数据感知过程之间差异的检测和解决

张学伟[1] 邢建春[1] 杨启亮[1] 宋 巍[2] 王洪达[1]
(解放军理工大学国防工程学院 南京 210007)[1] (南京理工大学计算机科学与工程学院 南京 210094)[2]

摘 要 业务驱动开发支持多名业务人员对一项流程进行设计。为获得一个标准的参考数据感知过程,需要将这些变种合并,而检测和解决变种之间的差异是不可或缺的一环。现有方法大多依赖于修改日志从控制流层面检测和解决过程模型之间的差异,而很少研究数据感知过程之间差异的问题。基于程序依赖图和对应关系,提出一种通过比较修改前后的数据感知过程来检测和解决差异的方法。该方法建立一个层次性修改日志,以满足用户友好性的要求。

关键词 数据感知过程,差异,程序依赖图,层次性修改日志

中图法分类号 TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.12.032

Detection and Resolution of Differences of Data Aware Processes

ZHANG Xue-wei[1] XING Jian-chun[1] YANG Qi-liang[1] SONG Wei[2] WANG Hong-da[1]
(College of Defense Engineering, PLA University of Science and Technology, Nanjing 210007, China)[1]
(Department of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China)[2]

Abstract Business-driven development favors the construction of a business process by different business-development staffers. In order to obtain a standard reference process to consolidate these variants, it is necessary to detect and resolve differences between data aware processes. Existing approaches focus on detection and resolution of differences between process models with relying on a change log at the level of control flow. However, few methods can study the problem of differences between data aware processes. Based on program dependence graphs and correspondences, a novel approach was proposed to detect and resolve differences by comparing before and after modified data aware processes. It builds a hierarchical change log and meets requirements of user-friendliness.

Keywords Data aware processes, Differences, Program dependence graphs, Hierarchical change log

在业务驱动开发中,许多公司或部门创建了大量的数据感知过程,而这些过程经过反复修改和过程演化又产生众多变种。在一些情形下需要将这些变种部分或完全地合并:当公司进行合并或重组时,为了消除冗余和创造协同效应,先前属于不同部门的数据感知过程需要被合并;当一个数据感知过程被多个业务人员设计时,会从原始过程中演变出多个变种,在某个特定的时刻,需要将它们整合到一个标准的参考过程中。这就需要检测和解决数据感知过程之间的差异。过程感知信息系统能够提供修改操作日志[1],将差异的检测直接引入到数据感知过程中。然而,在某些情况下却无法获得修改日志。面对如此境遇,差异的检测不得不通过比较修改前后的过程进行。对于检测的每一个差异,需要获得合适的修改操作,并重建修改日志。此外,数据感知过程建模工具必须满足用户友好性的要求:即使一个不精通该领域的用户也能够检测和解决它们之间的差异。

现有方法大多数都要依赖于修改日志,从控制流层面检测和解决过程模型之间的差异[2-4],而在没有修改日志的情形下,很少研究数据感知过程之间差异的问题。由于业务流程执行语言(BPEL)[5]是一种标准的描述数据感知过程组合的语言,因此本文以BPEL过程为对象,研究数据感知过程之间的差异。与过程模型相比,它不仅包含控制流信息,而且包含数据流信息。鉴于它与过程模型的本质区别,研究过程模型差异的方法可能并不适用于BPEL过程。本文提出了检测和解决BPEL过程之间差异的方法。该方法通过比较修改前后的两个BPEL过程,应用BPEL程序依赖图[6]和对应关系[7]检测和解决差异。根据对BPEL过程的块结构分解,将每种差异和对应的复合修改操作联系起来,建立了一个层次性修改日志[2,8]。案例分析表明,本文的方法能够很好地检测和解决BPEL过程之间的差异,并满足用户友好性的要求。

1 背景

在业务驱动开发中,BPEL过程变种的管理需要应用合适的修改操作将不同变种整合到一个标准的BPEL过程中。而将一个BPEL过程 P_1 经过一系列的修改操作转变成过程 P_2 是较常见的情景。对于大型BPEL过程,手工检测差异是一项相当耗时和易错的工作。在没有修改日志可用的前提

到稿日期:2015-02-11 返修日期:2015-04-01 本文受国家重点基础研究发展规划(2009CB320702),国家自然科学基金(61202003)资助。
张学伟(1988-),男,博士生,主要研究领域为过程技术、服务计算,E-mail:zxw19880707@163.com(通信作者);邢建春(1964-),男,博士,教授,博士生导师,主要研究领域为计算机控制、智能信息系统;杨启亮(1975-),男,博士,副教授,主要研究领域为软件工程、自适应软件;宋 巍(1981-),男,博士,副教授,主要研究领域为软件工程、服务计算;王洪达(1987-),男,博士生,主要研究领域为软件测试、服务计算。

下，一种最简单的方法就是找出所有被更改的元素，用修改操作原语将差异展示出来。但对于用户而言，修改操作原语和BPEL过程元素之间的关系不易被确定，且相当繁琐，因此它们难以被处理。

为了提供一种易于用户理解和掌握的方法以检测和解决BPEL过程之间的差异，本文的解决方案应该满足以下几点要求：(1)解决方案必须能够重建修改日志，呈现出BPEL过程之间转换的详细修改步骤；(2)为了提高用户的可用性，BPEL过程之间的差异应该被分类，并和它们发生的区域联系起来；(3)由于应用修改操作原语需要修改每一个被更改的元素，因此解决方案必须确保用户能够将一些修改操作原语整合成复合修改操作来解决差异；(4)解决方案能够让用户自主选择一些修改操作，并且以需要的修改顺序来完成这些操作。

基于上述要求，在没有修改日志可用情形下，给出一个检测和解决BPEL过程之间差异的技术框架，如图1所示。第一步，应用BPEL控制流程图(BCFG)描述两个BPEL过程，然后根据获得的BCFGs和保留的数据信息，采用BPEL程序依赖图(BPDG)[6]刻画BPEL过程活动间的控制依赖关系和数据依赖关系；第二步，基于BEPL程序依赖图和对应关系，检测出从原始过程转换成新过程的差异，并获得每个差异相对应的复合修改操作；第三步，根据BPEL过程结构树，建立一个由复合修改操作构成的层次性修改日志；第四步，根据用户的需求，规定复合修改操作执行顺序，应用迭代的方法确定每一步操作中的位置参数，最终解决BPEL过程之间的差异。

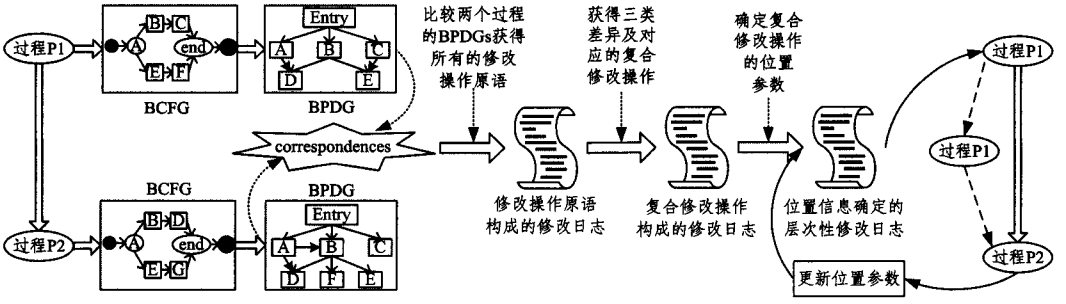

图1 检测和解决BPEL过程之间差异的技术框架

2 BPEL程序依赖图和对应关系

执行BPEL过程会产生很多执行信息，比如输入、输出变量等，如果用控制流程图描述BPEL过程，会将这些重要信息抽象掉。因此，本文采用BPEL程序依赖图(BPDG)[6]刻画BPEL过程。在BPDG中被同一控制活动所控制的所有子活动是一种无序的排列。而研究BPEL过程间的差异，无法从无序的排列中看出哪些子活动的位置发生了改变。因而，本文对BPDG中同一层次的所有活动，按照给定活动顺序从左至右进行排序。根据同一层次活动的序列及活动的层次关系，能确定控制流层面上所有活动的前后顺序关系。为了更好地检测BPEL过程之间的差异，根据BPEL过程中的控制活动对BPDG进行块结构分解。给定一个BPEL过程的BPDG，$B(G)$表示分解G获得的块结构集合，其中$b_1,b_2,\cdots\in B(G)$。在BPDG中分解的每个块结构都是一个非空子树，且同一层次的块结构之间不会发生重叠。

由于对应关系在检测和解决BPEL过程之间的差异时是必不可少的一环，因此它能够提供不同过程中的元素之间的链接。而元素的对应包含基本活动、块结构、数据依赖关系的对应。

定义1(对应关系，Correpondence) 设G_1,G_2是两个BPEL程序依赖图：

(1)令x,y分别是G_1,G_2中的活动，如果$(x,y)\in C^{N}_{1\text{-}1}(G_1,G_2)\subseteq N_1\times N_2$，那么$x$与$y$的对应关系为1:1；如果$(x,y)\in C^{N}_{1\text{-}0}(G_1,G_2)\subseteq N_1\times N_2$，那么$x$与$y$的对应关系为1:0；如果$(x,y)\in C^{N}_{0\text{-}1}(G_1,G_2)\subseteq N_1\times N_2$，那么$x,y$的对应关系为0:1，其中$\in$表示属于关系。

(2)令x,y分别是G_1,G_2中的块结构，如果$(x,y)\in C^{B}_{1\text{-}1}(B(G_1),B(G_2))\subseteq B(G_1)\times B(G_2)$，那么$x$与$y$的对应关系为1:1；如果$(x,y)\in C^{B}_{1\text{-}0}(B(G_1),B(G_2))\subseteq B(G_1)\times B(G_2)$，那么$x,y$的对应关系为1:0；如果$(x,y)\in C^{B}_{0\text{-}1}(B(G_1),B(G_2))\subseteq B(G_1)\times B(G_2)$，那么$x$与$y$的对应关系为0:1。

(3)令x,y分别是G_1,G_2中的数据依赖边，如果$(x,y)\in C^{E}_{1\text{-}1}(G_1,G_2)\subseteq E_1\times E_2$，那么$x$，$y$的对应关系为1:1；如果$(x,y)\in C^{E}_{1\text{-}0}(G_1,G_2)\subseteq E_1\times E_2$，那么$x$与$y$的对应关系为1:0；如果$(x,y)\in C^{E}_{0\text{-}1}(G_1,G_2)\subseteq E_1\times E_2$，那么$x$与$y$的对应关系为0:1。

由于G_2中存在与G_1中的一个元素x有着相同名称的元素y，因此我们认定元素x和y之间存在着1:1的对应关系；如果G_2中找不到与G_1中的元素x有相同名称的元素，我们认为是1:0的对应关系。反之亦然。此外，如果一个元素被细分成元素集合会产生1:n的对应关系，而多个元素被抽象为一个元素就构成n:1的对应关系。由于我们根据用户给定的活动的唯一标识检测活动之间的差异，因此后两种对应关系本文并没有考虑。

3 差异的检测和解决

3.1 差异

根据BPDGs和对应关系，BPEL过程之间控制流层面的差异能够直接被检测。其中，一类形式差异的出现是由于添加、删除、移动BPEL过程中的基本活动所引起的。

定义2(基本活动差异，Basic Activity Differences) 设x,y分别为BPEL程序依赖图G_1,G_2中的基本活动，b_1,b_2分别为包含x,y的块结构，$C^{N}_{1\text{-}1}(G_1,G_2)$，$C^{N}_{1\text{-}0}(G_1,G_2)$，$C^{N}_{0\text{-}1}(G_1,G_2)$为基本活动的对应集合，那么基本活动差异存在3种情形：

(1)删除基本活动(DeleteBasicActivity)的差异定义为$(x,\Phi)\in C^{N}_{1\text{-}0}(G_1,G_2)$，其中$\Phi$表示为空；

(2)插入基本活动(InsertBasicActivity)的差异定义为$(\Phi,y)\in C^{N}_{0\text{-}1}(G_1,G_2)$；

(3)移动基本活动(MoveBasicActivity)的差异定义为$(x,y)\in C^N(G_1,G_2)$,要么$(b_1,b_2)\notin C^B_{1\text{-}1}(B(G_1),B(G_2))$,要么$(b_1,b_2)\in C^B_{1\text{-}1}(B(G_1),B(G_2))$,但$x$,$y$的位置不一致。

对于移动基本活动引起的差异,则需要区分是在同一块结构移动基本活动引起的差异,还是在不同块结构之间移动基本活动引起的差异。对于后一种情况,通过比较所有1:1对应的活动、块结构进行检测。而前一种情况检测差异则需要比较对应块结构内的所有活动,找出活动顺序的改变。

除了基本活动差异,BPEL过程之间的差异有可能是通过删除或插入控制活动,也有可能是移动控制活动,还有可能是改变控制活动的类型而产生,这些修改都会引起过程中块结构的差异。下面给出块结构差异的定义。

定义3(块结构差异,Block Structure Differences) 设b_1,b_2分别为BPEL程序依赖图G_1,G_2中的块结构,$C^B_{1\text{-}1}(B(G_1),B(G_2))$,$C^B_{1\text{-}0}(B(G_1),B(G_2))$,$C^B_{0\text{-}1}(B(G_1),B(G_2))$为块结构的对应集合,块结构差异存在以下4种情形。

(1)删除块结构(DeleteBlockStructure)的差异定义为$(b_1,\Phi)\in C^B_{1\text{-}0}(B(G_1),B(G_2))$;

(2)插入块结构(InsertBlockStructure)的差异定义为$(\Phi,b_2)\in C^B_{1\text{-}0}(B(G_1),B(G_2))$;

(3)移动块结构(MoveBlockStructure)的差异定义为$(b_1,b_2)\in C^B(B(G_1),B(G_2))$,要么$(father(b_1),father(b_2))\notin C^B(B(G_1),B(G_2))$,要么$(father(b_1),father(b_2))\in C^B(B(G_1),B(G_2))$,但$b_1$和$b_2$的位置不一致;

(4)变换块结构(ConvertBlockStructure)的差异定义为块结构的类型被改变,或b_1中的子控制节点在b_2中没有对应的节点,反之亦然。

删除、插入、移动块结构的差异的检测类似于基本活动差异的检测。变换块结构的差异的检测涉及到所有1:1对应的块结构$(b_1,b_2)\in C^B(B(G_1),B(G_2))$,需要检查$b_1$和$b_2$的类型是否被改变或一个块结构中的某个子控制节点在另一个块结构中是否有对应的节点。

可执行的BPEL过程中活动的输入、输出变量或者控制活动的判定条件的改变,会导致两个活动之间的数据依赖关系发生改变,从而引起数据依赖关系的差异。

定义4(数据依赖关系差异,Data Dependence Differences) 设e_1,e_2分别为BPEL程序依赖图G_1,G_2中的数据依赖边,$C^E_{1\text{-}1}(G_1,G_2)$,$C^E_{1\text{-}0}(G_1,G_2)$,$C^E_{0\text{-}1}(G_1,G_2)$为数据依赖边的对应集合,数据依赖关系差异存在以下两种情形:

(1)删除数据依赖(DeleteDataDependence)的差异定义为$(e_1,\Phi)\in C^E_{1\text{-}0}(G_1,G_2)$;

(2)插入数据依赖(InsertDataDependence)的差异定义为$(\Phi,e_2)\in C^E_{0\text{-}1}(G_1,G_2)$。

数据依赖关系差异只与包含数据依赖的对应活动对有关,而与其是否处于同一块结构无关。如果属于不同块结构中的一对活动间的数据依赖发生改变,那么该差异所对应的修改操作发生在发起数据依赖的那个活动所处的块结构。

3.2 位置参数的确定

为了满足用户友好性的要求,BPEL过程之间的差异根据修改操作中的位置参数信息确定选择的修改操作的执行顺序,而如何确定操作中位置参数信息是我们必须要解决的问题。由于在BPDGs中引入了控制流层面上的活动前后顺序关系,BCFG所描述的控制流信息也能在BPDG中呈现,因此应用BPDGs确定复合修改操作中的位置参数。例如,图2展示了过程V_1和V_2的BPDGs。其中,在过程V_1中插入活动B和C,可获得过程V_2。于是,采用InsertBasicActivity(V_1,B,_,_)和InsertBasicActivity(V_1,C,_,_)操作。为了确保用户能够使用这两个操作,将它们中的位置参数确定为InsertBasicActivity(V_1,B,A,D)和InsertBasicActivity(V_1,C,A,D)。B和C任意一个作为确定修改操作的修改位置,都会引入修改操作之间的约束关系,即其中一个操作必须在另一个操作之前执行。

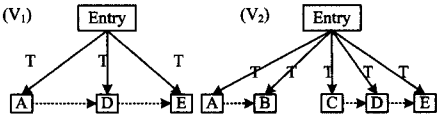

图2 计算复合修改操作中位置参数的示例

为了避免上述情况,对于给定的程序依赖图G_1,G_2和$C^N_{1\text{-}1}(G_1,G_2)$对应集合,本文用定点活动表示位置参数[5]。下面给出定点活动的定义。

定义5(定点活动,Fixpoint Activity) 设n_1,n_2为BPEL程序依赖图中的活动,如果对n_1,n_2未执行任何修改操作,那么n_1,n_2被称为定点活动,(n_1,n_2)被称为定点活动对。

例如,在图2中,(A,D)和(D,E)都是定点活动对。使用定点活动作为位置参数信息也确保了插入或移动的活动能够在控制流程图中自动与定点连接起来。在执行一个插入修改操作之后,将插入的基本活动或控制活动添加到定点活动集合中,然后重新计算位置参数以确保下一步操作能够被顺利执行。针对前文定义的3类差异以及每一种情形下的差异对应的复合修改操作,通过应用定点活动和迭代计算的方法,确定每个操作中的位置参数。基于这些位置参数确定的修改操作,用户能够以需要的修改顺序选择合适的复合修改操作解决BPEL过程之间的差异。为了让用户直观地看到解决差异的复合修改操作,下面将检测出的每一种情形的差异相对应的复合修改操作建立成一个层次性修改日志。

3.3 复合修改操作

根据3.1节定义的3类差异,能够建立每种情形下的差异对应的复合修改操作,即由几个关联的修改操作原语组合构成。其中,修改操作原语包括添加、删除活动和依赖边,即添加基本活动、添加控制活动、添加控制依赖边、添加数据依赖边、删除基本活动、删除控制活动、删除控制依赖边、删除数据依赖边。基本活动差异可以通过插入、删除或移动基本活动的修改操作解决。对于插入块结构的差异,根据插入块结构的类型,采用不同的复合修改操作,比如插入循环块结构、插入选择块结构等,而对于变换块结构的差异,我们引入转换块结构操作。对于数据依赖关系差异,其与基本活动差异类似,也能选择合适的复合修改操作解决。

3.4 层次性修改日志

根据BPDGs中活动和块结构的控制依赖所展现的层次关系,其对应的复合修改操作也能构建层次性修改日志。图3为BPEL过程的某种修改序列下的层次性修改日志。MoveBasicActivity(P_1,A_4,A_6,A_9),InsertDataDependence(P_1,e_a,A_{15},A_{19})等几个修改操作发生在根块结构b_1区域里。DelteBasicActivity(P_1,A_{11}),DelteBasicActivity(P_1,A_{13})

等操作分别发生在 b_3 的两个子块结构 b_4 和 b_5 中。在插入的选择块结构 b_6 中有 4 个插入基本活动和数据依赖边的操作，例如，InsertBasicActivity(P_1，A_{17}，A_{16}，A_{18})。而在并发块结构 b_2 中没有做任何的修改操作。使用层次性修改日志，用户很容易确定 BPEL 过程中哪一部分被修改过，提高了可用性。

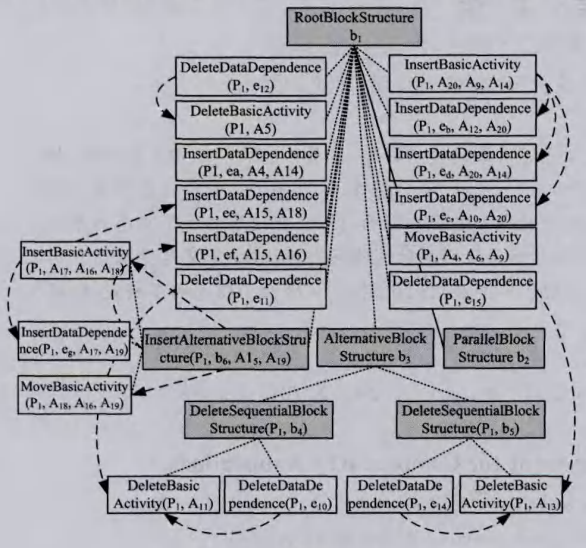

图 3　层次性修改日志

如前文所述，基于 BPDGs 用户可以按照自己需要的修改顺序选择合适的复合修改操作对 BPEL 过程进行修改。实际上，这种修改顺序也不能是任意选择的，因为一些修改操作之间存在某种约束关系，导致一个操作必须在另一个操作之前完成。比如，在过程 P 中插入基本活动 x，然后在 x 和 y 之间插入数据依赖边，反之则是不合适的。我们发现以下修改操作之间存在着约束关系：(1)首先执行删除数据依赖边操作，然后执行删除该依赖边所连接的活动的操作；(2)首先执行插入基本活动或块结构操作，然后执行插入该活动或块结构上的数据依赖边操作；(3)首先执行插入块结构操作，然后执行针对该块结构中的元素的操作。根据这些规则，修改日志中一些操作的执行序列能被确定，也能确保用户合理地选择修改操作的执行序列。

4　评估

为了展示提出的基于 BPDGs 和对应关系检测和解决 BPEL 过程之间差异的方法的适用性，本节将此方法应用于实际的 BPEL 过程中进行验证。该实例来自于 Oracle 公司 BPEL 过程管理库中的贷款服务流程。首先，将两个 BPEL 过程用 BCFGs 描述，并保留了数据信息。然后基于 BCFGs，获得了 BPEL 过程包含活动前后顺序关系的 BPDGs，并对其进行块结构分解，如图 4 所示。根据 BPDGs 和对应关系，我们发现过程 P_1，P_2 之间存在着多种差异。对于每一种情形下的差异，都能够建立与其相对应的复合修改操作。通过应用定点活动的概念和迭代计算，确定了每一步操作的位置参数信息。例如，根据定点 A_1 和 A_6，首先确定 InsertParallelBlockStructure(P_1，b_2，A_1，A_6)的位置参数，然后对位置参数重计算，确定了 InsertSequentialBlockStructure(P_1，b_3，A_7，A_6)的位置参数，以此类推，我们获得所有修改操作的位置参数。最后，根据操作之间的约束规则，找出复合修改操作之间的约束关系，建立了一个层次性修改日志，如图 5 所示。该案例分析进一步说明在用户没有可用修改日志的情形下，本文方法不仅能检测出控制流层面的差异，还能发现数据流层面的差异，并且每种差异都能找到合适的复合修改操作来解决。由复合修改操作建立的层次性修改日志很好地满足了用户友好性的需求。

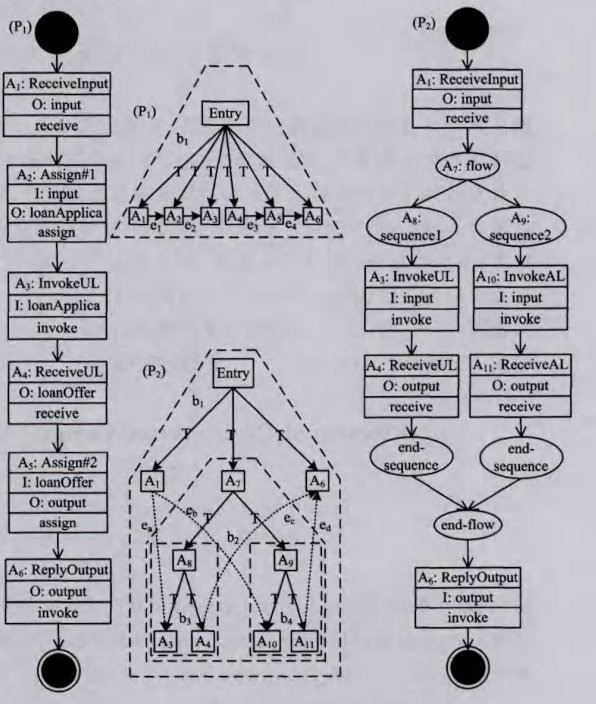

图 4　两个 BPEL 过程变种和 BPDGs

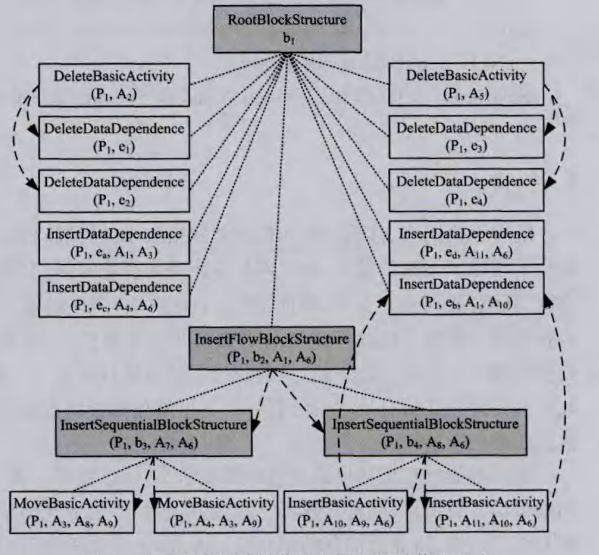

图 5　应用复合修改操作建立层次性的修改日志

5　相关工作

在业务过程开发和管理中，通常会涉及到如何检测和解决它们之间差异的问题。在模型驱动工程中，已有学者提出了检测和解决模型之间差异的一般化方法[9]。Alanen 等[4]给出了一种基于唯一标示符计算两个模型之间差异的算法。该算法类似于修改操作的重建，而本文工作主要是在与用户

(下转第 166 页)

参 考 文 献

[1] 吕建,马晓星,陶先平,等. 网构软件的研究与进展[J]. 中国科学(E辑),2006,36(10):1037-1080
Lv Jian, Ma Xiao-xing, Tao Xian-ping, et al. Research and progress on Internetware[J]. Science in China (Series E), 2006, 36(10):1037-1080

[2] Hamid R, Nezhad M, Stephenson B, et al. Virtual Business Operating Environment in the Cloud: Conceptual Architecture and Challenges [C]// OOER. 2009:501-514

[3] Garcia G, Octavio J. Agent-based cloud service composition [J]. Applied Intelligence, 2013, 38(3):436-464

[4] Zeng Liang-zhao, et al. QoS-Aware Middleware for Web Services Composition[J]. IEEE Transactions on Software Engineering, 2004, 30(5):311-327

[5] W3C. QoS for Web Services: Requirements and Possible Approaches [EB/OL]. http://www.w3c.or.kr/kr-office/TR/2003/ws-qos/

[6] 肖芳雄,李燕,黄志球,等. 基于时间概率代价进程代数的组合式软件建模和分析[J]. 计算机学报,2012,35(5):918-936
Xiao F X, Li Y, Huang Z Q, et al. Modeling and Analyzing Web Services Composition Using Timed Probabilistic Priced Process Algebra[J]. Chinese Journal of Computer, 2012, 35(5):918-936

[7] Object Management Group. UML profile for schedulability, performance and time [EB/OL]. http://www.omg.org/spec/SPTP/5

[8] Object Management Group. UML profile for modeling QoS and fault tolerance characteristics and mechanisms [EB/OL]. http://www.omg.org/spec/QFTP/

[9] Object Management Group. UML profile for modeling and analysis of real-time and embedded systems version1.0[EB/OL]. http://www.omg.org/spec/MARTE/1.0/

[10] Cambronero M E. Using UML Diagrams to Model Real-Time Web Services[C]// Proceedings of the Second International Conference on Internet and Web Applications and Services. 2007: 24-30

[11] Cambronero E M. RT-UML for modeling Real-Time Web Services[C]// Proceedings of the 3rd IEEE International Conference on Service Computing Workshops. Illinois, USA: IEEE Press, 2006:131-139

[12] 柳毅,麻志毅,何啸,等. 一种从UML模型到可靠性分析模型的转换方法[J]. 软件学报,2010,21(2):287-304
Liu Yi, Ma Zhi-yi, He Xiao, et al. Approach to Transforming UML Model to Reliability Analysis Model[J]. Journal of Software, 2010, 21(2):287-304

[13] Corellessa V, Pompei A. Towards a UML profile for QoS: A contribution in the reliability domain[J]. ACM SIGSOFT Software Engineering Notes, 2004, 29(1):197-206

[14] Bernardi S, Merseguer J, Petriu D C. Adding dependability analysis capabilities to the MARTE profile[C]// Czarnecki K, Ober I, Bruel J M, et al., eds. Proc. of the 11th Int'l Conf. on Model Driven Engineering Languages and Systems. Berlin: Springer-Verlag, 2008:736-750

[15] Cortellessa V, Singh H, Cukic B. Early reliability assessment of UML based software models[C]// Balsamo S, ed. Proc. of the 3rd Int'l Workshop on Software and Performance. New York: ACM Press, 2002:302-309

[16] Majzik I, Pataricza A, Bondavalli A. Stochastic dependability analysis of system architecture based on UML Models[C]// de Lemos R, Gacek C, Romanovsky A, eds. Architecting Dependable Systems. LNCS 2677, Berlin, Heidelberg: Springer-Verlag, 2003:219-244

[17] Rodrigues G, Rosenblum D, Uchitel S. Using scenarios to predict the reliability of concurrent component-based software systems [C]// Cerioli M, ed. Fundamental Approaches to Software Engineering, FASE 2005. Berlin: Springer-Verlag, 2005:111-126

(上接第 151 页)

最小程度的交互情况下解决过程之间差异的问题。事实上,许多编辑距离的概念也能被用于过程差异的检测。例如,Li等[10]提出了高层修改操作作为度量过程模型之间距离的基础。上述方法虽然能够计算过程之间差异的距离,却未给出具有层次性的修改操作。于是,Küster等[8]提出一种基于对应关系和单输入单输出分解的方法,并以一种用户友好的方式检测和解决差异。在此基础上,他们又提出了一种无需修改日志,应用复合修改操作和层次性分解解决过程模型之间差异的方法[2]。该方法虽然能够解决过程模型之间差异的问题,但并不适于用数据感知过程之间差异的检测和解决。

结束语 本文提出了一种检测和解决数据感知过程之间差异的方法。该方法通过比较修改前后两个数据感知过程检测出3类差异,根据每一类差异都能获得对应的复合修改操作。应用定点活动和迭代计算的方式确定所有复合修改操作中的位置参数,从而很好地解决了这些差异。下一步将在本文的基础上尝试解决数据感知过程的合并问题,并且进一步研究数据感知过程修改操作的语义和语法上的矛盾性问题。

参 考 文 献

[1] Weber B, Rinderle S, Reichert M. Change Patterns and Change Support Features in Process-Aware Information Systems[C]// Proceeding of CAiSE'07. LNCS 4495:574-588

[2] Küster J M, Förster A, Engels G. Detecting and Resolving Process Model Differences in the Absence of a Change Log[C]// Proceeding of the 6th BPM. LNCS 5240, 2008:244-260

[3] Dijkman R. Diagnosing Differences between Business Process Models[C]// Proceeding of the 6th BPM. 2008, LNCS 5240, 2008:261-277

[4] Fan S K, Zhao J L, Dou W C, et al. A framework for transformation from conceptual to logical workflow models[J]. Decision Support Systems, 2012, 54(1):781-794

[5] WS-BPEL 2.0 Specification (2007) [OL]. http://docs.oasisopen.org/wsbpel/2.0/wsbpel-v2.0.pdf

[6] Song W, Ma X X, Cheung S C, et al. Refactoring and publishing WS-BPEL processes to obtain more partners[C]// Proceeding of ICWS. 2011:129-136

[7] Pottinger R A, Bernstein P A. Merging models based on given correspondences[C]// Proceeding of the 29th VLDB. 2003:862-873

[8] Küster J M, Förster A, Engels G. Process Merging in Business-Driven Development: IBM Research Report, No. 3703 [R]. Switzerland: IBM Zurich Research Laboratory, 2008

[9] Herrmann C, Krahn H, Rumpe B, et al. An Algebraic View on the Semantics of Model Composition[C]// Proceeding of the 3rd ECMDA-FA, 2007. LNCS 4530:99-113

[10] Li C, Reichert M, Wombacher A. On measuring process model similarity based on high-level change operations[C]// Proceeding of Conceptual Modeling. LNCS 5231, Barcelona, 2008:248-264