息的元素:其中信息类型〈informationType〉定义了各种数据类型;〈variable〉定义了各角色拥有的变量。3)基于活动的元素:描述了

到稿日期:2008-11-10 本文受到重庆市自然科学基金重点项目"软件测试技术和方法研究"(CSTC,2006BA2003)支持。

梁智远(1983—),男,硕士研究生,主要研究方向为软件工程、形式语言,E-mail:alexleung1107@163.com;张为群(1950—),男,教授,硕士生导师,CCF会员,主要研究方向为形式语言、软件工程、软件测试;黄 娟(1983—),女,硕士研究生,主要研究方向为软件工程、Web软件测试。

各种控制结构，其中〈choreography〉表示一个完整的编排体；〈sequence〉、〈parallel〉和〈choice〉分别表示顺序、并行和选择活动；〈workunit〉表示工作单元；〈interaction〉表示交互活动，即两个参与方之间的信息传递；〈assign〉是在一个参与方内部的赋值活动。

3 形式化 WS-CDL

3.1 WS-CDL 模型的语法

正如上面提到的，一个编排包含了若干角色，每个角色上定义了若干操作和本地变量，另外交互活动所发生的场所——通道也是关联在角色上的，所以本文用角色来对所有声明进行抽象建模。假设 RN 是角色名字的集合，Op 是操作的集合，Var 是变量的集合，$Chan$ 是通道的集合，那么角色集 $R=\{(r,O,V,Ch)\mid r\in RN, O\in\wp(Op), V\in\wp(Var), Ch\in\wp(Chan)O_i\}$ 符号 $\wp(S)$ 表示集合 S 的幂集。对于 R 中的某一个角色 R_i，有 $R_i=(r_i,O_i,V_i,Ch_i)\in R$，其中 r_i、O_i、V_i 和 Ch_i 分别表示 R_i 的名字、在 R_i 上定义的操作集合、变量集合和通道集合（用 O_i、v_i 和 c_i 表示 R_i 上具体的某一个操作、变量和通道）。

WS-CDL 中的活动分为两类：原子活动和复合活动。任一活动 A 由下列文法定义：

$A::= Atom \mid g*A*p \mid A;A \mid A\sqcap A \mid g1:A[]g2:A \mid A||A$

$Atom::=\varepsilon \mid a_i \mid c[i,j]$

其中 $Atom$ 表示原子活动，g、p、$g1$ 和 $g2$ 都是布尔表达式名（在建模过程中，需要为每一个布尔表达式声明一个独立的名字），在 WS-CDL 中，g、$g1$ 和 $g2$ 又称作卫士条件（guard condition），p 又称作循环条件（repetition condition）。原子活动包括：(i)ε 表示空活动或与编排无关的哑活动；(ii)a_i 表示在角色 R_i 上的本地活动（如赋值活动）的名字，本文假设所有本地活动名均互不相同；(iii)$c[i,j]$ 表示在通道 c 上，角色 R_i 到 R_j 的交互活动。复合活动包括：(i)$g*A*p$ 表示工作单元，当条件 g 匹配不成功时结束该工作单元，否则执行 A，执行完毕后匹配条件 p，若匹配成功则重复该工作单元，否则结束该工作单元；(ii)$A;A$ 表示顺序执行活动；(iii)$A\sqcap A$ 表示不确定的选择执行活动；(iv)$g1:A[]g2:A$ 表示有条件的选择执行活动，且当 $g1$ 匹配成功时执行前面一个活动，否则匹配 $g2$，若成功则执行后面一个活动；(v)$A||A$ 表示并行执行活动。

定义 1 一个编排 C 是一个二元组 $\langle R,A\rangle$，其中 R 是角色的集合，A 是上述文法定义的一个活动，称作编排的活动。

3.2 WS-CDL 模型的语义

给出了语法定义之后，下面定义模型的语义。文献[4,6]中定义了 WS-CDL 模型的可操作语义（operational semantics）。这种可操作的语义是通过定义一个活动到另一个活动的推导规则来建立的。每一个推导过程是活动的一小步形如 $A\rightarrow A'$ 的变迁（transition）。定义这种可操作语义的目的在于验证 WS-CDL 所描述的编排所具有的性质，如一个编排系统不会发生死锁，也不会发生活锁[6]。

文献[7]中定义了另一种语义，称作轨迹语义（trace semantics）。一个轨迹 $t=\langle\alpha,\beta,\gamma,\cdots\rangle$ 列举了一个编排的执行序列，编排的语义就定义为所有可能的轨迹集合 T，推导规则就是轨迹的各种运算法则。下面对轨迹语义进行扩展，加入卫士条件和循环条件（统称为条件）。

定义 2 带条件的轨迹 t，满足以下文法：

$Trace::=\langle Sequence\rangle$

$Sequence::=\varepsilon \mid Act \mid Act, Sequence$

$Act::= a \mid c[i,j] \mid g1:Trace \mid \neg g1\wedge g2:Trace \mid g*Trace*p$

并且假设任一轨迹 t 对于带条件的选择结构都是具有良好形式的（well-formed），也就是说在结构 $g1:Trace$ 之后紧跟着一个形如 $\neg g1\wedge g2:Trace$ 的结构。例如：$t1=\langle g1:\langle\alpha1\rangle,\neg g1\wedge g2:\langle\alpha2\rangle\rangle$ 具有良好形式，而 $t2=\langle g1:\langle\alpha1\rangle,\alpha3,\neg g1\wedge g2:\langle\alpha2\rangle\rangle$ 则不具有良好形式，不属于我们讨论的范围。

然后定义轨迹的几种操作（t 表示轨迹，T 表示轨迹集合）：

(1)连接$\hat{}$：

若 $t1=\langle\alpha1,\alpha2,\cdots\rangle$，$t2=\langle\beta1,\beta2,\cdots\rangle$，则

$t1\hat{}t2\stackrel{def}{=}\langle\alpha1,\alpha2,\cdots,\beta1,\beta2,\cdots\rangle$；

$t\hat{}T\stackrel{def}{=}\{t\hat{}t'\mid t'\in T\}$；

$T\hat{}t\stackrel{def}{=}\{t'\hat{}t\mid t'\in T\}$；

$T\hat{}T'\stackrel{def}{=}\{t\hat{}t'\mid t\in T, t'\in T\}$

(2)投影↓：

若 S 是若干个活动的集合，则 $t'=t\downarrow S$ 定义为去除轨迹 t 中那些不属于 S 的活动，然后 t 中其他活动及其顺序予以保留所得到的轨迹；

$T\downarrow S\stackrel{def}{=}\{t\downarrow S\mid t\in T\}$

(3)交错⊠：

若 t 包含的活动集合记作 $S(t)$，且 $S(t1)\cap S(t2)=\emptyset$，则 $t1\boxtimes t2\stackrel{def}{=}\{t\mid S(t)=S(t1)\cup S(t2), t\downarrow S(t1)=t1, t\downarrow S(t2)=t2\}$；

$t\boxtimes T\stackrel{def}{=}\bigcup t\boxtimes t', (t'\in T)$；

$T\boxtimes t\stackrel{def}{=}\bigcup t'\boxtimes t, (t'\in T)$；

$T\boxtimes T'\stackrel{def}{=}\bigcup t\boxtimes t', (t\in T, t'\in T')$

(4)头部 $head(\ )$ 和尾部 $tail(\ )$，其中大写字母 A 和 B 表示任意活动：

$head(\langle\varepsilon\rangle)\stackrel{def}{=}\varepsilon$；$head(\langle\alpha\rangle)\stackrel{def}{=}\alpha$；

$head(\langle g*t*p\rangle)\stackrel{def}{=}g*t*p$；

$head(\langle g1:t_1,\neg g1\wedge g2:t_2\rangle)\stackrel{def}{=}g1:t_1,\neg g1\wedge g2:t_2$；

$head(\langle A,B\rangle)\stackrel{def}{=}head(\langle A\rangle), A\neq g1:t$；

对于 t，存在 t' 使得 $\langle head(t)\rangle\hat{}t'=t$，我们把 t' 记作 $tail(t)$。

定义 3 一个编排 C 中活动 A 的语义就是 A 的所有轨迹的集合，记作 $[[A]]$ 或 T_C，定义如下：

$[[\varepsilon]]\stackrel{def}{=}\{\langle\rangle\}$，$[[a]]\stackrel{def}{=}\{\langle a\rangle\}$，$[[c[i,j]]]\stackrel{def}{=}\{\langle c[i,j]\rangle\}$，

$[[g*A*p]]\stackrel{def}{=}g*[[A]]*p$，

$[[g1:A[]g2:B]]\stackrel{def}{=}(g1:[[A]])\hat{}(\neg g1\wedge g2:[[B]])$，

$[[A;B]]\stackrel{def}{=}[[A]]\hat{}[[B]]$，

$[[A \sqcap B]] \overset{def}{=} [[A]] \cup [[B]]$,

$[[A||B]] \overset{def}{=} [[A]] \boxtimes [[B]]$

3.3 把 WS-CDL 从全局视角映射到局部视角

上面是按照 WS-CDL 规范，从全局视角为编排建模（称作全局模型）。如前文所述，编排是从整体的角度来看 Web 服务组合的过程，将各个相互联系的角色统筹考虑。那么，站在某一个角色的立场上去看组合，会是一个什么样的情况呢？显然该角色看不到其他角色的状态或者所执行的活动。局部视角就是对单个角色而言的。要将 WS-CDL 映射到局部视角，我们就得先给出关于单个角色 R_i 的形式化模型。

定义 4 一个编排 C 在某一个角色上的局部模型定义为一个四元组 $\langle O, V, Ch, A\rangle$。对于一个特定的角色 R_i，我们把该四元组记为 $C_i = \langle O_i, V_i, Ch_i, A_i\rangle$，其中 O_i，V_i 和 Ch_i 的定义与上文相同，A_i 表示在角色 R_i 上执行的活动。

角色上的任一活动 A 由下列文法定义：

$A ::= Atom \mid g * A * p \mid A ; A \mid A \sqcap A \mid g1:A[]g2:A \mid A||A$

$Atom ::= \varepsilon \mid a \mid c! \mid c?$

其中 a 表示该角色的本地活动，$c!$ 表示该角色从通道 c 中发送消息，$c?$ 表示该角色从通道 c 中接收消息，其余各项与全局模型相同。在这里先假设 g，p，$g1$ 和 $g2$ 中只含有该角色的本地变量。该定义类似于 CSP，每个角色的所有活动（在 CSP 中称为进程）都只包含本地信息。

局部模型中活动 A 的语义 $[[A]]$ 与全局模型定义类似，特别地有 $[[c!]] \overset{def}{=} \{\langle c!\rangle\}$，$[[c?]] \overset{def}{=} \{\langle c?\rangle\}$。

定义 5 一个编排 C 到角色 R_i 上的局部模型 C_i 的映射（mapping）记为 M_i。对于 O，V 和 Ch 的映射过程只保留在 R_i 上有定义的那些操作、变量和通道。对于活动的映射规则如下（忽略布尔表达式中的变量归属，即假定 g，p，$g1$ 和 $g2$ 中包含的变量是任何角色都可见的）：

$(1) M_i(\varepsilon) \overset{def}{=} \varepsilon$

$$(2) M_i(a_j) \overset{def}{=} \begin{cases} a_j, & i=j \\ \varepsilon, & i \neq j \end{cases}$$

$$(3) M_i(c[j,k]) \overset{def}{=} \begin{cases} c^{[j,k]}!, & i=j \\ c^{[j,k]}?, & i=k \\ \varepsilon, & i \neq j \text{ and } i \neq k \end{cases}$$

$(4) M_i(g * A * p) \overset{def}{=} g * M_i(A) * p$

$(5) M_i(A;B) \overset{def}{=} M_i(A); M_i(B)$

$(6) M_i(A \sqcap B) \overset{def}{=} M_i(A) \sqcap M_i(B)$

$(7) M_i(g1:A[]g2:B) \overset{def}{=} g1:M_i(A)[]g2:M_i(B)$

$(8) M_i(A||B) \overset{def}{=} M_i(A)||M_i(B)$

由此可以看出从编排的全局模型到局部模型的转化步骤：首先，根据定义 1 为 WS-CDL 规范所描述的一个特定编排建立全局模型的语法规则，然后用定义 3 建立轨迹语义，接着根据定义 5 将全局模型的语法规则映射成局部模型的语法规则，最后建立局部模型的轨迹语义。显然，对于一个特定的编排 C，它的角色集合是有限的，并且编排活动的长度也是有限的。因此转化过程的每一步均可根据相应定义的有限条规则，在有限时间内完成。

4 WS-CDL 模型的执行过程与一致性问题

4.1 执行过程

编排过程只是参与方的业务分析和设计者就各个 Web 服务之间的合作事宜所达成的合约，是一种关于全局的交互流程和约束规则的共识，仅仅描述了保证跨业务体的服务之间的互操作性过程，而实际的实现决策则由参与方内部按照编排中的描述各自制定局部的执行过程，也就是将编排过程映射到局部视角，得到参与方各自独立的局部模型。将编排 C 中所有参与方的局部模型并行起来就得到 C 的一个全局执行。

下面分别定义两个局部轨迹的并行操作和所有角色的全局执行。

定义 6 设 $t1$ 和 $t2$ 是局部模型的两个轨迹，那么它们之间的并行执行 $t1||t2$ 定义为下面 5 条规则：

$(1) \langle\rangle||t1 \overset{def}{=} t1||\langle\rangle \overset{def}{=} t1$

(2) 若 $head(t1)=a$，则 $t1||t2 \overset{def}{=} \langle a\rangle^\frown(tail(t1)||t2)$，若 $head(t2)=a$，则 $t1||t2 \overset{def}{=} \langle a\rangle^\frown(t1||tail(t2))$

(3) 若 $head(t1)=c[i,j]?$，$head(t2)=c[i,j]!$，则 $t1||t2 \overset{def}{=} \langle c[i,j]\rangle^\frown(tail(t1)||tail(t2))$，

若 $head(t1)=c[i,j]!$，$head(t2)=c[i,j]?$，则 $t1||t2 \overset{def}{=} \langle c[i,j]\rangle^\frown(tail(t1)||tail(t2))$

(4) 若 $head(t1)=g*t1'*p$ 且 $head(t2)=g*t2'*p$，则 $t1||t2 \overset{def}{=} \langle g*(t1'||t2')*p\rangle^\frown(tail(t1)||tail(t2))$

(5) 若 $head(t1)=g1:t1'$，$\neg g1 \wedge g2:t1''$ 且 $head(t2)=g1:t2'$，$\neg g1 \wedge g2:t2''$，则 $t1||t2 \overset{def}{=} \langle g1:(t1'||t2'), \neg g1 \wedge g2:(t1''||t2'')\rangle^\frown(tail(t1)||tail(t2))$

定义 7 一个编排 C 的角色集 R 为 $\{R_1, R_2, \cdots, R_n\}$，各个角色上的局部模型分别为 $C_1, C_2, \cdots, C_n$，各个角色上的轨迹语义分别记作 $T_1, T_2, \cdots, T_n$，那么所有角色的全局执行 $T'_C \overset{def}{=} T_1||T_2||\cdots||T_n \overset{def}{=} \bigcup(t_1||t_2||\cdots||t_n)$，其中 $t_i \in T_i$，$i=1,\cdots,n$。

由定义 7 可知，所有角色 $\{R_1, R_2, \cdots, R_n\}$ 的全局执行 T'_C 由以下两步得到：1）对于每个角色，各取其轨迹语义中的一个轨迹，相互并行，反复利用定义 6 中定义的 5 条规则进行运算，直到消去并行符号，得到轨迹集合。2）取遍各个角色的轨迹语义中的轨迹进行第一步中的运算，将每次运算所得的轨迹集合做集合的并操作即得到 T'_C。显然这些运算会在有限的时间内得出最终结果。证明的关键在于验证第一步中的反复运算最后能消去并行符号，下面构造一个算法来消去并行符号（对于带条件的结构内部，可以单独运用此算法）：

(i) 若存在 $a=head(t)$，则运用定义 6 的规则(2)，提出本地活动，然后转到第(i)步；否则转到第(ii)步。

(ii) 若存在 $c[i,j]? = head(t)$，则必然存在且仅存在一个对应的 $c[i,j]! = head(t')$，那么运用定义 6 的规则(3)，提出交互活动，然后转到第(i)步；否则转到第(iii)步。

(iii) 若存在 $g*t'*p=head(t)$，则必然可以运用定义 6 的规则(4)，提出选择活动，然后转到第(i)步；否则转到第(iv)步。

（下转第 162 页）

ring, 2004:692-694

[6] Lee G L, Lo S C. Broadcast data allocation for efficient access of multiple data items in mobile environments[J]. Mobile Networks and Applications, 2003, 8(4):365-375

[7] Chang Y I, Hsieh W H. An efficient scheduling method for query-set-based broadcasting in mobile environments[C]// Proc. of the 24th Int. Conf. on Distributed Computing Systems Workshops. 2004:478-489

[8] 刘云生,杨进才,廖国琼. 移动环境中实时事务数据的广播调度[J]. 小型微型计算机系统,2004,25(4):531-534

(上接第 153 页)

(iv)若存在 $g1:t1'$, $\neg g1 \wedge g2:t1''=head(t)$,则必然可以运用定义 6 的规则(5),提出循环活动,然后转到第(i)步;否则转到第(v)步。

(v)此时必然满足定义 6 的规则(1),即可完全消去并行运算符号。

由此可以看出,定义 7 中所有角色的全局执行是可以在有限步内得出结果的。

4.2 一致性问题

一个按照各自的执行过程进行 Web 服务组合,然后再将各参与方并行执行,最终效果应该符合编排中所描述的那样,局部模型的并行组合之后的语义应该等价于全局模型的语义,即 $T'_C=T_C$,这就是一致性问题。

一致性表现在以下两个方面:(i)各个参与方并行执行之后,活动的执行顺序应该符合编排里所描述的全局过程;(ii)在某些特定的时刻,对于编排里所规定的某些信息(变量),各个参与方必须达成一致的共有的认识(common view),当参与方独立执行时,这个共有的认识也不能被破坏。由于各个参与方是相互独立而且异步地执行的,显然,不一致的情况会经常发生,例如:(i)有一个全局的编排活动 $A=a_1^1;a_2^1$,映射到 R_1 和 R_2 上的局部执行活动分别为 $M_1(A)=a_1^1$ 和 $M_2(A)=a_2^1$,显然两个局部活动的并行结果 $a_1^1||a_2^1$ 与全局的编排 $a_1^1;a_2^1$ 不一致;(ii)编排中规定了两个角色在交互中必须对某个变量达成一致,但是当独立执行时候,其中一个角色改变了该变量的值,而另一个角色还仍然保留着原来的值。一致性问题的处理非常重要,是因为 WS-CDL 规范仅仅用声明性的方法要求以交互信息校准(Interactive Information Alignment)的方式或编排协调(Coordination)的方式保证同步,但是并没有明确定义这些方式的执行机制。文献[8]中通过为全局模型和局部模型建立轨迹语义进一步探讨了一致性问题,提出了一些解决不一致问题的方案,如提出"支配选择"的概念,设定某个角色为"支配角色",由它来支配其余角色应该进入哪个选择分支。文献[9]也建立了模型,通过校准机制着重讨论了一致性问题。

本文中局部模型的建立以及全局模型到局部模型的映射过程都忽略了一个基本的事实,即卫士条件和循环条件中可能包含不同角色的变量。例如有一个编排 C 的活动 $A=g_1:P[]g_2:Q$,其中 $g_1=(v_1^1<8 \wedge v_2^1<7)$, $g_2=(v_1^1>8 \wedge v_2^1>7)$。此时映射到局部模型之后,对于 R_1,因为只能访问到其本地变量 v_1^1 而不能访问远程变量 v_2^1,所以必然不能采取正确的选择,对于 R_2 也是如此,这样不一致的情况就产生了。为解决这类问题,需要引入同步机制(synchronization),在选择之前通过加入一些额外的交互来达到同步。上述例子中对 A 加入同步交互之后变成 $A'=(c1[1,2]||c2[2,1]);g_1:P[]g_2:Q$,其中 $c1[1,2]$代表从 R_1 到 R_2 且通道名为 c1 的交互,将 R_1 的本地变量 v_1^1 的值传递到 R_2 中作为一个副本 $\tilde{v}_1^1$, $c2[2,1]$代表从 R_2 到 R_1 且通道名为 $c2$ 的交互,将 R_2 的本地变量 v_2^1 的值传递到 R_1 中作为一个副本 $\tilde{v}_2^1$。这样在进行选择的时候,就可以采取正确的方式达到一致性。

结束语 正如文献[1]中提到的那样,在 Web 服务的发展过程中,WS-CDL 规范似乎出来得太早。WS-CDL 规范没有形式化基础,规范的开发也没有基于任何一种可执行的程序语言。本文为 WS-CDL 建立了一个形式化模型,扩展了轨迹语义,加入条件,更好地模拟了编排的执行,通过对带条件的轨迹语义定义全局执行来讨论全局与局部的一致性问题。进一步的工作包括建立 WS-CDL 规范到全局模型的自动转换过程以及通过校准和同步来完善一致性问题的讨论。

参 考 文 献

[1] Barros A, Dumas M, Oaks P. A Critical Overview of the Web Services Choreography Description Language [OL]. http://www.bptrends.com, 2005

[2] Decker G, Zaha J M, Dumas M. Execution Semantics for Service Choreographies. Preprint # 4329, Faculty of IT, Queensland University of Technology, May 2006

[3] Foster H, Uchitel S, Magee J, et al. Model-based analysis of obligations in web service choreography[C]// Proceedings of International Conference on Internet and Web Applications and Services, 2006. 2006,149:19-25

[4] Busi N, Gorrieri R, Guidi C, et al. Towards a formal framework for Choreography[C]// Proceedings of 3rd International Workshop on Distributed and Mobile Collaboration, 2005. DMC'05

[5] Yang H L, Zhao X P, Qiu Z Y. A Formal Model for Web Service Choreography Description Language (WS-CDL)[C]// Proceedings of IEEE International Conference on Web Services (ICWS'06). September 2006:893-894

[6] Zhao X P, Yang H L, Qiu Z Y. Towards the Formal Model and Verification of Web Service Choreography Description Language [C]// Proceedings of Web Services and Formal Methods (WS-FM'06). September 2006:273-287

[7] Qiu Z Y, Cai C, Zhao X P, et al. Exploring into the Essence of Choreography. Preprint 2006-63 of Institute of Mathematics, Peking University [OL]. http://www.math.pku.edu.cn:8000/en/preindex.php

[8] Qiu Z Y, Zhao X P, Cai C, et al. Towards the theoretical foundation of choreography[C]// Proceedings of the 16th international conference on World Wide Web (WWW'07). May 2007:973-982

[9] Kazhamiakin R, Pistore M. Choreography Conformance Analysis: Asynchronous Communications and Information Alignment [C]// Proceedings of Web Services and Formal Methods (WS-FM'06). September 2006:227-241