Software Engineering and Applications 软件工程与应用, 2014, 3, 1-8
http://dx.doi.org/10.12677/sea.2014.31001 Published Online February 2014 (http://www.hanspub.org/journal/sea.html)

Research on Semantic Web Service Process Model Based on Process Ontology

Hao Hong[1], Yi Zeng[1,2], Bo Zhou[1], Cuiqin Wang[1]

[1]College of Computer Science, Chongqing University, Chongqing
[2]Chongqing Key Laboratory of Software Theory and Technology, Chongqing
Email: honghao3763876@163.com

Received: Dec. 21st, 2013; revised: Jan. 20th, 2014; accepted: Jan. 29th, 2014

Abstract: Aiming at the problem that Ontology Web Language for Services (OWL-S) process model lacks capacity for process information of Web Service discovery and expressing Quality of Service (QoS) attributes, this paper proposes an establishment method of Semantic Web Service (SWS) Process Model based on Process Ontology. First, the cycle progressive and stepwise refinement ontology modeling method is adopted to build process ontology which takes QoS attributes into account. Second, this paper gives a formalized definition of this process ontology and obtains the SWS process model. Finally, this paper designs a transformation algorithm, utilizes this algorithm to transforming SWS process model to process graph and detects the correctness of process graph after the conversion. At last, the availability and effectiveness of SWS process model are verified by experiments in the paper.

Keywords: Web Ontology Language for Service (OWL-S); Semantic Web Service (SWS); Ontology Modeling; Process Model; Process Graph

基于过程本体的语义 Web 服务过程模型的研究

洪　豪[1]，曾　一[1,2]，周　波[1]，王翠钦[1]

[1]重庆大学计算机学院，重庆
[2]软件理论与技术重庆市重点实验室，重庆
Email: honghao3763876@163.com

收稿日期：2013 年 12 月 21 日；修回日期：2014 年 1 月 20 日；录用日期：2014 年 1 月 29 日

摘　要：针对 Web 服务本体语言(OWL-S)过程模型存在 Web 服务发现时对过程信息和 QoS 属性表达能力不足的问题，本文提出了一种基于过程本体的语义 Web 服务(SWS)过程模型的建立方法。首先，采用循环递进、逐步求精的本体建模方法建立过程本体，并且在过程本体中考虑 QoS 属性；其次，对其进行形式化定义，得到语义 Web 服务过程模型；最后，设计了一种转换算法，利用该算法将语义 Web 服务过程模型转换为过程图，并检测转换后的过程图的正确性。本文最终通过实验验证了语义 Web 服务过程模型的可用性和有效性。

关键词：Web 服务本体语言；语义 Web 服务；本体建模；过程模型；过程图

1. 引言

随着 Web 服务的飞速发展，Web 服务本体语言(Ontology Web Language for Services, OWL-S)[1]也得到极大发展，但是由于所描述的是静态的 Web 服务，不包含任何有关服务执行过程的信息，并且 OWL-S Profile 不考虑 Web 服务的非功能属性，导致所发现的

服务不能较好地满足用户需求且查准率较低；OWL-S ServiceModel 对业务过程知识的表达能力还非常有限，不能充分描述过程间的关系、过程领域的继承规则，本质上也不具有过程分解和过程子活动依赖的能力，这都导致程本体论的语义不够丰富。

对于 OWL-S 的局限性，国内国外的研究者提出了很多解决的方法[2-4]。如文献 2 和 3 提出的基于 QoS 的语义 Web 服务发现方法使得用户可以根据自己的情况或个性化的需求进行服务选择；文献 4 通过考虑 Web 服务的过程信息，将得到的过程模型转化为过程图，从而通过过程图的匹配来达到 Web 服务发现的目的；上述方法在某种程度上均能提高 Web 服务匹配精度。但是，由于缺少对 Web 服务得到过程模型的准确性进行检验，或者单一的只是考虑 QoS 或是过程方面的信息，使得 Web 服务发现的查全率和查准率还是不尽如人意。

针对以上问题，本文在传统过程本体[5,6]的研究基础上，提出一种基于过程本体的语义 Web 服务过程模型的建立方法，在过程本体中考虑 QoS 需求信息，并对过程本体进行形式化定义，得到基于该过程本体的过程模型，设计了一种过程模型到过程图的转换算法，并利用该算法将过程模型转换成过程图，最后通过检测算法检测转换后过程图的正确性。

2. 过程本体

2.1. 顶层过程本体

过程本体是本文研究语义 Web 服务匹配工作的基础，

是动态选择 Web 服务的重要依据，是实现 Web 服务自动组合的保证，因此，设计过程本体模型的目的是为用户提供一个可共享的、统一的过程本体描述框架。同时，本文定义的过程本体是在扩展 OWL-S 的基础上，增强 OWL-S 对 QoS 的语义描述能力，为服务请求者提供最符合其 QoS 需求的服务，而且构造的过程模型具有动态交互和过程分解和过程子活动间依赖的能力。

Protégé 是斯坦福大学基于 Java 语言开发的流行的本体开发工具，属于开放源代码软件。主要用于语义网中本体的构建，现在最新版本为 4.3 版本。如图 1 所示给出由 Protégé 开发本文提出的过程本体的核心本体。

2.2. 过程本体的定义

本节将给出过程本体中的核心概念及概念关系的定义，这是后面建立过程模型的基础。

定义 1：过程本体(Process Ontology)为一个四元组，PO = {pC,pI,pP,pR}；其中：pC 表示过程类的集合；pI 表示过程实例集合；pP 表示过程属性集合；pR 表示过程的关系集合，

$$pR \subseteq (pC \cup pI \cup pP) \times (pC \cup pI \cup pP)$$

定义 2：过程(Process) P = {Pname,A,R,CS, QoS, Ptype}

Pname：过程的名称；

A：为完成预定目标或达到期望状态的活动的集合；

R：组成过程的活动之间的关系集合；

CS：组成过程的活动之间的约束集合；

QoS：表示过程或活动执行时需要满足的服务质量属性；

Ptype：表示过程的类型；Ptype = {“Atomic”, “Simple”, “Composite”}，其中：

Atomic 表示原子过程(Atomic Process)，指没有任何子过程的过程；

Simple 表示简单过程(Simple Process)，是一个抽象概念，作为原子过程的一种视图，它不能被直接调用，必须用原子过程调用；

Composite 表示组合过程(Composite Process)，是由若干个原子和组合过程构成的过程。

定义 3：子过程(SubProcess)关系

若过程 $P_1 = (\text{Pname}_1, A_1, R_1, CS_1, \text{Ptype}_1)$ 和 $P_2 = (\text{Pname}_2, A_2, R_2, CS_2, \text{Ptype}_2)$ 有

$$(\forall a_2 \in A_2 \Rightarrow a_2 \in A_1) \wedge (\forall r_2 \in R_2 \Rightarrow r_2 \in R_1)$$
$$\wedge (\forall cs_2 \in CS_2 \Rightarrow cs_2 \in CS_1) \wedge (\text{Ptype}_2 \prec \text{Ptype}_1)$$
$$\Rightarrow \text{SubProcess}(P_1, P_2)$$

表示 P_1 为 P_2 的子过程，其中 $\prec$ 表示 Ptype 的一种从属关系。

公理 1：子过程关系：具有传递性、自反性、不对称性，是偏序关系。

定义 4：端口(Ports) Port = (Name,Type,PA,Pre,E) 其中：

Figure 1. **Core process ontology**
图 1. **核心过程本体**

Name：端口名称；Type：端口类型，Type = {“I”, “O”}，“I”表示为输入端口，“O”表示为输出端口；PA：参数的集合；Pre：前置条件的集合；E：效果的集合。特别当 type = “I”时，E = ∅；type = “O”时，Pre = ∅。

定义 5：依赖(Depend)：Depend(A_1, A_2)表示活动集合 A_1 和活动集合 A_2 之间存在依赖关系，这种依赖关系是通过时序关系协调连接的，时序关系的描述包括：Before，Same，BeginAt，EndAt 等等，下面一一进行介绍。

时间点(TimePoint)是时间上的一个点，BeginAt (EndAt)是活动和时间点的关系，用来表示某个活动开始(结束)于某个时间点。如 BeginAt(A_1, t_1) 和 EndAt(A_1, t_1) 分别表示 A_1 活动在 t_1 时间开始或结束。

之前(Before)是时间点之间的关系，如果时间点 t_1 在时间点 t_2 之前，表示为 Before(t_1, t_2)。之前关系满足传递性、反对称性，不满足自反性。

同时(Same)是时间点之间的关系，如果时间点 t_1

和时间点 t_2 指的是时间轴上相同的一个点，表示为 $\text{Same}(t_1,t_2)$。

同时关系满足传递性、对称性、自反性.

重叠(Overlap)关系：

$$(\text{EndAt}(A_1,t_1))\wedge\text{BeginAt}(A_2,t_2)\wedge\text{Before}(t_2,t_1)\leftrightarrow\text{Overlap}(A_1,A_2)$$

覆盖(Overlay)关系：

$$(\text{Begin}(A_1,t_{11})\wedge\text{EndAt}(A_1,t_{12})\wedge\text{BeginAt}(A_2,t_{21})\wedge\text{EndAt}(A_1,t_{22})\wedge\text{Before}(t_{11},t_{21})\wedge\text{Before}(t_{22},t_{12}))\leftrightarrow\text{Overlay}(A_1,A_2)$$

相遇(Meet)关系:

$$(\text{EndAt}(A_1,t_1)\wedge\text{BeginAt}(A_2,t_2)\wedge\text{Same}(t_1,t_2))\leftrightarrow\text{Meet}(A_1,A_2)$$

同时(Equal)关系：

$$(\text{BeginAt}(A_1,t_{11})\wedge\text{EndAt}(A_1,t_{12})\wedge(\text{BeginAt}(A_2,t_{21})\wedge\text{EndAt}(A_2,t_{22})(\wedge\text{Same}(t_{11},t_{21})\wedge\text{Same}(t_{12},t_{22}))\leftrightarrow\text{Equal}(A_1,A_2)$$

早于(Earlier)关系：

$$(\text{EndAt}(A_1,t_1)\wedge\text{BeginAt}(A_2,t_2)\wedge(\neg\text{Before}(t_2,t_1)))\leftrightarrow\text{Earlier}(A_1,A_2)$$

定义 6：过程属性(ProcessAttribute) PA = (Pname, Constraint, Res, C, Exc)，其中 Pname 表示过程名；Constraint 是约束集，表示约束的公式的集合；Res 表示过程或活动发生时需用资源；C 表示过程类别，通过分类标准把相似语义(等价于具有相同目的的过程)的过程进行分类；Exc 表示过程执行时可能出现的异常，一般分为：运行时异常(RuntimeException)和受检查异常(Checked Exception)。

定义 7：继承(Specialization)：存在两个过程 p1，p2，若

$$\text{subProcess}(p_1,p_2)\wedge\exists p((p_1\ \text{hasProperty}\ p)\wedge\neg(p_2\ \text{hasProperty}\ p))$$

则 p_1 是过程式继承 p_2；其中 $p\in\text{subProcess}\cup pP$。

定义 8：组成(ComposedBy) ComposedBy(A_1,A_2)表示

A_2 是 A_1 的一个组成过程或活动，满足下列条件：

1) $\text{Overlay}(A_1,A_2)$；

2) 若 A_1， A_2 为过程，则 $\text{SubProcess}(A_2,A_1)$，显然

$$\text{ComposedBy}\subseteq\text{Process}\times(\text{Process}\cup\text{Activity})$$

定义 9：分解(Decompose) B = De (A)表示把过程 A 分解成过程(活动)集合 B，其中 $B=\{B_1,B_2,\cdots,B_n\}$，满足下列条件：

1) $\forall i\in\{i|0<i\le n\},\Rightarrow\text{ComposeBy}(A,B_i)$；

2) $\forall\{0<i,j\le n\},\text{且}i\ne j\Rightarrow\neg\text{Overlap}(B_i,B_j)$；

$\forall i\in\{i|0<i\le n\}$,

3) $\text{EndAt}(B_i,t_1)\wedge\text{BeginAt}(B_{i+1},t_2)$；

$\Rightarrow\text{Same}(t_1,t_2)$

4) $\text{BeginAt}(B_1,t_1)\wedge\text{EndAt}(B_n,t_2)$

$\Rightarrow\text{BeginAt}(A,t_1)\wedge\text{EndAt}(A,t_2)$。

2.3. 过程本体元模型的建立

一个过程被定义是通过指定它的分解(它所包含的原子活动或者原子过程)，接口(所包含的端口)，活动之间的依赖，每个过程实体的属性。每一个过程可以链接到它可以处理的各种异常情况，并在异常出现的情况下，可以转到处理过程对异常进行处理。过程与静态实体间的属性关系遵循传统本体中的相应定义，这里不再展开论述。如图 2 为过程本体元模型。

3. 基于过程本体的过程模型

根据 2.2 节过程本体的定义和 2.3 节建立的过程本体元模型，本节将给出过程模型的定义，同时对 Web 服务质量模型及过程图进行定义，最后实现过程模型到过程图的转化。

将一个 Web 服务描述成过程模型[7]的目的有两个，首先，给出一个详细的如何与服务进行交互视图，便于在以后研究服务请求者和服务提供者之间的过程匹配；其次，通过建立过程模型，可以清晰的知道过程的输入与前置条件和过程执行后产生的输出及过程的效果。

3.1. 过程模型的定义

定义 1：过程模型(ProcessModel) PM = (Node, CS, Link)，其中：Node 表示节点集合，包括活动节点(ANode)，连接节点(LNode)，空节点(Null)；CS 表示

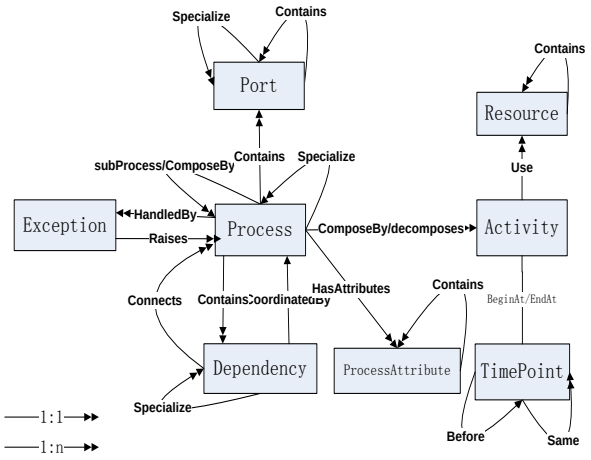

Figure 2. **The meta model of process ontology**
图 2. **过程本体元模型**

过程的控制结构，可以用来连接活动与过程；Link 表示二元关系， $Link \subseteq Node \times Node$ 。

定义 2：Web 服务质量模型 QoS = (Cost, ResponseTime, Reliability, Avaliabity, Security,…)是一个可扩充向量。由于本文主要研究过程本体下过程模型的 QoS，出于实用性和可测量性的考虑，所以从 QoS 本体[8]中选取 5 种最有代表性的 QoS 属性来考虑 Web 服务的服务质量。其中：

费用(cost)是描述服务请求者需要为使用服务而付出的费用；

响应时间(ResponseTime)客户端从提交服务请求到获得服务响应所花的时间，包括服务时间和往返通信时间；

可靠性(reliability)为成功执行次数与调用执行总次数的比率；

可用性(Availabity)指服务能够被其用户正常使用的概率；其计算公式为：MTTF/(MTTF + MTTR)；

安全性(Security)是 Web 服务质量的一个重要方面，通过验证涉及到的各方、对消息加密以及提供访问控制来提供机密性。

定义 3：(前驱节点、后继节点)N 是节点集合，$A \subseteq N \times N$ ， $\forall n \in N$ ， n 的前驱节点集合 $\cdot n = \{x \in N | (x,n) \in A\}$ ，n 的后继节点集合 $n \cdot = \{x \in N | (n,x) \in A\}$ 。

定义 4：过程图的边 $E = (v_i, v_j, w)$ ，其中 v_i 表示边的起点， v_j 表示边的终点，w 表示边上的权重，w 的值定义如下：

$$w = \frac{\min\{Qos(v_i), Qos(v_j)\}}{Qos(v_i) + Qos(v_j)} \quad (1)$$

定义 5：过程图[7]为五元组 $PG = (S, F, A, C, E)$ 其中：

a) S 表示起始节点， $|S| \geq 1$ 且

$$\forall s \in S : |s \cdot| = 1 \wedge |\cdot s| = 0 。$$

b) F 表示结束节点， $|S| \geq 1$ 且

$$\forall s \in S : |f \cdot| = 0 \wedge |\cdot f| = 1 。$$

c) A 表示所有活动节点构成集合，

$$\forall f \in F : |\cdot f| = 1 \wedge |f \cdot| = 1 。$$

d) C 表示所有连接节点构成的集合，

$$\forall c \in C : |\cdot c| = 1 \wedge |c \cdot| > 1 \vee |\cdot c| > 1 \wedge |c \cdot| = 1 。$$

e) $E = \{e_1, e_2, \cdots, e_n\}$ 表示所有连接边构成的集合，并且 $E \subseteq (\{S\} \cup \{E\} \cup A \cup C) \times (\{S\} \cup \{E\} \cup A \cup C)$ ，并且满足 $\forall n \in (F \cup A \cup C) : (n,n) \notin E$ ；
$\forall x, y \in (F \cup A \cup C) : |\{(x,y)\}| (x,y) \in E| = 1$ 。

3.2. ProcessModel 到过程图的转化

Control Construct 定义了复合过程中每个子过程的执行顺序。本文过程本体中定义的有 Sequence、Split、Split-Join、Unordered、Choice、If-Then-Else、Do-While 和 Repeat-Until。控制结构的具体转换规则如图 3 所示。

在上述 8 种单个控制结构转换方法的基础上，给出由 8 种控制结构嵌套而成 OWL-S ServiceModel 到过程图的转换算法。算

法的基本思想如下：

第一步：获取 ProcessModel 最外层的控制结构，如果该控制结构没有嵌套的控制结构，则根据该控制结构的转换规则直接转换，然后返回。

第二步：如果该控制结构内有嵌套的控制结构，则将内部的控制结构视为临时节点，并根据该控制结构的转换规则转换。

第三步：将第二步中得到的内部的控制结构作为最外层控制结构转第一步，直到所有的控制结构被转换结束为止。

由于在转化过程中将原子过程的过程名，输入参数集和输出参数集原样转化为活动节点的 3 个属性，

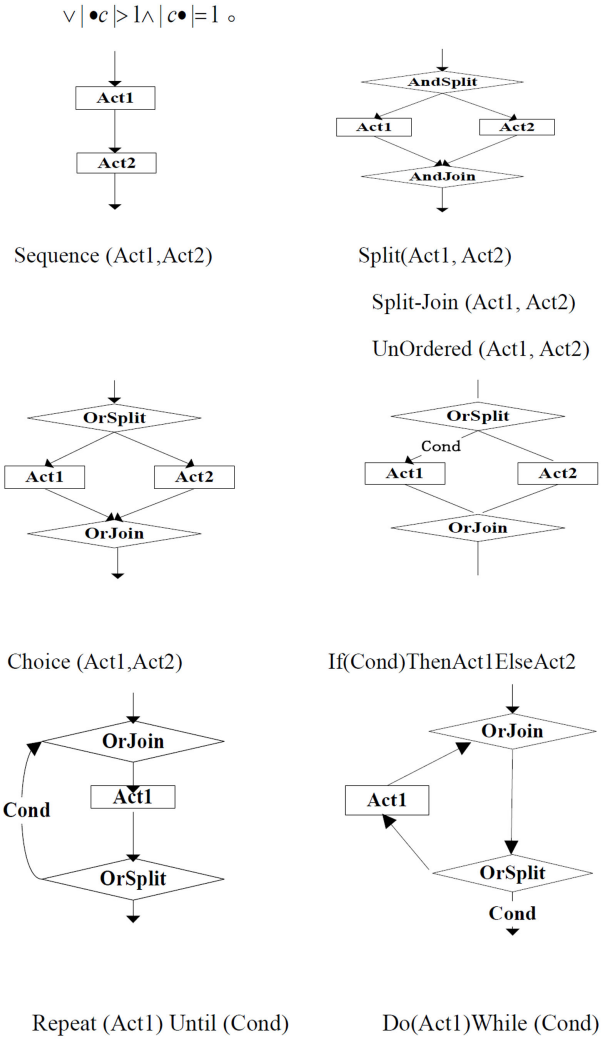

Figure 3. The transformation from control structure to process graph

图 3. 控制结构到过程图之间的转换

因而保证了过程图与 ProcessModel 语义上的一致性。

3.3. 过程模型正确性的检测

3.3.1. 过程模型的静态性质检测

通过检测过程模型中的过程的活动之间的活动连接和数据连接是否合理及它们之间的一致性，初步对过程模型进行静态性质性质检测，可以分为下面三种情况：

(1) 活动连接的合理性：具有 Sequence 关系的活动之间，前一个活动的 Effect 和后一个活动的 PreCondition 是一致的。

(2) 活动数据连接的合理性：活动之间的数据连接中，不能存在两个及两个以上的活动同时给一个活动提供相同变量的数据，且这些数据连接中定义的变量集合的交集不为空，这时就会发生数据冲撞，数据连接的定义不合理。

(3) 活动连接和数据连接的一致性：活动连接表示了过程中的活动的执行路由，执行的顺序需要和数据连接定义的数据供给—消费关系所蕴含的执行顺序一致。

3.3.2. 过程模型的动态性质检测

经过 3.3.1 节的静态性质检测后的过程模型还可能存在其它不合理的问题，而这些问题需要考虑过程的执行才能检测出来，因此下面需要对其进行动态性质检测。

本文的过程模型是一种有向图，在这里我们首先给出关于活动(节点)和活动连接(边)的状态的定义。

活动有五种状态：Ready、Enabled、Disabled、Running 和 Completed。

活动连接有三种状态：Ready，Active，Inactive。

为了直观、清晰的理解过程模型的执行过程中活动和边转换的关系，下面进行定义：

定义 1：设 $M=(A,E,\delta,S,F)$，满足如下条件的有向图称为 M 的状态转移图：

(1) a∈A⇔ a 是该有向图中的一个顶点。

(2) δ(a,e) = b 图中有一条从顶点 a 到顶点 b 标记为 e 的弧。

(3) 标记为 S 的箭头指出 M 的开始状态。

(4) F 表示终止状态，用双层圈标出。

假设当一个过程模型建立后，活动和活动连接都处于 Ready 状态，如图 4 表示活动和活动连接的状态转移图。

一个过程模型不正确，典型问题有：(a) 过程非正确结束，即过程执行到结束节点，但还有其他活动处于执行状态或者准备状态；(b) 存在永远不可能被执行的活动；(c) 发生死锁，即在过程未结束前，没有可以执行的活动。

下面给出一段检测的算法，本文直接使用定义 5 的过程图来进行检测。

算法 1：过程图的检测算法

输入：待检测的过程图输出：合理则返回 true，不合理则输出错误问题，返回 false。

(1) 初始化 Node 为有向图全体节点集合，执行节

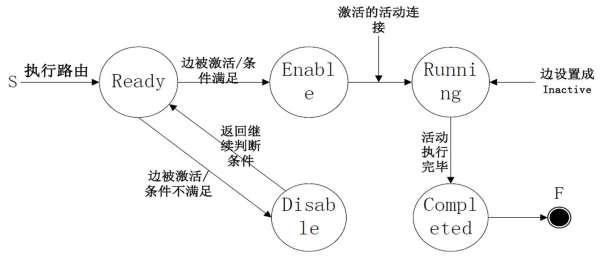

Figure 4. **The state transition diagram of M**
图 4. **M 的状态转移图**

点集合 Enode = {start}；

(2) 对于每一个 Enode 中节点，执行完毕，将节点的状态置为 Completed 并从 Enode 移去，加入执行完毕集合 Fnode；

(3) 刷新所有节点的状态，获取 Enode 的所有出边 e；

a) If (Condition(e) = true)将边的状态置为 Active，将 e 加入到 E 中，得到状态为 Active 的边的集合 E；

b) If (Condition(e) = false)将边的状态置为 InActive,将 e 边出发连接的后续节点加入 Fnode；

(4) If (E = Ø)输出“没有可激活的连接，存在死锁”，return false；否则获取{E}所激活的活动，将其中状态为 Enabled 活动的状态置为 Running，加入 Enode；继续步骤(2)；

(5) If (Endnode) and (Enode/Endnode ≠ Ø)输出“错误终结”，return false；

If (Node/(EnodeVFnode) ≠ Ø)输出“存在不可达活动”，return false。

4. 实验验证

实验实现了如图 5 的 ProcessModel 生成和转换框架：

本文通过网上购物 Web 服务的实现过程为例来验证上述框架的有效性。

首先，基于本文定义的过程本体，利用 Altova UModel 对网上购物过程进行建模，得到其 UML 活动图(如图 6)；

然后通过 Altova XMLSpy 提取 UML 活动图的 XMI 信息(如图 7)；最后对产生的 XMI 文件进行分析，解析，在 Java 平台下使用 DOM4J 技术实现，得到网上购物 Web 服务的 ProcessModel (如图 8)。

根据得到的网上购物 ProcessModel，按照 3.2 节

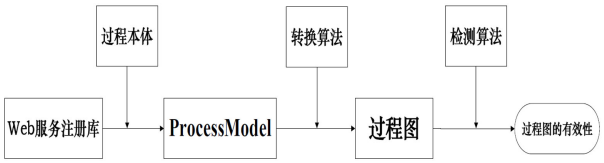

Figure 5. **The generation and transformation architecture of ProcessModel**
图 5. **ProcessModel 生成和转换框架**

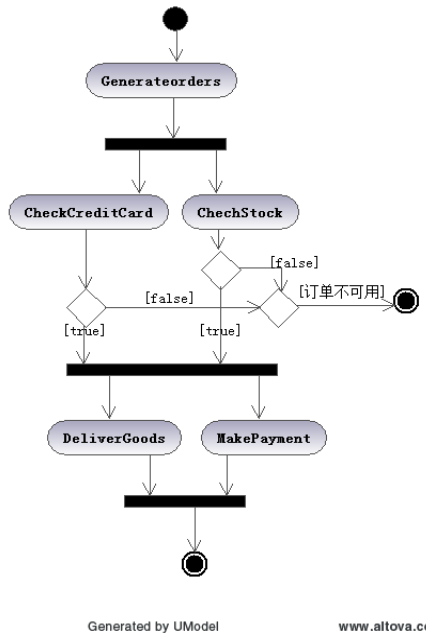

Figure 6. **UML activity diagram of shopping online process**
图 6. **网上购物过程的 UML 活动图**

Figure 7. **XMI information of shopping online model**
图 7. **网上购物模型的 XMI 信息**

提出的转换算法得到下面过程图，如图 9 所示。

采用 3.2 节的检测算法对图 8 所表示的过程图进行检测，根据检测的结果最终验证了过程图的正确性，从而验证了本文基于过程本体的过程模型的可行性

```
<process:ProcessModel rdf:ID="Shopping_On_ProcessModel">
    <process:hasProcess rdf:resource="#Shopping_Online_Composite" />
    <process:describes
        rdf:resource="http://www.cqu.edu.cn/his/BUY_Service#Shopping_Online_Service" />
</process:ProcessModel>
<process:CompositeProcess rdf:ID="Shopping_Online_Composite">
    <process:composedBy>
        <process:Sequence>
            <process:components rdf:parseType="Collection">
                <process:AtomicProcess rdf:about="#GenerateOrders" />
                <process:Decompose rdf:about="#Pay">
                <process:AtomicProcess rdf:about="#CheckCreditCard" />
                <process:AtomicProcess rdf:about="#MakePayment" />
            </process:Decompose>
                <process:CompositeProcess rdf:about="#DeliverGoods" />
            </process:components>
        </process:Sequence>
    </process:composedBy>
</process:CompositeProcess>
.....
<ResponseTime rdf:ID="ResponseTime_case">
    <hasMetric>
        <Metric rdf:ID="Metric_case">
            <hasUnit>
                <Unit rdf:ID="Unit_case">
                    <unit_value rdf:datatype="http://www.w3.org/2001/XMLSchema#string">s</unit_value>
                </Unit>
            </hasUnit>
            <value rdf:datatype="http://www.w3.org/2001/XML Schema#float">3.0</value>
        </Metric>
    </hasMetric>
    <hasMetricType>
        <operator rdf:datatype="http://www.w3.org/2001/XMLSchema#string">lessthan</operator>
    </hasMetricType>
</ResponseTime>
<process:Input rdf:ID="Credit_Card_ID_IN">
    <process:parameterName>Credit_Card_ID_IN</process:parameterName>
    <process:parameterType rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
</process:Input>
<process:Input rdf:ID="Credit_Card_Password_IN">
    <process:parameterName>Credit_Card_Password_IN</process:parameterName>
    <process:parameterType rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
</process:Input>
<process:Output rdf:ID="Credit_Card_authentication_OUT">
    <process:parameterName>Credit_Card_authentication_OUT</process:parameterName>
    <process:parameterType rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
</process:Output>
```

Figure 8. **The Web Service ProcessModel of shopping online**
图 8. **网上购物 Web 服务的 ProcessModel**

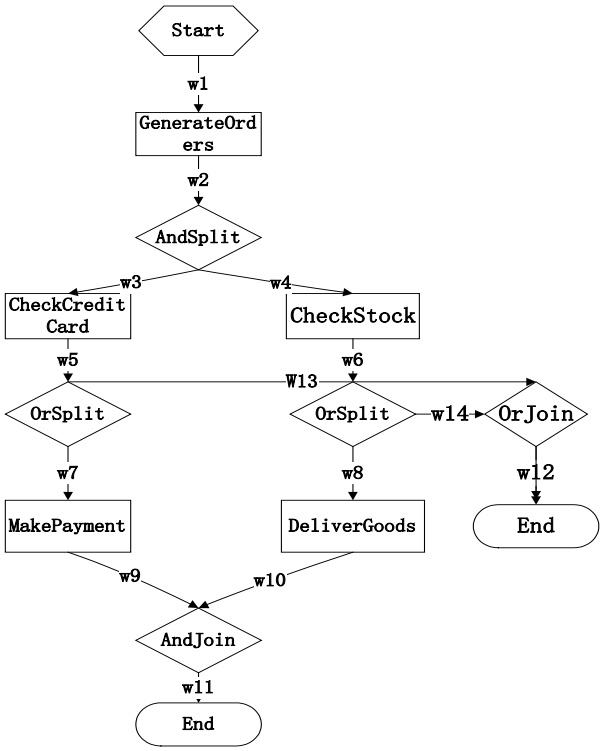

Figure 9. **The process graph transformed from shopping online ProcessModel**
图 9. **网上购物 ProcessModel 转换成的过程图**

与有效性。

5. 结语

本文提出一种基于过程本体的语义 Web 服务过程模型方法，得到的过程模型在考虑 Web 服务服务质量的同时也考虑过程行为的相关信息，并设计一种转换算法完成从过程模型到过程图的转换，同时设计了检测算法检验得到的过程图的正确性，达到了设计的预期目标，可以进一步提高 Web 服务匹配的精度。下一步工作将改进转换算法，提高其执行效率，并研究基于 QoS 权值的过程图之间的相似度匹配从而实现 Web 服务过程匹配。

参考文献 (References)

[1] W3C Member Submission (2004) OWL-S: Semantic markup for web services. http://www.w3.org/Submission/OWL-S/

[2] 邹金安 (2009) 基于 QoS 的语义 Web 服务发现研究. *计算机应用*, **29**, 2844-2846.

[3] 周娟, 李蜀瑜 (2011) 一种基于服务质量的语义 Web 服务发现框架. *计算机技术与发展*, **2**, 128-135.

[4] 曾一, 胡延强, 洪豪 (2012) 基于 OWL-S 过程模型的 Web 服务发现方法. *计算机工程*, **17**, 28-32.

[5] Aitken, S. and Curtis, J. (2002) A process ontology. In MGomezPerrz, A. and Benjamins, V.R., Eds., *Proceedings of EKAW*, Spring-Verlag, Berlin, 108-113.

[6] Yu, X., Yuan, F.L. and Zhang, Y. (2011) The modeling and application of process ontology in the field of space debris mitigation. *International Conference on Transportation, Mechanical, and Electrical Engineering* (*TMEE*), Changchun, 16-18 December 2011, 431-434.

[7] W3C (2004) Modeling Services as Processes. http://www.w3.org/Submission/OWL-S/#5

[8] 柯杨华, 郭红 (2006) 基于 QoS 的语义 web 服务组合研究. *计算机工程与应用*, **S1**, 75-77, 104.