软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2011,22(11):2698−2715 [doi: 10.3724/SP.J.1001.2011.03902]

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

Web 服务组合功能与 QoS 的形式化统一建模和分析*

肖芳雄[1+], 黄志球[1], 曹子宁[1], 屠立忠[2], 祝 义[1]

[1](南京航空航天大学 信息科学与技术学院,江苏 南京 210016)

[2](南京工程学院 计算机工程学院,江苏 南京 211167)

Unified Formal Modeling and Analyzing both Functionality and QoS of Web Services Composition

XIAO Fang-Xiong[1+], HUANG Zhi-Qiu[1], CAO Zi-Ning[1], TU Li-Zhong[2], ZHU Yi[1]

[1](College of Information Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China)

[2](School of Computer Engineering, Nanjing Institute of Technology, Nanjing 211167, China)

+ Corresponding author: E-mail: xiaofx@nuaa.edu.cn

Xiao FX, Huang ZQ, Cao ZN, Tu LZ, Zhu Y. Unified formal modeling and analyzing both functionality and QoS of Web services composition. *Journal of Software*, 2011,22(11):2698−2715. http://www.jos.org.cn/1000-9825/3902.htm

Abstract: This paper proposes a process algebra called PPPA (priced probabilistic process algebra) by extending an existing process algebra with QoS modeling capability. This paper presents its syntax and semantics to prove that it can model not only functionality, but also non-functionality, such as reliability, performance, and cost. Finally, this paper moves to model and analyze both functionality and QoS of Web services composition in a united way based on PPPA. The paper illustrates the effectiveness with an example to prove that PPPA can support formal modeling and analyzation of QoS of web services composition.

Key words: Web; service composition; unified modeling and analyzing; QoS; process algebra

摘 要: 进程代数是一种适合描述 Web 服务组合的形式建模语言,然而同样对 QoS 建模和分析的支持不足.在现有进程代数的基础上,提出了一种代价概率进程代数 PPPA(priced probabilistic process algebra),给出其语法和语义,证明其具有功能、概率和代价的统一建模和分析能力.给出了基于 PPPA 统一建模和分析 Web 服务组合功能和 QoS 的方法.实例建模和分析了 Web 服务组合的功能、可靠性、性能和代价,其结果表明,PPPA 可以有效地支持 Web 服务组合功能和 QoS 的形式化统一建模和分析.

关键词: Web;服务组合;统一建模和分析;QoS;进程代数

中图法分类号: TP311 **文献标识码**: A

Web 服务组合是一种基于 Web 服务协议组合已有 Web 服务、构建 Web 应用的软件构建新形态.由于被组合的 Web 服务是由不同组织发布、分布在 Internet 上、具有自治特征的软件组件,在开放、动态、难控的网

* 基金项目: 国家自然科学基金(60873025); 国家高技术研究发展计划(863)(2009AA010307); 航空科学基金(20085552023); 江苏省自然科学基金(BK2008 389); 江苏省高校自然科学基金(03KJD520105)

收稿时间: 2009-11-18; 修改时间: 2010-01-21; 定稿时间: 2010-06-09

络环境下实现各类Web服务资源集成和共享的Web服务组合,其QoS(quality of service)成为决定其能否成功的关键因素之一.在系统设计阶段,建立Web服务组合的形式模型并分析验证其功能正确性和QoS可满足性,是提高Web组合服务可信性的有效手段.

由于Web服务组合的主要特征在于组合,同样具有组合特征的进程代数成为适合建立Web服务组合形式模型的建模语言.进程代数是一种形式化建模语言,具有以自然的方式对并发和组合系统建模的特征[1].比如,提供了顺序、选择、并发、循环等丰富的组合算子和进程结构,可以将子系统组合成更大的系统,便于进行问题分解和组合层次建模;可以通过隐藏算子将系统不关心的活动隐藏为内部活动,便于进行问题抽象和简化;基于代数方法,具有简洁的语法和精确的语义,便于进行形式描述和验证;提供了互模拟概念,便于支持等价划分和问题简化.因此,近年来有不少研究者利用进程代数进行Web服务组合的研究,典型的有:Salaün在流程执行语言BPEL(business process excution language)[2]和进程代数CCS(calculus of communication system)[3]之间建立映射,从而可以借助CCS的建模方法和工具对BPEL流程程序的正确性进行验证[4];Koshkina提出运用CCS来建模和验证基于BPEL的Web服务协作[5];Lucchi给出了基于进程代数Pi-Culculus的BPEL的形式语义[6].这些研究的一个共同目的是要建立WS-BPEL与进程代数之间的映射.建立WS-BPEL与进程代数映射的目的是使WS-BPEL具有进程代数的抽象形式,其好处在于:一是在Web服务组合设计阶段,使用进程代数建立基于WS-BPEL的Web服务组合的形式模型,利用进程代数丰富的分析验证方法和工具来分析验证设计问题.经过验证的抽象模型可以从上到下精化为WS-BPEL流程程序,从而提高服务组合的可信性;二是对于未经验证的WS-BPEL流程程序,可以从下到上通过逆向工程映射为进程代数抽象形式模型,利用进程代数的分析方法和工具来分析存在的问题.另外,由于WS-BPEL已成为业界广为接受和认可的基于流程组合Web服务的事实标准,这样的映射使得WS-BPEL具有了进程代数的形式化基础,而进程代数具有以自然的方式对并发和组合系统建模的特征,因而进程代数适合作为Web服务组合的形式理论基础[7,8].

Web服务组合的QoS是其内在属性.从客观性和适用性的角度来看,Web服务组合研究中最为关注的QoS属性是时间、可靠性、性能、代价.QoS驱动的Web服务组合是一种重要的服务组合方法,它从多个符合功能需求的服务组合方案中选择QoS优化的服务组合[9−11].然而,上述利用进程代数的研究都集中在服务组合的功能建模和验证方面,忽视了服务组合的QoS建模,原因在于已有进程代数对QoS建模的支持不足.现有的进程代数包括[1]:经典进程代数CCS(calculus of communication system),CSP(communicating sequential processes)和ACP(algebra of communicating process),以及在经典进程代数上发展起来的支持移动计算建模的Pi-Calculus,支持实时系统建模的时间进程代数TPA(timed or temporal process algebra),支持性能建模的随机进程代数SPA(stochastic process algebra)等.虽然TPA和SPA分别可以对Web服务组合的时间和性能建模,但是尚不支持对Web服务组合的时间优化选择和可靠性优化选择.另外,对于代价这一重要的非功能属性,现有进程代数尚不支持对其建模.虽然Eberbach面向Agent领域在Pi演算上扩展了代价信息而提出了cost演算,但该cost演算是专为Agent定制的,语法复杂,没有用于且不适合用于Web服务组合[12].

实际上,对Web服务组合的形式建模不应只局限于功能上,而应是对功能和QoS的统一建模(为叙述简便,下文把功能和QoS统一建模简称为统一建模或统一模型).功能和QoS的统一建模在设计阶段就提前分析QoS属性,识别决定系统QoS指标的关键活动,指导系统实现,从而有助于降低开发风险,提高系统可信性.因此,有必要研究支持Web服务组合功能、时间、代价、概率统一建模的方法和工具.因为SPA侧重描述系统行为的不确定性及其对性能和可靠性的影响,而Web服务组合的性能和可靠性是重要的QoS属性.为此,本文尝试对SPA进行代价扩展,提出代价概率进程代数PPPA(priced probabilistic process algebra)并用于Web服务组合建模,以支持Web服务组合的功能、可靠性、性能和代价的建模和分析.

本文第1节提出PPPA,给出其语法和语义,证明其具有功能、概率、代价统一建模能力.第2节给出运用PPPA建模Web服务组合的实例.第3节给出算法构造代价概率空间以支持可靠性、性能和代价分析.第4节是相关研究.第5部分是结束语和下一步的工作.

1 PPPA

为了建立基于进程代数的功能和 QoS 统一建模语言,需要把概率和代价信息扩展到进程代数中.由于 SPA 是扩展了概率信息的进程代数,可把代价信息扩展到 SPA 中,从而形成一个支持功能、概率和代价的统一建模语言.经典的 SPA 包括:PEPA(performance evaluation process algebra)[13],TIPP(timed process and performance analysis)[14],EMPA(extended markovian process algebra)[15],IMC(interactive Markov chain)[16]等.它们的共同之处在于,把经典进程代数和连续时间马尔可夫链 CTMC(continue time Markov chain)相结合,把概率信息扩展到进程代数的动作上,从而对建立的模型既可运用经典进程代数进行功能验证,又可运用 CTMC 进行性能评价.它们的不同之处在于对并发同步动作延迟时间的处理:因为两个延迟时间 t_1 和 t_2 服从指数分布的动作 a_1 和 a_2,它们并发同步的延迟时间 t 应取 t_1 和 t_2 的最大值,即 $t=\max(t_1,t_2)$.但 t 不再是指数分布,破坏了马尔可夫性,对这个问题的不同处理形成了不同的 SPA.其中,PEPA 把 t 仍然作为指数分布处理,但其分布参数取 $r=\min(r_1,r_2)$,其中,r_1,r_2 分别为 a_1,a_2 的分布参数,从而保持了 PEPA 模型的马尔可夫性,同时又使得 PEPA 模型较为简洁,获得了大量应用.因此,本文参考 PEPA 来构建 PPPA.

接下来讨论如何进行代价扩展.首先引入概念:资源、价格、成本、代价.在 PPPA 中,资源是一个抽象的概念,泛指软件在完成任务的过程中消费的各种资源,包括但不限于处理器时间、内存、外存、功耗、带宽等;价格(price)指的是资源的价格,是静态的概念;代价(cost)指的是资源的消费,是动态的概念.价格是与动作相联系的.当动作没有执行时,声明为将消费的资源价格;当动作执行时,按照声明的价格产生代价.成本(cost)是与状态相联系的,系统到达某个状态是执行一系列动作的结果,在数量上等于这一系列动作累积的代价.由于进程代数是描述系统行为的语言,对系统行为的描述主要体现在动作和动作引起的状态迁移上,我们为 PPPA 中的动作联系将消费的资源价格.由于进程代数可以进行动作演算,在 PPPA 中带价格的动作演算可形成不同的联系了成本的状态.本文后面有时对价格和成本不作严格区分时,将两者通称为代价.由于进程代数中的进程(并非计算机程序中的进程)是形式建模的实体,首先对其扩展价格和成本信息,扩展了价格和成本信息的 PPPA 进程称为代价概率有限状态进程 PPFSP(priced probabilistic finite state process).下面给出 PPFSP 的形式定义.

定义 1. 一个 PPFSP 进程是一个五元组 $PPFSP=(S,s_0,Act,\rightarrow,l)$,其中,

(1) S 是非空有穷离散状态集.

(2) s_0 是初始状态.

(3) $Act=\{\tau\}\cup A$ 是动作集,其中,τ是进程的内部动作,对外不可见;A 是可观测动作集,对外可见.

(4) $\rightarrow\subseteq S\times Act\times\mathbb{R}_{\geqslant 0}\times\mathbb{R}_{\geqslant 0}\times S$ 是状态转移关系,其中,第 1 个非负实数域$\mathbb{R}_{\geqslant 0}$是动作引起的状态转移速率的值域,第 2 个非负实数域$\mathbb{R}_{\geqslant 0}$是动作价格的值域.

(5) $l:\rightarrow\mapsto(Act\times\mathbb{R}_{\geqslant 0}\times\mathbb{R}_{\geqslant 0})$是标记函数.

定义 2. 一个价格概率动作是一个三元组 $a_p=(a,r,p)$,其中,$a\in Act;r\in\mathbb{R}_{\geqslant 0}$,为动作 a 引起状态转移时,转移时间服从指数分布的参数;$p\in\mathbb{R}_{\geqslant 0}$,为动作 a 的价格.

定义 3. 一个成本概率状态是一个三元组 $s_c=(s,q_s,c_s)$,其中,$s\in S,q_s,c_s$ 分别是从 s_0 到达 s 的概率和成本.

定义 4. PPPA 的语法定义如下:

$$E::=0|(a,r,p).E|E;E|E+E|E|E|E\backslash L|A,$$

其中,(a,r,p)是价格概率动作,E 为 PPFSP 进程表达式,其余与 CCS 保持一致[3].之所以用 CCS 的风格来表示 PPPA 的语法,是因为 CCS 具有简洁的形式.

经典进程代数的语义是用标记迁移系统 LTS(labeled transition system)来表示的[3],本文引入代价概率标记迁移系统 PPLTS(priced probabilistic labeled transition system)来表示 PPPA 的语义.

定义 5. PPPA 的语义定义为代价概率标记迁移系统 $PPLTS=(S_c,s_{c_0},Act,\{\xrightarrow{(\alpha,r,p)}\})$, 其中,

(1) S_c 为成本概率状态集;

(2) $s_{c_0}=(s_0,1,0)$ 为初始成本概率状态;

(3) Act 为动作集;

(4) $\xrightarrow{(\alpha,r,p)} \in S_c \times Act \times \mathbb{R}_{\geqslant 0} \times \mathbb{R}_{\geqslant 0} \times S_c$ 为代价概率迁移关系.

经典进程代数的引导图 DG(derivation graph)对应其 LTS,可用于描述进程的行为演化[3].在 DG 的基础上,引入代价概率引导图 PPDG(priced probabilistic derivation graph),用于描述 PPPA 进程的行为演化.

定义 6. 任一 PPFSP 的 PPDG 对应一个代价概率标记迁移系统 $PPLTS=(S_c,s_{c_0},Act,\{\xrightarrow{(\alpha,r,p)}\})$.

在此,需要区别几个概念:在 PPDG 中,任意一条路径的概率指的是该路径上所有边(对应动作迁移)的概率之积;而路径的成本指的是该路径上所有边的价格之和;任意一个状态的概率指的是到达该状态的可能性,其值为到达该状态的所有路径的概率之和;状态的成本指的是到达该状态的路径的成本,沿不同路径到达该状态的成本不同,按照最小代价语义,其值为最小成本的路径的成本.

下面给出 PPPA 算子的结构化操作语义,每个 PPPA 算子的语义都由 3 个部分组成:

1) 动作执行的经典结构化操作语义[3],这部分与经典进程代数的语义保持一致;
2) 动作执行的概率语义,这是本文对 SPA 语义的进一步发展,SPA 只给出了动作执行的迁移速率,而本文进一步给出了动作引起的迁移状态的概率;
3) 动作执行的代价语义,这是本文为进程代数的代价扩展而提出的语义.

为表示方便,令 s_b,s_e 分别表示算子执行前、后的状态;令 $c_{b\_\min}$,$c_{e\_\min}$ 分别表示算子执行前后的最小成本;令 $Path(S)$表示从状态 S 出发的迁移集合.

定义 7. PPPA 的算子结构化操作语义如下:

(1) Prefix:

1) $\overline{a.E \xrightarrow{(a,r,p)} E}$;

2) $Pr(s_e)=Pr(s_e)+Pr(s_b)\cdot\dfrac{r}{\sum\limits_{j\in Path(s_b)} r_j}$;

3) $c_{e\_\min}=\mathrm{Min}(c_{b\_\min}+p,c_{e\_\min})$.

Prefix 算子语义表明:

1) 动作前缀算子可以无条件地执行.
2) 后状态的新概率=后状态的原概率+(前状态的概率 x 该动作在多个迁移中赢得“竞争”的概率).这一规则源于:① 概率计算的算法(将在第 4 节介绍)将把除初始状态(概率为 1)外的所有状态概率初始化为 0;② 如果到达某个状态存在多个路径,则该状态的概率为这些路径的概率之和.
3) 后状态的最小成本的计算需比较动作价格与前状态的成本之和.

(2) Choice:

1.1) $\dfrac{E_1 \xrightarrow{(a_1,r_1,p_1)} E_1'}{E_1+E_2 \xrightarrow{(a_1,r_1,p_1)} E_1'}$, if a_1 executed;

1.2) $Pr(s_e)=Pr(s_e)+Pr(s_b)\cdot\dfrac{r_1}{\sum\limits_{j\in Path(s_b)} r_j}$;

1.3) $c_{e\_\min}=\mathrm{Min}(c_{b\_\min}+p_1,c_{e\_\min})$;

2.1) $\dfrac{E_2 \xrightarrow{(a_2,r_2,p_2)} E_2'}{E_1+E_2 \xrightarrow{(a_2,r_2,p_2)} E_2'}$, if a_2 executed;

2.2) $Pr(s_e)=Pr(s_e)+Pr(s_b)\cdot\dfrac{r_2}{\sum\limits_{j\in Path(s_b)} r_j}$;

2.3) $c_{e\_\min}=\mathrm{Min}(c_{b\_\min}+p_2,c_{e\_\min})$.

Choice 算子语义表明:

1) 对 E_1,E_2 不确定选择,如果执行 a_1,则选择 E_1;如果执行 a_2,则选择 E_2.

2) 如果执行 a_1,则 r_1 参与概率计算;如果执行 a_2,则 r_2 参与概率计算.

3) 如果执行 a_1,则 p_1 参与代价计算;如果执行 a_2,则 p_2 参与代价计算.

(3) Parallel:

1.1) $\dfrac{E_1 \xrightarrow{(a_1,r_1,p_1)} E_1'}{E_1|_{\{\varnothing\}}\ E_2 \xrightarrow{(a_1,r_1,p_1)} E_1'|_{\{\varnothing\}}\ E_2}$;

1.2) $Pr(s_e)=Pr(s_e)+Pr(s_b)\cdot\dfrac{r_1}{\sum_{j\in Path(s_b)} r_j}$;

1.3) $c_{e\_\min}$=Min($c_{b\_\min}+p_1$,$c_{e\_\min}$);

2.1) $\dfrac{E_2 \xrightarrow{(a_2,r_2,p_2)} E_2'}{E_1|_{\{\varnothing\}}\ E_2 \xrightarrow{(a_2,r_2,p_2)} E_1|_{\{\varnothing\}}\ E_2'}$;

2.2) $Pr(s_e)=Pr(s_e)+Pr(s_b)\cdot\dfrac{r_2}{\sum_{j\in Path(s_b)} r_j}$;

2.3) $c_{e\_\min}$=Min($c_{b\_\min}+p_2$,$c_{e\_\min}$).

Parallel 算子语义表明:

1) E_1 和 E_2 异步并发执行.

2) 异步并发中如果执行 a_1,则 r_1 参与概率计算;如果执行 a_2,则 r_2 参与概率计算.

3) 异步并发中如果执行 a_1,则 p_1 参与代价计算;如果执行 a_2,则 p_2 参与代价计算.

(4) Cooperation:

1) $\dfrac{E_1 \xrightarrow{(a,r_1,p_1)} E_1', E_2 \xrightarrow{(a,r_2,p_2)} E_2'}{E_1|_{\{a\}}\ E_2 \xrightarrow{(a,\min(r_1,r_2),p_1+p_2)} E_1'|_{\{a\}}\ E_2'}$;

2) $Pr(s_e)=Pr(s_e)+Pr(s_b)\cdot\dfrac{\min(r_1,r_2)}{\sum_{j\in Path(s_b)} r_j}$;

3) $c_{e\_\min}$=Min($c_{b\_\min}+p_1+p_2$,$c_{e\_\min}$).

Cooperation 算子语义表明:

1) E_1 和 E_2 同步并发执行.

2) 用于概率计算的同步并发动作的速率为 min(r_1,r_2).

3) 参与同步并发的两个动作的代价都被计算到状态成本中.

(5) Hiding:

1) $\dfrac{E \xrightarrow{(a,r,p)} E'}{E\setminus L \xrightarrow{(a,r,p)} E'\setminus L}, a\notin L$;

2) $Pr(s_e)=Pr(s_e)+Pr(s_b)\cdot\dfrac{r}{\sum_{j\in Path(s_b)} r_j}$;

3) $c_{e\_\min}$=Min($c_{b\_\min}+p$,$c_{e\_\min}$).

Hiding 算子语义表明:

1) 动作集合 L 中被隐藏的动作不被系统外所见.

2) 限制算子不影响概率计算.

3) 限制算子不影响代价计算.

(6) Constant:

1) $\dfrac{E\xrightarrow{(a,r,p)}E'}{A\xrightarrow{(a,r,p)}E'}$, $A\Leftarrow E$;

2) $Pr(s_e)=Pr(s_e)+Pr(s_b)\cdot\dfrac{r}{\sum_{j\in Path(s_b)}r_j}$;

3) $c_{e\_\min}=\mathrm{Min}(c_{b\_\min}+p,c_{e\_\min})$.

Constant 算子语义表明:

1) 表达式中的常量 A 可以用其他表达式替代,常量被替代后将具有替代表达式相同的执行能力.

2) 替代算子不影响表达式概率计算.

3) 替代算子不影响表达式代价计算.

进程代数的互模拟用于刻画不同进程的行为等价性,是进程代数的核心概念之一,具有重要的作用,如,在设计验证中用于检验设计与规约的一致性;在模型检测中用于互模拟等价类划分,约简状态空间以解决状态空间爆炸问题等.互模拟有强弱之分,弱互模拟考察不同进程的外部行为等价性,是在实际应用中常使用的互模拟概念,下面给出弱互模拟的形式定义.

定义 8. 对称二元关系 $\mathcal{R}\subseteq S\times S$ 称为弱互模拟,如果对任意 $(s_1,s_2)\in\mathcal{R}$ 蕴涵:若 $s_1\xrightarrow{\alpha}s_1'$,存在 s_2' 使得 $s_2\overset{\hat{a}}{\Rightarrow}s_2$,且 $(s_1',s_2')\in\mathcal{R}$.其中,$\overset{\hat{a}}{\Rightarrow}$ 为包含外部动作 a 和 0 到多个内部动作 τ 的动作系列.

下面讨论扩展代价的完备性和合理性.完备性是指任一 PPFSP 进程的 PPDG 描述了该进程所有的成本概率状态.合理性是指任一成本概率状态对应的概率是到达该状态的路径概率之和,对应的代价是到达该状态的最小路径代价.

定理 1. 任一 PPFSP 进程的 PPDG 描述了该进程所有的成本概率状态及其迁移,且任一成本概率状态对应的概率是到达该状态的路径概率之和,对应的成本是到达该状态的最小路径代价.

证明:完备性证明.DG 描述了进程所有的状态及其迁移,PPDG 是 DG 的概率和代价扩展,因而 PPDG 也描述了进程所有的状态及其迁移.同时,在 PPDG 中,所有的状态和迁移都扩展了概率和代价信息,因而 PPDG 描述了进程所有的成本概率状态及其迁移.

合理性证明.PPDG 中任一成本概率状态中的概率和成本,都是按照定义 5 和定义 7 计算出来的,这就保证了对应的概率是到达该状态的路径概率之和,对应的成本是到达该状态的最小路径成本.证毕. □

下面讨论 PPPA 和 PEPA 的关系,为此引入派生类的概念.证明 PPPA 是 PEPA 的派生类,即 PPPA 扩充了 PEPA 的代价建模能力.

定义 9. 如果 A 类进程代数的任意一个进程可接受的语言都能够被 B 类进程代数的某个进程所接受,那么称 B 类进程代数是 A 类进程代数的派生类.

定理 2. PPPA 是 PEPA 的派生类.

证明:一个 PEPA 进程可接收的语言为其可执行带延迟参数的动作序列集合,形如 $\{\delta|\delta=(a_1,p_1)(a_2,p_2)\ldots(a_n,p_n)\}$;一个 PPPA 进程可接收的语言为其可执行价格概率动作序列集合,形如 $\{\sigma|\sigma=(a_1,r_1,p_1)(a_2,r_2,p_2)\ldots(a_n,r_n,p_n)\}$.令 ϕ 是 PEPA 中任意一个进程,ϕ 能接受的语言为 L_ϕ;φ 是 PPPA 中对应 ϕ 的一个进程,φ 能接受的语言为 L_φ.L_ϕ 为 ϕ 能接受的所有动作序列;L_φ 为 φ 能接受的所有价格概率动作序列.对于 L_ϕ 中任意一个动作序列 $\delta=(a_1,p_1)(a_2,p_2)\ldots(a_n,p_n)$,则唯一对应 L_φ 中一个价格概率动作序列 $\sigma=(a_1,r_1,p_1)(a_2,r_2,p_2)\ldots(a_n,r_n,p_n)$,即 φ 也能接受 L_ϕ.所以,PPPA 是 PEPA 的派生类.证毕. □

定理 2 证明了 PPPA 是 PEPA 的派生类,即 PPPA 在 PEPA 基础上扩充了代价建模能力.实际上,一个经典进程可接收的语言为其可执行动作序列集合,形如 $\{\rho|\rho=(a_1,a_2,\ldots,a_n)\}$.因此,同理可证 PEPA 是经典进程代数的派生类,在经典进程代数基础上扩充了概率建模能力.所以,PPPA 在经典进程代数功能性建模能力基础上扩充了概率和代价建模能力,从而能够支持功能、概率、代价统一建模.如果令所有价格概率动作中的动作价格为 0,

则 PPPA 退化为 PEPA;如果进一步令所有价格概率动作中的动作延迟概率参数和动作价格都为 0,则 PPPA 退化为经典进程代数.

2 基于 PPPA 的服务组合建模

用 PPPA 来建模基于 BPEL 的 Web 服务及其编制(orchestration)组合,首先需要建立起 PPPA 和 BPEL 之间的映射.参考文献[6]中的映射方法,表 1 给出 PPPA 与 WSDL/BPEL 之间的对应关系.根据表 1 所示映射规则,把 WSDL 描述的 Web 服务映射为 PPPA 的进程,把 Web 服务的消息交互行为映射为 PPPA 的动作,把 BPEL 的顺序、选择、并发、循环等结构映射为 PPPA 的相应结构.经过上述映射,BPEL 描述的 Web 服务组合就可以映射为 PPPA 的进程演算,进而运用进程代数的方法和工具进行分析和验证.需要说明的是,虽然上述映射是 PPPA 的经典动作和结构与 BPEL/WSDL 之间的映射,但 PPPA 的经典动作绑定了概率信息和代价信息,从而可以在进程代数的抽象层面进行功能、可靠性、性能、代价等的建模和分析验证.而经过统一分析验证的 PPPA 模型可以作为 BPEL 流程实现的参考模型,经过映射和精化后进一步可成为 Web 服务组合的 BPEL 流程,从而有助于提高服务组合的设计正确性.同时,PPPA 模型层面的 QoS 分析能够帮助检查设计是否满足 QoS 要求,识别决定系统 QoS 的关键活动,用于指导服务组合的实现.

Table 1 Mapping between PPPA and BPEL/WSDL

表 1 PPPA 与 BPEL/WSDL 的对应关系

CCS	BPEL/WSDL
Actions	WSDL: message, portType, operation, partnerLinkType; BPEL: receive, reply, invoke
Sequence '.'	Sequence
Choice '+'	Pick, switch
Parallel composition '|'	Flow
Restriction set '\'	Interactions, assign
Termination '0'	Empty
Recursive	While
Dull action	Assign, interactions
Process	Service

下面以文献[2]中的旅行订票系统为基础,省略了与 PPPA 建模分析无关的环节,得到如图 1 所示简化的旅行订票系统,以此为例来说明基于 PPPA 的 Web 服务组合建模.该旅行订票系统(BookingSystem)由旅行代理服务(TAgent)、宾馆服务(Hotel)和航空服务(AirlineA 和 AirlineB)等子服务组合而成.本文假设 BookingSystem 提供服务会消耗两种资源:一是子服务之间消息交互的带宽资源;二是各子服务的计算资源.为简化模型,假设不考虑消息的大小和通道的种类等,收发一个消息所占用的带宽资源的价格都是 1.各子服务的计算资源的价格假设以该子服务的使用费用表示,如为该子服务在 UDDI(universal description discovery and integration)上声明的价格,且假设为如图 1 所示:TAgent 的价格为 5,Hotel 的价格为 2,AirlineA 的价格为 3,AirlineB 的价格为 4.因为进程代数是以动作来描述系统行为的,需把这两种资源价格映射进动作中.为此,把价格概率动作中的价格细化为形式 $a_p=(a,r,p_{b'},p_s)$,其中,a 为动作,r 为动作引起的迁移的概率(该参数可通过统计 Web 服务的实际历史数据得到),p_b 为动作占用的带宽资源的价格,p_s 为动作所映射的原子服务的价格.相应地,成本概率状态细化为形式 $s_c=(s,q_s,c_b,c_s)$,其中,s 为状态,q_s 为到达该状态的概率,c_b 为到达该状态的累积带宽成本,c_s 为到达该状态的累积服务成本.

图 1 中 TAgent 是服务组合的核心,为此,我们从 TAgent 的角度来描述服务组合的流程.与 CCS 的进程动作表示一致,TAgent 发送消息的动作有前缀"'",而接受动作无此前缀.

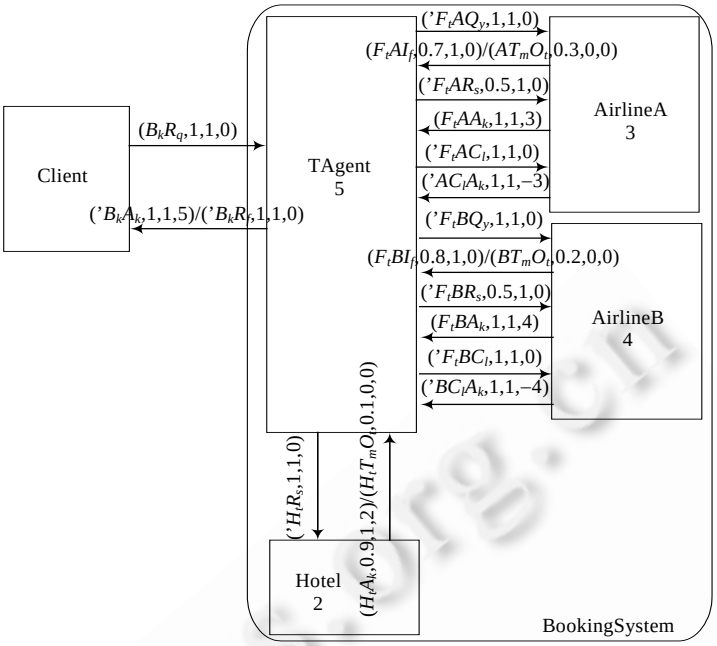

Fig.1 Travel booking system

图 1 旅行订票系统

1) TAgent 平时处于等待提供服务状态,收到 Client 的订票请求(B_kR_q,1,1,0).其中,第 1 个 1 是指 TAgent 在等待提供服务状态下接受消息 B_kR_q 的迁移概率为 1.因为 TAgent 在等待提供服务状态下必须经历该迁移,否则其他迁移将无法发生.第 2 个 1 是指接受消息 B_kR_q 的带宽价格为 1;0 指该消息没有映射服务价格.

2) TAgent 并行地向 AirlineA 和 AirlineB 发出票务查询('F_tAQ_y,1,1,0)和('F_tBQ_y,1,1,0).为简化建模,我们采用('F_tAQ_y|'F_tBQ_y,1,2,0)统一表示这两个动作.其中,'F_tAQ_y|'F_tBQ_y 指并行发送查询消息,1 指 TAgent 在收到 B_kR_q 的状态下发送'F_tAQ_y|'F_tBQ_y 的概率为 1,2 指并行发送'F_tAQ_y|'F_tBQ_y 的带宽价格为 2,0 指该消息没有映射服务价格.

3) TAgent 并行地接收 AirlineA 和 AirlineB 的响应.AirlineA 的响应分为(F_tAI_f,0.7,1,0)/(AT_mO_t,0.3,0,0)两种情况,其中,(F_tAI_f,0.7,1,0)表示返回票务信息的概率为 0.7,(AT_mO_t,0.3,0,0)表示超时无响应的概率为 0.3.其中,AT_mO_t 表示 TAgent 的内部超时动作,不占用带宽.AirlineB 的响应分为(F_tBI_f,0.8,1,0)/(BT_mO_t,0.2,0,0)两种情况.为简化建模,我们统一 AirlineA 和 AirlineB 的响应为 4 种情况:

3.1) AirlineA 和 AirlineB 都返回票务信息($F_tAI_f|F_tBI_f$,0.56,2,0)(其中,0.56=0.8×0.7;带宽价格为 2;不映射服务价格.在此状态下,TAgent 将基于某些标准(比如 QoS 等)来选择服务,假设 TAgent 将以概率 0.5 向 AirlineA 发送订票请求'F_tAR_s,或以概率 0.5 向 AirlineB 发送订票请求'F_tBR_s).

3.2) AirlineA 返回票务信息而 AirlineB 超时无响应($F_tAI_f|BT_mO_t$,0.14,1,0)(其中,0.14=0.7×0.2;带宽价格为 1;不映射服务价格.在此状态下,TAgent 将以概率 1 向 AirlineA 发送订票请求'F_tAR_s).

3.3) AirlineA 超时无响应而 AirlineB 返回票务信息($AT_mO_t|F_tBI_f$,0.24,1,0)(其中,0.24=0.3×0.8;带宽价格为 1;不映射服务价格.在此状态下,TAgent 将以概率 1 向 AirlineB 发送订票请求'F_tBR_s).

3.4) AirlineA 和 AirlineB 都超时无响应($AT_mO_t|BT_mO_t$,0.06,0,0)(其中,0.06=0.3×0.2;带宽价格为 0;不映射服务价格.在此状态下,TAgent 将以概率 1 向 Client 发送服务失败消息('B_kR_f,1,1,0)).

4) 一旦 TAgent 向 AirlineA 发送'F_tAR_s,在此状态下,则假设它们之间的通道在一次服务组合事务过程中不再出现故障,则 TAgent 收到 AirlineA 订票成功的概率为 1,也即动作(F_tAA_k,1,1,3)(其中,3 表示该动作映射了 AirlineA 的服务价格).

5) 相应地,一旦 TAgent 向 AirlineB 发送'F_tBR_s,在此状态下,则假设它们之间的通道在一次服务组合事务过程中不再出现故障,则 TAgent 收到 AirlineB 订票成功的概率为 1,也即(F_tBA_k,1,1,4)(其中,4 表示该动作映射了 AirlineB 的服务价格).

6) 订票成功后,TAgent 向 Hotel 发送订房间请求('H_tR_s,1,1,0).

7) Hotel 的响应分为(H_tA_k,0.9,1,2)/($H_tT_mO_t$,0.1,0,0)两种情况:

7.1) (H_tA_k,0.9,1,2)表示订房间成功的概率为 0.9,该动作映射了 Hotel 服务的价格 2.在此状态下,TAgent 将以概率 1 向 Client 发送服务成功消息('B_kA_k,1,1,5),其中,5 表示该动作映射了 TAgent 的服务价格.

7.2) ($H_tT_mO_t$,0.1,0,0)表示超时无响应的概率为 0.1.在此状态下,还需:

7.2.1) 如果向 AirlineA 订过票,则 TAgent 还需向 AirlineA 发送取消订票消息('F_tAC_l,1,1,0),并接收取消确认消息('AC_lA_k,1,1,−3)(其中,−3 表示该动作取消 AirlineA 的服务价格计算.因为组合服务是以成功提供服务来计算费用的,不成功提供服务将不计算费用,这可以看作是 Web 服务事务中的补偿机制),之后向 Client 发送服务失败消息('B_kR_f,1,1,0).

7.2.2) 相应地,如果向 AirlineB 订过票,则 TAgent 需向 AirlineB 发送取消订票消息('F_tBC_l,1,1,0),并接收取消确认消息('BC_lA_k,1,1,−4)(其中,−4 表示该动作取消 AirlineB 的服务价格计算),之后向 Client 发送服务失败消息('B_kR_f,1,1,0).

本文着重于阐述运用 PPPA 对服务组合建模和分析的原理,对上述实例适当作了一些简化,省略了部分细节问题,并做如下适当假设:

1) 子服务的外部行为是可知的,体现在其接口的消息交互中;

2) 子服务之间的通信是同步通信;

3) 异常处理表现为预订房间失败情况下对预订机票的服务事务补偿.

对上述系统,我们把 Tagent,Hotel,AirlineA,AirlineB,BookingSystem 分别建模为 PPFSP.另外需要说明的是,Client 不是组成 BookingSystem 的子服务,而是作为 BookingSystem 的需求者来验证服务组合是否满足需求.因为 Client 是与 BookingSystem 交互的,如果 BookingSystem 满足 Client 需求,则 Client 与 BookingSystem 可以看作是一对互补的系统,且客户对偶 RClient(即与 Client 互补的满足需求的理想进程,对 Client 进程表达式中所有的动作取相应的补动作,可直接得到 RClient 的表达式)与 BookingSystem 在外部行为上应是一致的,这正好可以通过 RClient 与 BookingSystem 是否弱互模拟来验证.因此,我们把 Client 和 RClient 也建模为 PPFSP.下面给出它们的 PPPA 表达式.

代理服务 TAgent 的进程表达式如下:

$proc$ TAgent = (B_kR_q,1,1,0).('F_tAQ_y |'F_tBQ_y,1,2,0).TAgent1

$proc$ TAgent1 = (F_tAI_f | BT_mO_t,0.14,1,0).TAgent2 + (F_tAI_f | F_tBI_f,0.54,2,0).TAgent3+
(AT_mO_t | F_tBI_f,0.24,1,0).TAgent4 + (AT_mO_t | BT_mO_t,0.06,0,0).TAgent5

$proc$ TAgent2 =('F_tAR_s,1,1,0).TAgent6

$proc$ TAgent3 =('F_tAR_s,0.5,1,0).TAgent6+('F_tBR_s,0.5,1,0).TAgent7

$proc$ TAgent4 =('F_tBR_s,1,1,0).TAgent7

$proc$ TAgent5 =('B_kR_f,1,1,0).TAgent

$proc$ TAgent6 =(F_tAA_k,1,1,3).('H_tR_s,1,1,0).TAgent8

$proc$ TAgent7 =(F_tBA_k,1,1,4).('H_tR_s,1,1,0).TAgent9

$proc$ TAgent8 =(H_tA_k,0.9,1,2).TAgent10+($H_tT_mO_t$,0.1,0,0).TAgent11

$proc$ TAgent9 =(H_tA_k,0.9,1,2).TAgent10+($H_tT_mO_t$,0.1,0,0).TAgent12

$proc$ TAgent10 =('B_kA_k,1,1,5).TAgent

$proc$ TAgent11 =('F_tAC_l,1,1,0).(AC_lA_k,1,1,−3).TAgent5

$proc$ TAgent12 =('F_tBC_l,1,1,0).(BC_lA_k,1,1,−4).TAgent5

说明:TAgent 的表达式是对上述服务组合中 TAgent 流程的建模.

航空服务 AirlineA 的进程表达式如下:

$$proc\ \ \text{AirlineA} = (F_tAQ_y,0.5,1,0).('F_tAI_f,1,1,0).\text{AirlineA}+(F_tAR_s,0.4,1,0).('F_tAA_k,1,1,3).\text{AirlineA}+(F_tAC_l,0.1,1,0).('AC_lA_k,1,1,-3)\text{AirlineA}$$

说明:AirlineA 在等待提供服务的状态下可能存在 3 种行为:$(F_tAQ_y,?,1,0),(F_tAR_s,?,1,0),(F_tAC_l,?,1,0)$.这 3 种行为的概率可通过统计历史运行数据得到.比如,在 N 次运行中统计这 3 种行为的次数,则可把它们在 N 次运行中的发生的比例分别视为它们的概率.本文假设它们发生的概率分别为 0.5,0.4 和 0.1.需要注意的是,这 3 个动作是从 AirlineA 的角度而言的,不同于 TAgent 中的动作.

航空服务 AirlineB 的的进程表达式如下:

$$proc\ \ \text{AirlineB} = (F_tBQ_y,0.5,1,0).('F_tBI_f,1,1,0).\text{AirlineB}+(F_tBR_s,0.4,1,0).('F_tBA_k,1,1,3).\text{AirlineB}+(F_tBC_l,0.1,1,0).('BC_lA_k,1,1,3).\text{AirlineB}$$

说明:AirlineB 的情形与 AirlineA 的类似.

宾馆服务 Hotel 的进程表达式如下:

$$proc\ \text{Hotel}=(H_tR_s,1,1,0).('H_tA_k,1,1,2).\text{Hotel}$$

说明:Hotel 的行为本应与 AirlineA,AirlineB 类似,但为了使建模实例简化,Hotel 的情形做了简化处理,仅考虑了一种情形.

组合服务 BookingSystem 的进程表达式如下:

$$\text{set In} = \{F_tAQ_y,F_tBQ_y,F_tAI_f,F_tBI_f,F_tAR_s,F_tBR_s,F_tAA_k,F_tBA_k,F_tAC_l,F_tBC_l,AC_lA_k,BC_lA_k,H_tR_s,H_tA_k,AT_mO_t,BT_mO_t,H_tT_mO_t\}$$

$$proc\ \ \text{BookingSystem} = (\text{TAgent}\mid\text{AirlineA}\mid\text{AirwayB}\mid\text{Hotel})\setminus\text{In}$$

说明:BookingSystem 由子服务 Tagent,AirlineA,AirlineB 和 Hotel 并发组合而成.动作集合 In 中的动作是 BookingSystem 内各子服务之间的交互动作,对用户 Client 是不可见的,应作为内部动作使用隐藏算子隐藏起来.

客户 Client 的进程表达式如下:

$$proc\ \ \text{Client} = ('B_kR_q,1,1,0).\text{Client1}$$

$$proc\ \ \text{Client1} = (B_kA_k,?,?,?).\text{Client}+(B_kR_f,?,?,?).\text{Client}$$

说明:$(B_kA_k,?,?,?)$和$(B_kR_f,?,?,?)$表示从 Client 的角度而言,接收到 B_kA_k(也即服务组合成功)或 B_kR_f(也即服务组合失败)的概率、带宽成本、服务成本未知,需要在服务组合的设计验证阶段求解,这也是本文需要解决的问题之一.

需要注意的是,这两个动作与图 1 中标注的动作$('B_kA_k,1,1,5)$和$('B_kR_f,1,1,0)$是不同的,后两个动作是从 TAgent 的角度而言,其中,$('B_kA_k,1,1,5)$表示 TAgent 在预订机票和房间都成功的状态下必然向 Client 发送成功消息,概率为 1;而$('B_kR_f,1,1,0)$表示 TAgent 在预订机票失败或预订房间失败的状态下必然向 Client 发送失败消息,概率为 1.

客户对偶 RClient 的进程表达式如下:

$$proc\ \ \text{RClient} = (B_kR_q,1,1,0).\text{RClient1}$$

$$proc\ \ \text{RClient1} = ('B_kA_k,?,?,?).\text{RClient1}+('B_kR_k,?,?,?).\text{RClient1}$$

说明:RClient 是 Client 的对偶进程,表示对服务组合的外部行为需求.其中,“?”的含义与 Client 表达式中的相同.

3 基于 PPPA 的服务组合分析

可以对上述基于 PPPA 建立的服务组合模型进行如下统一分析:

3.1 功能分析

如果把上述模型中的概率和代价信息忽略,也即忽略所有动作 $a_p=(a,r,p_{b'},p_s)$中的$(r,p_{b'},p_s)$,则可得到一个功能模型.可运用进程代数自动验证工具 CWB-NC[17]对功能模型进行了如下功能分析和验证(在使用 CWB 工具验证时,各进程表达式需按照 CWB 的符号规则进行表示,如空进程用 nil 表示等,在此不再赘述):

a) 安全检查:可运用死锁检测命令 fd 对各个子服务和组合服务进行死锁检查和活性检查.

b) 活动观察:可运用 sim 命令单步跟踪各进程行为,观察分析各子服务和组合服务的进程行为是否符合预期,对可能存在的问题进行调试.

c) 功能验证:可运用互模拟验证命令 eq-S obseq 验证 RClient 和 BookingSystem 功能模型的进程表达式是否弱互模拟,也即验证组合服务是否满足客户功能需求.如果两者不弱互模拟,则可继续进行上述活动 b),对存在的问题进行调试,不断调整 BookingSystem 的组合设计,直至通过验证.

功能分析已有相关研究[4−6,17],在此不再赘述.

3.2 QoS分析

本文中,基于 PPPA 的 QoS 分析包括两部分:基于概率信息的分析和基于代价信息的分析.而基于概率信息的分析又可分为可靠性分析和性能分析.为此,首先需要生成带有概率和代价信息的状态空间.由上述模型可以看出,BookingSystem 由子服务 Tagent,AirlineA,AirlineB 和 Hotel 并发组合而成,BookingSystem 的状态空间应是 4 个子服务状态空间的笛卡尔积.根据已有算法,可以求解出 BookingSystem 的状态空间,但是算法的时间和空间复杂度将随子服务的增多呈指数级增长,并可能带来状态空间爆炸问题[3].为此,本文提出一种简化处理方法来避免此问题.考虑基于 BPEL 的编制(orchestration)服务组合特点,这样的服务组合总存在一个核心子服务,该核心子服务参与组合子服务的所有行为,体现了组合服务的功能.比如,运行在 BPEL 执行引擎上用于组合子服务的 BPEL 流程程序可以看作是组合服务的核心子服务.因此,对组合服务状态空间的分析可以转化为对核心子服务状态空间的分析,从而降低状态空间处理的时间和空间复杂度.

本文中的 TAgent 可以看作是 BookingSystem 的核心子服务,对 BookingSystem 概率和代价的分析可以转化为对 TAgent 概率和代价的分析.下面按照代价概率空间构造、概率分析、代价分析 3 个部分进行介绍.

3.2.1 代价概率空间构造

根据 PPPA 的代价概率迁移语义,PPPA 进程表达式对应的代价概率空间实际上是一个代价概率引导图 PPDG.如前所述,由于 TAgent 可看作是组合服务 BookingSystem 的核心子服务,对 BookingSystem 的分析可以转化为对 TAgent 的分析.对 TAgent 中的并发动作使用并发算子“|”表示后,并发动作可看作是一个动作.比如,TAgent 进程表达式中的$('F_tAQ_y|'F_tBQ_y,1,2,0)$可看作是一个价格概率动作.由于 TAgent 是单个原子服务,在把并发动作使用并发算子“|”表示成单个动作后,其程序逻辑上仅存在 3 种基本结构:顺序、选择、循环.不妨设其进程表达式为 $E=a_{p_1}E+a_{p_2}E+...+a_{p_n}E$,其中, a_{p_i} (i=1,2,…,n)为价格概率动作;E 为以递归的形式定义的只含有顺序动作前缀算子和选择算子两种基本结构的进程表达式,而循环结构体现在表达式的等式结构中.进程的初始表达式对应进程的初始状态,在 PPDG 中对应根节点,蕴含了进程所有可能的演化.进程执行一个动作对应一次状态迁移,在 PPDG 中对应一条边.进程从初始状态开始,在执行动作的过程中会形成不同的状态,这些状态对应进程演化过程中不同的进程表达式,在 PPDG 中对应不同的状态节点.算法从根节点开始,以进程表达式中的“+”算子为分界,为每个动作执行构造一条边,为动作执行后形成的表达式构造节点,深度优先递归地构造出 PPDG.下面给出构造 PPDG 的算法.

算法 1. Generate PPDG for PPPA expression.

(1) *GeneratePPDG*(*E*,& *ptRootT*)

Input: *E*, a PPPA expression.

Output: *ptRoot*, a pointer pointing to a node.

Function: create *E*'s PPDG whose root node pointed by *ptRoot*.

Steps:

1) Register names for all *E*'s process expression agents prefixed with "proc"; //For example, register names for TAgent, TAgent1, TAgent2,…
2) Create the root node $s_{c_0}=(s_0,1,0,0)$ and tag its name to the original process expression, i.e., TAgent, and let *ptRoot* point to s_{c_0};
3) Call *GenerateNodesEdges*(s_{c_0},*E*).

(2) *GenerateNodesEdges*(s_{c_k},*E*).

Input: s_{c_k}, a priced probabilistic node,

E, a PPPA expression.

Function: create PPDG starting from node s_{c_k} and meeting *E*.

Steps:

1) Scan *E*;
2) For every $\alpha_i E$, i=1,2,…,n, separated by choice operator '+' do
3) { If (*E*=0) or (α_i=0) then continue;
4) If (*E* does not match any registered process agent) or (*E* matches to TAgent)
5) Create a node s_i for *E*;
6) Elseif the registered process agent matched to *E* has not been tagged with a node name
7) Create a node s_i for *E* and tag name s_i for the registered process agent;
8) Create edge $s_k \xrightarrow{(a,r,p_b,p_s)} s_i$;
9) Justify (q_s,c_b,c_s) of $s_{c_i}=(s_i,q_s,c_b,c_s)$ according to the minimun cost semantics of PPPA;
10) If (*E* matches to TAgent)
11) Create edge $s_k \xrightarrow{\text{tagged with recursive edge}} s_0$;
12) Call *GenerateNodesEdges*(s_{c_i},*E*) recursively;}

算法 1 首先对进程表达式中以关键字"proc"标注的进程表达式代理(比如 TAgent,TAgent1,TAgent2,…)进行注册.然后在扫描进程表达式生成状态节点的过程中,如果首次遇到注册过的进程表达式代理,则为其创建一个状态节点,并把创建的状态节点名标记到该进程表达式代理.以后再扫描到该进程表达式代理时,则无须再为其创建状态节点,仅需为其创建一条边,并根据代价概率迁移语义调整状态节点上的概率和代价数值.算法 1 以进程表达式中的"+"算子为分界,深度优先递归地构造 PPDG.因为假设只有一个初始状态,所构造出的 PPDG 只有一个初始节点.设表达式含有 n 个动作,因为算法需要为每个动作创建一条边,则算法 1 的时间复杂度为 $O(n)$.

如图 2 所示为基于算法 1 按照代价和概率语义构造出的 PPDG.在此对图 2 进行简要说明:组合服务初始化时,处于初始状态的概率为 1,成本为 0,对应 TAgent 的初始表达式,也对应 PPDG 根节点(s_0,1,0,0).执行动作(B_kR_q,1,1,0)后迁移到状态节点(s_1,1,1,0).之后,执行动作('F_tAQ_y|'F_tBQ_y,1,2,0)迁移到状态节点(s_2,1,3,0),对应 TAgent1 表达式.在此状态下,从 AirlineA 和 AirlineB 得到的响应分为 4 种情形,对应 4 种迁移,分别迁移到状态(s_3,0.14,4,0)(仅有 AirlineA 响应的状态)、(s_{13},0.56,5,0)(AirlineA 和 AirlineB 都有响应的状态)、(s_{19},0.24,4,0)(仅有 AirlineB 响应的状态)、(s_9,0.154,3,0)(AirlineA 和 AirlineB 都无响应的状态).对于状态(s_4,0.42,5,0)(TAgent 向 AirlineA 发出订票请求的状态),可由状态 s_3(s_3,0.14,4,0)或 s_{13}(s_{13},0.56,5,0)迁移而来.在状态 s_3,只能向 AirlineA 发出订票请求,因而 s_3 向状态 s_4 迁移的概率为 1;而在状态 s_{13},可以向 AirlineA 或 AirlineB 发出订票请求,且向两者发出订票请求的概率各为 0.5,则到达 s_4 的概率为 0.14×1+0.56×0.5=0.42.但根据最小代价语义,状态 s_4 的成本 5 是根据状态 s_3 中的成本计算得来.其余的迁移过程类似,不再赘述.状态(s_8,0.846,9,10)是组合成功状态,状态(s_{10},0.154,4,0)是组合失败状态,它们都是组合结束状态,通过循环表达式返回到初始状态.

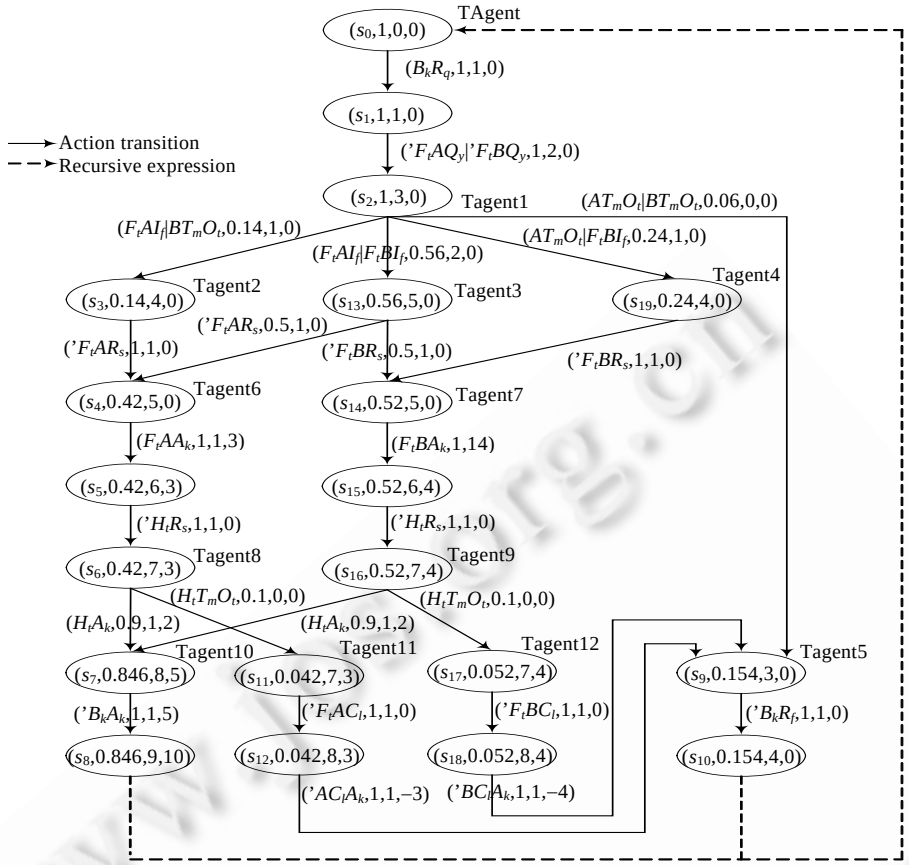

Fig.2 Priced probabilistic derivative graph

图 2 代价概率引导图

3.2.2 基于概率信息的分析

基于概率信息的分析分为可靠性分析和性能分析.

3.2.2.1 可靠性分析

Web 服务的可靠性在大多数 Web 服务研究中指的是成功提供服务的概率[9−11].本文沿用这样的 Web 服务可靠性定义.可靠性分析包括在服务组合的设计验证阶段分析:

1) 设计出的组合服务是否满足可靠性要求?为此需要求出服务组合成功的概率.
2) 哪条路径导致服务组合成功或失败的可能性最大,以便对该路径上的活动采取相应保障措施?为此需要求出导致服务组合成功或失败概率最大的路径;
3) 存在多种服务组合方案时,实施哪种服务组合方案的可能性最大?

可以在算法 1 生成的 TAgent 的 PPDG 中查找组合服务成功的成本概率状态,该状态上标记的概率即为服务组合成功的概率.TAgent 向 Client 发送成功消息'B_kA_k 引发迁移后的状态即是服务组合成功的状态,该消息可以看做是特定的消息.因此,通过在 PPDG 中查找该消息及其引发迁移后的状态,就可以从该状态中获得服务组合成功的概率.下面给出深度优先遍历 PPDG 的算法 2,该算法在 PPDG 中查找'B_kA_k 及其引发迁移后的状态,返回该状态中的概率值.由于到达服务组合成功状态可能存在多条路径,在服务组合设计验证阶段,用户可能还会对到达服务组合成功状态的不同路径的概率感兴趣,算法 2 还给出了到达服务组合成功状态的所有路径及其概率.算法 2 开始时初始化一个堆栈 S 和一个路径数组 *arrPath*,在算法深度优先遍历 PPDG 的过程中把经过的

状态节点压入堆栈 S 中,并把只有 1 个出分支的状态节点和所有出分支都遍历过状态节点标记为已遍历,当遇到特定消息 aOK(本例对应'B_kA_k)时,其引发迁移后的状态即对应服务组合成功的状态,其概率即为所求概率.此时,堆栈 S 中的状态节点序列即为一条从初始状态出发到达服务组合成功状态的路径,这条路径上所有边的概率的乘积即为该条路径的概率.此时,算法把服务组合成功状态的概率赋值给出参 rP,并把堆栈 S 复制作为出参 $arrPath$ 的一个元素.算法然后对 S 中的节点出栈,对未标记已遍历的节点继续深度优先遍历,从而求出所有从初始状态出发到达服务组合成功状态的路径,并把这样的路径复制到 $arrPath$.

算法 2. Reliability analysis algorithm.

Reliability_Analysis(*ptRoot*,*aOK*,& *rP*,& *arrPath*)

Input: *ptRoot*: a pointer pointing root of PPDG,

aOK: a specific message indicating services composition success.

Output: *rP*: the probability of services composition success,

arrPath: an array of pathes by which to reach the success composition state.

Function: In a PPDG whose root pointed by *ptRoot*, there is a success composition state in which *aOK* is sent. *rP* is the probability of reaching to the state. This function returns *rP* and put all pathes in *arrPath* by which to reach the state.

Steps:

```
1)  Initialize Stack(S); Initialize Array(arrayPath);
2)  Mark ptRoot travelled; Push(ptRoot,S);
3)  While not Empty(S) do
4)  {   ptParent=GetTop(S);
5)      if (all children of ptParent have been travelled) {mark ptParent travelled; ptChild==NIL}
6)      else {ptChild=Get non-travelled child of ptParent;}
7)      if (ptChild==NIL) {Pop(S); continue;}
8)      else {mark ptChild travelled;}
9)      While (ptChild!=ptRoot)
10)  {   Push(ptChild,S);
11)      if (ptChild has only child) {mark ptChild travelled;}
12)      if (aOK is the message of the edge between ptParent and ptChild)
13)         {rP=ptParent.r; copy S as an element of arrPath; break;}
14)      else {ptChild=Get a child of ptParent; continue;}
15)  }
16) }
```

算法 2 中存在两个循环:外层循环用于标记遍历的节点,内层循环用于遍历以匹配特定消息.设 PPDG 中顶点数为 n,则算法 2 的时间复杂度为 $O(n^2)$.

把算法 2 应用于 TAgent 的 PPDG,从而得知特定消息'B_kA_k 引发迁移后的状态为(s_8,0.846,9,10).该状态上的概率 0.846 即是从初始状态出发到达服务组合成功状态的概率,也即组合服务的可靠性是 0.846.从初始状态出发到达服务组合成功状态的路径有 4 条,存放在 $arrPath$ 中,分别为(容易根据 $arrPath$ 中状态节点序列求解边序列及其路径概率,限于篇幅,本文省略了这部分处理步骤的描述):

$$Path1 = s_0 \xrightarrow{1} s_1 \xrightarrow{1} s_2 \xrightarrow{0.14} s_3 \xrightarrow{1} s_4 \xrightarrow{1} s_5 \xrightarrow{1} s_6 \xrightarrow{0.9} s_7 \xrightarrow{1} s_8,$$

边上标注的数字为迁移概率,路径概率为 $Pr(Path1)=1\times1\times0.14\times1\times1\times1\times0.9\times1=0.126$;

$$Path2 = s_0 \xrightarrow{1} s_1 \xrightarrow{1} s_2 \xrightarrow{0.56} s_{13} \xrightarrow{0.5} s_4 \xrightarrow{1} s_5 \xrightarrow{1} s_6 \xrightarrow{0.9} s_7 \xrightarrow{1} s_8,$$

路径概率为 $Pr(Path2)=1\times1\times0.56\times0.5\times1\times1\times0.9\times1=0.252$;

$$Path3 = s_0 \xrightarrow{1} s \xrightarrow{1} s_2 \xrightarrow{0.56} s_{13} \xrightarrow{0.5} s_{14} \xrightarrow{1} s_{15} \xrightarrow{1} s_{16} \xrightarrow{0.9} s_7 \xrightarrow{1} s_8,$$

路径概率为 Pr(*Path*3)=1×1×0.56×0.5×1×1×0.9×1=0.252;

$$Path4 = s_0 \xrightarrow{1} s_1 \xrightarrow{1} s_2 \xrightarrow{0.24} s_{19} \xrightarrow{1} s_{14} \xrightarrow{1} s_{15} \xrightarrow{1} s_{16} \xrightarrow{0.9} s_7 \xrightarrow{1} s_8,$$

路径概率为 Pr(*Path*4)=1×1×0.24×0.5×1×1×0.9×1=0.216.

这 4 条路径概率之和为 0.846,正好是服务组合成功状态 s_8 的概率.概率最大的路径是 *Path*2 和 *Path*3,其概率都是 0.252,这是因为本文假设 TAgent 接收到 AirlineA 和 AirlineB 查询响应的情况下,向它们发送订票消息的概率相等.在这 4 条路径中,*Path*1 和 *Path*2 的概率之和为 0.378,为通过 Tagent,AirlineA,Hotel 组合成功的概率;*Path*3 和 *Path*4 的概率之和为 0.468,为通过 Tagent,AirlineB,Hotel 组合成功的概率.即通过 Tagent,AirlineB,Hotel 组合服务的概率大.同理,如果取'B_kR_f为特定消息,则运用算法 2 可以求解到达服务组合失败状态的概率,以及到达该状态的所有路径及其概率,不再赘述.

3.2.2.2 性能分析

基于概率的性能分析分为稳态分析和瞬态分析[13].本文现阶段的工作考虑稳态分析.性能稳态分析需要求解系统到达稳定情况下各个状态的概率,然后为各个状态联系一个性能参数(如响应时间、资源利用率等),从而可求出系统的平均性能.实际上,算法 1 和算法 2 适用于性能分析.例如,根据算法 1 和算法 2 容易求得到达服务组合成功和失败状态的概率分别是 0.846,0.154,假如到达服务组合成功和失败状态的响应时间分别是 0.3s,0.7s(考虑失败重试的时间),则组合服务的平均响应时间为 0.3616(即 0.846×0.3+0.154×0.7).

3.2.3 基于代价信息的分析

本文的代价分析包括在服务组合的设计验证阶段:

1) 检查状态的成本是否满足成本约束.

容易对算法 2 稍加调整,在算法 2 遍历 PPDG 的过程中把状态的成本与成本约束进行比较,则可检查状态的成本是否满足成本约束.例如,如果要求到达服务组合成功状态的成本约束是(9,11)(前一成本为带宽成本,后一成本为服务费用),由于 Tagent,AirlineA,Hotel 与 Tagent,AirlineB,Hotel 两种组合方案的成本分别为(9,10)和(9,11),都满足成本约束;如果要求到达服务组合成功状态的成本约束是(9,10),则只有 Tagent,AirlineA,Hotel 的组合方案满足成本约束;如果要求到达服务组合成功状态的成本约束是(9,9),则不满足成本约束.

2) 求解从初始状态出发到达特定状态的所有路径及其成本,用于指导成本优化的服务组合选择.

算法 2 实现了求解从初始状态出发到达特定状态的所有路径,容易对算法 2 稍加调整,对各条路径的边的代价求和,则可得出各条路径的成本.如前所述,到达服务组合成功状态的路径有 4 条,其成本分别为

Cost(*Path*1)=(9,10),*Cost*(*Path*2)=(10,10),*Cost*(*Path*3)=(10,11),*Cost*(*Path*4)=(9,11).

在这 4 条路径中,*Path*1 的代价最小.该路径上的动作序列将引导成本最小的服务组合,即 Tagent,AirlineA,Hotel 的组合.但同时我们也看到,*Path*1 的概率在 4 条路径中也最小,因此可以按照概率优先或成本优先来控制服务组合的选择,或基于决策理论中多属性决策方法[18]综合两者来控制服务组合的选择,或考虑代价最小为目标的 Markov 决策过程[19].限于篇幅,对多属性决策方法和 Markov 决策过程不再详述.

4 相关工作

QoS 驱动的 Web 服务组合依据 QoS 指标从多个符合功能需求的组合方案中选择 Web 服务,以构建 QoS 优化的 Web 组合方案.代表性的工作是 Zeng 等人提出的在服务组合的实施阶段,按执行时间、代价、可靠性、有效性和信誉等 QoS 指标选择 Web 服务进行服务组合[9−11].这类工作依据流程来组合服务,并依据 QoS 指标来选择服务进行组合.然而,组合服务的功能和 QoS 并没有经过设计阶段的模型分析和验证.而本文的工作则主要是在 Web 服务组合的模型层面,在设计阶段通过对 Web 服务组合的形式化分析,提前对组合服务的功能和 QoS 属性都进行了分析和验证,并提供 QoS 优化的组合方案,从而在模型层面保证服务组合设计的功能和 QoS 都满足需求,进而指导系统的实现,有助于降低开发风险,提高系统的可信性.

在系统模型层面统一建模和分析方面的相关工作主要有自动机、Petri 网、进程代数等方面的工作.在自动

机领域,Behrmann 等人提出了价格时间自动机,为位置和边联系 price 参数,表示所占用的系统资源的价格,主要用于实时系统中与成本相关的调度分析[20].Kwiatkowska 等人在概率时间自动机中为位边联系概率参数,表示动作迁移的概率,主要用于网络协议分析[21].但这些扩展的自动机都没有讨论组合问题,且没有在 Web 服务组合上应用.在 Petri 网领域,随机 Petri 网是一种经典的概率模型系统,可用于性能和可靠性的评价,但目前未能建模代价[22].Liu 等人提出价格时间 Petri 网,这些扩展的 Petri 网为每个变迁联系时间和价格参数,讨论了与成本相关的可达性问题,没有讨论组合问题和概率问题,且没有在 Web 服务组合上应用[23].由于 Web 服务组合的主要特征在于组合,因而同样具有组合特征的进程代数成为适合建立服务组合形式模型的建模语言.然而,目前现有进程代数对服务组合 QoS 建模和分析的支持尚不充分.比如,时间进程代数虽然支持对功能和时间统一建模,但主要用于实时系统建模,不支持路径分析,没有用于服务组合建模[1].现有随机进程代数 SPA 可对功能和概率统一建模,其概率分析方法主要是基于 Markov 的稳态分析和瞬态分析,采用的是数学解析方法,可求解到达状态的概率,但不能求解到达状态的所有路径及其概率[13−16].本文提出的 PPPA 是对 PEPA 的代价扩展,PEPA 是一种经典的 SPA,因而 PPPA 也具有上述 SPA 的性质.SPA 的分析方法主要是基于连续时间 Markov 链的数值分析方法,可以求解到达特定状态的概率.而 PPPA 提供了代价概率空间生成方法以及基于代价概率空间的路径分析方法,除了可以求解到达特定状态的概率和代价,还可以求解到达特定状态的路径及其概率和代价,这一特征更适用于从多种组合方案中选择 QoS 优化的 Web 服务组合分析.另外,现有进程代数对代价的建模和分析更少,Eberbach 虽然面向 Agent 领域在 Pi 演算上扩展了代价信息而提出了 cost 演算,但该 cost 演算是专为 Agent 定制的,语法复杂,没有用于且不适合用于 Web 服务组合[12].在软件体系结构研究领域,Mei 总结了软件体系结构设计中对功能和非功能进行统一建模的相关研究[24],这些研究主要是从模型、方法、框架(framework)的角度关注功能和非功能统一建模.与这些工作不同的是,本文提出了一种支持功能和非功能性统一建模的建模语言 PPPA 以及基于该语言的建模方法和分析方法.PPPA 支持功能、可靠性、性能和代价的统一建模和分析.

5 结束语和下一步工作

本文针对目前 QoS 属性在 Web 服务组合的系统建模中研究不足的现状,提出了基于进程代数支持功能、概率和代价统一建模的建模语言 PPPA,给出了 PPPA 的语法和语义,给出计算代价概率空间的算法以及可靠性、性能和代价分析的算法,并通过实例说明了其有效性.我们认为,本文尝试提出的 PPPA 及其统一建模和分析方法是对进程代数研究领域的有益补充,同时也为 Web 服务的研究提供了一种新的视角和途径.目前正在进行的工作包括实现文中所给出的算法,并集成到现有的进程代数工具中.

本文假定服务之间的通信是同步通信,而实际还存在异步通信,对异步通信的建模和分析是本文下一步考虑的工作.同时,本文建模了 Web 服务组合的 BPEL 补偿机制及其代价,而实际还存在异常处理机制、事件处理机制等复杂的处理,如何建模这些处理也是本文下一步考虑的工作.另外,本文提出的 PPPA 虽然可对功能、概率、代价等进行统一建模,但还没有支持确定时间的建模.PPPA 中的概率是由连续时间 Markov 链中的概率时间引入的,如何在统一模型中既支持概率时间又支持确定时间,从而建立功能、概率、代价、时间的统一建模语言并用于 Web 服务组合建模,也是本文下一步考虑的工作.

致谢 向对本文的工作给予支持和建议的同行,尤其是作者所在教研室讨论班上的老师和同学表示感谢.

References:

[1] Baeten JCM. A brief history of process algebra. Theoretical Computer Science, 2005,335(2-3):131−146. [doi: 10.1016/j.tcs.2004.07.036]

[2] Juric MB, Mathew B, Sarang P. Business Process Execution Language for Web Services. 2nd ed., Birmingham: Packt Publishing, 2006.

[3] Milner R. A Calculus of Communicating Systems. Berlin: Springer-Verlag, 1980.

[4] Salaün G, Bordeaux L, Schaerf M. Describing and reasoning on Web services using process algebra. In: Zhang LJ, Zhang J, eds. Proc. of the 2nd IEEE Int'l Conf. on Web Services (ICWS 2004). San Diego: IEEE Computer Society, 2004. 43−50. [doi: 10.1109/ICWS.2004.46]

[5] Koshkina M, van Breugel FV. Modelling and verifying Web service orchestration by means of the concurrency workbench. ACM SIGSOFT Software Engineering Notes, 2004,29(5):1−10. [doi: 10.1145/1022494.1022526]

[6] Lucchi R, Mazzara M. A pi-calculus based semantics for WS-BPEL. Journal of Logic and Algebraic Programming, 2007,70(1): 96−118. [doi: 10.1016/j.jlap.2006.05.007]

[7] Milner R. Theories for the global ubiquitous computer. In: Proc. of the 7th Int'l Conf. on Foundations of Software Science and Computation Structures (FOSSACS 2004). Barcelona: Springer-Verlag, 2004. 5−11. [doi: 10.1007/978-3-540-24727-2_]

[8] Lü J, Ma XX, Tao XP, Xu F, Hu H. Research and progress on Internetware. Science in China (Series E), 2006,36(10):1037−1080 (in Chinese with English abstract).

[9] Zeng LZ, Benatallah B, Ngu AHH, Dumas M, Kalagnanam J, Chang H. QoS-Aware middleware for Web services composition. IEEE Trans. on Software Engineering, 2004,30(5):1−17. [doi: 10.1109/TSE.2004.11]

[10] Haddad JE, Manouvrier M, Ramirez G, Rukoz M. QoS-Driven selection of Web services for transactional composition. In: Zhang LJ, Zhang J, eds. Proc. of the 6th IEEE Int'l Conf. on Web Services (ICWS 2008). Beijing: IEEE Computer Society, 2008. 653−660. [doi: 10.1109/ICWS.2008.116]

[11] Fan XQ, Jiang CJ, Wang JL, Peng SC. Random-QoS-Aware reliable Web service composition. Journal of Software, 2009,20(3): 546−556 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/20/546.htm

[12] Eberbach E. $-Calculus of bounded rational agents: Flexible optimization as search under bounded resources in interactive system. Fundamenta Informaticae, 2005,68(1-2):47−102.

[13] Hillston J. A compositional approach to performance modelling [Ph.D. Thesis]. Cambridge: Cambridge University Press, 1996.

[14] Hermanns H, Herzog U, Mertsiotakis V. Stochastic process algebras as a tool for performance and dependability modeling. In: Proc. of the Int'l Computer Performance and Dependability Symp. on Computer Performance and Dependability Symposium (IPDS'95). Washington: IEEE Computer Society, 1995. 102−111. [doi: 10.1109/IPDS.1995.395813]

[15] Bernardo M, Gorrieri R. A tutorial on EMPA: A theory of concurrent processes with nondeterminism, priorities, probabilities and time. Theoretical Computer Science, 1998,202(1-2):1−54. [doi: 10.1016/S0304-3975(97)00127-8]

[16] Hermanns H. Interactive Markov Chains: And the Quest for Quantified Quality. Berlin: Spring-Verlag, 1998.

[17] Cleaveland R, Li T, Sims S. The Concurrency Workbench of the New Century (Version 1.2). Carolina: North Carolina State University, 2000. 1−10.

[18] Kksalan M, Zionts S. Multiple criteria decision making, multiattibute utility theory: The next ten years. Management Science, 1992, 38(5):645−654. [doi: 10.1287/mnsc.38.5.645]

[19] Doshi P, Goodwin R, Akkiraju R, Verma K. Dynamic workflow composition using Markov decision processes. Int'l Journal of Web Services Research, 2005,2(1):576−582. [doi: 10.1109/ICWS.2004.1314784]

[20] Behrmann G, Fehnker A, Hune T, Larsen KG, Pettersson P, Romijn J, Vaandrager FW. Minimum-Cost reachability for priced timed automata. In: Benedetto MDD, Sangiovanni-Vincentelli A, eds. Proc. of the 4th Int'l Workshop on Hybrid Systems: Computation and Control. LNCS 2034, Berlin: Springer-Verlag, 2001. 147−161. [doi: 10.1007/3-540-45351-2_15]

[21] Kwiatkowska M, Norman G, Sproston J, Wang FZ. Symbolic model checking for probabilistic timed automata. Information and Computation, 2007,205(7):1027−1077. [doi: 10.1007/978-3-540-30206-3_21]

[22] Bause F, Kritzinger PS. Stochastic Petri Nets. Vieweg Braun-Schweig: Bause and Kritzinger, 2002.

[23] Liu WD, Song JX, Lin C. Modeling and analysis of grid computing application based price timed Petri net. Acta Electronica Sinica, 2005,33(8):1416−1420 (in Chinese with English abstract).

[24] Mei H, Shen JR. Progress of research on software architecture. Journal of Software, 2006,17(6):1257−1275. (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/17/1257.htm [doi: 10.1360/jos171257]

附中文参考文献:

[8] 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.

[11] 范小芹,蒋昌俊,王俊丽,庞善臣.随机 QoS 感知的可靠 Web 服务组合.软件学报,2009,20(3):546−556. http://www.jos.org.cn/1000-9825/20/546.htm

[23] 刘卫东,宋佳兴,林闯.基于价格时间 Petri 网的网格计算应用模型及分析.电子学报,2005,33(8):1416−1420.

[24] 梅宏,申峻嵘.软件体系结构研究进展.软件学报,2006,17(6):1257−1275. http://www.jos.org.cn/1000-9825/17/1257.htm [doi: 10.1360/jos171257]

肖芳雄(1971－),男,广西桂林人,博士,主要研究领域为服务计算,软件工程.

屠立忠(1968－),男,博士,副教授,主要研究领域为软件工程.

黄志球(1965－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为软件工程,数据工程.

祝义(1975－),男,博士,主要研究领域为软件工程.

曹子宁(1972－),男,博士,教授,主要研究领域为软件工程,形式化方法.