# BPEL 流程间死锁检测研究

黄俊飞 杨学红 宫云战

(北京邮电大学网络与交换技术国家重点实验室 北京 100876)

**摘 要** BPEL 流程的正确性问题越来越受到广泛关注.作为一种支持并发机制的 Web 服务组合语言,不正确地使用并发和同步操作或不正确的交互都可能会使单一流程内部或流程间产生死锁.死锁问题是一类常见的并发缺陷,可能严重影响系统的可靠性与可用性.更为严重的是,对于业务流程组合这种可靠性和安全性要求极高的系统,一旦发生死锁,将会对业务系统产生严重的后果.为此作者在分析 BPEL 特性和死锁特征的基础上,提出了一种可在 BPEL 流程设计阶段检测流程死锁的方法,从而提高流程的可靠性,减少系统的维护代价.文中提出了一种从全局出发进行分析的方法,结合流程间执行的上下文信息,可以有效检测流程间的通信死锁,该方法也适用于其它与流程执行上下文相关的缺陷检测.

**关键词** BPEL;死锁;流程摘要;组合服务

**中图法分类号** TP311 **DOI 号**: 10.3724/SP.J.1016.2011.02427

## Research on Deadlock Detecting of BPEL Inter-Processes

HUANG Jun-Fei YANG Xue-Hong GONG Yun-Zhan

(*State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing* 100876)

**Abstract** The correctness of BPEL process is getting more and more attentions. As a concurrent supported language, improper or incorrect use of concurrent and synchronization may cause intra-processes or inter-processes deadlock, which is a kind of common concurrent defects seriously affect reliability and availability of system. Even more serious, for the critical-task system, such deadlock will cause a serious consequence. So, based on analysis of BPEL and deadlock characters, we propose a detecting method in the design phase, reducing system maintenance costs, improving process reliability. This paper presents a method from the overall analysis view to resolve the above problems, combing the execution context information among processes, so the communication deadlock can be detected. This method is also applicable to other defects that related to the inter-processes.

**Keywords** BPEL; deadlock; process summary; service composition

## 1 引 言

操作系统中"死锁"的概念是指:两个或多个并发进程各自占有某种资源而又都等待别的进程释放它们所占有的资源.死锁产生的原因包括:一是系统竞争临界资源.当系统所拥有的某种临界资源个数比各个进程要求该资源的总数要少时,则有可能引起多个并发进程对资源的竞争而产生死锁;二是进程推进顺序不当.进程运行过程中,由于请求和释放资源的顺序不当,而产生死锁.不管是在操作系统中,还是其它并发程序语言中,死锁都是人们不希望

收稿日期:2011-06-11;最终修改稿收到日期:2011-10-22.本课题得到国家自然科学基金(91018002)资助.**黄俊飞**,男,1977 年生,博士,讲师,主要研究方向为软件测试与云计算.E-mail: huangjunfei@bupt.edu.cn.**杨学红**,女,1983 年生,博士研究生,主要研究方向为软件测试与软件工程.**宫云战**,男,1962 年生,教授,博士生导师,主要研究领域为软件测试与高可信计算.

见到但又不可回避的系统特性.在并发系统的运行过程中,为保证系统的运行安全可靠,必须有效地解决死锁问题.要想有效地解决死锁,首要的任务就是发现死锁.因此,死锁检测十分必要.然而 BPEL 的特殊性也决定了这些传统的死锁检测方法很难应用于 BPEL 的死锁检测.在基于 BPEL 的组合流程中,各参与的伙伴服务与流程之间通过 SOAP 消息进行通信,因此消息接收的正确性是流程死锁的一个最为突出要素,它可使整个流程瘫痪,无法进行自动化组合服务执行.从目前来看,解决 BPEL 流程死锁的数学工具有图理论方法、Petri 网方法和进程代数方法等,图论尽管能够检测出控制 link 造成的控制依赖死锁,但不能表达系统的动态运行行为.采用 Petri 网方法能够表达流程动态运行行为,并且能检测出单一流程内部的死锁,但当前的 Petri 死锁检测方法,并没有说明如何处理流程间的死锁.进程代数的方法采用移动工作台也只能检测出单一流程内部的控制依赖死锁,并且这些方法很难应用于大规模的流程验证.

BPEL 流程通过 invoke 活动调用参与的伙伴服务,该调用方法与传统程序语言中的函数调用有相似之处.传统的函数间分析通过计算函数摘要[7-8]的方法提取函数信息,在具体的函数调用使用函数摘要进行替换.与此类似,我们提出了流程摘要的概念,提高静态流程间死锁检测的可靠性和完备性.

## 2 BPEL 的同步和异步交互

BPEL 流程与外部流程之间的交互有两种方式:同步和异步.如果某业务流程请求能够在较短的时间内被处理,则流程可以使用同步机制与外部服务进行通信,即请求-响应操作.通常有如下两种方式进行同步操作.

(1)通过 receive 和 reply 活动对实现,如流程使用 receive 活动等待客户端的购买订单流程,然后使用同步的 reply 活动发送发票信息.

```
〈receive partnerLink="customer"
 portType="purchaseOrderPT"
 operation="sendPurchaseOrder" variable="PO"/〉
〈reply partnerLink="customer"
 portType="purchaseOrderPT"
 operation="sendInvoke" variable="Invoice"/〉.
```

(2)使用 invoke 活动实现服务的同步调用,如银行贷款中调用风险评估的伙伴服务.

```
〈invoke inputVariable="request"
 name="InvokeLoanAssessor" operation="check"
 outputVariable="risk" partnerLink="assessor"
 portType="riskAssessmentPT"/〉.
```

BPEL 还支持长时间的业务交互,通过异步消息交换实现,例如上例中的 reply 活动可用异步交互的 invoke 活动取代.

```
〈invoke partnerLink="customer"
 portType="InvoiceCallbackPT"
 operation="sendInvoice" variable="Invoice"/〉.
```

业务流程在执行过程中,可能被多次实例化,因此必须要确保异步交互的返回消息发送到正确的业务流程实例.BPEL 提供了应用程序级的机制实现了此功能,定义了关联集合属性,关联集合在其语义上类似于延迟绑定的常数,关联集合的绑定由特别标记的消息发送或接收操作来触发.比如在一个旅行订票流程中,当流程启动之后,用户需要能够查询该流程的状态,并能取消该流程,这就需要关联集合的支持来确保后续的请求消息能够绑定到相同的流程实例中.

## 3 BPEL 死锁分析

### 3.1 BPEL 死锁特征

死锁是指系统处于这样的状态:任何一个进程(或线程)都保持现有的状态且无法前进.对死锁的认识由来已久,最早是 Dijkstra 在 1965 年提出的[2].后来 Coffman 又提出了死锁发生的 Coffman 条件[3],但该条件只是死锁发生的必要条件,因为即使系统不满足这些条件,可能也发生死锁.例如一个系统内有两个进程都处于等待状态,它们的推进都依赖于对方进程的资源,这时就产生了死锁.在 Web 服务世界里,如果流程等待的通信消息永远不会发生,也可认为发生了死锁,即流程的执行处于一种无限等待状态,它的推进需要伙伴发送相应的 SOAP 消息,此时的伙伴服务可能由于某种原因已经结束而没有返回任何 SOAP 消息,也就是说流程等待未来时刻内不会发生的事件,这样就造成了流程处于一种死锁状态.因此死锁是流程的一种缺陷和错误,不同于以往的并发语言,BPEL 死锁可能存在于单一流程内部,也可能存在于不同的流程之间的消息通信.下面对 BPEL 流程的死锁进行分类.

#### 3.1.1 单一流程内部的死锁

BPEL 使用 link 进行并发活动之间的同步,不正确的使用 link 结构,会使流程中存在死锁:如果 BPEL 流程存在控制依赖环,说明流程中的 link 使用引入依赖环路,也就是流程中存在控制依赖死锁,

当前大多数 BPEL 开发环境都支持 link 产生的控制依赖死锁的验证，如 ActiveBPEL. 图 1 给出了 3 种不同的控制依赖死锁结构，在与这些 link 连接的活动中，每一个活动都在等待集合中另一活动的结束才开始执行，所有活动都处于无终止的等待状态，从而导致并发结构的不可完成.

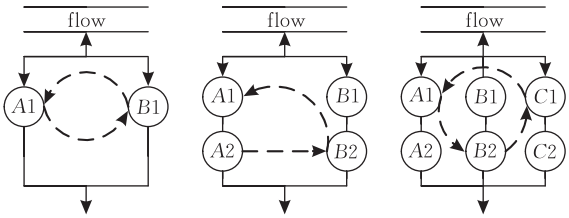


图 1　link 结构死锁

### 3.1.2　流程间的消息通信死锁

如果两个流程间的交互活动形成环，如图 2 所示，虚线代表的是流程间的通信依赖造成的环，左边的流程等待右面 order 发送相应的消息，然后发送 confirm 消息；而右面的流程等待左面流程先发送 confirm 消息，然后再发送 order 消息. 由于通信依赖形成了循环等待环，说明流程间存在死锁. 流程间的资源循环等待环与流程内部的控制依赖死锁环类似，但是环并非流程死锁的充要条件，如果流程内部不存在控制依赖环，或流程间不存在通信依赖环，仍然可能导致流程间存在死锁.

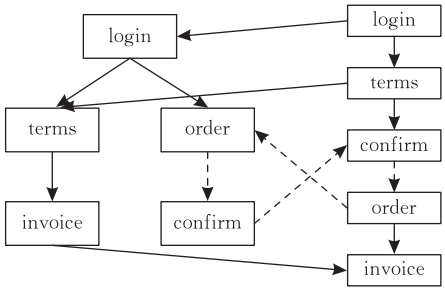


图 2　流程间循环等待死锁

在流程间的通信过程中，必须要确保流程中的每一个 receive 活动都能够成功地接收到伙伴流程提供的输入消息（主要是对异步调用而言），否则将会导致流程发生死锁. BPEL 通过 SOAP 消息与外部伙伴 Web 服务进行交互，参与交互的 Web 服务可能是简单的原子 Web 服务，也可能是组合 Web 服务，如图 3 所示是一个存在死锁的流程间交互实例.

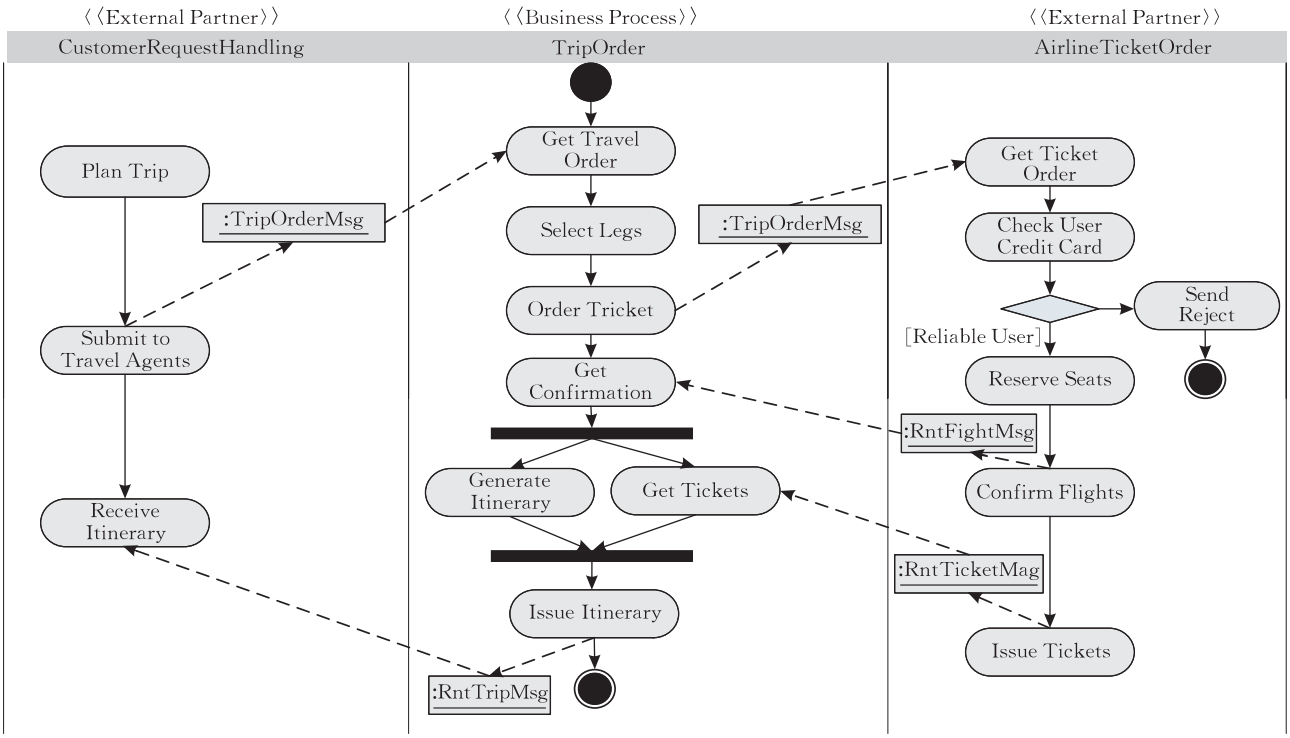


图 3　订票流程实例

该订票流程包括两个伙伴外部流程，虚线表示流程间的通信. 整个业务流程的执行过程是：首先业务流程（TripOrder）接收到一个外部请求（TripOrderMsg），再根据接收到的请求消息调用 Web 服务（AirlineTicketOrder），最后将结果（RtnTripMag）返回给请求者（CustomerRequsetHanding），从而结束整个过程. 注意，在流程接收到 TripOrder 消息 TripOrderMag 后，一方面，流程 TripOrder 将接受

到的消息发送到 AirlineTicketOrder 流程;另一方面,活动 GenerateItinerary 为客户返回一个恰当的旅行计划.另外,当活动 CheckUserCreditCard 判定客户是一个可靠用户时,通过 IssueeTickets 发送一个数据 RtnTicketMag 给 TripOrder 流程,从而使 IssueIterary 能够执行,结束整个订单流程.

通过对上述流程的分析可知,当订单流程从一个不可靠的客户那里接收到消息,并且 AirlineTicketOrder 活动对用户的信用度进行评价,使得该伙伴流程不能返回 TicketOrder 流程需要的消息,从而使 ReceiveItinerary 和 GetConfirmation 活动处于无限等待状态.我们将这种导致流程处于无限等待的现象也称之为死锁.

单一流程内部的死锁通过判断 link 连接的目标活动是否包含在其直接前驱活动中很容易进行检测.如果是,则说明存在死锁.而流程间的死锁则相对复杂,流程间的循环等待死锁也可以通过建立流程间的依赖图判断是否存在环而检测,但对于图 3 所示的消息接收导致的死锁,则不能简单地通过环的形式检测,需要有新的检测方法.本章在基于缺陷的 BPEL 静态检测过程中,通过引入"流程摘要"来解决流程间通信敏感的缺陷问题.

### 3.2 流程摘要

对组合流程的分析包括两部分:单个流程内的分析以及流程间的分析.流程间的分析可以从两个方面进行研究:上下文敏感和上下文不敏感分析.上下文敏感的分析方法需要根据流程调用上下文信息计算流程调用结果.由于组合流程中的伙伴 Web 服务的具体实现细节是隐藏的、对外透明的,因此进行上下文敏感的流程间分析需要对调用流程进行抽象.

所谓流程摘要和传统程序语言分析中的函数摘要有异曲同工的作用,流程间敏感方法根据不同组合流程调用的上下文信息分别计算流程的调用效果,非流程间敏感方法则不区分流程间的调用信息.本文提出一种基于流程摘要的流程间敏感的分析方法,将流程间的缺陷检测问题转化为流程内部的缺陷检测问题.流程摘要是对流程的部分抽象,通常只描述和缺陷相关的信息.不同的缺陷需要记录不同的流程摘要.流程间分析的一个非常重要的属性是确保流程间交互不存在死锁,因为流程间的相互阻塞的现象是不允许的.

Web 服务通过 WSDL 向外部暴露其所能提供的功能,BPEL 作为一种组合服务,对外也通过 WSDL 供其它服务调用.除此之外不提供任何的服务实现细节,对用户来说是透明的.如图 3 中的例子,如果不考虑流程内部的实现细节对流程调用间的影响,伙伴流程 AirlineTicketOrder 总会向主流程中返回消息,这样就导致存在一个死锁的漏报问题.

基于缺陷的流程间静态分析,根据不同的分析目标属性(也成缺陷特征)生成不同的流程摘要信息.流程摘要可以定义为三元组 $PSummary=\langle precondition, defect\text{-}feature, postcondition\rangle$:包含了调用前上下文环境需满足信息,即调用前应检查并满足的前置条件集合;包含调用后对上下文环境所带来的影响信息的后置条件集合;包含和分析目标属性相关的缺陷特性信息集合.每一个前置条件、后置条件、函数特性信息都带有相应的条件信息.同样采用了变量的抽象取值区间进行表示.

**定义 1.** 缺陷特征($df$).给定缺陷模式 $fsm$,它可以检测某一程序属性 feature 是否违反了程序语法或语义规则,该 feature 对应的程序变量即为缺陷模式 $fsm$ 的缺陷特征,即为 $df(fsm)$.

缺陷特征可以理解为缺陷检测是否与程序中某个变量相关,例如初始化变量(UIV)模式必然与一个变量相关,Link 非法使用(IUL)模式必然与一个同步 link 相关,流程间死锁由于等待的消息无法到达而处于无限等待状态.这里的消息也就是 BPEL 变量,因此也是与变量相关的.在基于缺陷的流程间死锁检测过程中,当遇到流程(在不引起混淆的情况下,也称为服务)调用时,它只关心该被调流程的执行是否会引起当前的缺陷状态发生变化.例如对于流程间通信死锁(Inter-Process Communication Deadlock,IPCD)来说,它只关心被调流程是否会返回调用流程等待的消息.对于 UIV 来说,只关心是否对使用的消息进行了有效赋值.也就是说流程摘要与具体的检测缺陷相关.

### 3.3 基于流程摘要的死锁检测

基于 BPEL 的服务组合通过发送 SOAP 消息调用提供具体功能的服务(单一 Web 服务或组合 Web 服务),被调服务的具体实现细节对调用流程来说是透明的,因此为了准确获得流程调用的上下文信息,根据抽象解释理论,可以通过流程摘要对服务调用的上下文信息和约束信息进行抽象.对组合中的每一个 BPEL 流程进行单独分析的同时,生成该流程相应的流程摘要,当流程被调用时,可用流程摘要进行具体的分析.

#### 3.3.1 流程依赖关系

若流程 A 调用了流程 B,则说流程 A 依赖于流程 B,这种依赖关系可以用调用图的形式进行表示,

如图 3 中流程间的调用关系可以用图 4 中调用关系进行表示.

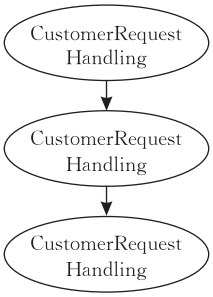


图 4　流程调用图

流程摘要是对流程内容的抽象，尽管使用流程摘要比直接使用其提供的 WSDL 接口信息相对复杂，但却可以提高流程的检测精度，比如上面提到的流程间的通信死锁. 通过流程依赖图可以获得流程调用之间需要满足的条件.

3.3.2　流程摘要分析

首先按照流程调用关系获得流程调用的拓扑关系，然后按照拓扑逆序的方法计算每一个流程的流程摘要，流程摘要的计算取决于待检测的缺陷特征. 例如对于未初始化变量缺陷来说，图 3 中，AirlineTicketOrder 流程前置条件(pre-condition)，即{TicketOrderMsg:[initialized]}. 而对通信死锁而言，为了避免死锁，正确返回调用流程等待的信息，需要保证客户的风险等级是可靠的，因此 pre-condition={risklevel:[reliable]}. 也就是说当 TripOrder 流程调用 AirlineTicketOrder 流程时，检查发送的 SOAP 消息中相应客户的风险等级，如果是非可靠用户，则会导致调用该流程时产生流程间通信死锁缺陷. 也就是说按照待检测缺陷特征(defect-feature)产生前置条件.

为了准确获得缺陷相关的流程摘要，每一个前置条件和后置条件都定义了相应的抽象域，该约束域取决于缺陷特征 $df$. 对于任意 $vi \in df$，前置条件可以表示为 $constraints = \{\langle value(v_1), C_1\rangle, \langle value(v_2), C_2\rangle, \cdots, \langle value(v_n, C_n)\rangle\}$，$value(v_i)$ 表示当条件 $C_i$ 为 true 时，$v_i$ 的取值范围，用扩展的区间进行描述. 同样可以计算相应的后置条件信息.

为了进行直观表示，这里以文中给出的 3 种死锁情况中响应消息丢失的 IPCD 为例进行说明，其缺陷状态机包含 6 个状态{start，request，response，unresponse，error，end}，图 5 是相应状态之间的转换图. 每一个状态机代表一个异步响应调用，request 代表发送请求消息，response 代表被调用流程返回给发送请求的流程的响应，而 Unresponse 代表没有发送响应消息，这样便会使调用流程出于一种无限等待状态，即我们认为的死锁.

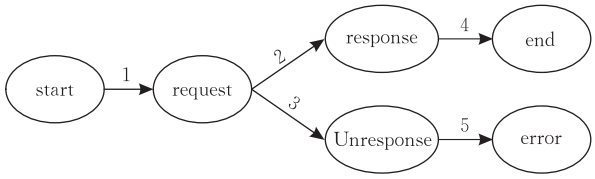


图 5　流程间通信死锁状态转换图

算法思想：考虑服务间通信形式，如果是同步调用(invoke 活动同时具有输入输出参数)，则流程间的通信不可能存在死锁. 但如果是异步调用(invoke 活动只具有输入参数)，则需要对该异步调用消息建立相应的 IPCD 状态机，如果流程间的通信不能确保始终返回响应消息，则说明存在通信死锁故障. 简单的循环等待通信消息可以通过建立流程依赖图的形式进行判断，复杂的死锁需要“流程摘要”的支持.

3.3.3　流程间死锁检测算法

流程间的死锁检测首先是获得被调流程的流程摘要，算法 1 给出了流程摘要的生成算法，算法依赖于流程依赖图(CFG)，首先判断 DCFG 中是否存在环，如果存在，则报告一个死锁，然后采用遍历算法获得依赖图的拓扑结构队列.

**算法 1.**　流程摘要生成算法.

```
procedure processSummary (DCFG dcfg){
1.    if dcfg contain cycles
2.          report deadlock error;
3.    get the topology list from dcfg;
4.    reverse the list;
5.    while (!list,isempty()){
6.          p=list.delete();
7.          old=p.getSummary();
8.    analysis p locally according to defect feature;
9.          new=generate new summary for p;
10.          if(old!=new)
11. add all the callprocess which invoke p to the list;
          }
      }
```

流程间的死锁检测，主要是当流程通过 invoke 活动调用外部伙伴服务时，需要考虑流程间上下文敏感的分析，算法 2 所示给出了当遇到异步调用的时候流程间敏感的分析算法片段.

**算法 2.**　流程间的死锁检测.

```
...
1.  if a is a asynchronous invocation
2.  create the fsm for a;
4.  get summary of callee process;
```

5. *precondition*=*summary*.getPrecondition();
6. if *a*.*fsm* violates the *precondition*
7. then report an *error*;
8. *postcondition*=*summary*.getPostcondition();
9. merge the *postcondition* to the current intra-process analysis
10. …

3.3.4 实例及分析

利用 IPCD 状态机和流程摘要对图 3 中的订票流程通信实例(TripOrder)及航空旅行订票流程(Airline)图 6 进行了具体分析,其中在 Airline 中,实线箭头代表的是同步请求响应操作,其它的箭头是异步请求操作,箭头上的名字是流程间传递的消息.结果如表 1 所示,分别给出了带有流程摘要和不带流程摘要的分析结果.在 Airline 流程中,由于不存在使流程终止的活动,每个被调流程都能够成功地返回消息,因此不存在流程间的通信死锁.

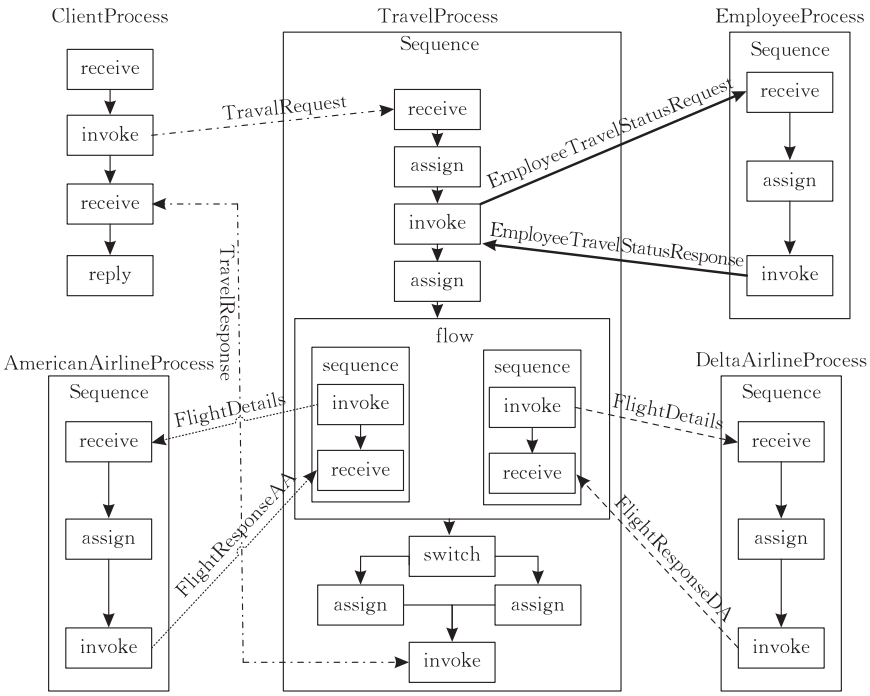


图 6 航空旅行订票流程

**表 1 对比分析结果**

| 待测流程 | 参与的流程个数 | 不带流程摘要的分析结果 | 带流程摘要的分析结果 |
|---|---|---|---|
| TripOrder | 3 | 0 | 2 |
| Airline | 5 | 0 | 0 |

## 3.4 相关工作

近年来已经出现了大量有关死锁的检测方法和工具[18-21],按照是否执行代码可以分为静态检测和动态检测,静态分析验证技术,不需要运行软件,通过对代码或模型的分析验证就能发现部分潜在的缺陷,大大缩短了死锁的发现时间周期.

基于类型约束的死锁静态检测:类型系统定义了每个变量和表达式的类型,所谓类型即属性或语义.类型检查是指检查代码中的变量和表达式的类型是否满足类型系统的定义.文献[1]通过静态分析将 Java 源代码转换成一组谓词,再通过定理证明检测其是否满足规约,从而验证死锁.FindBugs[6]是一款针对 Java 基于静态程序分析的工具,它通过检查程序是否含有"缺陷模式"来发现缺陷,主要技术是在语法层面上将程序与已知的可能导致缺陷的代码模式进行比较.死锁的存在严重影响了程序质量,文献[17]提出了基于 FSM/Petri 网的分析方法,文献[16]基于移动工作台 MWB 的 Π 演算的形式验证 BPEL 死锁.

流程间的死锁是一种隐含死锁,因此需要对流程间交互进行深入研究[13-15].有关 BPEL 的死锁研究:文献[9]通过对 CWB 以 PAC 的方法修改支持 BPE 演算,从而对 BPEL 流程进行分析,能够对死锁进行验证,但这种方法复杂度较高,有可能出现状态空间爆炸的问题.文献[10]采用 Petri 网技术对 BPEL 的控制流进行了分析,对流程间的通信进行了建模,给出了造成死锁的可能情况:(1)接收请求者发送的消息的活动没有执行;(2)请求者发送的消息被接收,但是由于某些原因,向请求者发回消息

的活动未执行(例如执行返回活动之前,流程被迫终止);(3)请求响应消息在传输过程中丢失.这些情况的出现都将会导致流程间的通信死锁,即请求流程执行死锁.另外如果请求流程发送的消息不能达到接收流程,则将会导致接收流程死锁.然而由于没有对数据流信息进行分析,缺少对交互活动的执行条件的计算.文献[11]给出了企业间业务流程 IEBP 的建模和死锁验证方法,采用两步抽象技术:首先使用符号抽象图(Symbolic Observation Graph,SOG)技术对每一个局部流程进行抽象,这样做有两个好处,一是可以将原流程的分析转化为 SOG 的分析;二是可以隐藏流程的内部结构和组织形式,符合 IEBP 的上下文需求.其次,通过组合局部 SOGs 获得 IEBP 的抽象,从而可以有效进行全局抽象分析,如死锁检测.

我们的方法与文献[11]提出的方法类似,通过流程摘要对流程进行抽象,并且流程摘要是缺陷相关的,不同缺陷的流程摘要是不同的.流程摘要只关心和具体缺陷相关的流程信息,而隐藏了其它流程内部实现细节,避免了对流程调用图进行全局分析的局限性,并在实验室自主研发的 BPEL 测试工具中[22]验证了该方法的可行性和有效性.

## 4　结束语

精确的流程间分析对缺陷检测来说是非常重要的,因为复杂的业务流程通常需要进行流程间的通信.本文提出了"流程摘要"的概念用来有效解决流程间的分析问题."流程摘要"包含了 3 个属性:前置条件用来记录调用流程对被调流程时序安全属性的影响约束,当违反这一约束时,将会导致被调流程的执行存在缺陷.特征属性用来记录特定缺陷相关的信息,而后置条件记录了被调流程的返回信息对调用流程的影响.通过使用"流程摘要"能够有效地检测流程间通信造成的死锁.

将来的工作主要是分析影响流程间缺陷检测的精度的因素,提高检测效率.

## 参　考　文　献


[1] Koshkina M, Breugel F. Modelling and verifying Web service orchest ration by means of the concurrency workbench. ACM SIGSOFT Software Engineering Note, 2004, 29(5): 1-10

[2] Dijkstra E W. Solutions of a problem in concurrent programming control. Communications of the ACM, 1965, 8(9): 569

[3] Coffman E G et al. System deadlocks. ACM Computing Surveys(CSUR), 1971, 3: 67-78

[4] Flanagam C, Leino K M, Lillibridge M et al. Extended static checking for java//Proceedings of the ACM SIGPLAN 2002 Conference on Programming Language Design and Implementation(PLDI'02). 2006: 234-245

[5] Farchi E et al. Concurrent bug patterns and how to test them//Proceedings of the 17th International Symposium on Parallel and Distributed Processing. Nice, France, 2003: 286

[6] Hovemeyer D, Pugh W. Finding bugs is easy. SIGPLAN Not, 2004, 39(12): 92-106

[7] Yorsh G, Yahav E, Chandra S. Generating precise and concise procedure summaries//Proceedings of the 35th Annual Acm Sigplan-Sigact Symposium on Principles of Programming Languages. San Francisco, USA, 2008: 221-234

[8] Gulwani S, Tiwari A. Computing procedure summaries for interprocedural analysis//Proceedings of the Programming Languages and Systems. Braga, Portugal, 2007: 253-267

[9] Fu X, Bultan T, Su J. Analysis of interacting BPEL Web services//Proceedings of the 13th International Conference on World Wide Web. New York, NY, USA, 2004: 621-630

[10] Ouyang C et al. Formal semantics and analysis of control flow in WS-BPEL. Science of Computer Programming, 2007, 67: 162-198.

[11] Lohmann N, Massuthe P, Stahl C, Weinberg D. Analyzing interacting BPEL processes//Dustdar S, Fiadeiro J L, Sheth A P eds. Proceedings of the International Conference on Business Process Management, BPM 2006. Vienna, Austria. Lecture Notes in Computer Science 4102. Springer-Verlag, 2006: 17-32

[12] Klai K, Tata S, Desel J. Symbolic abstraction and deadlock-freeness verification of inter-enterprise processes//Proceedings of the 7th International Conference on Business Process Management. Lecture Notes in Computer Science 5701. 2009: 294-309

[13] Lohmann N, Massuthe P, Stahl C, Weinberg D. Analyzing interacting WS-BPEL processes using flexible model generation. Data and Knowledge Engineering, 2008, 64(1): 38-54

[14] Cheng Y, Wang Z. Research on reachability verification of Web service composition. World Congress on Software Engineering, 2009, 233-237

[15] Aalst W M, Mooij A J, Stahl C, Wolf K. Service interaction: Patterns, formalization, and analysis. Formal Methods for Web Services, 2009, 88

[16] Sone Yan, Gao Chun-Ming. Deadlock checking of BPEL4WS based on mobility workbench. Computer Engineering, 2007, 33(1): 92-94, 97(in Chinese)
(宋艳,高春鸣. 基于移动工作台的 BPEL4WS 死锁验证. 计算机工程, 2007, 33(1): 92-94, 97)

[17] Fu Jian-Ming, Han Guan-Peng, Zhu Fu-Xi. Two logical methods of deadlock analysis. Journal of Wuhan University (Natural Science Edition), 1999, 45(3): 291-294(in Chinese)
(傅建明，韩光鹏，朱福喜. 两种死锁分析的逻辑方法. 武汉大学学报(自然科学版)，1999，45(3)：291-294)

[18] Tetsuya Maruta, Senichi Onoda, Yoshitomo Ikkai et al. A deadlock detection algorithm for business processes workflow models//Proceedings of the 1998 IEEE International Conference on Systems, Man, and Cybernetics. San Diego, CA, USA, 1998: 611-616

[19] Senichi Onoda, Yoshitomo Ikkai, Takashi Kobayashi et al. Definition of deadlock patterns for business processes work on models//Proceedings of the Annual Hawaii International Conference on System Sciences. Maui, Hawaii, 1999: 1-4

[20] Ezpeleta J, Colom J M, Martinez J. A Petri net based deadlock prevention policy for flexible manufacturing systems. IEEE Transactions on Robotics and Automation, 1995, 11(2): 173-184

[21] Chen Sheng, Bao Liang, Chen Ping et al. Study of algorithm on data race and deadlock detection for BPEL process. Journal of Xidian University, 2008, 35(6): 1056-1063(in Chinese)
(陈胜，鲍亮，陈平等. BPEL 流程数据竞争和死锁检测算法研究. 西安电子科技大学学报，2008，35(6)：1056-1063)

[22] Yang Xue-Hong, Huang Jun-Fei, Gong Yun-Zhan. Research on the static defect detecting in BPEL. Journal of Beijing University of Posts and Telecommunications, 2011, 34(2): 108-112(in Chinese)
(杨学红，黄俊飞，宫云战. BPEL 静态缺陷检测方法研究. 北京邮电大学学报(自然科学版)，2011，34(2)：108-112)

**HUANG Jun-Fei**, born in 1977, Ph. D., lecturer. His main research interests include software testing and cloud computing.

**YANG Xue-Hong**, born in 1983, Ph. D.. Her main research interests include software testing and service computing.

**GONG Yun-Zhan**, born in 1962, professor, Ph. D. supervisor. His main research interests include software testing and dependable computing.

**Background**

Deadlock is an important aspect that obstructs the correctness of BPEL process. Previous work focuses on defecting deadlocks by only using formal methods (such as automata, process algebra and Petri net) or graphic theory, this paper analyzes the deadlock situations existed in intra-processes or inter-processes.

Be different from the above works, this paper focuses on the deadlock inter-processes and aims to propose a defect-based method to detect the deadlocks existed in BPEL process, the state machine is used to express the deadlocks and the "processes summary" is used to record execution information of process when detecting deadlocks inter-processes. According to this method, the process dependency graph is constructed to express relations among processes, if there is a cycle, it can say deadlock occurs. Besides this, if the message is not returned from the callee process to the caller process, there is also a deadlock, in order to avoid the state explosion; "process summary" can be used to instead of analyzing the whole process. This work is supported by National Natural Science Foundation of China under Grant No. 91018002.