SOArTester:一个基于精简用例的组合服务自动化测试系统

金若凡,孙海龙,刘旭东,李 翔
(北京航空航天大学计算机新技术研究所,北京 100191)

摘 要: 测试是保证组合服务功能正确性、性能和可靠性的必要手段.基于 BPMN 的组合服务流程具有多参数多取值的特征,参数的取值组合数量巨大.本文提出了一种基于组合精简技术的测试用例精简方法,用于检测输入参数间的相互作用对组合服务执行可能产生的影响,设计并实现了基于 BPMN 规范的组合服务自动化测试系统 SOArTester.该系统支持自动化的测试环境部署、测试执行和监控,生成的用例数量与参数个数之间呈现对数增长关系.实验表明 SOArTester 系统提高了组合服务测试的自动化程度,降低了组合服务测试的开销,因而提高了测试的效率.

关键词: 测试;测试用例精简;组合服务;业务流程建模标注
中图分类号: TP393.06 **文献标识码:** A **文章编号:** 0372-2112 (2010) 2A-065-06

SOArTester: An Automatic Composite Service Testing System Based on Test Case Reduction

JIN Ruo-fan, SUN Hai-long, LIU Xu-dong, LI Xiang
(*The Institute of Advanced Computing Technology, Beihang University, Beijing* 100191, *China*)

Abstract: Testing is necessary for the functional correctness, performance, and reliability. The BPMN-based composite services have large number of input parameters and the assigned values of individual parameters, the number of corresponding combinations of input parameters is huge. This paper described a new approach based on combinatorial designs to generate reduced test cases that detect the mutual effect of parameters and based on the method, designed and implemented the SOArTester system, which helps the processes of test cases deployment, execution and monitoring, and the number of test cases generated grows logarithmically in the number of parameters. Experiments demonstrated that SOArTester enhanced the automation level of composite service testing, reduced the testing cost, and improved the efficiency of testing.

Key words: testing; test case reduction; service composition; BPMN

1 引言

Web 服务作为一种分布式计算模型,是解决互联网松耦合环境下异构应用之间的互操作和集成问题的有效手段.随着相关标准的不断完善和支撑平台的不断成熟,服务组合成为一种高效的应用开发方法.业务过程建模是基于服务组合进行开发的重要内容.相对于 BPEL4WS(Business Process Execution Language for Web Service)等缺乏对多方参与分布式业务流程支持的流程描述语言而言,BPMN[1](Business Process Modeling Notation)作为一种业务流程建模语言,可以有效地描述多方参与的协作业务流程.BPMN 提供统一的建模图元,利用类似流程图的形式描述业务流程,使建模人员无需关心底层的实现细节,在服务组合领域得到了越来越广泛的应用.

为保证组合服务的正确性和可靠性,测试成为组合服务开发过程中的一个重要阶段.统计资料[11]表明,软件测试上的开销占到了总成本开销的 30% ~ 50%.而在测试工作的所有开销中,约 40% 花费在测试用例的设计上.因而提高测试的自动化程度,特别是提高测试用例的生成效率,在实践中被认为是降低测试开销,提高测试质量,进而提高软件整体开发效率的重要手段.

学术界针对组合服务测试工作开展了广泛的研究.文献[2,3]提出了一种基于 WSDL 文档的自动测试用例

收稿日期:2009-05-22;修回日期:2010-01-13
基金项目:国家 863 高技术研究发展计划(No.2007AA010301, No.2006AA01A106, No.2009AA01Z41)

生成方法,但该方法只支持单个服务的测试.文献[4]提出了基于 BPEL 组合服务的单元测试框架,但对于测试中的关键问题,即测试用例如何生成没有涉及.文献[5]提出了将 BPEL 转化为 CP 网的方法,但更多关注流程的形式化验证问题.在测试用例方面,文献[6,7]提出了基于 HPN 的 BPEL 流程测试用例精简方法,对业务模型要求较高,且没有提及原始输入数据如何获得.文献[11]提出了基于 BPEL 的测试用例生成方法,但未讨论其精简问题.

针对上述问题,本文提出了一种基于精简用例的组合服务测试方法,并在此基础上设计并实现了自动化测试系统 SOArTester,提高了在组合服务测试中用例生成、环境部署、脚本管理执行等阶段的自动化程度.针对组合服务流程的复杂输入参数,实现了基于组合精简模型的测试用例生成算法,能够对参数间相互作用对系统可能产生的影响进行检测.

2 组合服务测试用例精简方法

BPMN 规范规定了一套表示业务流程元素的统一图元,便于业务流程中不同角色的人员之间的交流,也有利于不同业务实体之间的交流.由业务人员建立的 BPMN 流程是一种概念模型,与 BPEL 相比较,由于 BPMN 模型具有执行流任意流动的特点,为了提高 BPMN 模型测试用例的生成效率,必须对业务流程的执行流进行分析.

2.1 组合服务执行流分析

为了分析流程执行流,首先需要将 BPMN 流程转化为有向图.通过分析有向图,获得流程全部可能执行路径.然后通过对路径上节点执行语义的分析,取得节点条件约束信息,最终获得路径条件约束链.最后根据约束链信息,求出输入参数的约束区间,给出输入参数各区间的候选值以备进一步精简.为了描述 BPMN 转换为有向图的方法,下面给出 BPMN 执行模型的相关定义:

定义 1 流对象(Flow Object,FO),指 BPMN 流程中基本可执行节点,包括各种事件(Event)、活动(Activity)和网关(Gateway)等,将其表示为四元组 FO = (Id, IngoingDegree, OutgoingDegree, Type).

其中,Id 唯一地标识了一个流对象,即 BPMN 图中的一个节点. IngoingDegree 表示 BPMN 流对象的入度;OutgoingDegree 是流对象 OBJ 的出度;Type 是该流对象的类型.

BPMN 中的流对象(FO)分为若干种,每一种 FO 又包含若干不同的类型.例如对于类型为事件(Event)的,有 Start Event 和 End Event 等类型,一般将 Start Event 作为执行的开始,将 End Event 作为流程终点;对于 Gateway,又分为 AND、OR、XOR 等分支聚合结构,为了适应多样的流程结构,本文在路径分析中对各种分支进行统一处理.

定义 2 顺序流(Sequence Flow,SF),指 BPMN 中的流程执行控制链路,表示为四元组 SF = (Id, sourceRef, targetRef, conditionExpression).

其中,Id 唯一地标识了一个顺序流,即 BPMN 流程图中的一条边;sourceRef 是该 SF 源 FO 的 Id,targetRef 是该 SF 指向的 FO 的 Id. conditionExpression 表示该 SF 从 sourceRef 转移到 targetRef 的条件约束.如果没有约束,则 conditionExpression 为空.

根据以上定义,可以将一个 BPMN 流程转换为有向图 $G(V,E)$.其中 V 为节点集合,$V = \{FO\}$,E 为有向边,$E = \{SF\}$,方向由 sourceRef 指向 targetRef,构造出有向图后,下面给出了着色路径搜索算法(Colored Path Searching,CPS),该算法对流程执行流进行遍历,从而得到流程的执行树,进而获得业务流程执行路径.

算法中白色节点为未被搜索节点,灰色为初次发现节点,当以某节点为根的路径树遍历完成后,置该节点为黑色.当子节点的邻接节点为灰色时,则该路径为环路.当某节点的邻接节点为黑色时,则存在一条直连路径.CPS 算法时间复杂度为 $\Theta(V+E)$.

算法 1 着色路径搜索算法

输入:有向图 $G(V,E)$
输出:树 T

```
CPS(G(V,E))
1.for each vertex u∈V[G]
2.   do color[u] ← WHITE   //初始化
3.         u.parent ← NULL
4.         u.children ← NULL
5.for each vertex u∈V[G]
6.   do if color[u] = WHITE   //对未遍历节点进行搜索
7.         then CPS-VISIT(u)          //调用递归函数
8.   else continue

CPS-VISIT(u)
1.color[u] = GRAY   //该节点初次被搜索到
2.for each v∈Adj[u]   //遍历 SF,找到相邻 FO
3.   do if color[v] = WHITE
4.         then PUSH(u.children, v)
5.              v.parent ← u
6.              CPS-VISIT[v]
7.   else if color[v] = GRAY
8.         then continue   //存在执行环路
9.color[u] ← BLACK   //完成该 FO 搜索
```

2.2 路径条件约束链分析

对测试用例的设计是测试阶段最关键的技术问题之一[8].流程中输入参数的条件约束对测试数据的有

效选取有直接指导作用.为了确定输入参数的条件约束,利用2.1节中的着色路径搜索算法(CPS)中获得的执行路径信息,通过分析路径节点的执行语义,剔除与输入变量无关的约束,将条件约束信息综合,最终确定输入变量定义域.路径条件约束链分析算法 ConditionParse 描述如下,算法输入为路径起始流对象 FO:

算法 2　路径条件约束链分析算法

输入:路径起始事件 StartEvent

输出:参数候选值数组

```
ConditionParse(StartEvent)
1. FO ← StartEvent
2. while type(FO) ≠ EndEvent
3.    FO ← nextFO()   //获得路径中下一执行节点
4.    expressions ← SF(FO)
5.    if condition (FO) ≠ NULL
      and inputVariable∈ condition (FO)   //剔除无关变量
6.    v ← expressionParser(expr)
                        //表达式分析函数,分析条件约束间断点
7.    push(oddPoint, v)
8.    generateInputData(oddPoint)
                        //对根据间断点对等价区间给出候选值
```

利用 CPS 算法,可以获得业务流程的所有可执行路径.对于每条路径,依据 ConditionParse 算法.以类型为 start event 的流对象作为路径起始点,以类型为 end event 流对象作为终点,依次遍历路径上所有边,可以得到该路径中条件约束链信息.expressionParser 分析涉及输入参数条件约束的间断点,根据间断点划分输入区间等价类.为每个等价区间设置候选输入参数值.该候选值可由程序按照预定规则给出,也可由测试人员手工指定.

2.3　测试用例算法

2.3.1　精简的基本思路

设组合服务流程具有 n 个输入参数,每个参数有 k 种取值情况,则总共有 k^n 种输入参数组合.可以预见,随着待测模型复杂度的提高,可能的参数组合数量将成几何级数增长.本文拟采用组合精简模式对测试用例数量进行精简.

以一个实现了网上购书功能的组合服务业务模型为例,该模型的有四个输入参数分别为:客户类型 = {普通,高级,VIP},付款方式 = {货到付款,在线支付,银行转账},物流方式 = {邮局,快递,EMS},状态 = {有货,预定,缺货};对于该模型,不同取值组合定义了不同的测试输入.本例中每个参数均具有三种不同取值,因此该模型定义了共 $3^4 = 81$ 种不同的情况,若进行全覆盖测试,代价较高.一种常见的测试用例优化方法是为每个参数选择一个缺省值,并在每次测试时改变一个参数的取值,直到所有参数的所有取值都被覆盖.该方法将测试用例的数量减少到 $9(=2\times4+1)$,并可覆盖了所有参数的所有取值.该方法与第一种全覆盖方式相比有了较大进步,但该方法生成的用例集合,仅覆盖了所有 $C_4^2\cdot C_3^1\cdot C_3^1 = 54$ 种参数间两两组合中的30种.而在实践中,通过观察和经验发现测试中大约70%的故障是由两个以下参数的相互作用引起的[10],由此可见该覆盖率的价值.同时,双参数组合交互又是分析多参数交互作用的基础.因此,本文提出了一种基于组合精简(Combinatorial Reduction,CR)模式的参数组合覆盖方法,对参数间的组合交互情况进行覆盖.下面给出了使用该方法生成的9个测试用例:

{{普通,在线支付,EMS,缺货},
{高级,银行转账,邮局,缺货},
{VIP,货到付款,快递,缺货},
{普通,银行转账,快递,有货},
{高级,货到付款,EMS,有货},
{VIP,在线支付,邮局,有货},
{普通,货到付款,邮局,预定},
{高级,在线支付,快递,预定},
{VIP,银行转账,EMS,预定}}.

与上面利用缺省值组合的方法不同的是,这9种测试用例覆盖了参数值的所有两两组合(pair-wise)情况.文献[9]证明,满足覆盖 n 个因子交互所需的测试用例数量,随参数个数的增加以对数趋势增长.

2.3.2　组合精简算法

根据算法2给出的各输入参数的候选值,为了生成覆盖候选参数值两两组合(pair-wise)的精简用例集合,我们设计并实现了一种基于贪婪思想的组合精简算法,描述如下:

设待测流程有 n 个输入参数,定义为 $P=\{P_1,\cdots,p_n\}$,每个参数存在 k 种取值,将其定义为 $I=\{I_1,\cdots,I_k\}$,并假定当前已选择了 m 个测试用例,则通过如下算法选择第 $m+1$ 个测试用例.

(1)选择一个在所有尚未被之前 m 个用例覆盖的参数值对(Pair)中出现次数最多的参数 p 及其取值 i.

(2)令 $P_1 = p$,并将剩余参数随机排列,将得到所有 n 个参数的一个序列:$P_1,\cdots,P_n$.

(3)设对于步骤(2)中的参数序列,当前已选择了 j $(j<n)$ 个参数的取值 $P_1,\cdots,P_j$,且对任意 $1\leqslant i\leqslant j$,参数 P_i 的取值为 V_i.则通过如下方法为参数 P_{j+1} 选择 V_{j+1}:

(a)对 f_{j+1} 每个可能的取值 v,生成 pair 集 $\{f_{j+1}=v,f_i=v_i,1\leqslant i\leqslant j\}$,并与未覆盖集合取交集,将该集中出现最多的取值作为 v_{j+1};

(b)重复步骤(a),直到所有参数都被取值.

(4)如尚有 Pair 未被覆盖,返回步骤(1),否则记录下生成的用例总数.

该算法生成每个测试用例的过程中,每个参数只被处理一次,且当为 f_{j+1}选择取值时,可能的取值仅与 j 个已经选择的参数值进行比较,因此该算法能够收敛.但由于该算法的随机性特征,无法保证单次运行即可生产最优结果,故需要重复运行上述步骤 M 次,之后从 M 次生成的结果中选择生成的用例总数最少的一次.我们对该算法进行了多次实验,发现当 $M>10n$(n 为输入参数个数)时即可保证良好的精简效果.

举例来说明该算法的处理过程:令模型三个参数为 a,b,c,参数对应取值分别为{a1},{b1,b2},{c1,c2,c3},则初始参数值对(pair)集合为{a1,b1},{a1,b2},{a1,c1},{a1,c2},{a1,c3},{b1,c1},{b1,c2},{b1,c3},{b2,c1},{b2,c2},{b2,c3}.选择出现次数最多的 a1,取参数的一个随机排列 a,c,b.为选择参数 c 的取值,对其所有可能取值 c1,c2,c3,与 a1 进行组合,得到{a1,c1},{a1,c2},{a1,c3},选择 c1.重复该步骤,选择参数 b 的值 b1,此时各参数均已取值,可得到一个用例为 a1,b1,c1.

经过第一轮选择,未覆盖的值对为:{a1,b2},{a1,c2},{a1,c3},{b1,c2},{b1,c3},{b2,c1},{b2,c2},{b2,c3},此时选择出现次数最多的 b2,取参数的一个随机排列 b,a,c,得到 pair 集{a1,b2},选择 a1,得到 pair 集{a1,c2},{a1,c3},{b2,c1},{b2,c2},{b2,c3},可得到一个测试用例为 a1,b2,c2.

此时未覆盖的值对为:{a1,c3},{b1,c2},{b1,c3},{b2,c1},{b2,c3},此时选择出现次数最多的 c3,取参数的随机排列为 c,a,b,得到 pair 集{a1,c3},选择 a1,得到 pair 集{b1,c3},{b2,c3},可得到一个测试用例为 a1,b1,c3.

重复上述步骤可得到精简的测试用例集合,如表 1 所示,可以看到该集合覆盖了参数间的所有两两组合.

表 1 根据组合精简方法生成的用例

	a	b	c
1	a1	b1	c1
2	a1	b2	c2
3	a1	b1	c3
4	a1	b1	c2
5	a1	b2	c3
6	a1	b2	c1

3 SOArTester 系统的设计与实现

SOArTester 是一个基于精简用例的组合服务自动化测试系统,该系统是面向服务的软件设计与生产平台 SOArWare(SOA Based Software Design and Production Platform)中的重要组件.SOArWare 的系统结构如图 1 所示,SOArWare 平台包括 Java 遗留系统封装工具 J2WS,服务资源库 SOArBase,以及 BPMN 的可视化业务建模工具 BEIDE 等组件,协同完成面向服务的软件设计和生产任务.

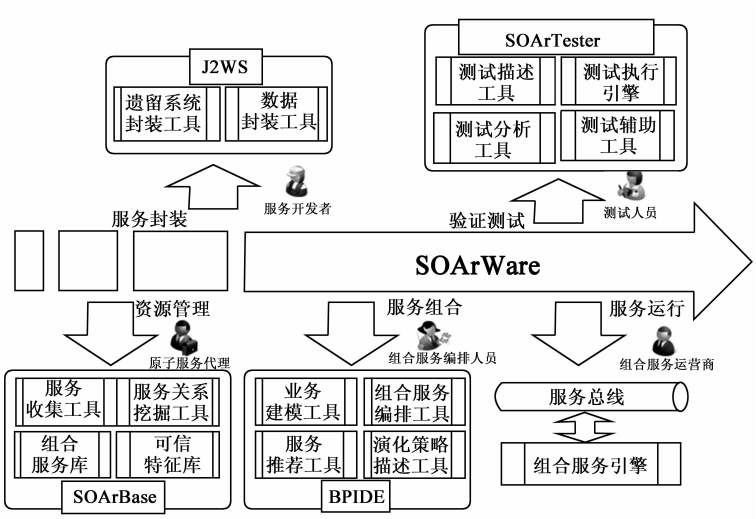


图1 SOArWare系统结构

作为 SOArWare 中的测试组件,SOArTester 为测试人员对业务流程模型进行测试提供支持,其系统结构如图 2 所示.SOArTester 包含测试环境部署、业务流程解析、测试用例生成和执行等子模块,各模块协同工作,实现了前文中的路径分析、条件约束链分析和用例精简方法.

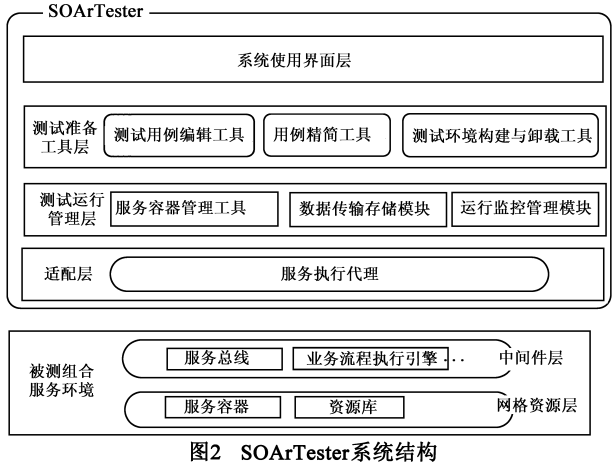


图2 SOArTester系统结构

在测试中,为了描述虚拟的测试执行环境,SOArTester 定义了一种可通过图形化建模生成的分布式的网络服务拓扑描述文件.如图 3 所示,文件中描述了各服务容器的 IP 和端口,以及容器中服务部署的信息,根据这些信息,SOArTester 中的测试环境部署工具可以实时地将服务部署到分布式的远程服务容器中,以搭建满足业务流程测试运行所需的虚拟测试支撑环

境.并在测试执行完毕后,对搭建的虚拟测试环境进行清理,对现有网络环境不造成任何影响,实现了测试环境的隔离.

```
<?xml version="1.0" encoding="UTF-8"?>
<topology>
    <bus name="http://192.168.3.178:8080/axis2/services/BusService">
        <container ip="http://192.168.0.1" port="8070">
                <service name="LoginService"></service>
                <service name="BankingService"></service>
                <service name="DeliveryService"></service>
        </container>
        <container ip="http://192.168.3.1" port="8060">
                <service name="FindBookService"></service>
                <service name="FindPriceAfterDiscount"></service>
        </container>
    </bus>
</topology>
```

图3　测试环境拓扑描述

测试执行模块根据测试脚本提供的信息,对测试任务的执行进行控制,根据测试脚本数据构造执行请求,顺序或并发地向总线提交流程执行,它是测试应用执行的中心控制器.实现了批量测试用例的自动化处理.

监控和故障定位模块,如图 4 所示,可以实时图形化显示流程的执行状态,在分布式的服务执行环境中实现了状态监控和故障定位功能.该模块采用集中式的控制方式,采用注册、接收机制,被动接收引擎推送的流程执行状态信息,避免了状态轮询的开销.当流程执行过程出错时,故障节点和错误信息都将被及时反馈和记录,便于对历史数据进行查询和分析,这种机制对于测试人员排查故障十分有益.

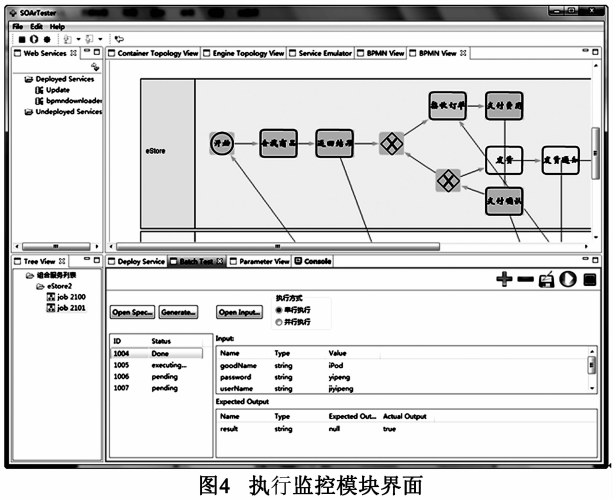


图4　执行监控模块界面

图 5 是 SOArTester 系统执行一次测试的完整工作流程。

①流程应用开发人员利用图形化建模工具完成流程建模和编排;②将流程提交测试人员;③测试人员利用 SOArTester 测试系统,编辑测试网络拓扑环境脚本,生成测试用例;④合成环境脚本和测试用例,生成测试脚本,并提交测试执行引擎;⑤测试引擎解析输入的用例,向总线提交作业;⑥引擎收到作业后,通过对脚本的解析,动态部署测试环境,并可在负载合成工具和服务仿真工具的协同下,完成流程的测试执行任务;⑦在测试执行完成后,返回测试结果持久化到数据库中以备查询;⑧将测试结果反馈给测试人员;⑨通过错误信息和故障定位模块,测试人员可以发现流程中存在的问题;⑩测试人员将分析结果反馈给应用开发人员进行修改和完善.

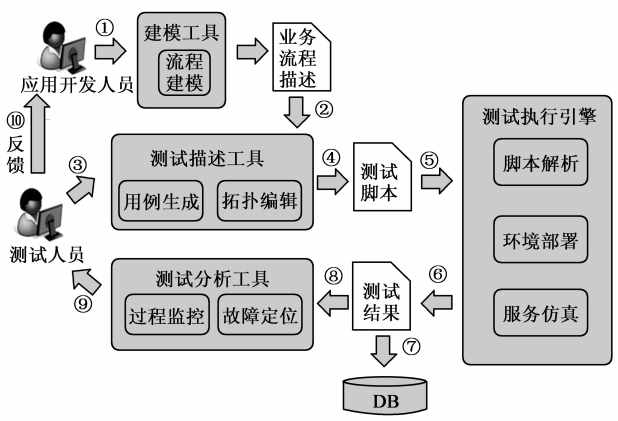


图5　测试流程

4　实验分析

为了验证基于组合精简（Combinatorial Reduction）算法的有效性,选择了多种不同数量的参数作为算法输入,最终生成的测试用例数及相关数据如表 2 所示.

表 2　测试结果数据

参数个数	生成用例数量	执行 10000 次时间开销	全覆盖用例数	精简率
4	9	1.562s	$3^4=81$	9/81 = 11.1%
7	14	6.922s	$3^7=2187$	14/2187 = 0.64%
10	16	22.437s	$3^{10}=59049$	0.027%
15	19	76.438s	3^{15}	0.001%
20	21	187.816s	3^{20}	$<10^{-9}$
30	23	586.944s	$>10^{14}$	$<10^{-13}$

从表 2 的结果数据中可以看出,采用参数完全组合覆盖方式生成测试用例时,全覆盖用例数为 3^n（n 为参数数量),呈指数形式增长,即使对于每参数只有 3 种取值的情况,当参数个数超过 10 以后,需要生成用例数量已经上升至 5 万个以上,执行开销过大.

从图 6 中可以看出,当采用组合精简算法方法生成精简测试用例时,生成的用例数量与参数个数之间呈现出对数增长趋势(证明略),该算法对于具有较多输入参数的 BPMN 业务流程测试十分有利,测试人员可以不必担心输入参数和对应候选值的增多会导致测试用例数剧增.而从图 7 中描述的测试运行时间与参数个数关系可以看到,由于在用例精简过程中需要进行大量

的搜索、比较和筛选运算，算法的运行时间随参数个数的增长呈指数增长趋势.

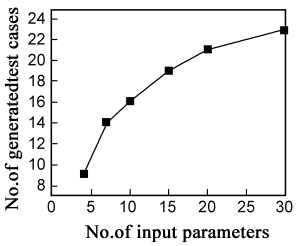


图6 用例与参数个数关系

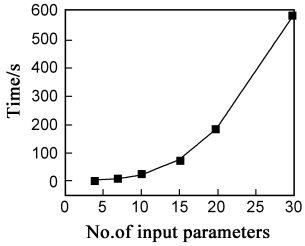


图7 运行时间与参数个数关系

5 结论

为了满足对复杂的组合服务流程进行高效测试的需求，本文针对 BPMN 业务流程的特点对测试用例精简的技术进行了研究，提出了基于组合精简的测试用例生成方法，基于该方法开发了组合服务测试系统 SOArTester，能够在精简测试用例同时，实现对测试环境的自动远程部署和测试用例自动执行分析反馈功能，从而提高了测试的自动化程度.通过实验表明，基于组合精简算法的组合服务自动化的测试工具 SOArTester，可以大大降低组合服务测试的开销，从而提高测试效率.

SOArTester 系统目前只是在分析组合服务业务流程描述文件的基础生成测试用例，对于流程调用的原子服务，由于计算环境的分布性及原子服务的独立性，暂时未对其分析.在今后的工作中，将结合原子服务的测试的分析方法，进一步提高测试用例生成的有效性.

参考文献：

[1] BPMI org, OMG. Business Process Modeling Notation (BPMN) 1.1[OL]. http://www.omg.org/spec/BPMN/1.1/,2008.

[2] Tsai W T, et al. Extending WSDL to facilitate web services testing[A]. Proceedings of the 7th IEEE International Symposium on High Assurance Systems Engineering[C]. IEEE Press, 2002.171 – 172.

[3] Bai Xiaoying, et al. WSDL-based automatic test case generation for Web Services testing[A]. Proceedings of the IEEE International Workshop on Service-Oriented System Engineering (SOSE)[C]. IEEE Press, 2005.215 – 220.

[4] Mayer P, D Lubke. Towards a BPEL unit testing framework [A]. Proceedings of the 2006 workshop on Testing, analysis, and verification of web services and applications[C]. New York, USA: ACM Press, 2006.33 – 42.

[5] Yan Ping Y, et al. Transformation BPEL to CP-nets for verifying web services composition in next generation web services practices[A]. Proceedings of International Conference on Next Generation Web Services Practices (NWeSP)[C]. IEEE Press, 2005.137 – 142.

[6] W Dong, H Yu, Y Zhang. Testing BPEL-based web service composition using high-level Petri nets[A]. Proceedings of the 10th IEEE International Enterprise Distributed Object Computing Conference (EDOC)[C]. Washington, DC, USA: IEEE Computer Society, 2006.441 – 444.

[7] Dong W L. Test case reduction technique for BPEL-based testing[A]. Proceedings of the 2008 International Symposium on Electronic Commerce and Security (ISECS)[C]. Washington, DC, USA: IEEE Computer Society, 2008.814 – 817.

[8] Rapps S, E J Weyuker. Selecting software test data using data flow information[J]. IEEE Transactions on Software Engineering, 1985, 11(4): 367 – 375.

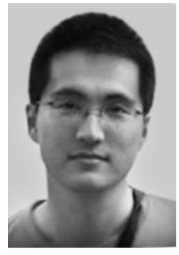
[9] D M Cohen, et al. The AETG system: an approach to testing based on combinatorial design[J]. IEEE Transactions on Software Engineering, 2000, 23(7): 437 – 444.

[10] Kuhn D R, Reilly M J. An investigation of the applicability of design of experiments to software testing[A]. Proceedings of the 27th Annual NASA Goddard Software Engineering Workshop (SEW – 27′02)[C]. Washington, DC, USA: IEEE Computer Society, 2002.91 – 95.

[11] 朱少民.软件测试方法和技术[M].北京：清华大学出版社，2005.

[12] 曾云峰，周航，黄志球.BPEL 的测试用例生成研究[J].计算机工程与设计，2008，29(20)：5243 – 5246，5249.

作者简介：



金若凡 男，1984 年出生于北京.2003 年毕业于北京航空航天大学计算机科学与技术系，2007 进入在北航计算机新技术研究所，现为硕士研究生，从事服务计算相关研究工作.
E-mail: jinrf@act.buaa.edu.cn

孙海龙 男，1979 年出生于河北承德.讲师、IEEE 和 ACM 会员.主要研究领域为服务计算. E-mail: sunhl@act.buaa.edu.cn