

SOArTester4RT:一种面向组合服务演化的 高效回归测试工具

蒋俊彦^{1,2}, 李先贤¹, 李建欣¹, 吴 茜¹

(1 北京航空航天大学 计算机学院, 北京 100191; 2 中国人民解放军 96669 部队, 北京 102206)

摘 要: 提出了一种基于有向图关键路径选择的组合服务回归测试方法. 通过分析组合服务执行流, 识别受影响的执行流, 选择适当的回归测试路径, 并生成相应的回归测试用例. 基于该方法, 在 SOArWare 平台上实现了 SOArTester4RT 工具. 实例分析和测试结果表明: 该工具可以对 BPMN 的各种控制结构加以分析, 产生必需的测试用例, 有效提高了组合服务演化后的回归测试效率.

关键词: BPMN; 组合服务; 回归测试; 关键路径选择

中图分类号: TP393.06

文献标识码: A

文章编号: 1000-7180(2010)08-0181-05

SOArTester4RT: An Efficient Regression Testing Tool for Composite Service Evolution

JIANG Jun-yan^{1,2}, LI Xian-xian¹, LI Jian-xin¹, WU Qian¹

(1 School of Computer Science and Engineering, Beihang University, Beijing 100191, China;

2 The 96669th Force of PLA, Beijing 102206, China)

Abstract: This paper described a regression testing approach based on a Critical Path Selection for composite service, by parsing BPMN execution flow, identified the affected execution flow, selected the appropriate path for regression testing, and generated the regression testing cases. Based on this method, designed and implemented the SOArTester4RT tool, which is based on the SoarOArWare. Experiments demonstrated that SOArTester4RT can parse the BPMN, generate test cases, efficient improved the efficiency of regression testing for composite service evolution.

Key words: BPMN; composite service; regression testing; critical path selection

1 引言

当前针对 Web 服务回归测试方法大致分两类, 一种仅仅针对单个 Web 服务, 如文献[1]研究单个 Web 服务回归测试用例选择的算法; 另一种就是面向组合服务的, 文献[2]通过构造 XBF 研究面向组合服务的回归测试方法, 但不能识别因流程结构变化而受影响的差异路径, 文献[3]提出了一种基于变更表(Change Table)的面向业务流程的回归测试方法, 但该方法对业务流程的分析局限于单选和同步结构, 而且并没有考虑回归测试用例的生成, 文献

[4]将组合服务作为一个整体研究其回归测试, 对重构(Refactor)的组合服务进行全面测试, 测试成本较高, 文献[5]研究基于代数式(Algebraic)的回归测试方法, 与控制流图模型相比, 代数式模型不能完全表达复杂的业务流程结构. 此外, 文献[2]对比了从不同测试执行方实施 Web 服务回归测试的代价和局限性.

为解决如何对演化后的组合服务进行回归测试的问题, 特别是为减少回归测试路径, 降低回归测试成本, 提高回归测试效率, 文中首先构建组合服务执行流描述模型, 通过执行流识别技术对该模型进行

执行流分析,找出 BPMN 流程的可执行路径,然后对比演化前后的可执行路径,通过关键路径选择技术找出受演化影响的可执行路径即回归测试路径,再针对回归测试路径生成相应的回归测试用例,最后在 SOArWare 平台上设计并实现了回归测试工具 SOArTester4RT.

2 面向组合服务演化的回归测试方法

面向组合服务演化的回归测试方法的主要思路:首先将 BPMN 流程文件转化为基于有向图的组合服务执行流描述模型,通过执行流识别技术对该模型进行执行流路径分析,找出业务流程文件的所有可执行路径,然后对比演化前后的可执行路径,通过关键路径选择技术选出回归测试路径,最后针对此回归测试路径通过回归路径生成技术生成相应的回归测试用例.

2.1 组合服务执行流描述模型

文献[6]通过构造 BPMN 有向图实现了一种基于图形搜索的 BPMN 流程测试方法,将控制流测试技术应用于 BPMN 流程测试,但对 BPMN 业务流程的所有分支路径进行统一处理,不能完整地表达 BPMN 业务流程.为了识别组合服务演化前后可执行流的差异,本文对有向图进行必要的扩展,对各分支路径单独处理,形成 BPMN 有向图 BDG(BPMN Directed Graph),从而构造组合服务执行流描述模型,以满足回归测试的需要,下面给出组合服务执行流描述模型的相关定义:

定义 1 流对象(Flow Object, FO):指 BPMN 中的可视化节点(graphicalElements),包括各种事件(Event)、活动(Activity)和网关(Gateway)等,将其表示为七元组 $FO = (ID, Type, InDegree, OutDegree, inMessageRef, outMessageRef, Webservice)$.

其中, ID 唯一标识一个 FO,对应 BDG 中的一个节点. Type 是该 FO 的类型, BPMN 中的 FO 包含三种类型: Event、Activity 和 Gateway. 其中 Event 又分为 Start Event、Intermediate Event 和 End Event,一般将 Start Event 作为流程的起始节点,将 End Event 作为流程的终点节点; Activity 又分为 task、sub-processes 和 wrapper,其中 task 又分为 Send Task、Receive Task 和 Service Task; Gateway 又分为 ParallelGateway、InclusiveGateway 和 ExclusiveGateway. 为准确表达 BPMN 流程,将 ParallelGateway 映射为并发节点(PN),分并发分支节点和并发归并节点,所有边都必须被触发;将 InclusiveGateway 映射

为多选节点(IN),分为多选分支节点,可以有任意数目的边被触发;将 ExclusiveGateway 映射为单选节点(EN),分为单选分支节点(InDegree 等于 1, OutDegree 大于 1)和单选归并节点(InDegree 大于 1, OutDegree 等于 1),有且只有一条边被触发,除 Gateway 外的其它可视节点统一映射为普通节点(NN). InDegree 表示该 FO 的入度; OutDegree 表示该 FO 的出度. inMessageRef 表示 Type 为 ServiceTask 的 FO 的输入信息,如果 Type 不是 ServiceTask,则 inMessageRef 为空. outMessageRef 表示 Type 为 ServiceTask 的 FO 的输出信息,如果 Type 不是 ServiceTask,则 outMessageRef 为空. Webservice 表示 Type 为 ServiceTask 的 FO 所绑定的 web 服务,如果 Type 不是 ServiceTask,则 Webservice 为空.

定义 2 顺序流(Sequence Flow, SF):指 BPMN 流程的可执行控制链路(sequenceFlows),表示为四元组 $SF = (ID, SourceRef, TargetRef, ConditionExpression)$.

其中, ID 唯一标识一个 SF,对应 BDG 中的一条有向边; SourceRef 是该 SF 源 FO 的 Id, TargetRef 是该 SF 所指向的 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,可以构造组合服务执行流描述模型,该模型基于有向图 BDG (N, E, s, f) . 其中 N 为节点集合, $N = \{FO\} = PN \cup IN \cup EN \cup NN$, E 为有向边集合, $E = \{SF\}$, 方向由 SourceRef 指向 TargetRef, s 为起始节点,即 FO 中 Type 为 StartEvent 的节点, f 为终止节点,即 FO 中 Type 为 EndEvent 的节点.

2.2 执行流路径生成

由于 BPMN 流程中的顺序流(SF)含有 SourceRef 和 TargetRef 及 ConditionExpression 等能准确反映业务流程可执行路径的信息,因此,可以用 BDG 边的集合表示可执行路径 P ,各元素的顺序关系隐含在 SF 的 SourceRef 和 TargetRef 中. 分析组合服务执行流描述模型生成执行流路径集合 P 的执行流识别方法是:

(1) 将具有相同 SourceRef 的边映射到 hashmap 表中,其结构如下:

SourceRef	TargetRef	Type	TargetRef	Type
-----------	-----------	------	-----------	------

(2) 遍历搜索 FO,查找起始节点(s)的位置,记

录下 s 在节点表中的行号和个数;

(3) 遍历搜索 SF , 查找以 s 为源节点的边(e), 如果有 m 条这样的边, 则生成 m 条测试路径, 向每条路径中添加 e ;

(4) 对于 m 条路径中的每一条路径, 如果目标节点是普通节点, 则将以此目标节点为源节点的边添加进路径中; 如果目标节点是单选节点, 具有 k 个分支, 则将路径复制 $k-1$ 次, 并将以单选节点为源节点的边添加至路径和复制的 $k-1$ 次路径中; 如果目标节点是多选节点, 假设该节点的出度为 m , 则这 m 条边中同时被执行的数目介于 1 和 m 之间, 共有 $(2^m - 1)$ 种, 因此需要将路径复制 $(2^m - 2)$, 再分别按这 $(2^m - 1)$ 种组合将这些边添加到对应的路径中; 如果目标节点是并发节点, 文中将各并发路径统一为一条路径, 将以此目标节点为源节点的边添加到路径. 遇到终止节点 f 表示路径结束. 为方便后面的回归测试路径的识别与选择, 可以用此执行流识别方法遍历搜索组合服务业务流程的所有流对象(FO)生成节点集合.

2.3 回归测试路径的识别与选择

传统的回归测试之所以测试代价高, 就因为对组合服务演化后的所有可执行路径进行全面的测试, 这其中必然会包括那些不受演化影响的路径. 针对此问题, 文中提出一种关键路径选择 (Critical Path Selection, CPS) 方法, 该方法通过对比演化前后的可执行流路径, 选出受演化影响的可执行流路径 (即关键路径), 从而由所有关键路径组成回归测试路径, 回归测试时只测关键路径. 目前常见的组合服务演化操作有: 替换、增加、删除和结构调整等. 因此, 回归测试路径 P_{rt} 由两部分组成: 由于业务流程变更 (包括增加、删除节和结构调整) 影响的可行路径集合 P_{rt1} 以及由于原子服务绑定变化 (包括替换操作) 影响的可行路径集合 P_{rt2} , 即 $P_{rt} = P_{rt1} \cup P_{rt2}$.

2.3.1 回归测试路径 P_{rt1} 的识别与选择

根据 2.2 介绍的执行流路径生成方法分别可以找出流程的所有可执行节点和可执行路径, 设组合服务演化前后的可执行节点集分别为 N 和 N' , 设组合服务演化前后的可执行路径集分别为 P 和 P' . 为方便回归测试路径的识别与选择算法的描述, 特给出如下定义:

定义 3 新增节点 (Add Nodes, AN): 在 $BDG(N, E, s, f)$ 中, 如果节点 $n \in N' \&\& n \notin N$, 则

称 n 为新增节点 (AN).

定义 4 删除节点 (Deleted Nodes, DN), 在 $BDG(N, E, s, f)$ 中, 如果节点 $n \in N' \&\& n \notin N$, 则称 n 为删除节点 (DN).

定义 5 结构调整路径 (Structure Adjustment Path, SAP), 设路径 $P = \{n_1, e_1, \dots, n_i, e_i, \dots, n_j, e_j, \dots, n_n, e_n\}$, 其中, $n_1, \dots, n_n \in N, e_1, \dots, e_n \in E, n_1$ 为起始节点 s, n_n 为终止节点 f , 如果路径 $P' = \{n_1, e_1, \dots, n_j, e_i, \dots, n_i, e_j, \dots, n'_n, e'_n\}$, 则称 P' 为 SAP.

回归测试路径 P_{rt1} 的生成方法为: 遍历新增节点 AN 列表, 如果以该新增节点为起始节点或目标节点的边包含在 P' 中的某条子路径 p' 中, 则将子路径 p' 添加到 P_{rt1} 中; 遍历删除节点 DN 列表, 如果 P 中除去以 DN 节点为起始节点或目标节点的边后的路径 p 在 P' 中, 则将此路径 p 添加到 P_{rt1} 中; 如果 P' 中有 SAP, 则将 SAP 直接添加到 P_{rt1} 中. 关键路径选择算法 1 描述为:

算法 1 SelectRTPath4Prt1 (P, P', AN, DN, SAP)

//选择因流程变更影响的可行执行路径

输入: P, P', AN, DN, SAP

输出: P_{rt1}

SelectRTPath4Prt1 (P, P', AN, DN, SAP)

```
{
    Prt1 = NULL; //初始化
    for(an: AN) //遍历新增节点
    {
        for(p': P') //遍历演化后可执行流路径集
        {
            if ((e|e.SourceRef=an|| e.TargetRef=an) ∈ p')
                Prt1 = Prt1 ∪ {p'};
        }
    }
    for(p: P)
    {
        for(dn: DN) //遍历删除节点
        {
            if ((e|e.SourceRef=dn|| e.TargetRef=dn) ∈ p
                && (p-e) ∈ p')
                Prt1 = Prt1 ∪ {p-dn};
        }
    }
    if(SAP ≠ NULL) //如果 P' 中有 SAP
        Prt1 = Prt1 ∪ SAP;
    return Prt1;
}
```

2.3.2 回归测试路径 P_{rt2} 的识别与选择

BPMN 流程中 Type 为 ServiceTask 的流对象

(FO)通过属性值 WebService 绑定原子服务,在非可视化元素集 (invisibleElement) 下的 supportingElement 节点显示该原子服务的具体绑定地址,通过对比该绑定地址,可以发现原子服务的绑定是否有变化。

设组合服务演化前 Type 为 ServiceTask 的 FO 节点集为 SN,演化后 Type 为 ServiceTask 的 FO 节点集为 SN',演化后的路径集为 P', $N_{all} = N \cup N'$ 。

绑定有原子服务的 FO 节点搜索方法:

(1) 搜索 FO,将 Type 为 ServiceTask 的 FO 保存至 hashMap, id 为 FO 的 ID,映射为 WebServiceID;

(2) 搜索 BPMN 文件 supportingElements 下元素,提取 Type 为 WebService 的元素中的属性 url 值,将 url 值保存为 table 中的一行;

(3) 新建 hashMap, ID 为步骤(1)中节点的 ID,映射为步骤(2)中 url 值。

因绑定变化而产生的回归测试路径 P_{r2} 的生成方法为:

(1) 遍历 N_{all} 列表,判断 N_{all} 列表中节点 node 是否包含在演化前和演化后的 WebService 绑定 hashMap 中;

(2) 如果都包含,则将绑定不一致的节点保存至 diff_node,执行步骤(3);

(3) 遍历 diff_node 中的每个节点,查找 P' 中包含以该节点为起始节点或目标节点的边 e 的所有路径,判断包含 e 的新路径是否已生成,如果没有,生成新的回归测试路径 P_{r2} 。关键路径选择算法 2 描述如下:

算法 2 SelectRTPath4 $P_{r2}(N_{all}, SN, SN', P')$

//选择因绑定变化影响的可执行路径

输入: N_{all}, SN, SN', P'

输出: P_{r2}

SelectRTPath4 $P_{r2}(N_{all}, SN, SN', P')$

```

{
    diff_node = NULL,  $P_{r2}$  = NULL; //初始化
    for(n:  $N_{all}$ ) //遍历所有节点
    {
        if( $n \in SN \ \&\& \ n \in SN'$ )
        {
            if( $SN.url \neq SN'.url$ )
            {
                diff_node = diff_node + n;
            }
        }
    }
    for(n: diff_node) //遍历所有绑定发生变化的节点

```

```

{
    for( $p': P'$ )
    {
        if(( $e | e.SourceRef = n | e.TargetRef = n$ )  $\in p'$  &&
            ( $e | e.SourceRef = n | e.TargetRef = n$ )  $\in P_{r2}$ )
        {
             $P_{r2} = P_{r2} \cup \{e | e.SourceRef = n | e.TargetRef = n\}$ 
        }
    }
    return  $P_{r2}$ ;
}

```

2.4 回归测试用例生成

对测试用例的设计是测试阶段最关键的技术问题之一^[7],针对回归测试路径 P_{rt} 生成相应的回归测试用例同样是回归测试领域最受关注的问题之一。为生成有效的、针对性强的回归测试用例 (Regression Test Case, RTC),文中提出一种有效的回归测试用例生成方法,其主要思想为:对于每条可执行路径,以 s 作为路径起始点, f 作为路径终点,依次遍历路径上所有边 e,可以得到该路径中条件约束链信息 (ConditionExpression, CE),该条件约束链信息由两部分组成:一部分是 FO 中 ServiceTask 节点的输入信息 (inMessageRef) 和输出信息 (outMessageRef);另一部分是 SF 中的条件转移信息 (ConditionExpression)。如果每条路径所生成的条件约束链信息相同,则它们的输入参数和转移条件都相同,则可以选择原有测试用例 (TestCase, TC) 进行回归测试,否则就根据条件约束链信息生成新的测试用例 (NewTestCase, NTC) 作为回归测试用例。而每个输入参数的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。因此,回归测试用例由原有测试用例或新的测试用例组成,即 $RTC = \{TC || NTC\}$ 。

3 SOArTester4RT 系统设计与实现

基于以上研究成果,作者在面向服务的软件生产线 SOArWare 平台上设计实现了一个面向组合服务演化的回归测试工具 SOArTester4RT。SOArWare 的系统结构如图 1 所示。SOArWare 平台包括原子服务开发工具 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。

作为 SOArWare 中的测试组件,SOArTester4RT 为测试人员对演化后的业务流程模型进行测试提供支持,其系统结构如图 2 所示。

SOArTester4RT 包含测试环境部署、回归测试

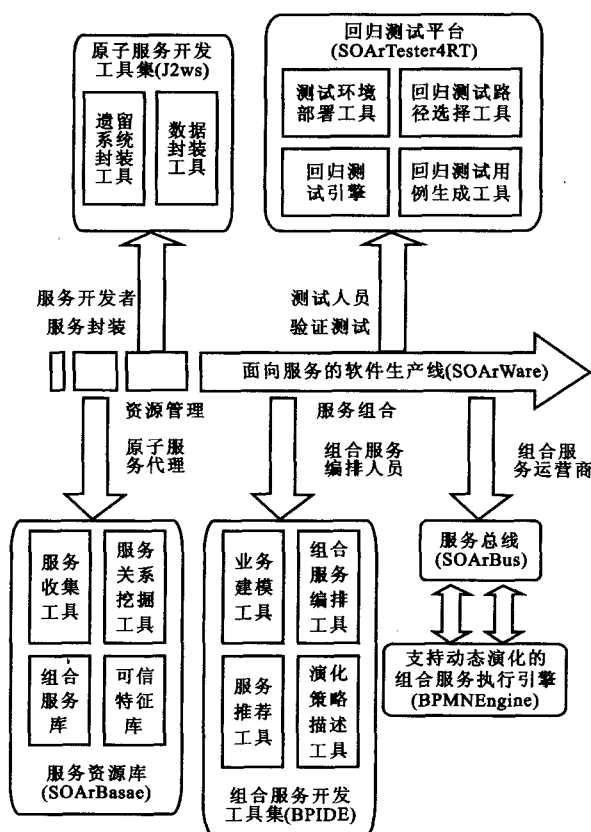


图1 SOArWare 系统结构图

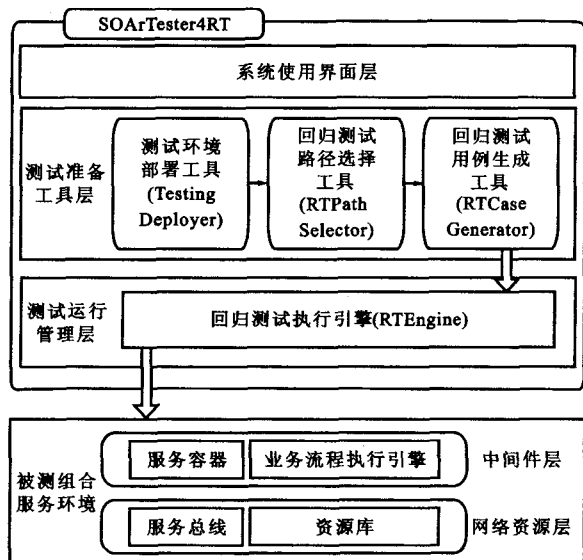


图2 SOArTester4RT 系统结构图

路径选择、回归测试用例生成和回归测试执行引擎等子模块,各模块协同工作,实现了前文所述的组合服务执行流描述、执行路径生成、回归测试路径的识别与选择和回归测试用例生成等技术。从图2可以看出,工具由四部分组成:

(1) TestingDeployer:测试环境部署器通过指定

的网络拓扑描述文件将服务分布式的部署在远程服务容器上,构建测试所需要的虚拟测试环境,测试执行完后通过反部署服务销毁服务容器,从而实现测试环境的清理。同时该部署器还完成对被测组合服务执行流描述模型的构建。

(2) RTPPathSelector:回归测试路径选择器对TestingDeployer构建的组合服务执行流描述模型进行执行流路径分析,先生成所有可执行路径,然后对比可执行路径,通过关键路径选择技术识别变更路径,最后得到回归测试路径 RTPPath;

(3) RTCaseGenerator:测试用例生成器接收RTPPathSelector选出的回归测试路集 RTPPath,生成回归测试用例集 RTCases;

(4) RTEngine:回归测试引擎解析回归测试用例,向服务总线提交作业,最后由服务总线调用组合服务执行引擎(BPMNEngine)完成对组合服务流程的测试执行。

通过实例分析表明,文中提出的回归测试方法能准确发现因演化受影响的节点及执行流路径,适用于所有用 BPMN 进行流程描述的业务流程,具有较好的可操作性和通用性。

4 结束语

文中针对传统回归测试测试成本高的缺陷,提出了一种面向组合服务演化的回归测试方法,该方法基于关键路径选择技术,以 BPMN 业务流程文件作为组合服务执行流建模依据,通过分析组合服务执行流模型,生成执行流路径,再通过关键路径选择技术识别因组合服务演化影响的可执行流路径,选择回归测试路径,并通过分析同一路径的条件约束链信息,生成相应的回归测试用例。基于以上方法开发的组合服务回归测试工具 SOArTester4RT,实现了对组合服务演化的回归测试。通过实验表明,基于关键路径选择技术的回归测试工具 SOArTester4RT 可以有效降低回归测试成本,提高回归测试效率。

文中的方法虽对 BPMN 业务流程的各种分支情况进行了分类讨论,但对于像多选分支这样的复杂分支情况的分析不够全面,在今后的工作中,将结合复杂分支分析方法进一步提高测试工具的通用性。

参考文献:

- [1] Michael Ruth, Shengru Tu. Concurrency issues in automating RTS for web services[C]// ICWS 2007. USA: Sdt Lake City, 2007.

(下转第 189 页)

程中,随着序列长度的增大,主从处理机之间所需的通信开销在减少,算法总的处理时间(通信时间和计算时间)在降低;(2)算法执行时间随着串联重复串最小长度的增加而减少,原因是如果串联重复串最小长度越小,那么算法搜索空间就越大,搜索时间也就越长^[8].

表3 串联重复串最小长度为2的并行算法性能

序列编号	序列长度	并行执行 时间(s)	串行执行 时间(s)	加速比
DQ198799.1	8108	0.22	0.26	1.18
AY584531.1	12061	0.70	0.85	1.21
AP006270.1	113220	5.69	7.92	1.39
NC_010368.1	284332	22.17	33.75	1.52
BA000002.3	691920	49.36	78.79	1.60

5 结束语

文中给出的并行算法将序列后缀划分为若干组,可以对每组独立地进行长序列最大串联重复识别,从而得到完整序列的最大串联重复.算法分析与实验结果表明该方法具有很好的适应性和可扩展性,可以提高长序列有最大串联重复识别处理时间,并有效解决内存容量受限制问题.

参考文献:

- [1] Benson G. Tandem repeats finder, a program to analyze DNA sequences [J]. Nucleic Acid Research, 1998(27): 573-580.

- [2] Sonnhammer E L L, Durbin R. A dot-matrix program with dynamic threshold control suited for genomic DNA and protein sequence analysis[J]. Gene, 1995, 167(1/2): GC1-GC10.
- [3] Gusfield D, Stoye J. Linear time algorithms for finding and representing all the tandem repeats in a string[R]. Dept. of Computer Science, University of California, Davis, 1998.
- [4] Parisi V, Fonzo VD, Aluffi-Pentini F. String: finding tandem repeats in DNA sequences [J]. Bioinformatics, 2003, 19(14): 1733-1738.
- [5] 王晓敏, 王正志. 基于后缀列的基因序列最大串联重复查找技术[J]. 生物信息学, 2006(2): 72-75.
- [6] Karkkainen J, Sanders P. Simple linear work suffix array construction[C]// Proceedings of Automata, LINGA-GS and Programming. Berlin: Springer-Verlag, 2003: 943-955.
- [7] Kasai T, Lee G. Linear-time longest-common-prefix computation in suffix arrays and its applications [C]// Proc. of Annual Symposium on Combinatorial Pattern Matching. Berlin: Springer-Verlag, 2001: 181-192.
- [8] 钟诚, 陈国良. PRAM 和 LARPBS 模型上的近似串匹配并行算法[J]. 软件学报, 2004, 15(2): 159-169.

作者简介:

陆向艳 女, (1973-), 硕士, 讲师. 研究方向为计算机网络与并行分布计算.

钟 诚 男, (1964-), 博士, 教授, CCF 高级会员. 研究方向为计算机网络与并行分布计算.

(上接第 185 页)

- [2] Canfora G, Penta M D. Testing services and services-centric systems: challenges and opportunities[C]// IT Pro March. Italy, 2006: 10-17.
- [3] Yonez HH, Liu, Li ZJ, Zhu J, et al. Business process regression testing [C]// ICSOC 2007. Austria: Vienna, 2007.
- [4] Cesare Pautasso. JOpera: an agile environment for web service composition with visual unit testing and refactoring [C]// VL/HCC'05. USA: Dallas, 2005.
- [5] Ginige J A, Sirinivasan U, Ginige A. A mechanism for efficient management of changes in BPEL based business processes: an algebraic methodology [C]// ICEBE'06. China: Beijing, 2006.
- [6] 金若凡, 孙海龙, 刘旭东, 等. SOArTester: 一个基于精简用例的组合服务自动化测试系统[C]// NASAC'09. 中

国: 沈阳, 2009.

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

作者简介:

蒋俊彦 男, (1980-), 硕士研究生, 工程师. 研究方向为服务计算、服务测试.

李先贤 男, (1969-), 教授, 博士生导师. 研究方向为服务计算、可信计算、信息安全.

李建欣 男, (1979-), 博士, 讲师. 研究方向为服务计算、可信计算、信息安全.

吴 茜 女, (1987-), 硕士研究生. 研究方向为服务计算、服务测试.