服务组合BPEL测试序列生成研究

张 亚
(江苏自动化研究所 连云港 222006)

摘 要 为更好地对Web服务组合的控制流进行分析和验证,提出一种映射转换模型和测试序列生成算法。首先解析基于BPEL的Web服务组合流程描述文件,然后对流程描述文件进行图形化转换,形成业务流程编排控制流程图,再采用路径推导算法对控制流程图进行解析,获取基于BPEL的服务组合的所有测试执行路径,最后通过一个服务组合实例证明模型及算法的有效性。该方法解决了测试路径的自动生成和全覆盖问题,保证了测试的充分性,提高了路径生成效率。

关键词 Web服务,组合测试,映射转换,BPEL流程,组合路径
中图法分类号 TP311.5 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2017.01.039

Research on BPEL Test Sequence Generation for Web Services Combination

ZHANG Ya
(Jiangsu Institute of Automation, Lianyungang 222006, China)

Abstract A kind of concurrent-based mode of control flow was proposed for the analysis and verification of interactions of Web services composition, which contains many complex concurrent behaviors using BPEL language. A formal for concurrent control flow and an efficient algorithm were also presented to generate BPEL testing sequence. First, we discussed many possible situations in Web services process by analyzing BPEL files, and translated the BPEL process source code into concurrent flow diagram to simplify the model. Then, the algorithm of executive paths of services composition process was discussed, and the algorithm can find out the sum of all executive paths and node passed by the paths. Finally, an example of composite service was given to proof the usability. The algorithm is the basis of full-scale test of Web services composition process, the research and implement of web services composition.

Keywords Web services, Composition testing, Mapping conversion, BPEL process, Composition route

1 引言

Web服务是面向服务架构(Service-Oriented Architecture,SOA)的一种主流实现形式,是一种自治、开放及平台无关的网络构件。SOA的出现也使得分布式系统向一种更灵活的体系结构进化。Web服务组合整合已有的异构Web服务,产生具有新功能的组合服务,实现服务的复用。BPEL(Business Process Execution Language)是业界认可的标准,也是SOA实现组合服务和服务编排的重要技术基础,其规范和标准已经十分完备。与普通仅有WSDL接口描述的原子服务不同,组合服务除了具有WSDL接口描述,还具有一个能够反映其内部结构的BPEL执行过程描述。BPEL类似于程序中的代码,有了代码即可考虑从白盒角度来依据代码生成测试用例。

近年来,WS-BPEL成为了Web服务组合的应用标准,研究人员花费大量精力进行相关研究和应用。芬兰Eindhoven University of Technology的研究小组做了许多工作,并开发了相应的测试工具。在文献[1,2]中,W. M. P. van和C. Ouyan将BPEL流程协议转化为Petri网,把相关的SOAP消息在Petri网连接线上展现,以此验证一致性,并分析了多服务交互中的某个参与方是否符合约定的服务行为。在文献[3]中,R. Hinz等人将BPEL流程映射为Petri网,用CTL逻辑检查验证Petri网的标准属性。H. Verbeek等人在文献[4]中定义一种特殊的Petri网(工作流网)用于验证标准属性,如结点的不可达性、工作流网的可终止性,同时考虑BPEL中的〈link〉,〈join condition〉和DPE。德国Humboldt University Zu Berlin的研究小组使用Petri网在BPEL建模、验证方面做了许多工作,并开发了相应的测试工具。K. Schimidt和C. Stahl在文献[5-9]中通过给出几个示例的方法讨论了BPEL到Petri网的映射。N. Lobmarul在文献[10]中针对BPEL的2.0版本,完成了用Petri网表示BPEL的语义。国内近年开展了相关研究,文献[11-15]从Petri网建模等方面考虑服务组合测试。

从上述研究现状来看,虽然目前存在一些Web服务组合测试方法,但多数侧重于BPEL流程的建模和形式化验证,仅提供了BPEL语言的部分覆盖,且没有对〈Flow〉结构的特殊性进行充分考虑,关于测试路径生成和覆盖等方面的研究还是存在一些不足,如BPEL包含多种类型的流程,包括可同时执行多个动作、非同步执行的事件处理器及错误补偿处理等。一个复杂流程的执行路径可能相当多,系统中可能存在非常多的流程需被测试,由于其非顺序执行的特性,导致找出所有可能的执行路径非常困难。因此如何自动生成可能执行路径,

到稿日期:2015-12-19 返修日期:2016-03-12 本文受总装十二五预研(513150502)资助。
张 亚 硕士,工程师,主要研究领域为可靠性测试、故障注入技术等,E-mail:zhya100200@163.com。

并实现全覆盖，对整个测试工作的进行十分重要。

本文针对服务组合测试路径生成存在的问题，通过分析 BPEL 文件建立控制流程图，充分考虑 BPEL 结构化活动因素间的交互关系，采用路径合并搜索和并行路径搜索结合的路径推导算法，最终给出测试路径集合，进一步降低服务组合测试的代价，提高测试覆盖率。本文首先使用基于 DOM4J 技术的 BPEL 映射转换方法，对基于 BPEL 的 Web 服务组合的流程描述文件进行图形化转换，形成文件结构图，得到 Web 服务组合业务的控制流程图；其次，使用路径合并改进搜索和并行路径搜索的算法推导 Web 系统服务组合的所有测试序列。

2 基于 DOM 树的 BPEL 映射转换

2.1 活动映射转换

BPEL 存在以下基本活动：调用某个 Web 服务的操作(invoke)；等待一条消息来响应由某人从外部进行调用服务接口的操作(receive)；生成输入/输出操作的响应(reply)；等待一段时间(wait)；把数据从一个地方复制到另一个地方(assign)；指明某个地方出错(throw)；终止整个服务实例(terminate)；或者什么也不做(empty)。通过使用 BPEL 语言所提供的结构化活动，可将这些原语活动组合成更复杂的算法。这些结构化活动提供的能力包括：定义一组步骤有序的序列(sequence)；使用“case-statement”方法来产生分支(switch)；定义一个循环(while)；执行几条可选路径中的一条(pick)；指明一组应该并行执行的步骤(flow)；在并行执行的一组活动中，可通过使用链接(link)来指明执行顺序的约束，并允许递归地组合结构化活动，以表达任意复杂的算法，这些算法表示了服务的实现。

这里提出一种 BPEL 映射转换方法，图 1 是 XML 格式的流程描述文件的 BPEL 映射转换示意图。

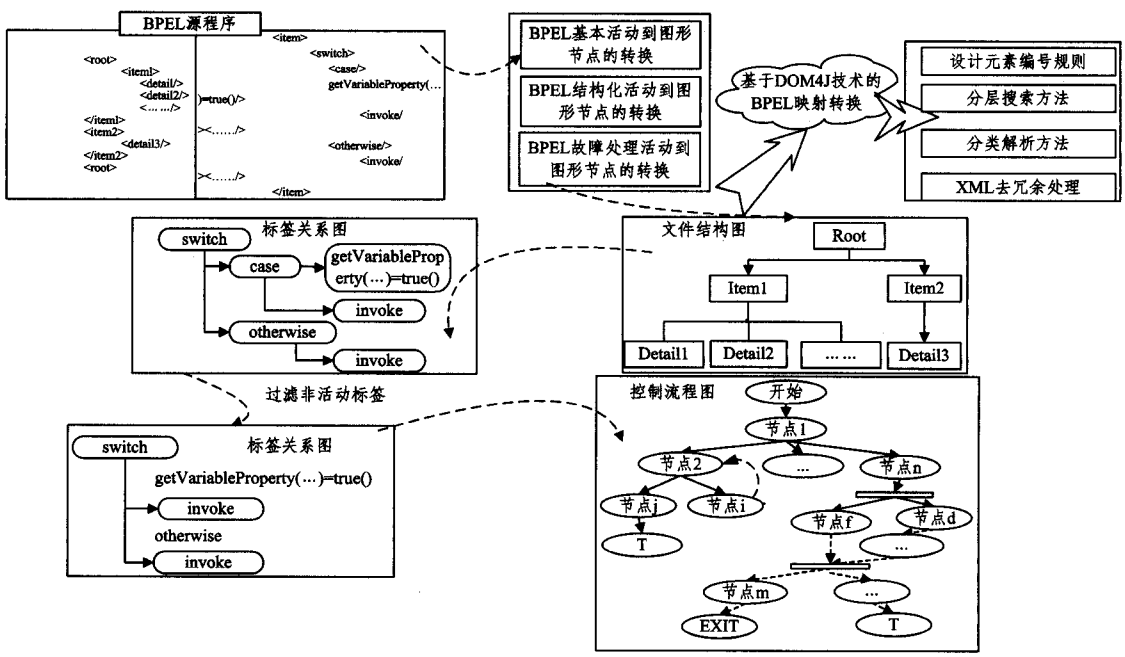

图 1 服务组合的测试路径生成工作流

为生成组合服务的控制流图，首先需要对反映组合服务编排逻辑的 BPEL 描述文件进行解析。解析过程首先获得反映 BPEL 文件结构的 DOM 树。根据 BPEL 文件的模式特征和 DOM 树中各个节点的 tag 标记，可以将 DOM 树节点解析为 BPEL 描述中的 import，partnerLink，variable，activity 等实体。实体间通过层次嵌套关系形成了组合服务的文档结构树。

文档结构图包含了 BPEL 组合服务的所有信息，但是并非该图中所有成分都是实际执行的实体，诸如描述信息导入(import 语句)、变量声明等结构主要提供执行所需的辅助信息而不是可执行的单元。为支持组合流程分析和测试用例生成，系统将辅助信息记录到一系列表格中，形成编译中类似“符号表”的实体。同时，提取可执行的活动单元，即 BPEL 中各类可执行的 Activity，构成标签关系图。该图主要反映了组合服务可执行活动之间的关联关系，记录了可执行语句之间的前后关联、从 if 语句到各个分支的联系等。

定义 1 定义二元组 $G=(N,E)$。其中 N 是可执行节点集，BPEL 中的可执行节点主要包括赋值语句、调用语句、if 语句、switch-case 语句、repeat-until 语句、foreach 语句等。E 是有向边的集合，每条有向边反映 BPEL 执行过程中的一个可能控制流转移。

控制流转移包括顺序语句逐条向下执行过程中的转移、if 语句到真假分支的转移、循环语句返回到循环条件的回退跳转转移等。

2.2 控制流程图转换算法

测试路径推导方法：

(1)对于不含并行流程〈flow〉的 BPEL，直接根据 BPEL 结构即可构建控制流图，遍历控制流图，可以获得所有服务执行路径。

(2)对于包含并行流程〈flow〉的 BPEL，首先分析〈flow〉活动中的所有 link 关系，找出 link 的源和目标获得。Link 关系表达了一种同步约束，对于普通算法构造出的控制流图，需要根据这些同步约束调整控制转义边，使得 link 关系的目标获得脱离原来的前驱节点，挂接到 link 的源活动下。如此，可得到一个考虑同步的控制流图。对该控制流图进行分析扫描，可以推导出所有可行的程序路径。分析过程首先将 Flow 获得块收缩为一个节点或者整个程序的路径组合，然后将 Flow 节点展开，考虑其中可能存在的并行执行关系，获得一

组并行路径。将 Flow 展开后的路径和 Flow 收缩前的路径进行笛卡尔乘积运算,可以得到所有路径。

(3)对于不含并行关系的一条 BPEL 中的执行路径,可以利用符号执行技术找出该路径对应的输入约束,求解约束,可以获得路径对应的测试输入。

(4)对于包含并行关系的一条 BPEL 中的执行路径,分析后展开节点,递归执行,直至最后不含并行关系的路径,利用符号执行技术求解约束,获得路径对应的测试输入。

下面给出基于控制流程图的转换算法。

```
输入:The bpelCode under test
输出:BPEL 初始控制流程图(MSG)
Arithmetic bpelCode To MSG(){//bpel 源代码向 MSG 转换
node E=bpelCode. CurTag;//节点 E 取 BPEL 代码当前行标签
While(E!="</Process>")
{//转换标签 process 内的活动模块
  node E =bpelCode. NextTag;//读取下一行标签
  Switch E:
    Case "<sequence>" node:/ * 处理 sequence 结构/ *
      squenceTransform();
      break;
    Case "<Pick>" node:/ * 处理 Pick 结构/ *
      PickTransform();
      break;
    Case "<Switch>",or "<if>",or"<While>" node:/ * 处理 branch
        和 While 结构/ *
      WhileTransform();
      break;
    Case "<flow>" node:/ * 处理 flow 结构/ *
      flowTransform();
      break;
    Default:break;
}

squenceTransform(){//顺序活动转换
  node E=bpelCode. NextTag;
  while(E!="</squence>")
  {//如果没有到该结构末尾
  If E="<receive>"node{//如果是基本结构,将其转换为活动节点,
    同时增加活动的末尾状态
  Sequence_Transform_MSG ();//构造顺序关系
  Else
  {//如果是复杂活动,则根据活动类型循环处理
  Construct MSG using E recursively;
  }
  node E =bpelCode. NextTag;//取下一个标签
}

PickTransform();{//Pick 结构转换
  node E=bpelCode. NextTag;
  while(E!="</Pick>")
  {//如果没有到该结构末尾
  If E="<onMessage>"node{//如果是基本结构,将其转换为活动节
    点,同时增加活动的末尾状态
  Exclusive_Transform_MSG ();//构造独占关系
  Else
  {//如果是复杂活动,则根据活动类型循环处理
  Construct MSG using E recursively;
  }
  node E=bpelCode. NextTag;//取下一个标签
}

WhileTransform(){//branch 和 While 结构转换
  node E=bpelCode. NextTag;
  while(E! ="</Switch>",or "</if>",or"</While>")
  {//如果没有到该结构末尾
  If E="<receive>"node{//如果是基本结构,将其转换为活动节点,
    同时增加活动的末尾状态
  Exclusive_Transform_MSG ();//构造顺序关系
  Else{//如果是复杂活动,则根据活动类型循环处理
  Construct MSG using E recursively;
  }
  node E =bpelCode. NextTag;//取下一个标签
}
flowTransform(){//flow 结构转换
  node E=bpelCode. NextTag;
  while(E!="</flow>") {//如果没有到该结构末尾
  If E !="</link>" //并行中不含有 Link
  Elseif E="<receive>"node{//如果是基本结构,将其转换为活动节
    点,同时增加活动的末尾状态
  Paratactic _Transform_MSG ();//构造并行关系
  Else{//如果是复杂活动,则根据活动类型循环处理
  Construct MSG using E recursively;
  }
  Else //并行中含有 Link
  Sequence_Transform_MSG ();//构造顺序关系
  node E =bpelCode. NextTag;//取下一个标签
}
```

3　测试路径推导算法

根据 2.2 节的算法能够得到 Web 服务组合的 BPEL 控制流程图。本文基于控制流程图,通过路径合并搜索和并行路径搜索算法获取测试路径集合。首先分析 BPEL 服务的控制流图,找出所有执行路径,进而采用推导算法为每条路径自动构造测试输入。

由于循环甚至递归结构的存在,组合服务的可行执行路径可能存在无限对,为所有路径生成测试用例显然不可行。因此,本文考虑的"所有"路径主要是无环路径。通过适当扩展,也可以枚举有一定上限的有循环路径。

路径枚举采用服务组合测试路径推导算法实现,用一个栈跟踪从控制流图沿着控制转移边向下遍历的过程,当遍历到一定深度,发现新遇到的节点已经处理过,出现循环回边时,将新路径导出,并加入到路径列表中。系统将遍历所得的所有路径汇报给用户,经用户筛选后的路径将反馈给后续算法,以生成测试用例。

下面给出测试路径推导算法的整体思路:

(1)首先在路径的第一条语句执行前,根据 BPEL 中包含的变量信息和 WSDL 文件中给出的变量类型信息,为每个输入变量分配一个符号值。如图 2 的例子中,为 x,y,z 分别分配了符号值 A,B 和 C。这些符号值有占位标记的作用,其对应的输入变量到底取何种值未知。一些服务的输入是复杂数据类型(complexType)。对这些类型,将其逐层展开,直到每个基本类型成员。而每个基本类型成员将当作独立的变量来

处理。算法最后生成的测试输入是基本类型成员按照其原先在复杂类型中的排布重新组装的结果。

(2)在给定的符号值下,系统将逐条执行路径中的每条语句,对于计算过程中获得的中间结果,在无法获得具体值的情况下,用输入变量上的符号表达式表示。比如图 2 的例子中,在执行完前两条语句后,x 的取值用符号表达式 $A-1$ 表示,而 y 的取值用 $B+1$ 表示。

(3)对于 if/switch/while 等判定语句,首先用符号表达式代替判定条件中的所有变量。图 2 中 $x==y$ && $y>z$ 的条件将被替换为 $(A-1)==(B+1)$ && $(B+1)>C$。如果程序走真分支,则将判定条件加入到整个路径的条件(path condition)中,表明要使程序走真分支,则该条件必须得到满足。对于图 2 的例子,路径条件 PC 为 $(A-1)==(B+1)$ && $(B+1)>C$。

(4)如果有对另外的服务操作 S 的调用,则显然调用实参是已知取值表达式的变量或常量,而 S 的输出结果未知。系统将也为 S 的输出结果分配一个符号标识。如果用户已经在整个待测项目的约束建模中,建立了从 S 服务的输出到输入的约束关系,则将该约束关系加入到路径条件中。

(5)在整个路径执行结束后,将得到由路径中所有判定条件、服务调用产生的一组约束,即前述路径条件。

(6)利用 Z3 约束求解工具可以获得一个满足路径约束的解。以该解作为组合服务的输入,可以保证服务按照预设的路径执行,达到期望的覆盖目标。

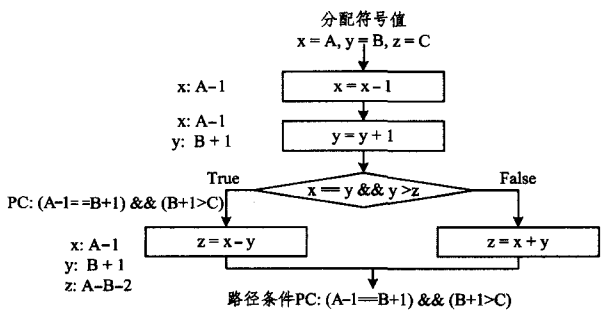

图 2 符号执行的约束关系示意图

本文中的约束关系采用线性不等式和布尔公式组合来表达,在原子约束的基础上,通过 AND,OR,NOT 3 种算子组合成更复杂、更庞大的约束组。

每个原子约束是一个线性不等式或布尔表达式。表达式由约束变量、常量和运算符构成。约束变量的形式如下:

value(svc\operation\message_xpath)

其中,value 是变量名,表示一个服务操作参数的取值。svc 是服务地址标识,用以区分不同的服务,系统允许定义与 C 语言中宏相似的结构来简化服务表达式的书写。定义为:

svc=http://test.com/axis2/services/Add

其中,operation 是操作标识,用以区分同一个服务的不同操作。message_xpath 是到基本类型数据的 xpath 路径表达,用以表征一个服务操作的输入中不同的子参数。一个约束变量可以用来表达服务输入参数或者输出数据的取值,而这些取值又可以表达数据、状态等信息。系统支持整数、位向量、布尔值、字符串枚举类型上的约束变量。

[1] http://sable.github.io/soot/, soot.toolkits.graph.UnitGraph

在约束变量的基础上,定义形如:value(svc\add\response\return)=value(svc\add\input\a)+value(svc\add\input\b),value(svc\add\response\return)="success"等的原子约束公式。该约束公式可以表达真实业务中,服务输出与服务输入之间的关系。这组关系能够支持组合服务中路径条件的生成,从而导出测试用例。

4 系统设计

由上文分析可知,系统主要提供测试执行路径生成功能,主要步骤如下:

(1)读入 XML 格式的 BPEL 源代码文件,自动生成业务流程的文件结构树。

(2)从测试的角度重点关注业务流程中的活动及其关系,以文件结构树为基础,过滤非活动标签,形成 BPEL 的标签关系图。

(3)利用服务组合的流程测试执行路径推导算法,给出业务流程所有的可能测试执行路径。

系统主要包括源码展示、服务文件结构树展示、服务标签关系图展示、服务控制流程图展示、测试路径推导展示共 5 个子模块。

(1)源码展示:以 TextEditor 控件呈现 WSDL 或 BPEL 的 XML 源码,通过源码清晰显示待测试服务。

(2)服务文件结构树展示:以 TreeView 的方式呈现 XML 文件的层次结构。结构树反映 XML 中元素之间的嵌套关系。文件结构树直接通过 XML 的 DOM 解析器获得,使用 Eclipse 的 TreeViewer 控件进行展示。

(3)服务标签关系图展示:过滤 XML 描述中的非活动标签,保存活动标签,比如 if,else,形成标签关系图。构建过程如图 3 所示。

图 3 标签关系图构建过程

(4)服务控制流程图展示:控制流程图类似程序流图一样展现 BPEL Process 的活动过程。参考 Java 程序分析工具 Soot[1],构建控制流图生成算法和数据结构。

运用 jgraphx 展示控制流图,bpel 组合服务中的多个服务的调用关系和执行路径,以及每条执行路径的触发条件都在控制流图上得到展示。

5 验证分析

本文选择某型号面向服务的舰船指控系统作为原型系统的验证实例。选取作战指挥模块的软件子功能作为测试对象,获取其服务组合业务流程描述 BPEL 文件,如图 4 所示,使用专家系统分析获取测试路径信息,整个业务流程共有 $90\times18=1620$ 条可执行路径。对于 While 结构的处理,将其转换为 IF 结构,不考虑循环次数对路径的影响。

结合研究团队十二五课题的相关工作,本文研制了 Web 服务组合测试路径自动生成原型系统 SOATest。本工具支持 Web 服务组合测试路径的自动生成,图 5 示出了 BPEL 文档结构树。

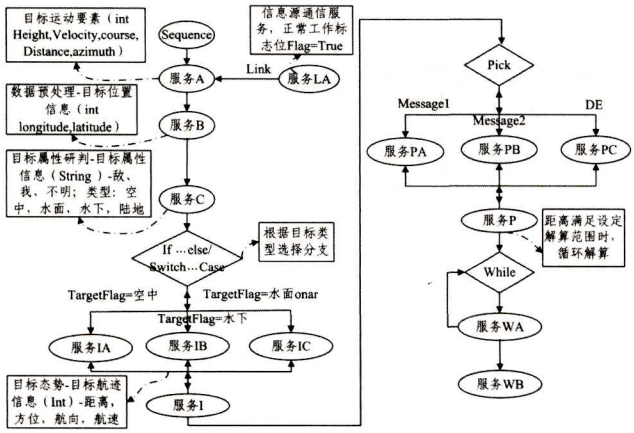

图 4　测试路径集合生成流程

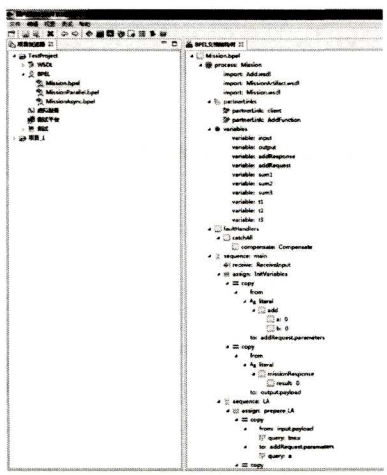

图 5　算法工具原型 BPEL 文档结构树

图 6 示出了 BPEL 标签关系图。

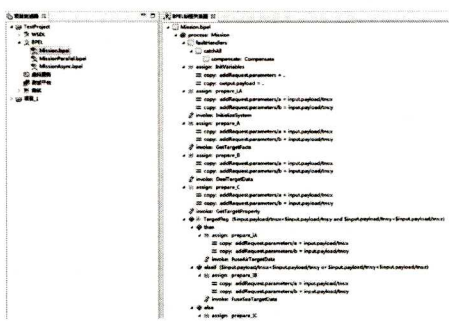

图 6　算法工具原型 BPEL 标签关系图

使用原型工具，解析 BPEL 文件，获取控制流程图和测试路径集合，如图 7 所示。

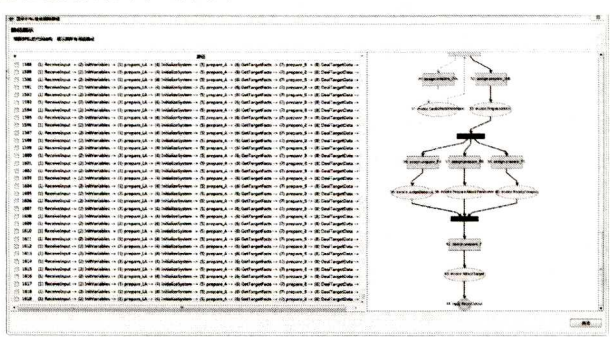

图 7　算法工具原型生成测试路径集合

为了证明本文提出的方法在测试路径自动生成和覆盖率上的优势，使用结构化活动和基本活动任意组合编排的业务流程作为验证对象，分别使用专家系统人工分析获取测试路径，使用原型工具自动生成测试路径，比较两者对实际存在可执行路径的覆盖情况。

表 1 给出了运用工具为某型号面向服务的舰船指控系统软件服务组合业务流程的 5 个案例生成测试路径的相关信息，包含了案例对应的服务个数、流程所饮食的活动类型、生成的路径数。实验表明，受制于专家水平和人为因素，随着组合服务规模的增大，专家获取的测试路径的实际覆盖大幅变差，而原型工具能够正确、高效地生成测试路径，且获取的测试路径集合能够完全覆盖实际执行路径。因此，本文提出的算法在测试路径覆盖率和自动生成方面具有优势。

表 1　测试路径结果信息

服务组合案例	服务个数	包含结构活动类型	路径总数		
			人工分析（多位专家）	原型工具	实际可执行路径
目标跟踪显示业务	12	sequence,Switch,flow	208,225,238	240	240
目标态势处理业务	10	sequence,While	158,160,160	160	160
敌我识别业务	8	sequence,flow	80,80,80	80	80
情报处理业务	16	sequence,Switch,Pick ,flow	680,656,617	720	720
目指打击业务	18	sequence,Switch,While,Pick,flow	1218,1460,1371	1620	1620

结束语　本文提出了一种基于 BPEL 的 Web 服务组合测试路径自动生成方法，通过将基于 DOM 树的控制流程图映射方法和基于符号执行技术的测试路径推导相结合，解决了服务组合的测试路径自动生成和全覆盖问题，排除测试人员水平等人为因素的影响，保证了测试的充分性，提高了生成效率。

在后期工作中，将结合考虑组合流程的控制流和数据流，对数据流层次进行详尽的研究，进一步解决 Web 服务组合的测试用例自动生成和自动测试问题。

参 考 文 献

[1] VAN DER ALST WM P,DUMAS M,OUYANGETA C. Choreography Conformance checking:An Approach based on BPEL and Petri Nets[C]//Proceedings of the 4th International Conference on Business Process Management. Vierula,Austria,2006.

[2] OUYAJLG C,VERBEEK E,VAN DER ALST W M P,et al. Formal semantics and analysis of control flow in WS-BPEL[J]. Science of Computer Programming Archive,2007,67(23):162-198.

[3] FARAHBOD R,GLSSER' U,VAJIHOLLAHI M. A formal semantics for the Business Process Execution Language for Web Services[C]//Web Services and Model-Driven Enterprise Information Services. 2005.

[4] VERBEEK H,VAN DER ALST W M P. Analyzing BPEL processes using Petri-Nets[C]//Second International Workshop on Application of Petri-Nets to Coordination Workflow and Business Process Management. 2005.

[5] SCHMIDT K,STAHL C. A Petri net semantic for BPEL4WS Validation and application[J]. Proceedings of the 11th Workshop on Algorithms and Tools for Petri-Nets. Paderbom,2004,75(21):1-6.

（下转第 225 页）

green energy in data-processing frameworks[C]// Proceedings of the 7th ACM European Conference on Computer Systems. ACM, 2012: 57-70.

[16] CARDOS M, SINGH A, PUCHA H, et al. Exploiting spatio-temporal tradeoffs for energy efficient MapReduce in the cloud [R]. Department of Computer Science and Engineering, University of Minnesota, 2010.

[17] CHE Y, GANAPATHI A, KATZ R H. To compress or not to compress-compute vs. io tradeoffs for mapreduce energy efficiency[C]// Proceedings of the First ACM SIGCOMM Workshop on Green Networking. ACM, 2010: 23-28

[18] SONG J, LI T T, ZHU Z L, et al. Benchmarking and analyzing the energy consumption of cloud data management system [J]. Chinese Journal of Computers, 2013, 36(7): 1485-1499. (in Chinese)
宋杰，李甜甜，朱志良，等. 云数据管理系统能耗基准测试与分析[J]. 计算机学报，2013，36(7)：1485-1499.

[19] LIAO B, YU J, SUN H, et al. Energy-efficient algorithms for distributed storage system based on data storage structure reconfiguration[J]. Journal of Computer Research and Development, 2013: 50(1): 3-18. (in Chinese)
廖彬，于炯，孙华，等. 基于存储结构重配置的分布式存储系统节能算法[J]. 计算机研究与发展，2013，50(1)：3-18.

[20] LIAO B, YU J, ZHANG T, et al. Energy-efficient algorithms for distributed file system HDFS[J]. Chinese Journal of Computers, 2013, 36(5): 1047-1064. (in Chinese)
廖彬，于炯，张陶，等. 基于分布式文件系统 HDFS 的节能算法[J]. 计算机学报，2013，36(5)：1047-1064.

[21] LIN J C, LEU F Y, CHEN Y. Impact of MapReduce Policies on Job Completion Reliability and Job Energy Consumption[J]. IEEE Transactions on Parallel and Distributed Systems, 2015, 26(5): 1364-1378.

[22] LIAO B, YU J, ZHANG Tao, et al. Energy-Efficient Algorithms for Distributed Storage System Based on Block Storage Structure Reconfiguration [J]. Journal of Network and Computer Applications, 2015, 48(2): 71-86.

[23] WANG H X, WU B, LIU Y. Parallel graph data analysis system based on Spark[J]. Journal of Frontiers of Computer Science and Technology, 2015, 9(9): 1066-1074. (in Chinese)
王虹旭，吴斌，刘旸. 基于 Spark 的并行图数据分析系统[J]. 计算机科学与探索，2015，9(9)：1066-1074.

[24] QIU R C. Parallel design and application of CURE algorithm based on Spark platform[D]. South China University of Technology, 2014. (in Chinese)
邱荣财. 基于 Spark 平台的 CURE 算法并行化设计与应用[D]. 广州：华南理工大学，2014.

[25] WANG Z Y, WANG H J, XING H L, et al. Ant colony optimization algorithm based on Spark[J]. Journal of Computer Applications, 2015, 35(10): 2777-2780. (in Chinese)
王诏远，王宏杰，邢焕来，等. 基于 Spark 的蚁群优化算法[J]. 计算机应用，2015，35(10)：2777-2780.

[26] ZHENG F F, HUANG W P, JIA M Z. Matrix factorization recommendation algorithm based on Spark[J]. Journal of Computer Applications, 2015, 35(10): 2781-2783. (in Chinese)
郑凤飞，黄文培，贾明正. 基于 Spark 的矩阵分解推荐算法[J]. 计算机应用，2015，35(10)：2781-2783.

[27] YAN Y L, DONG Y H, HE X M, et al. FSMBUS: A frequent subgraph mining algorithm in single large-scale graph using spark[J]. Journal of Computer Research and Development, 2015 (8): 1768-1783. (in Chinese)
严玉良，董一鸿，何贤芒，等. FSMBUS：一种基于 Spark 的大规模频繁子图挖掘算法[J]. 计算机研究与发展，2015(8)：1768-1783.

[28] SAMANTHULA B K, JIANG W. Secure Multiset Intersection Cardinality and its Application to Jaccard Coefficient[J]. IEEE Transactions on Dependable & Secure Computing, 2016, 13(5): 1.

(上接第 207 页)

[6] HINZ S, SCHMIDT K, STAHL C. Transforming BPEL to Petri Nets[C]// International Conference on Business Process Management. 2005: 220-235.

[7] STAHL C. A Petri Net Semantics for BPEL: Technical Repert 188[R]. Humboldt University Zu Berlin, Institut for Informatik, 2005.

[8] HOLZMANN G J. The Spin Model Checker: Primer and Reference Manual[D]. Addison-Wesley, Boston, MA, USA: 2004.

[9] SCHMIDT K. LoL A-a low level analyser[C]// Proceedings of the 21st International Conference on Application and Theory of Petri Nets, Volume1 825 of Lecture Notes in Computer Science. Aarhus, Demnark: Springer, Verlag, 2000: 465-474.

[10] LOHMANN N. A Feature-Complete Petri Net Semantics or WS-BPEL2. 0[C]// Web Services and Formal Methods International Workshop(WSFM07). 2007: 77-91.

[11] SUN Xi-long. Research on Web Service Composition Testing Based on BPEL[D]. Beijing: Beijing University of Technology, 2009. (in Chinese)
孙喜龙. 基于 BPEL 的 Web 服务组合测试研究[D]. 北京：北京工业大学，2009.

[12] YU Bo. Application of Petri Net to Improve the Correctness of BPEL Program[J]. Application Research of Computers, 2011 (28): 3348-3350. (in Chinese)
余波. 应用 Petri 网改进 BPEL 程序的正确性[J]. 计算机应用研究，2011(28)：3348-3350.

[13] LUO Xiang-yu, TAN Zheng, SU Kai-le, et al. A Web Service Composition Verification Method Based on Cognitive Model Detection[J]. Chinese Journal of Computers, 2011 (34): 1041-1061. (in Chinese)
骆翔宇，谭征，苏开乐，等. 一种基于认知模型检测的 Web 服务组合验证方法[J]. 计算机学报，2011(34)：1041-1061.

[14] SUN Lin, LIU Jiu-fu, YANG Zhen-xing. Software Test Case Generation Method Based on Petri Net[J]. Computer Measurement & Control, 2010(18): 2019-2022. (in Chinese)
孙琳，刘久富，杨振兴. 基于 Petri 网的软件测试用例生成方法[J]. 计算机测量与控制，2010(18)：2019-2022.

[15] MOU Xiao-ling. Research on Service Composition Testing Based on Extended Colored Petri Nets[D]. Chongqing: Southwest University, 2012. (in Chinese)
牟小玲. 基于扩展着色 Petri 网的服务组合测试研究[D]. 重庆：西南大学，2012.