一种支持领域变化性的 Web Services 组装方法*)

赵俊峰 张 路 王亚沙 谢 冰
(北京大学信息学院软件研究所 北京 100871)

摘 要 随着 Web Services 及其相关技术的成熟,软件开发逐渐从基于构件的软件开发转向基于服务的软件开发。因此如何支持适应用户需求的 Web Services 的组装是当前研究的热点之一。本文提出了一种支持领域变化性的 Web Services 组装方法,该方法可以较好地适应用户需求的更改。文中引入了领域组装模型,使之能对一族具有领域共性和变化性的系统进行定义与描述。接着论述了如何根据用户的需求来确定 Web Services 应用的系统组装模型,并转换为可执行的组装描述语言的过程和方法。本文在最后给出了一个应用实例。

关键词 Web services,Web services 组装,领域工程

A Web Service Composition Method with Domain Variability Support

ZHAO Jun-Feng ZHANG Lu WANG Ya-Sha XIE Bing
(Software Institute,School of Electronics Engineering and Computer Science,Peking University,Beijing 100871)

Abstract With the development of Internet and the technology of Web services,more and more applications have changed the way to develop,that is from component-based software development to service-based software development. This paper puts forward a method of Web services composition to support the specific domain variability. It can automatically comply with the changing requirements of customer. In the paper,we give a description language to describe the domain composition model. Then we give the mapping algorithm and mapping rule to convert the domain composition model together with the requirement of user into an executable Web services composition language. In the end,we give an example to demonstrate.

Keywords Web services,Web services composition,Domain engineering

1 引言

Web Services 的兴起标识了未来 Internet 软件之间的集成将是一种面向服务(大粒度、松耦合、动态绑定)的模式。Web Services 是由 URI(Uniform Resource Identifier)标识的软件应用,该应用的接口和绑定可通过 XML 制品进行定义、描述和发现,同时,该应用可通过基于 Internet 的 XML 消息协议与其它软件应用直接交互[1]。Web Services 采用简单、易理解的标准协议作为接口描述和交互描述的规范,完全屏蔽了不同软件平台的差异,为企业之间进行跨平台的应用组装提供了技术基础。Web Services 技术促进了软件复用技术的发展,其中,对于 Web Services 组装技术的研究成为了目前研究热点之一,各大 IT 公司纷纷制定 Web Services 的组装标准,如 IBM 先后推出的 WSFL(Web Services Flow Language)[2]和 BPEL4WS(Business Process Execution Language for Web Services)[3],微软推出的 XLANG[4],以及 SUN 和 BEA 主推的 WSCI(Web Services Choreography Interface)[5]等。

与传统构件组装技术相比,Web Services 组装是基于过程的组装,在 Web Services 语境下,术语"组装"是指将 Web Services 或 Web Services 细粒度的操作组织起来构造一个可执行的商业过程[6]。上述介绍的组装标准实际上是 Web Services 的组装描述语言标准,用以描述 Web Services 的组装模型,即描述构成应用系统的 Web Services 及其交互关系。其描述结果是一个可执行的组装描述,可通过相应的工具进行解析,形成一个运行的应用系统。这些解析工具有 IBM 的 BPEL4J,微软的 BizTalk 等。

目前 Web Services 已被应用到许多领域中,对同一领域而言,领域内的应用系统既具有共性,又有差异性,且同一应用系统在不同阶段、不同环境中均会有所变化。在软件复用中,领域共性为应用系统集成组装提供了基础,变化性则确定了个体系统特性,在一个特定的系统中领域共性与变化性共生。所以分析、提炼领域共性,控制变化性是成功而有效地实施组装技术的关键。然而,对于 Web Services 的组装,当前的研究仅仅考虑了 Web Services 之间交互关系的逻辑,没有将领域的变化性考虑进去,这样在进行 Web Services 组装时缺乏一定的灵活性。为此,本文提出了一种支持领域变化性的 Web Services 的组装方法,该方法能较好地适应用户变化的需求,提高了 Web Services 组装的灵活性。

2 领域特性及领域组装模型

领域工程的作用是对一个领域中的若干系统进行分析,识别这些系统的共同特征和可变特征,对这些特征进行抽象,形成领域模型,依据领域模型产生出领域中一类应用系统共同具有的构架,并以此为基础识别、开发和组织可复用构件[7]。其中"领域"是指一组具有相似或相近软件需求的应用系统所覆盖的功能区域。领域模型是针对领域的需求规约模型,表达了一族具有共性和变化性的系统。考察领域中现有系统的需求时,会发现这些需求体现出一定的变化性。一些需求是被所有系统共同具有的,一些需求只被个别系统具有,而其它需求被部分系统具有。可以将这些需求分为以下三类[8]:

◇必须的(Mandatory):即领域中所有系统都必须具有

*)基金项目:国家"八六三"高技术研究发展计划项目(No. 2001AA3070)资助课题。赵俊峰 讲师,博士研究生。

的本质的且共同的需求，它体现了该领域中系统的共同性。

◇可选的(Optional)：部分现有系统具有这类需求，但并非全部系统都具有。它体现了领域中系统间的变化性。

◇ 多选一的(Alternative)：即领域中的系统在多个需求中，必须选且只能选其中一项的需求。这是一组互相之间存在着互斥关系的需求。假设需求 p、q、r 是这样的一组需求，当单独地考察每项需求时，它们都是可选的需求，但是，一个特定的系统必须具有其中的一项需求，又只能具有其中的一项，即，$\langle p,q,r\rangle$形成互斥集。这类需求体现了系统间的变化性。

另外，领域中具有变化性的需求间还包括依赖、互斥关系。

◇ 依赖关系是指只有在需求 p 存在的情况下，才能存在需求 q，这时称需求 q 依赖于需求 p。

◇ 互斥关系是指需求 p 和 q 不能同时存在于一个系统中。上面讨论的多选一的需求是具有互斥关系的需求的一种特殊情况。

领域模型是面向问题域的。它是领域中所有实体的行为表示，说明了哪些功能被执行，哪些数据、信息和实体在功能间流动。领域模型包括了业务模型、业务过程和应用系统功能，但它不表示过程和功能如何实现。其中，业务模型反映了业务策略或操作概念。业务过程定义了达到反映在业务模型中的目标所需的业务功能以及功能之间的流。应用系统功能是为实现业务过程所需的更加详细、更加技术的功能。

由于 Web Services 组装是基于过程的组装，为此我们将领域模型中的业务过程抽取出来，同时将反映领域变化性的特征加入，形成领域组装模型。领域组装模型代表了该领域内一族具有共性和变化性系统的业务过程，通过领域组装模型和用户的具体需求可以确定该应用系统中哪些服务是必须的、可选的和多选一的，并且可以根据需求之间的依赖和互斥关系来确定服务之间的关系，用以指导应用系统的组装。

这样，当开发同一领域中的应用系统时，可以根据用户需求和领域组装模型，确定应用系统的需求规约，并以此为基础进行应用系统的组装。结合领域组装模型进行应用系统的组装，会给组装带来很大的灵活性和适应性。在用户需求经常变更的情况下，采用领域工程的方法可以大大提高系统的可维护性。

3 支持领域变化性的 Web Services 组装

为了支持领域变化性，需利用领域模型指导 Web Services 组装，本文讨论的重点在于如何利用领域工程已经得到的结果对 Web Services 组装进行指导，因此对于如何利用领域工程得到该领域模型的过程不是本文的重点，在此不作阐述，具体的实施方案可参见文[9,10]。除此而外，在进行 Web Services 组装过程中，很重要的一步是进行 Web Services 的发现，本文假设所需的 Web Services 均已发现(如采用 UDDI 等技术)，直接就可利用这些 Web Services 的访问地址进行绑定运行。

3.1 支持领域变化性的 Web Services 组装过程

图 1 所示的是支持领域变化性的 Web Services 组装过程。首先需要对某领域实施领域工程，获得该领域的领域模型，并从中得到领域组装模型，为了有效地描述领域组装模型，我们引入了领域组装模型的描述语言 DSWSCL(Domain Specific Web Services Composition Language)对领域工程所得结果进行充分的描述。接着结合用户具体需求生成系统组装模型，系统组装模型主要描述的是构成某种具体应用的服务及服务之间相互关系。最后我们将此组装模型转换成可执行的组装描述，利用相应工具进行解析，生成可运行的应用系统。在这一过程中，起着关键作用的是：

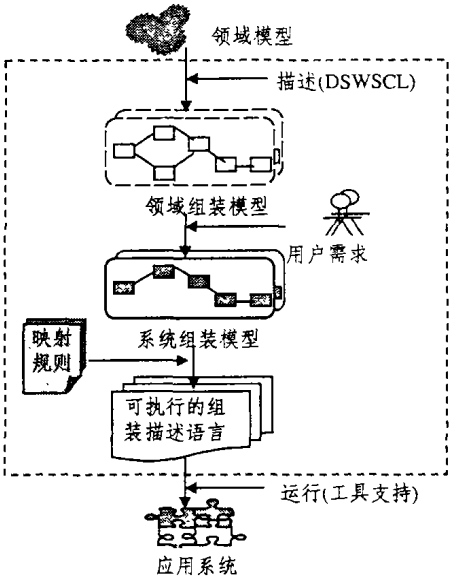

图 1 支持领域变化的 Web Services 的组装过程

· 领域组装模型描述语言 DSWSCL——有效地描述领域组装模型；

· 领域组装模型和用户需求的解析——结合用户需求，固定领域变化性，形成系统组装模型；

· 映射规则——将系统组装模型转为可执行的组装描述。这样利用相应工具解析，最终成为可运行的系统。

下面我们将详细介绍这几个部分。

3.2 领域组装模型描述语言

为了更好地将领域工程的结果应用到系统组装中，我们引入 DSWSCL 对领域组装模型进行描述，用以指导用户进行应用系统的组装。在 DSWSCL 中关键是对服务和服务之间的关系进行描述，我们采用 BNF 范式的规则进行描述，其中带引号的部分表明是关键字，尖括号的部分表明此部分还需展开说明，小写字母的名词表明具体的值，服务及其关系的描述规则分别见表 1、表 2。我们现在给出 DSWSCL 中的基本元素——服务和服务之间的关系定义：

· 服务：指完成某种功能的 Web Services 形态的构件。在服务中描述了服务的名称、功能概述、提供者等基础信息，还描述了运行该服务所需的参数及输出结果。除此而外，我们在服务的描述中加入了服务属性的描述，用“SERVICE PROSPERITY”标识，服务属性的取值是在“Mandatory”、“Optional”、“Alternative”中选择其一，用以标定该服务在该领域中的特性，反映了领域工程的结果。对于属性是“Mandatory”的服务，用户在构造系统时必须包含该服务。对于属性是“Optional”的服务，用户在构造系统可以根据具体需求进行选择，对于属性是“Alternative”的服务，用户不仅要根据具体需求进行选择，还要淘汰掉与该服务互斥的服务。为了指明服务之间的相互作用关系，在服务的描述中还加入描述服务行为的前置和后置条件：

· 前置条件：用以指明正确运行该服务所需要的条件。

· 后置条件：表示服务正确运行后的状态。

在服务接口中，前置条件和后置条件的描述规则分别用 PRE_CONDITION 和 POST_CONDITION RULE 标识。

表 1 服务的 DSWSCL 描述(部分)

```
<SERVICE>::= "SERVICE" serviceName ";"
             "SERVICE PROSPERITY" "(" < prosperity > ")" ";"
             "SERVICE PROVIDER" "(" providerName
               [,providerAddress]")" ";"
             "DESCRIPTION" "(" descriptionInfo ")" ";"
             "REQUIRE_PARA" ["(" <parameter list> ")"] ";"
             "PROVIDE_PARA" ["(" <parameter list> ")"] ";"
             ["PRE_CONDITION" "(" <condition list> ")"] ";"
             ["POST_CONDITION" "(" < condition list > ")"] ";"
             ......
             "END SERVICE" ";"
<prosperity>::= "MANDATORY" | "OPTIONAL" | "ALTERNATIVE"
<parameter list>::= "VOID" | <parameter> {", " <parameter>}
<parameter>::= [<parameterType>] parameterName
<parameterType>::= "INT" | "STRING" | "BOOL" | "DATE" | ......
......
<condition list>::= <expression> {"AND" | "OR" <expression>}
<expression>::= varName <OP> Value |
<OP>::= < | <= | > | >= | = =
......
```

· 服务之间的关系:文中 1.1 节已提到,需求之间的关系分为依赖关系和互斥关系,在领域模型中我们也将服务之间的关系定义为依赖关系和互斥关系,这是从整体的角度来定义服务之间全局交互关系,并且该关系与服务属性定义结合在一起可用以定义系统的整体结构,及系统中应出现的服务及服务之间的关系。而服务中所包含的前置和后置条件定义的是两两服务之间的交互关系,即定义的是服务的执行序列。下面我们给出对于这些关系约束的描述规则:

"CONSTRAINT TYPE"代表服务之间的关系的类别,"DEPENDENCE"表示依赖关系,此时在"SERVICE LIST"中以一个二元组来表示服务之间的关系,其中此二元组的形式为(被依赖方,依赖方),当被依赖方未被选定时,其依赖方也不能被选定。"ALTERNATIVE"代表互斥关系,此时在"SERVICE LIST"中以一个多元组来表示服务之间的关系,其中"SERVICE LIST"中的元素个数不得少于两个,此中的形式为(互斥方 1,…互斥方 n)。

表 2 服务关系的 DSWSCL 描述(部分)

```
RULE ::= "RULE" rule_id ";"
           "CONSTRAINT TYPE" "DEPENDENCE"
         | "ALTERNATIVE" ";"
           ["CONDITION" <condition> [{";"<
      condition> "}"]] ";"
           "SERVICE LIST" "(" service name1, service
      name2 "{" "," service name "}" ")" ";"
           "END RULE" ";"
condition::= "IF" "(" <expression list> ")" "THEN" statement
expression list::=<expression> {"AND" | "OR" <expression>}
expression ::= conName <OP> conValue
OP::= < | <= | > | >= | = =
......
```

"CONDITION"代表约束条件,这是一个条件布尔表达式,表示在满足某条件的情况下,选择某种特定的服务。

利用领域工程进行 Web Services 组装时,需要将变化性固定下来,领域组装模型为变化性的固定过程提供依据。可根据用户的具体需求将领域模型中可变的部分确定下来,得到具体的系统组装模型,上述我们阐述了领域组装描述语言 DSWSCL 是如何对领域工程的结果进行描述的,下面我们介绍如何结合用户的需求,导出采用某种组装描述语言描述的 Web Services 可执行的组装描述,最终完成满足用户需求的应用。

3.3 领域组装模型的解析

领域组装模型的解析主要是根据领域模型中的规则对用户的需求进行判定,固定变化性,得到一个符合用户需求的系统组装模型。

在进行解析时,主要进行两部分的判定,首先要在满足用户需求的所有服务中,根据领域组装模型判定哪些服务是必须的服务,哪些服务是可选的服务,哪些服务之间是互斥关系,哪些服务之间是依赖关系,从而形成待组装系统的服务集合。我们可以根据服务的属性先进行筛选,对于是必须的服务直接选入系统组装模型中,而对于可选的或互斥的服务就要进一步判断了,对于在同一互斥集中的服务需要根据用户具体需求和服务之间的互斥关系判定应去除哪些服务,若用户的需求不足以进行判定的话,则提示用户提供更精确的限制条件。对于具有依赖关系的服务,应将依赖方与被依赖方全部都包含在待组装系统的服务集合中。其次需要将对服务的前置条件和后置条件进行解析,形成 Web Services 组装模型中服务之间的关系,该关系主要描述的是服务之间的调用或激活关系。这样,通过以上对领域组装模型的解析就形成了特定于用户需求的系统组装模型。系统组装模型是领域组装模型的具体化和实例化,当用户需求更改时,可以根据用户新的需求对领域组装模型进行解析,形成新系统的组装模型。这样,大大提高了应用系统的可维护性。

表 3 所示的是将领域组装模型具体化和实例化的解析判定算法部分片断,该算法的输入是用户的需求和领域组装模型的描述规约,输出是确定了变化性的系统组装模型的具体描述规约。我们在该算法中引入两个集合用以存放判定后的结果,ServiceSet 存放根据领域组装模型和用户需求得到的系统组装模型中的服务列表;relationshipSet 存放系统组装模型中服务之间的关系,这一部分是依据用户需求对领域组装模型中服务之间的关系判定而生成的。

3.4 组装模型的映射

映射规则是将系统组装模型映射为采用某种组装描述语言描述的可执行组装描述,如 WSFL。映射规则由一组映射关系组成,此映射规则的形式为:被映射方→映射方,被映射方主要是由组装模型中的描述成分构成,而映射方由具体的可执行组装语言的成分构成。我们将此规则写成一个配置文件,这样我们只需替换配置文件就可获得采用不同描述语言所描述的可执行组装描述,进一步提高了转换的自动性,为用户提供了方便。在本文中我们以 WSFL 为例阐明映射的过程,我们将 serviceSet 中的元素映射为 WSFL 活动(activity),relationshipSet 映射为活动之间的控制链(controlLink),依次类推,组装模型就可转换成 WSFL 描述的可执行组装描述。下面我们以 WSFL 举一个实例来说明。

表 3 领域模型和用户需求解析过程的片断

```
.....
if 约束规则判定集不为空                    //确定服务及服务之
间的关系
      则取一条未判定的规则 rule (i)
      if CONSTRAINT TYPE==DEPENDENCE
      将 rule (i) 中依赖方与被依赖方放入 serviceSet
      else CONSTRAINT TYPE==ALTERNATIVE
            if rule(i).condition==true
                  从 rule(i).services_list 中选出放入
            serviceSet
.....
if serviceSet 中还有服务未被判定          //得到服务的
执行序列
      取出此服务的前置后置条件进行判定
      判定是否存在与其他服务的交互关系,若有则形
      成一个二元组<源方,目的方>放入 relationshipSet
      .....
```

4 实例

4.1 领域组装模型的描述

例如需要一个旅程预订的应用，该应用根据用户的需求进行交通票的预订、住宿预订、购买保险等活动，最后生成一个旅程安排计划表提交给用户。通过领域工程，我们得到该领域组装模型，其中的业务功能包括预定机票服务、预定住宿服务等，采用 DSWSCL 描述的预定飞机票服务如下所示：

```
. SERVICE bookPlaneTicket;
    SERVICE PROSPERIT (MANDATORY);
    SERVICE PROVIDER (ticketCompany,
    http://www.ticket.com);
    DESCRIPTION (to book the plane ticket);
    REQUIRE_PARA (DATE takeOffTime, STRING
    plane type);
    PROVIDE_PARA (BOOL bookSuccessOrNot);
    PRE_CONDITION (the tripTime < = 3days AND
    completet_task(bookHotel) = =TRUE);
END SERVICE;
```

在该组装模型中，还描述了服务之间的关系，一种为依赖关系，如声明只有购买飞机票后才能进行购买航空保险；另一种为互斥关系，如阐明在进行交通票预定时，到达同一目的地的选择方式只能在飞机、轮船和火车之间选定一种，这两种关系的示例表示如下：

```
RULE rule_1;
CONSTRAINT TYPE DEPENDENCE;
SERVICE LIST(bookPlaneTicket,bookInsurance);
END_RULE;
RULE rule_2;
CONSTRAINT TYPE ALTERNATIVE;
CONDITION IF (用户的往返天数<3天 AND 经费)
2000元) THEN CHOOSE 预定飞机票
SERVICE LIST (预定飞机票,预定轮船票,预定火车
票);
END_RULE;
```

在规则1中表明，若有购买航空保险的服务存在，则相关的购买飞机票服务也须存在。在规则2中，CONDITION 为约束条件，若用户不满足该条件则不能选择预定飞机票服务。

4.2 可执行组装描述的生成

依据上述的领域模型，现假设某一用户提出如下的旅程需求：

{… 往返时间<3天，需购买航空保险，需住宿一天，经费不限，…}

那么根据用户需求，利用该方法可知，购买航空保险的服务的被依赖方是购买飞机票服务，而用户的条件满足选用购买飞机票服务的条件，通过解析可以确定的服务必有购买飞机票和购买航空保险服务，它们之间的关系也可利用领域组装模型得出，最后再采用映射规则导成采用 WSFL 描述的 Web Services 的执行模型描述，其部分描述如下：

```
〈flowModel name="Schedule Itinerary" 〉
 〈activity name=" bookPlaneTicket"〉
  〈input message="tns: TicketOrder"〉
  〈output message="tns:TicketInfo"〉
      〈performedBy serviceProvider="ticketCompa-
       ny"〉
  〈/activity〉
   〈activity name="bookInsurance"〉
    〈input message="tns: TicketInfo"〉
    〈output message="tns: InsuranceInfo"〉
    〈performedBy serviceProvider = " Insurance-
     Company"〉
  〈/activity〉
    ……
    〈controlLink source=" bookPlaneTicket" tar-
     get="buyInsurance"/〉
 ……
〈/flowModel〉
```

若用户的需求有所改动，用户不再考虑时间因素，而考虑经费的因素，变更如下：

{…，往返时间>3天，需购买航空保险，需住宿一天，经费<1000元，…}

此时通过对用户的需求和领域组装模型的解析可知，由于预订飞机票的条件不满足了，因此取消预订飞机票的服务，而购买航空保险的服务是依赖于预订飞机票的，那么此时应取消购买航空保险的服务，它们之间的关系也不存在了。此时根据用户的需求，选用预订火车票的服务，再根据映射规则，最终得到的 WSFL 描述如下：

```
〈flowModel name="Schedule Itinerary" 〉
 〈activity name="bookTrainTicket"〉
  〈input message="tns: trainTicketOrder"〉
  〈output message="tns:trainTicketInfo"〉
  〈performedBy serviceProvider = " trainTicket-
  Company"〉
 〈/activity〉
 ……
〈/flowModel〉
```

从这两个例子可以看出，采用了支持领域变化性的组装模型可以更好适应用户的需求，提供了一定程度上的灵活性。

结束语 对于 Web Services 组装，如何灵活地适应用户不断变化的需求是目前所要解决的关键问题之一。领域工程的作用是针对一个应用领域，对领域中的若干系统进行分析，识别这些系统的共性和变化性成分，用以指导系统的组装。本文结合领域工程所得到的知识，给出了一种支持领域变化性的 Web Services 的组装方法，通过该方法可以自动完成用户需求到可执行的组装描述之间的转换，从而完成用户所需的应用系统。对于此方法的后续研究集中在如何根据服务质量进行服务的选择，为用户提供一个更强大更灵活的 Web Services 组装技术基础。

参 考 文 献

1 Austin D, Barbir A, Garg S. Web Services Architecture Requirements. http://www.w3.org/TR/2002/WD-wsa-reqs-20020429, April 2002

2 Leymann F. Specification: Web services Flow Language (WSFL 1.0). IBM Software Group, May 2001

3 Andrews T, et al. Specification: Business Process Execution Language for Web Services Version 1.1, 05 May 2003

4 Thatte S. Specification: LANG-Web Services for Business Process Design. Microsoft Corporation, 2001

5 Askary S, Fordin S, Orchard D, et al. Specification: Web Service Choreography Interface 1.0, BEA Systems. Sun Microsystems, Intalio Co., SAP, 2002

6 Mcgovern J, et al. Java Web Services Architecture, Morgan Kaufmann Publishers, 2003

（下转第30页）

15 Buchholz P. Equivalence and aggregation of GSPNs with labeled transitions. In: Proc. of the 9thPNPM, 2001. 81～90
16 Trivedi K S. SHARPE 2002: Symbolic Hierarchical Automated Reliability and Performance Evaluator. In: Proc. of the Int'l Conf. on Dependable Systems and Networks, 2002. 544
17 Deavours D, Clark G, Courtney T, et al. The Mobius framework and its implementation. IEEE Trans on Software Engineering, 2002, 28(10): 956～969
18 Franceschinis G, Gribaudo M, Iacono M, et al. Towards an object based multi-formalism multi-solution modeling approach. In: Proc. of the 2nd Workshop on Modeling of Objects, Components and Agents, 2002. 47～66
19 Boucherie R J. A characterization of independence for competing Markov chains with applications to Stochastic Petri nets. IEEE Trans Software Engeneering, 1994, 20(7): 536～544
20 Balbo G, Bruell S C, Sereno M. Embedded processes in Generalized Stochastic Petri nets. In: Proc. of the 9th PNPM, 2001. 71～80
21 Balbo G, Bruell S C, Sereno M. On the relations between BCMP queueing networks and product form solution Stochastic Petri nets. In: Proc. of the 10th PNPM, 2003. 103～113
22 刘道斌, 林闯, 陆维明. 非乘积解随机 Petri 网的乘积形式近似求解. 计算机学报, 2001, 24(6): 588～595
23 Chiola G, Anglano C, Campos J, et al. Operational analysis of timed Petri nets and application to the computation of performance bounds. In: Proc. of the 5th PNPM, 1993. 128～137
24 Allmaier S C, Kowarschik M, Horton G. State space construction and steady state solution of GSPNs on a shared-memory multiprocessor. In: Proc. of the 7th PNPM, 1997. 112～121
25 Ciardo G. Distributed and structured analysis approaches to study large and complex systems. Formal Methods and Performance Analysis, LNCS 2090, 2001. 344～374
26 林闯, 曲扬, 郑波, 等. 一种随机 Petri 网性能等价化简与分析方法. 电子学报, 2002, 30(11): 1620～1623
27 Haverkort B R, Ost A. Steady-state analysis of infinite Stochastic Petri nets: comparing the spectral expansion and the matrix-geometric method. In: Proc. of the 7th PNPM, 1997. 36～45
28 Ciardo G. Discrete-time Markovian Stochastic Petri nets. In: Proc. of the 2nd Int'l Workshop on Numerical Solution of Markov Chains, 1995. 339～358
29 Jones R, Ciardo G. On Phased Delay Stochastic Petri nets: definition and an application. In: Proc. of the 9th PNPM, 2001. 165～174
30 Ajmone Marsan M, Chiola G. On Petri nets with deterministic and exponentially distributed firing times. Advances in Petri Nets, LNCS 266, 1987. 132～145
31 Choi H, Kulkarni V G, Trivedi K S. Markov regenerative stochastic Petri nets. Performance Evaluation, 1994, 20: 337～357
32 Telek M, Pfening A. Performance analysis of Markov regenerative reward models. Performance Evaluation, 1996, 27: 1～18
33 Bobbio A, Puliafito A, Telek M. A modeling framework to implement preemption policies in non-Markovian SPNs. IEEE Trans on Software Engineering, 2000, 26(1): 36～54
34 German R, Lindemann C. Analysis of stochastic Petri nets by the method of supplementary variables. Performance Evaluation, 1994, 20: 317～335
35 German R, Logothetis D, Trivedi K S. Transient analysis of Markov Regenerative Stochastic Petri nets: a comparison of approaches. In: Proc. of the 6thPNPM, 1995. 103～112
36 German R, Telek M. Formal relation of Markov renew theory and supplementary variables in the analysis of Stochastic Petri nets. In: Proc. of the 8th PNPM, 1999. 64～73
37 Trivedi K S, Kulkarni V G. FSPNs: Fluid Stochastic Petri Nets. In: Proc. of the Petri Nets'93, 1993. 24～31
38 Horton G, Kulkarni V G, Nicol D M, et al. Fluid stochastic Petri nets: theory, applications and solution. European Journal of Operation Research, 1998, 105(1): 184～201
39 Gribaudo M, Sereno M, Bobbio A. Fluid Stochastic Petri Nets: An extended formalism to include non-Markovian models. In: Proc. of the 8th PNPM, 1999. 74～82
40 Wolter K. Second Order Fluid Stochastic Petri Nets: an extension of GSPNs for approximate and continuous modeling. In: Proc. of the WCSS'97, 1997. 328～332
41 Wolter K. Jump transitions in second-order FSPNs. In: Proc. of the IEEE MASCOTS'99, 1999. 156～163
42 Alla H, David R. Continuous Petri nets. In: Proc. of the Petri Nets'87, 1987. 275～294
43 David R. Modeling of hybrid Systems using Continuous and Hybrid Petri Nets. In: Proc. of the 7th PNPM, 1997. 47～57
44 David R, Alla H. On Hybrid Petri nets. Discrete Event Dynamic Systems, 2001, 11(1): 9～40
45 Horton G. Computation of the distribution of accumulated reward with Fluid Stochastic Petri nets. In: Proc. of the 2nd IEEE IPDS, 1996. 90～95
46 Wolter K, Zisowsky A. On Markov reward modeling with FSPNs. Performance Evaluation, 2001, 44: 165～186
47 Tuffin B, CHEN D S, Trivedi K S. Comparison of hybrid systems and Fluid Stochastic Petri nets. Discrete Event Dynamic Systems, 2001, 11(1): 77～95
48 German R, Gribaudo M, Horva'th A, et al. Stationary analysis of FSPNs with mutually dependent discrete and continuous parts. In: Proc. of the 10th PNPM, 2003. 30～39
49 Gribaudo M, Horva'th A. Fluid Stochastic Petri nets augmented with flush-out arcs: a transient analysis technique. IEEE Trans on Software Engineering, 2002, 28(10): 944～955
50 Horva'th A, Gribaudo M. Matrix geometric solution of Fluid Stochastic Petri nets. In: Proc. of the 4th Int'l Conf. on Matrix-Analytic Methods in Stochastic models, 2002. 37～49
51 Tuffin B, Trivedi K S. Importance sampling for the simulation of Stochastic Petri nets and Fluid Stochastic Petri nets. In: Proc. of High Performance Computing, 2001. 228～235
52 Ciardo G, Nicol D M, Trivedi K S. Discrete-event simulation of Fluid Stochastic Petri nets. IEEE Trans on Software Engineering, 1999, 25(2): 207～217

(上接第20页)

7 Arango G, Prieto-Diaz R. Domain Analysis Concepts and Research Directions. In: Domain Analysis and Software System Modeling, R. Prieto-Diaz and G. Arango, eds. IEEE Computer Society Press, Los Alamitos, California, 1991
8 Matskin M, Rao Jinghai. Value-Added Web Services Composition Using Automatic Program Synthesis. In: CAiSE 2002 Intl. Workshop, WES 2002, Toronto, Canada, May 2002
9 李克勤. 面向对象的领域工程方法研究. [博士论文]. 北京大学, 2000
10 Tracz W, Coglianese L. Domain-Specific Software Architecture Engineering Process Guidelines (Version 2.1). ADAGE-IBM-92-02B, April 1994
11 Zeng L, Dumas M. Quality Driven Web Services Composition. ACM1-58113-680-3/03/0005
12 Benatallah B, Sheng Q Z. The Self-Serv Environment for Web Services Composition. Published by the IEEE Computer Society, 2003
13 Narayanan S, et al. Verification and Automated Composition of Web Services, ACM1-58113-449-5/02/0005
14 张文娟, 赵俊峰, 谢冰, 杨芙清. 一种支持变化性的构件模型 JBCOM/E. 电子学报, 2003(6): 899～902
15 Allen R J. A Formal Approach to Software Architecture. [PhD Thesis]. Carnegie Mellon Univ., CMU Technical Report CMU-CS-97-144, May 1997
16 张文娟. 支持变化性的软件构件模型其相关技术研究. [博士论文] 北京大学, 2002