# 业务流程驱动的半自动语义Web服务组合方法研究及应用

熊丽荣 余 晖 范 菁 董天阳
(浙江工业大学计算机学院 杭州 310014)

**摘 要** 随着Web服务数量的不断增多,如何利用Web服务组建松散耦合的应用系统以满足不断变化的业务需求,成为一个研究热点。传统的业务流程驱动的服务组合通常需要预定义组合流程,当业务流程改变时则需重新定义流程,因此不能适应动态业务需求。针对上述问题,提出了一种业务流程驱动的半自动语义Web服务组合方法。该方法在业务流程驱动的服务组合基础上,将局部动态流程用抽象服务描述,当流程执行阶段不存在能满足抽象服务需求的单个服务时,采用自动服务组合方式将抽象服务替换成组合服务,并执行自动组合方法。该方法由于采用了基于回溯树的服务组合方法,并在回溯树中增加语义信息,还通过计算组合流程的QoS来实现组合服务的选取,因此能有效解决因缺乏语义导致查全率低的问题。最后给出了半自动化的服务组合框架,该框架兼顾了可用性与动态性的需求,具有一定的实用价值。

**关键词** 流程驱动,语义Web服务,半自动服务组合,QoS
**中图法分类号** TP311 **文献标识码** A

## Process-driven Method of Semi-automatic Semantic Web Service Composition and its Application

XIONG Li-rong YU Hui FAN Jing DONG Tian-yang
(College of Computer, Zhejiang University of Technology, Hangzhou 310014, China)

**Abstract** As the number of Web services increases quickly, how to compose Web services into loosely-coupled system to fulfill flexible business requirements becomes a popular research topic. Traditional process-driven based service composition methods and frameworks predefine the composite process. When the local business-process changes, the whole composite process must be redefined again which lacks of flexibility. This paper introduced a semi-automatic semantic service composition method. The abstract service is used to describe the local dynamic process and will be automated replaced by composite service at execution phase when there is no candidate concrete service suitable for the abstract service. A backward-tree based service composition method was proposed for the replacement of the abstract service and semantic information was applied to improve the recall rate of Web service. Considering the selection of composite processes, the QoS evaluation method was introduced. Based on business process method, this paper proposed a semi-automatic semantic service composition framework. OWL-S technology is used for the description of the dynamic process. Finally, an application of credit evaluation case is shown which presents the availability and flexibility of the framework.

**Keywords** Process-driven, Semantic Web service, Semi-automatic service composition, QoS

## 1 引言

Web服务由于其互操作性和重用性,成为面向服务的体系架构(SOA)的核心支撑技术。如何利用数量众多的Web服务组建松散耦合的企业应用系统以满足不断变化的业务需求成为一个研究热点。

业务流程驱动的服务组合方式一直备受工业界青睐,主要是因为其直接从业务需求的角度建模,可以非常准确地刻画企业用户的真实需求。基于业务流程的服务组合方法,需要预先制定抽象的组合过程。然后在运行时,根据一定的限制条件来动态选取合适的服务并绑定选取的服务。

对于企业应用而言,其核心业务流程很少发生变化,经常变更的只是局部子流程,但局部流程的变更将导致整个流程的重新定义与部署,因此,业务流程的组合方法存在着需要大量的人工参与、自动性不足、无法处理柔性多变的业务流程的局限性。

为创建具有较好适应性的服务组合业务流程,同时满足企业级应用业务需求,本文提出一种业务流程驱动的半自动化语义Web服务组合框架,在业务流程驱动的服务组合基础上引入局部的自动服务组合,即用基于语义描述的抽象服务

到稿日期:2013-02-08 返修日期:2013-03-25 本文受浙江省重点科技创新团队(2009R50009),重大科技专项重大工业项目(2012C11026-2)资助。

熊丽荣(1973—),女,硕士,副教授,CCF会员,主要研究方向为服务计算和软件中间件,E-mail:lilybear@zjut.edu.cn;余 晖(1987—),男,硕士生,主要研究方向为软件中间件技术;范 菁(1969—),女,博士,教授,博士生导师,CCF理事,主要研究方向为虚拟现实及可视化、软件中间件技术,E-mail:fanjing@zjut.edu.cn(通信作者);董天阳(1977—),男,博士,副教授,主要研究方向为软件构件技术。

定义局部的动态子流程。在组合流程执行阶段，通过基于语义回溯树的服务自动组合方法生成组合服务替换抽象服务，并提出一种可配置的组合流程 QoS 计算方法，以解决当存在多个组合服务满足需求时的服务选取问题。该框架保证流程整体正确可靠，在满足商业需求的基础上可以提高其动态能力，具有一定的实用价值。

## 2 相关研究工作

Web 服务组合的方法可以大致分成两类[1]：业务流程驱动的 Web 服务组合和语义驱动的 Web 服务组合。

业务流程驱动的服务组合主要以工作流技术为基础，从具体的业务场景出发，用流程建模语言 BPEL 等对业务流程进行建模，通过为流程中的每一个节点选取并绑定 Web 服务来实现组合服务。流程驱动的服务组合间的交互与执行完全受控于业务流程，灵活性和动态性不足，因此许多学者提出了改进方法。惠普实验室[2]提出了一种基于模板的服务组合，即将流程中的某些节点定义为服务模板，在流程执行前利用服务发现技术将模板替换成具体的 Web 服务。Deng 等人[3]提出一种柔性工作流的服务组合方法，将不确定子流程用“黑盒”表示，在流程执行阶段用基于规则的组合方法替换“黑盒”执行，该方法具有良好的灵活性，但未考虑规则可能存在冲突的问题。针对工作流建模灵活性不足、易于出错的问题，Song 等人[4]提出一种工作流服务组合框架，引入人工智能规划方法生成组合流程，然后将服务选择问题表示成 CSP(Constraint Satisfaction Problem)加以解决。方其庆等人[5]提出一种结合 AI 规划和工作流的动态服务组合方法，其有效克服了采用单种方法所造成的不足。可见将 Web 服务语义化，引入 AI 规划等方法解决工作流灵活性不足的问题是流程驱动服务组合研究的趋势。

基于语义的 Web 服务组合强调 Web 服务自描述的特性，用 OWL-S 等本体论形式化的语言对 Web 服务进行标注，采用形式化推理、AI 规划的方法动态形成组合方案。Narayanan等人[6]提出将 OWL-S 转化成情景演算，使用 Petri 网的可达性分析来确定合成服务，但 Petri 网的可达性分析存在状态空间爆炸的问题。Evren 等人[7]提出了基于 HTN 的规划器 SHOP2 来实现自动 Web 服务组合，在 HTN 规划中，任务分解的概念和服务组合过程的分解相似，很多的控制结构都可以通过 SHOP2 明确表示，但是 SHOP2 存在着假定有确定的动作和静态的环境局限。由于人工智能中包括 AI 规划问题都可以表示为问题状态空间中从起始状态到目标状态的图搜索问题，因此图搜索技术也可用于解决 Web 服务组合问题。如刘峰等人[8]根据服务的输入输出构造服务依赖关系图，提出一种图搜索算法进行服务合成，但该方法搜索复杂度较高，且获得的组合方案不能保证全局最优。邓水光等人[9]提出一种基于完备回溯树的服务自动组合方法，其属于图搜索范畴，通过建立完备回溯树的方式来获取有效的生存路径，最后将路径合成为组合服务，但该方法缺乏对 Web 服务的语义描述，导致服务组合成功率降低。

基于语义的 Web 服务组合和业务流程驱动的服务组合各有其局限性。从实际应用上来看，业务流程驱动的服务组合具有较好的可操作性，但可扩展性与灵活性较差。基于语义的组合方法虽然自动化程度高，但存在着组合方法复杂度较高、组合方案偏离具体业务逻辑的缺点，离实际应用尚有距离。

当存在多个满足抽象服务需求的组合服务时，通常根据服务的 QoS 来选择最佳组合服务。组合服务的 QoS 计算通常将组合流程分解成基本组合流程，如顺序、并行、循环等，通过计算基本组合流程的 QoS 得到组合流程的 QoS[10-12]。Cardoso 等人[10]提出一种预言式的 QoS 计算模型，其引入服务被执行的概率来提高 QoS 精确度，但该模型无法处理复杂流程计算。Dumas 等人[11]提出的 QoS 计算模型能处理复杂的、非结构化的组合流程，但对服务执行的概率考虑不足，没有考虑 QoS 的下界。Zheng 等人[12]提出一种系统化的 QoS 计算方法，其综合考虑 Web 服务的 QoS 以及被执行的概率，使得组合服务的评估更为全面，但该方法无法很好地处理生命周期较短或执行次数较少的 Web 服务。

## 3 业务流程驱动的半自动语义 Web 服务组合方法

流程是一个具有起点和终点的活动序列，它由一个事件引起，然后对信息进行转换，并产生相应的输出。流程驱动的服务组合利用服务组合问题与工作流模型的相似性，利用成熟的建模工具和方法对业务过程进行建模，然后将流程中的活动替换成具体的 Web 服务，从而获得可执行的组合方案。

传统的业务流程驱动的服务组合分成两个阶段：流程建模阶段和流程执行阶段。在流程建模阶段，流程设计人员根据具体业务逻辑，通过可视化的建模工具建立组合流程，流程中的活动被绑定至一个具体的 Web 服务。在流程执行阶段，流程执行引擎载入流程并逐个调用 Web 服务。有些原型系统支持通过服务执行阶段的服务发现来获得满足流程活动节点的需求，但是这些方法都需要组合流程在执行前被完全预定义[2,13]。有些方法使用执行期的服务组合来获得服务，但未考虑语义和 QoS，无法获取同时满足抽象服务功能性和非功能性需求的最佳组合服务[3]。

在本文提出的业务流程驱动的半自动语义 Web 服务组合中，流程建模阶段如图 1 所示，由建模人员利用建模工具建立组合流程。流程中的每一个活动被绑定到一个具体的 Web 服务上，或为其建立抽象服务。

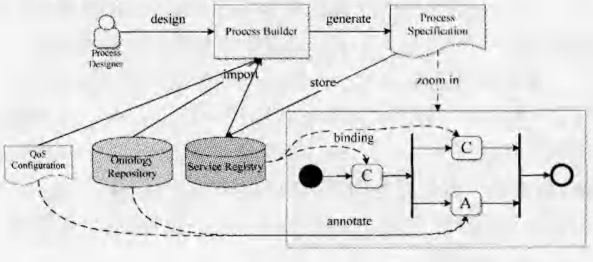


图 1 组合流程建模阶段

抽象服务是组合流程中的“占位符”，当不存在单个具体服务能满足业务流程的需求，或存在不能被预定义的动态子流程时，通过抽象服务的自动组合替换成组合服务执行。

在组合流程执行阶段(见图 2)，执行引擎接受服务请求者(通常是一个外部应用程序)的一次请求，通过服务名称查找组合服务，并载入组合流程调用执行，若组合流程中存在抽象服务，则调用 SBT-ASC Composer 进行自动的服务组合以获取满足抽象服务功能性需求且 QoS 最佳的服务。SBT-

ASC Composer 实现了基于语义回溯树的抽象服务自动组合算法，将在第 4 节详细描述。

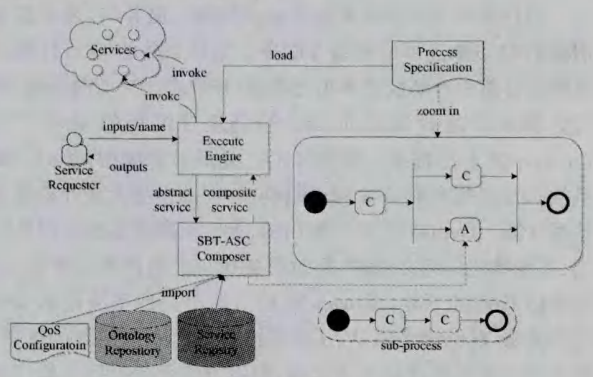


图 2　组合流程执行阶段

该半自动服务组合方法的优点在于流程建模人员在建模时无需预定义完整流程，抽象服务封装了组合流程中的不确定、动态部分。而抽象服务的自动组合将被推迟到流程执行阶段，以获得满足抽象服务功能性和非功能性需求的组合服务。

## 4　形式化描述

为了进一步详细说明抽象服务的自动组合，下面给出相关概念的形式化定义和组合流程的 QoS 计算方法。

### 4.1　形式化定义

**定义 1**(原子服务)　一个原子服务是不可分割、可被直接调用执行的 Web 服务，可被描述成一个七元组 $S_A=\langle n,\eta,\lambda,I,O,\omega,Q\rangle$，其中：$n$ 为服务名称，是服务的唯一标识；$\eta$ 为服务功能性描述；$\lambda$ 为服务提供者信息；$I=\{i_1,i_2,\cdots,i_m\}$ 为服务的输入集合；$O=\{o_1,o_2,\cdots,o_n\}$ 为服务的输出集合；$\omega$ 为服务的访问节点，描述了服务的调用方式，如 OWL-S 中的 Service Grounding，WSDL 中的 portType；$Q=\{q_1,q_2,\cdots,q_k\}$ 为服务质量的集合，如服务执行时间、费用等。

**定义 2**(抽象服务)　一个抽象 Web 服务是一个六元组 $S_R=\langle n,\eta,I,O,\alpha,W_Q\rangle$，其中：$n$ 为服务名称；$\eta$ 为服务功能性描述；$I=\{i_1,i_2,\cdots,i_m\}$ 为用户能提供的输入集合；$O=\{o_1,o_2,\cdots,o_n\}$ 为用户期待的输出集合；$\alpha$ 为用户最低服务相似度需求；$W_Q=\{\omega_{q1},\omega_{q2},\cdots,\omega_{qk}\}$ 为服务质量参数的权重需求。

**定义 3**(组合服务)　一个组合服务是一个五元组 $S_C=\langle n,I,O,\rho,Q\rangle$，其中：$n$ 为服务名称；$I=\{i_1,i_2,\cdots,i_m\}$ 为服务的输入集合；$O=\{o_1,o_2,\cdots,o_n\}$ 为服务的输出集合；$\rho$ 为组合流程，描述了组合服务内部的执行逻辑；$Q=\{q_1,q_2,\cdots,q_k\}$ 为组合服务的服务质量，由组合流程 $\rho$ 唯一决定，即存在函数 $q_i=f_{q_i}(\rho),q_i\in Q$。

一个组合服务是一个“黑盒”，是对一段业务流程 $\rho$ 的封装，流程 $\rho$ 对外不可见。

**定义 4**(数据绑定)　一个数据绑定是一个五元组：$b=\langle S_s,S_d,p_s,p_d,\alpha\rangle$，其中：$S_s$ 为源服务；$S_d$ 为目标服务；$p_s$ 为源服务 $S_s$ 中的一个参数；$p_d$ 为目标服务 $S_d$ 中的一个参数；$\alpha=sim(p_s:C,p_d:C)$ 为 $p_s$ 和 $p_d$ 之间的概念相似度。

数据绑定描述了服务间的数据流可以分成两类：一类是原子服务之间的数据绑定，如图 3(a)所示，服务 $S_d$ 的输入参数 $i$ 接受 $S_s$ 的输出参数 $o$ 作为其数据来源。原子服务间的数据绑定也隐式地指出了目标服务 $S_d$ 的执行依赖源服务 $S_s$ 的执行完成。另一类是抽象服务与原子服务之间的数据绑定，如图 3(b)、(c)所示，表明了抽象服务与原子服务间的输入参数的绑定依赖关系或输出参数的绑定依赖关系，表明原子服务的输入输出语义上满足抽象服务的输入和输出需求。

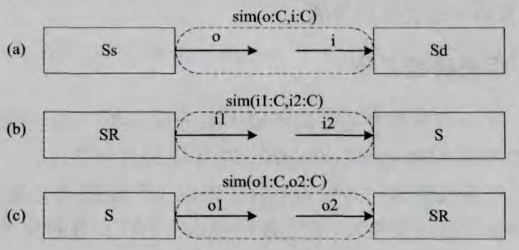


图 3　数据绑定示例

**定义 5**(组合流程)　组合流程由服务控制流和数据流组成，为了形式化描述服务控制流，引入以下 4 种流程控制符号(CF，Control Flow)：

(1)$a>b$：流程 $a$、$b$ 顺序执行。

(2)$a+b$：流程 $a$、$b$ 并发执行。

(3)$e?a:b$：布尔表达式 $e$ 为真，则执行流程 $a$，否则执行流程 $b$。

(4)$e*a$：循环执行 $a$ 直至布尔表达式 $e$ 为假。

应用以上流程控制结构，即可构造组合服务控制流。如 $a>((e_1*b)+(e_2?c:d))$ 的执行过程如图 4 所示。

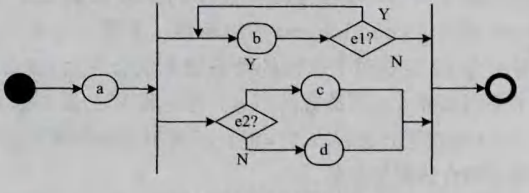


图 4　组合流程示例

一条组合流程可以描述为 $\rho=S\vee\langle CF,D,B\rangle$，即组合流程既可退化为单个原子服务，也可以为嵌套的流程集合，其中 $D=\{\rho_1,\rho_2,\cdots,\rho_n\}$ 是 CF 调用的子流程集合，$B=\{b_1,b_2,\cdots,b_m\}$ 是服务绑定的集合，描述了服务间的消息传递。

### 4.2　组合流程的 QoS

组合流程的 QoS 计算基于 4 种基本流程控制结构 CF 的 QoS 计算公式。假设 $f_{q_i}^{\rho.CF}$ 为 $q_i$ 的聚合函数，其中 $CF\in\{>,+,?,*\}$，$q_i$ 为第 $i$ 维 QoS 属性，则

$$f_{q_i}(\rho)=\begin{cases}q_i, & \text{if } \rho=S\wedge q_i\in S.Q\\ f_{q_i}^{\rho.CF}(\rho.D), & \text{if } \rho\neq S\end{cases}$$

为组合流程 $\rho$ 的 QoS 计算公式，任意复杂的组合流程的 QoS 都可递归应用该公式计算。

表 1　组合流程 $\rho$ 的时间 $T$、费用 $C$ 的聚合公式

| CF | $f_T^{\rho.CF}$ | $f_C^{\rho.CF}$ |
|---|---|---|
| $>$ | $\sum_{\rho_i\in\rho.D}T(\rho_i)$ | $\sum_{\rho_i\in\rho.D}C(\rho_i)$ |
| $+$ | $\max_{\rho_i\in\rho.D}(T(\rho_i))$ | $\sum_{\rho_i\in\rho.D}C(\rho_i)$ |
| $?$ | $\sum_{\rho_i\in\rho.D}\lambda*T(\rho_i)$ | $\sum_{\rho_i\in\rho.D}\lambda*C(\rho_i)$ |
| $*$ | $n*T(\rho_i),\rho_i\in\rho.D$ | $n*C(\rho_i),\rho_i\in\rho.D$ |

例如，假设组合服务的执行时间、费用的 4 种基本聚合公式如表 1 所列($\lambda$ 表示流程 $\rho_i$ 被执行的概率，$n$ 为 $\rho_i$ 的平均执

行次数),那么图4所示组合流程的执行时间为 $T(\rho)=T(a)+\mathrm{Max}\{(n*T(b),(\lambda_1 T(c)+\lambda_2 T(d)))\}$,费用为 $C(\rho)=C(a)+(n*T(b)+(\lambda_1 T(c)+\lambda_2 T(d)))$,其中 $n$ 为子流程 $b$ 的执行次数,$\lambda_1$、$\lambda_2$ 分别为子流程 $c$、$d$ 的执行概率。

组合平台提供者只需指定QoS属性指标的4种基本聚合公式,所有复杂流程的QoS都可递归计算得出,这种可配置的QoS计算模型增加了QoS指标体系的可扩展性。

## 5 基于语义的抽象服务自动组合方法

抽象服务是对业务流程中动态子流程的一种抽象,需要在执行期自动合成。基于抽象服务的语义服务组合自动合成问题可以描述为:给定抽象服务,找到能匹配其输入输出的组合服务集合,并按照抽象服务权重需求选取集合中QoS最高的组合服务的过程。

本文在文献[9]的回溯树Web服务组合算法的基础上,为服务参数添加语义信息,提出一种基于语义回溯树的改进的Web服务组合方法SBT-ASC。

**定义6**(语义回溯树) 一棵语义回溯树是一个三元组:$SBT=\langle root,V,E\rangle$,其中:

(1)$root=\{o\}$是回溯树的根节点,$o\in S_R.O$;

(2)$V=\{v_1,v_2,\cdots,v_n\}$是回溯树节点的集合,$V$是所有语义Web服务输入输出参数的子集;

(3)$E=V\times V\times Set_B$是回溯树边的集合,其中$Set_B$是数据绑定的集合。每一条边$e=\langle v_s,v_d,b\rangle$都是从树的底层节点$v_s$指向高层节点$v_d$的边,$b$是边相关联的数据绑定。

每一棵回溯树都描述了对于抽象服务$S_R$的一个输出$o$,所有生成$o$的路径的集合。回溯树边中的数据绑定集合存储了服务之间参数的语义匹配信息。

SBT-ASC的Web组合方法分成3个步骤:

Step1 建立可用的数据绑定集合

首先遍历服务库和抽象服务,根据抽象服务中最低语义相似度需求,建立可用的数据绑定集合。如下所示,FIND-AVAILABLE-DATABINDINGS函数用于查找服务库$SR$与抽象服务$S_R$之间的可用数据绑定集合。函数分成两部分,1—6步骤计算原子Web服务与抽象Web服务之间的数据绑定集合,7—10步骤计算原子Web服务之间的数据绑定集合。

```
Function:FIND-AVAILABLE-DATABINDINGS
Input:The abstract service request SR,service repository SR
Output:Available data binding set BA
1. for each S∈SR{
2.   if ∃i∈SR.I,i'∈S.I,let β=sim(i:C,i':C)≧SR.α
3.     create b=⟨SR,S,i,i',β⟩,BA.add(b);
4.   if ∃o∈S,o'∈SR,let β=sim(o:C,o':C)≧SR.α
5.     create b=⟨S,SR,o,o',β⟩,BA.add(b);
6. }
7. for each S1,S2∈SR and S1≠S2{
8.   if ∃o∈S1,i∈S2,let β=sim(o:C,i:C)≧SR.α
9.     create b=⟨S1,S2,o,i,β⟩,BA.add(b);
10. }
11. return BA;
```

图5为可用数据绑定集合的示例,若存在抽象服务$S_R$的输入、输出集合为$(\{i\},\{o\})$,服务库$SR=\{S_1,S_2,S_3,S_4\}$,其中$S_1$、$S_2$、$S_3$、$S_4$的输入输出集合分别为$(\{i_1\},\{o_1\})$,$(\{i_2\},\{o_2\})$,$(\{i_3,i_5\},\{o_3\})$,$(\{i_4\},\{o_4\})$,遍历$S_R$、$SR$,若存在$\alpha_1=\mathrm{sim}(o_1:C,o:C)$、$\alpha_2=\mathrm{sim}(o_2:C,o:C)$、$\alpha_3=\mathrm{sim}(o_3:C,o:C)$、$\alpha_4=\mathrm{sim}(i:C,i_1:C)$、$\alpha_5=\mathrm{sim}(i:C,i_4:C)$、$\alpha_6=\mathrm{sim}(i:C,i_3:C)$,$\alpha_7=\mathrm{sim}(o_4:C,i_2:C)$大于$S_R.\alpha$,则得到的可用的数据绑定集合如图5所示。

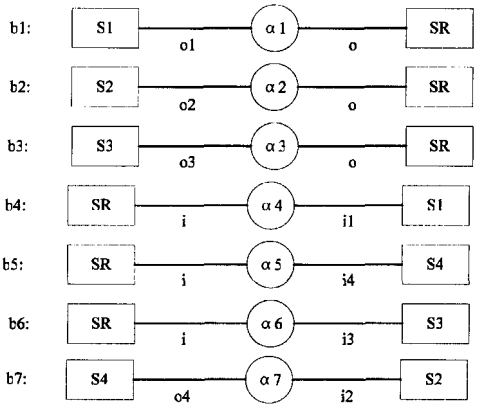


图5 服务库中可用的数据绑定集合

Step2 为抽象服务中每一个输出构造语义回溯树

函数SBT-CONSTRUCTOR接受数据绑定$B_A$和抽象服务的一个输出$o$作为函数的输入,返回一棵以$\{o\}$为根节点的语义回溯树作为输出。SBT-CONSTRUCTOR从根节点$\{o\}$开始自顶向下逐层构造语义回溯树,队列*tQueue*用于存放待处理的叶子节点集合。对于队列*tQueue*中的每一个子目标*tNode*,步骤10查询其是否是抽象服务输入集合的子集,若是则跳出该次迭代,否则继续构造SBT。构造SBT的过程中,条件$b.S_S.I\cap(\bigcup_{e\in l}e.b.o_d)==\emptyset$避免了在数据绑定集合$B_A$中的循环搜索;当发现$B_A$中没有数据绑定能产生子目标$o'$时,则移除$o'$所在的节点及其对应的生成路径分支(步骤25—28),这样保证了所有从叶子节点到根节点的路径都为目标$\{o\}$有效的生成路径。

```
Function:SBT-CONSTRUCTOR
Input:Data binding set BA,the output o∈SR.O
Output:A SBT tree to*=⟨root,V,E⟩
1. create a new SBT tree to*=⟨root,V,E⟩;
2. create a new FIFO queue tQueue;
3. create a new SBT node tNode;
4. create a boolean variable flag;
5. set root={o};
6. V.add({o});
7. tQueue.enquque(root);//根节点入队
8. while(!tQueue.isEmpty()){
9.   tNode=tQueue.dequque();
10.  if tNode⊆SR.I continue;
11.  set O^tNode=tNode-(tNode∩SR.I);
12.  for each o'∈O^tNode{
13.  set l=⟨e1,e2,⋯,ek⟩ as the path from tNodeto the root;
14.    set flag=false;
15.    for each b∈BA{
16.      if(b.od==o'&&b.SS.I∩(⋃e∈l e.b.od)==∅){
17.        set flag=true;
18.        create a node cNode=(tNode-(Ss.O∩tNode))∪b.Ss.I;
```

```
19.          tQueue. enqueue(cNode);
20.          V. add(cNode);
21.          create an edge e=⟨cNode,tNode,b⟩;
22.          E. add(e);
23.     } }
24.   if(!flag){//当不存在一个能满足 o′的数据绑定时
25.     for i=1 to l. length{
26.       E. remove(e_i);
27.       V. remove(e_i. v_s);
28.       if(e_i. v_d. hasChildren()) break;
29. }}}}
30. return t_o^* ;
```

图 6 是以图 5 所示数据绑定集合为例的语义回溯树的构造过程，(a)为初始情况，$o$ 为抽象服务 $SR$ 的一个输出；如(b)所示，可用数据绑定集合中有 $b1$、$b2$、$b3$ 这 3 条数据绑定能输出 $o$；如(c)所示，可用数据绑定集合中分别存在 $b4$、$b7$ 能输出 $i1$、$i2$，且$\{i\}$是抽象服务 $SR$ 输入集合的子集，因此 $b4-b1$ 为目标 $o$ 的有效生成路径；如图(d)所示，数据绑定集合中不存在能满足$\{i3,i5\}$的数据绑定，因此移除$\{i3,i5\}$节点和 $b3$ 所在的生成路径，而数据绑定集合中存在 $b5$ 生成$\{i4\}$，且$\{i\}$是抽象服务 $SR$ 输入集合的子集，因此 $b5-b7-b2$ 也是目标 $o$ 的有效生成路径。

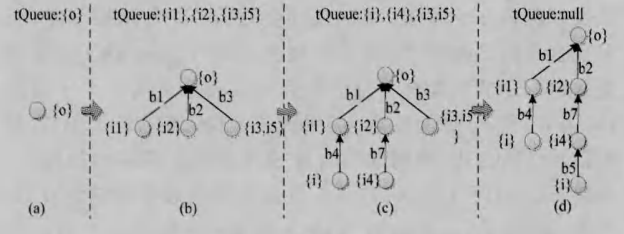


图 6 SBT 的构造过程

Step3 合成 QoS 最佳的组合服务返回

语义回溯树的每一条从叶子节点到根节点的路径都是从抽象服务的输入到抽象服务输出中一个输出的生成路径。若回溯树中存在多条生成路径，则选取 QoS 最高的生成路径进行合成。提取其数据绑定的集合即可得到一条顺序执行的组合流程($S_1>S_2>\cdots>S_n$)，应用组合流程计算公式计算其 QoS($f_{q_i}(\rho)$)并按抽象服务的权重计算($\sum W_{q_i} * f_{q_i}(\rho)$)QoS 最高的服务操作链合成即可得到组合流程的集合。图 6(d)所示回溯树合成的组合服务如图 7 所示，布尔表达式 $e$ 比较两条执行路径的 QoS 大小，为真时执行服务 S1，否则执行顺序流程($S_4>S_2$)。

$$e=\sum_{q_i\in S_1.Q} W_{q_i} * q_i > \sum W_{q_i} * f_{q_i}^{>}(S_4,S_2)$$

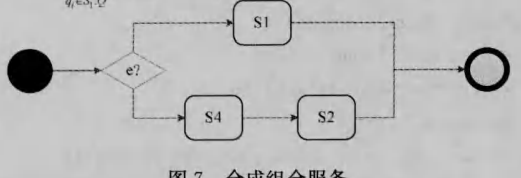


图 7 合成组合服务

与手动地自上而下建立组合流程不同，SBT-ASC 方法查询语义 Web 服务之间是否存在着可绑定关系，自底向上地构造组合流程，这是一个完全自动的过程，提高了固定流程的灵活性。

## 6 半自动 Web 服务组合框架的设计和案例仿真

### 6.1 服务组合框架的设计

考虑到服务注册库中大部分的 Web 服务是基于 WSDL 描述的，我们采用本体标注语言 OWL-S 对基于 WSDL 的 Web 服务进行语义标注，并对组合流程进行建模。OWL-S 上层本体由 3 部分组成：Service Profile 描述了服务的轮廓，提供了服务的输入输出、描述、服务提供者等信息，其中服务的每个输入输出都会关联到一个 OWL-DL 概念上。Service Model 描述了服务的内部执行流程，其中的 Atomic Process 描述了一个原子流程，对应 WSDL 中的一个 Operation；Composite Process 描述了一个组合流程，提供了 Sequence、Split-Join 等流程控制结构(ControlConstruct)用于描述组合流程的控制流和 Binding 节点用于描述服务的数据流。Service Grounding 用于描述 OWL-S 与 WSDL 的映射关系，每一次 OWL-S Web 服务的执行实际都是对于底层 WSDL 描述的 Web 服务的调用，如图 8 所示。

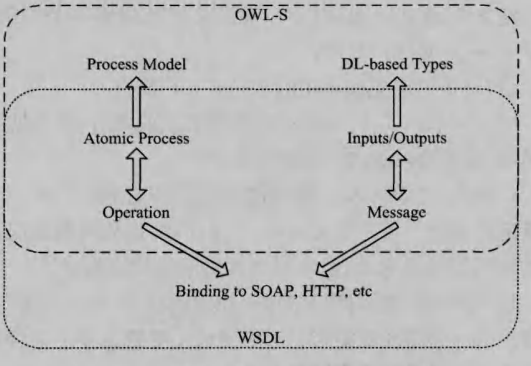


图 8 OWL-S 与 WSDL 的映射关系[14]

业务流程基于可视化的流程建模环境，我们将抽象服务表示成只含 Service Profile 的 OWL-S 描述，将组合流程的控制流映射成 OWL-S 的 ControlConstruct 元素，将数据绑定映射成 Binding 元素。

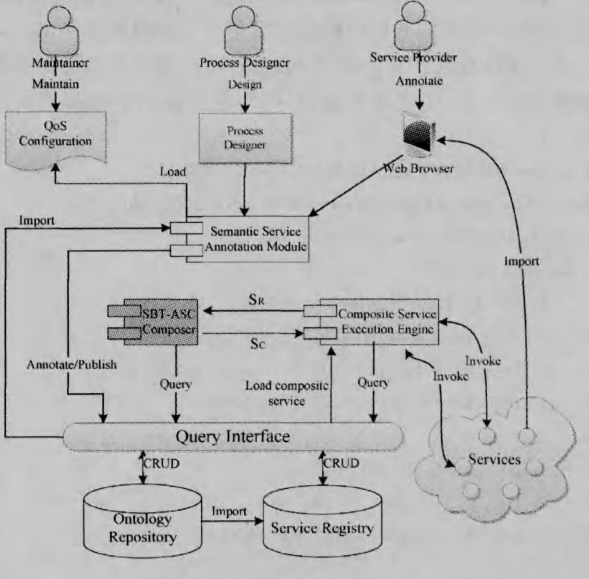


图 9 服务组合系统的体系结构

半自动服务组合框架的体系结构如图 9 所示。半自动服务组合框架的实现分为 3 个层次：数据持久层提供了服务数

据的持久化机制，并提供透明的查询接口为上层服务；核心逻辑层实现了组合服务的执行、抽象服务的自动组合机制，是整个半自动服务组合框架的自动部分；用户交互层提供了用户交互的入口，是半自动组合框架的手动部分。组合框架涉及3种用户角色：服务提供者发布并标注原子 Web 服务；组合流程设计者设计组合流程；组合平台提供者维护 QoS 指标体系。

(1)数据持久层

数据持久层是组合框架中进行数据持久化的部分，存储了 OWL-S 描述的语义 Web 服务描述以及对应的 WSDL 描述、基于 OWL-DL 的相关本体。采用 Apache Tomcat 作为 Web 容器、Axis2 作为 Web 服务引擎、Jena 提供语义支持和 MySql 作为数据库存储。主要包含以下 3 个模块：

• Service Registry：注册了所有 OWL-S 标注的语义 Web 服务，包括组合服务和原子服务。组合服务通过服务名和版本号唯一确定。

• Ontology Repository：存储了所有语义 Web 服务所需的 OWL-DL 本体，提供推理、标注等语义支持接口。

• Query Interface：该模块基于外观(Facade)模式，对访问持久层的上层模块提供一致透明的接口，并提供了语义缓存机制，即当服务注册中心的服务数量增多时，FIND-AVAILABLE-DATABINDINGS的时间复杂度为 $O(n^2)$，会非常耗时。在数据库启动时，Query Interface 就会对所有语义 Web 服务进行遍历并将其中存在的参数语义相似度信息进行缓存。这样进行 SBT-ASC 自动组合时只需遍历一次服务注册中心即可，大大减少了组合时的查询时间。

(2)核心逻辑层

核心逻辑层是服务组合框架的关键部分，包含以下两个模块：

• SBT-ASC Composer：实现了基于语义回溯树的改进的 Web 服务组合方法 SBT-ASC，对于接收的每一个抽象服务请求，进行抽象服务的自动合成。

• Service Execution Engine：对服务调用者发起的每次组合服务调用，Service Execution Engine 调用具体服务(WSDL 描述的服务)进行执行，并在运行期调用 SBT-ASC Composer 进行局部的抽象服务自动合成。

(3)用户交互层

用户交互层提供了不同用户与底层进行交互的接口，包含 3 个模块：

• Process Designer：为流程设计人员提供一个可视化的建模环境。流程设计人员用拖拽的方式可视化地建模并生成 OWL-S 描述的组合流程。

Semantic Service Annotation Module：用户交互层的核心，包含以下功能：(a)对 Web 服务进行语义标注；(b)读取数据库中的本体和语义 Web 服务提供给建模人员和服务提供者；(c)将数据持久化至相应数据库。

• QoS Configuration：提供可配置的 QoS 属性列表及其基本计算公式。

### 6.2 案例仿真

信用评估是指评估机构利用专家判断、数学分析等方法对个人和企业履约能力和信誉程度进行评价的方法。其中，企业信用评估作为重要组成部分日益受到政府部门的重视，银行、保险等金融机构都建立了各自的信用评估系统。

下面以企业信用评估业务流程为目标应用场景，给出半自动 Web 服务组合框架的仿真案例。该仿真案例中的 Web 服务来源于《信用服务模型的可视化集成开发环境》和项目《基于 Web Service 的信用构件库》[15-17]。这些构件以 Web 服务的形式存在，分成两大类：信用领域专有构件服务和通用构件服务，如表 2 所列。

表 2 信用领域专有构件服务和通用构件服务

| | | |
|---|---|---|
| 信用领域专用构件服务 | 信用评估模型构件 | 制造业上市公司 z 模型构件、制造业非上市公司的 z 模型构件、非制造业公司的 z 模型构件、理查德-托夫勒 z 计分模型构件、张玲 Z 记分模型构件、巴萨得模型构件、营运资产分析模型构件、CHESSER 信用评分模型构件、基于加权平方和法的外贸信用评级模型构件、CreditRisk+模型构件 |
| | 信用评估建模构件 | 通用加权平方和建模构件、层次分析建模构件、模糊层次分析建模构件、BP 神经网络建模构件 |
| | 模型检验构件 | ROC 曲线检验构件、K-S 统计量检验构件 |
| | 指标计算构件 | 个人资质评价指标计算构件、资产评价指标计算构件、支出情况指标计算构件、历史信用指标计算构件 |
| | 指标量化构件 | 常用指标量化方法 |
| | 工作流构件 | 信用卡申请工作流构件、通用信用评估工作流构件、报销工作流构件 |
| 通用构件服务 | 统计描述计算构件 | 相关系数计算构件、协方差的计算构件、分位数计算构件、峰度计算构件、众数计算构件、全距计算构件、偏度计算构件、标准化 z 分数计算构件、方差标准差计算构件、频率计算构件 |
| | 多元统计分析构件 | 因子分析构件、主成分分析、聚类分析构件 |
| | 表示层构件 | 折线图显示构件、饼图显示构件、柱状图显示构件、表格显示构件、树型显示构件、表排序标签构件 |

一个企业信用评级业务流程包含以下几个顺序活动："提交企业数据信息"、"预处理数据信息"、"选择评级模型"、"执行评估得到评估结果"、"发布评估报告"。这些活动中，有些是相对固定的流程，如"提交企业数据信息"活动，对于所有企业信用评级系统来说都是必需的；有些活动则是相对柔性的业务，如"选择评级模型"活动，根据企业的性质(如公司所处行业、上市与否)等上下文信息选择不同的评估服务。

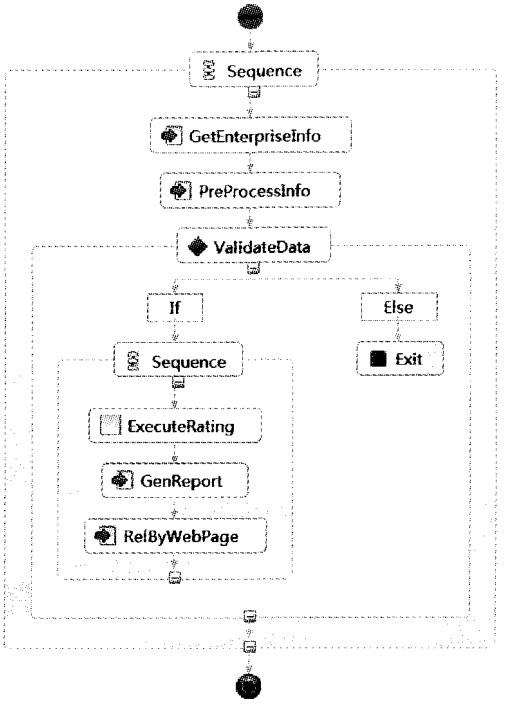


图 10 EnterpriseCreditRating 流程定义

企业信用评级组合服务“EnterpriseCreditRating”的流程定义如图 10 所示，其中 ExecuteRating 是抽象服务，需要根据企业信息来选择评估模型，属于局部动态子流程，需要在运行时动态合成。我们选取服务执行时间和服务价格为 QoS 的指标，对其权重设置为(0.8,0.2)。

其中，抽象服务 ExecuteRating 的 OWL-S 描述片段如图 11 所示。我们选择一组评估数据来进行测试(见图 12)，得到了一个评估结果(见图 13)。其中通过抽象服务自动合成的组合服务流程 ExecuteRating 的片段如图 14 所示。组合服务首先并行调用了 CalcWorkingCapital 和 CalcNetProfit 计算营运资本和净利润，然后调用了非制造业 z 模型构件服务计算得到评估结果。

```
〈profile:Profile rdf:ID="ExecuteRatingProfile"〉
  〈service:isPresentedBy rdf:resources="ExecuteRatingService"/〉
  〈profile:serviceName xml:lang="en"〉
    executeRatingService
  〈/profile:serviceName〉
  〈profile:textDescription xml:lang="en"〉
    execute a enterprise credit rating.
  〈/profile:textDescription〉
  〈profile:hasInput rdf:resource="#TotalAssets"/〉
  〈profile:hasInput rdf:resource="#TotalProfits"/〉
  〈profile:hasInput rdf:resource="#CurrentAssets"/〉
  〈profile:hasInput rdf:resource="#CurrentLiabilities"/〉
  〈profile:hasInput rdf:resource="#TotalLiabilities"/〉
  〈profile:hasInput rdf:resource="#OwnerEquity"/〉
  〈profile:hasInput rdf:resource="#InterestExpense"/〉
  〈profile:hasInput rdf:resource="#DepreciationCost"/〉
  〈profile:hasOutput rdf:resource="#CreditRatingResult"/〉
〈profile:Profile〉
```

图 11　抽象服务的 OWL-S 描述片段

| | |
|---|---|
| 资产总额: | 30000000.00 |
| 利润总额: | 12000000.00 |
| 流动资产: | 8000000.00 |
| 流动负债: | 500000.00 |
| 未分配利润: | 3000000.00 |
| 负债总额: | 5000000.00 |
| 所有者权益: | 3000000.00 |
| 折旧费: | 400000.00 |
| 利息支出: | 500000.00 |

图 12　测试数据

调用非制造业Z-Score评级模型计算，得到评估值为8.7356;评估级别为AAA级。

企业的Z-Score与信用等级的对应关系如下：

| | |
|---|---|
| AAA | >=8.15 |
| AA | >=7.30 |
| A | >=6.65 |
| BBB | >=5.85 |
| BB | >=4.95 |
| B | >=4.15 |
| CCC | >=2.50 |
| D | >=0 |

图 13　评估结果

```
〈process:CompositeProcess rdf:ID="ExecuteRatingProcess"〉
  〈process:hasInput rdf:resource="#TotalAssets"/〉
  〈process:hasInput rdf:resource="#TotalProfits"/〉
  〈process:hasInput rdf:resource="#CurrentProfits"/〉
  〈process:hasInput rdf:resource="#CurrentLiabilities"/〉
  〈process:hasInput rdf:resource="#UndistributedProfits"/〉
  〈process:hasInput rdf:resource="#TotalLiabilities"/〉
  〈process:hasInput rdf:resource="#OwnerEquity"/〉
  〈process:hasInput rdf:resource="#InterestExpense"/〉
  〈process:hasInput rdf:resource="#DepreciationCost"/〉
  〈process:hasOutput rdf:resource="#CreditRatingScroe"/〉
  〈process:composeOf〉
    〈process:Sequence〉
      〈process:components〉
        〈process:ControlConstructList〉
          〈list:first〉
            〈process:Split-Join〉
              〈process:components〉
                〈process:Perform rdf:about="http://127.0.0.
                  1:8080/services/statistics/CalcWorkingCapi-
                  tal"/〉
                〈process:Perform rdf:about="http://127.0.0.
                  1:8080/services/statistics/CalcNetProfits"/〉
              〈/process:components〉
            〈/process:Split-Join〉
          〈/list:first〉
        〈/list:rest〉
          〈process:Perform rdf:about="http://127.0.0.1:
            8080/services/credit/ZValueForNonManufacture"/〉
        〈/list:rest〉
      〈/process:ControlConstuctList〉
    〈/process:components〉
  〈/process:Sequence〉
〈/process:composeOf〉
〈process:Perform rdf:about="http://127.0.0.1:8080/services/
 credit/ZValueForNonManufacture"〉
  〈process:hasDataFrom〉
    〈process:InputBinding〉
      〈process:valueSource〉
```

图 14　生成的组合服务描述片段

**结束语**　针对传统业务流程驱动的服务组合动态性不足的问题，本文提出一种业务流程驱动的语义 Web 服务半自动组合方法，即将局部动态流程用抽象服务描述，并在流程执行阶段采用基于语义回溯树的自动服务组合得到满足抽象服务需求的组合服务。本文还给出了成功应用组合方法的仿真案例，该案例表明该半自动服务组合方法能在保证整体流程正确的同时兼具灵活性。

本文提出的组合方法缺乏一定的容错性和事务机制，下一步的工作是研究在组合算法中加入容错和事务机制以适应更加复杂的网络环境。

## 参 考 文 献


[1] 倪晚成，刘连臣，吴澄. Web 服务组合方法综述[J]. 计算机工程，2008,34(4):79-81

[2] Casati F, Ilnicki S, Jin L J, et al. Adaptive and dynamic service composition in eFlow[C]// Advanced Information Systems Engineering. Springer Berlin/Heidelberg, 2000:13-31

[3] Deng S, Li Y, Xia H, et al. Exploring the flexible workflow technology to automate service composition[J]. The Semantic Web-ASWC 2006, 2006, 4185:444-458


(下转第 196 页)

监控、自适应决策和演化实施提供了良好的基础；(3)将监控逻辑与业务功能逻辑的实现相分开，通过采用独立于应用程序的外部单元(监控单元、诊断单元以及自适应重配置修复单元)来实现系统的运行时监控和诊断，更利于系统的维护和管理，同时也更符合软件复用的思想。

**结束语** 本文提出了一个面向自适应重配置的运行时监控方法。本方法与传统的模型检测方法的不同之处在于它是一种动态的监控和诊断方法。本方法以目标模型为基础，通过对系统运行时的功能性行为进行监测，诊断分析软件系统的行为是否满足需求规约，并根据分析结果对系统进行自适应调整，以保证系统运行的正确性，可有效地提高软件系统的可信性。本文的主要贡献是：从监控系统的监控事件的定义，到如何生成和编织监控代码，再到如何诊断分析监控数据，并响应监控结果，进行自适应调整，给出了一个运行时监控、诊断和自适应重配置的整体概念框架。我们将在此基础上进一步开展相关支撑工具的实现工作。

## 参 考 文 献


[1] Kephart J O, Chess D M. The Vision of Autonomic Computing [J]. Computer, 2003, 36(1): 41-50

[2] van Lamsweerde A. Goal-Oriented Requirements Engineering: A Guided Tour[C]// Proceedings RE'01, 5th IEEE International Symposium on Requirements Engineering. Toronto, 2001: 249-263

[3] van Lamsweerde A, Letier E. Handling Obstacles in Goal-oriented Requirements Engineering[J]. IEEE Transactions on Software Engineering, 2000, 26(10): 978-1005

[4] Clarke E M, Grumberg O, Peled D A. Model Checking[M]. The MIT Press, 2001

[5] Taher L, Basha R, Khatib H E. Qos Information & Computation (QoS-IC) Framework for QoS-based Discovery of Web services [J]. The European Journal for the Informatics Professional, 2005, 6(4)

[6] Mani A, Nagarajan A. Understanding quality of service for Web services [OL]. http://www-128. ibm. com/developerworks/webservices/library/ws-quality. html

[7] Dalpiaz F, Giorgini P, Mylopoulos J. An architecture for requirements-driven self-reconfiguration [C] // Proceedings, CAiSE, Volume 5565 of LNCS. Springer, 2009: 246-260

[8] Lee I, Ben-Abdallah H, Kannan S, et al. A monitoring and checking framework for run-time correctness assurance[C] // Proceedings of the 1998 Korea-U. S. Technical Conference on Strategic Technologies. 1998

[9] Li Zheng, Jin Yan, Han Jun. A Runtime Monitoring and Validation Framework for Web Service Interactions[C] // Proceedings of ASWEC'06 Proceedings of the Australian Software Engineering Conference. IEEE Computer Society, 2006: 70-79

[10] Chen Feng, Rosu G. MOP: An Efficient and Generic Runtime Verification Framework [R]. Technical report UIUCDCS-R-2007-2836. March 2007

[11] Simmonds J, Ben-David S, Chechik M. Monitoring and Recovery of Web Service Applications [C] // The Smart Internet 2010. Springer, 2010: 250-288

[12] Amin A, Colman A, Grunske L. Using Automated Control Charts for the Runtime Evaluation of QoS Attributes[C] // Proceedings of the 13ht IEEE International High Assurance Systems Engineering Symposium. IEEE Computer Society, 2011: 299-306

[13] Delgado N, Gates A Q, Roach S. A Taxonomy and Catalog of Runtime Software-Fault Monitoring Tools [J]. IEEE Transactions on Software Engineering, 2004, 30(12): 859-872

[14] Avgustinov P, Bodden E, Hajiyev E, et al. Aspect for trace monitoring[C] // Proc of Formal Approaches to Testing Systems and Runtime Verification(FATES/RV 2006). LNCS 4262, 2006: 20-39

[15] Bodden E. A lightweight LTL runtime verification tool for Java [C] // Proc of OOPSLA. 2004: 306-307


(上接第 180 页)


[4] Song X, Dou W, Chen J. A workflow framework for intelligent service composition[J]. Future Generation Computer Systems, 2011, 27(5): 627-636

[5] 方其庆，彭晓明，刘庆华，等. 结合 AI 规划和工作流的动态服务组合框架研究[J]. 计算机科学，2009，36(9)：110-114

[6] Narayanan S, McIlraith S A. Simulation, verification and automated composition of Web services[C] // Proceedings of the 11th international conference on World Wide Web. ACM, 2002: 77-88

[7] Evren D W, Wu D, Sirin E, et al. Automatic Web services composition using shop2[C] // Workshop on Planning for Web Services. 2003

[8] 刘峰，谭庆平，杨艳萍. 基于图论的 Web 服务合成算法[J]. 华中科技大学学报：自然科学版，2005，33：202-204

[9] 邓水光，吴健，李莹，等. 基于回溯树的 Web 服务自动组合[J]. 软件学报，2007，18(8)：1896-1910

[10] Cardoso J, Sheth A, Miller J, et al. Quality of service for workflows and Web service processes[J]. Web Semantics: Science, Services and Agents on the World Wide Web, 2004, 1(3): 281-308

[11] Dumas M, García-Bañuelos L, Polyvyanyy A, et al. Aggregate quality of service computation for composite services[J]. Service-Oriented Computing, 2010, 6470: 213-227

[12] Zheng H, Zhao W, Yang J, et al. Qos analysis for Web service compositions with complex structures[J]. IEEE Transactions on Service Computing, 2013, 6(3): 373-386

[13] Sirin E, Hendler J, Parsia B. Semi-automatic composition of Web services using semantic descriptions[C] // Web Services: Modeling, Architecture and Infrastructure workshop in ICEIS. 2003

[14] Martin D, Burstein M, Hobbs J, et al. OWL-S: Semantic markup for Web services[J]. W3C Member submission, 2004, 22

[15] 范菁，刘韬，熊丽荣. 信用构件的刻面分类及检索方法研究[J]. 计算机系统应用，2008(6)

[16] 范菁，杨冰，熊丽荣. 基于功能语义的构件描述和检索研究[J]. 计算机系统应用，2009(4)：26-31

[17] 董天阳，李文杰，范菁，等. 业务流程驱动的森林仿真构件组装技术及应用研究[J]. 计算机科学，2012，39(9)：126-132