# 一种 BPEL 结构演化算法研究*

孙 晶，李东方
（北方工业大学 计算机学院，北京 100144）

**摘 要：** 针对 BPEL 并发同步引发的数据竞争、流程阻塞等问题，提出了一种 BPEL 结构演化算法。该算法的基本思想是利用 BPEL 基于 XML 编写的事实，使用 XML 解析工具对 BPEL 的结构进行调整，达到规避失败的同步过程，删除冗余 link 结构的目的。利用自主开发的转换工具实现 BPEL 到形式化描述语言 LOTOS 的转换，通过对演化前后的 BPEL 流程作模型检测，来验证该演化方法的可行性。实验结果表明，BPEL 结构演化算法能够有效缓解 BPEL 并发同步带来的数据竞争、流程阻塞问题。

**关键词：** 服务演化；并发；BPEL 结构
**中图分类号：** TP311；TP301.6 **文献标志码：** A **文章编号：** 1001-3695(2016)09-2637-04
doi:10.3969/j.issn.1001-3695.2016.09.016

## Study of algorithm on evolution for BPEL structure

Sun Jing, Li Dongfang
(*College of Computer Science, North China University of Technology, Beijing* 100144, *China*)

**Abstract:** For the problems of the BPEL process blockage and data competition caused by concurrent and synchronization, the thesis proposed an algorithm about BPEL structure evolution. The basic idea of this algorithm was intended to adjust BPEL structure via XML parsing tools to avoid the synchronization failure and deleted the redundant link structure, based on the fact that BPEL is coded with XML language. Then using self-developed conversion tool to implement the transformation from the BPEL to the formal description language LOTOS, verifying the feasibility of the algorithm by conducting the model checking on the BPEL. Experimental results show that this algorithm can relieve the problems of the BPEL process blockage and the data competition caused by concurrent and synchronization.

**Key words:** service evolution; concurrent; BPEL structure

## 0 引言

Web 服务是一种独立开放的应用程序组件，它具有完好的封装性、高度互操作性、动态性等特点。Web 服务组合通过对基本的 Web 服务进行整合来提供新的服务，使系统的功能可以灵活地扩展。通常，Web 服务的组合方式分为服务编制和服务编排[1]。服务编制通常用于专用业务流程中，它使用一个可执行的中心流程来协同内部及外部的 Web 服务交互；服务编排更多地强调协同工作和业务合作能力，通过消息的交互序列来控制各个部分资源的交互。

为了适应用户需求、运行平台以及外部环境的不断变化，服务组合需要不断地进行演化。文献[2]对常见的服务组合演化类型作出了较全面的总结，为应对演化带来的特殊要求提出了一个面向演化的服务生命周期方法学。文献[3]以基于服务组合的可信软件为研究对象，提出了基于服务组合的可信软件动态演化机制，从演化操作、演化结果、演化实施过程等方面保证了服务组合演化机制的可信性。文献[4]提出了语义 Web 环境下的一种服务组合演化方法。该方法将演化周期划分为三个部分六个阶段，从系统工程的角度分析服务组合演化过程，在实现演化需求的同时保证了描述服务的语法和语义的一致。文献[5]从体系结构的角度关注服务的动态演化，使用 SOA 代数模型描述服务体系结构，将服务的体系结构演化转换成代数系统的变换，从而使用数学的方法分析服务体系结构动态演化中遇到的问题。此外还提出了服务内部与服务之间两种服务动态演化模式。文献[6]提出了一个服务组合动态演化过程框架，定义了此框架下的实例可迁移性标准，此标准允许更多的服务组合实例迁移到目标流程下，并且在迁移后不会引入死锁等动态演化的错误。文献[7]针对服务演化的一致性判定问题，从服务的结构层和非功能层两个角度建立了基于变化的服务描述模型，并在该模型的基础上引入了演化一致度对演化一致性进行度量。文献[8]提出了一种基于服务的信息系统机构，将服务演化问题分解到原子服务处和组合服务层上的问题，并给出了两个层次上保证透明性的基本原理以及具体实现方法。文献[9]提出了一种基于反射的 BPEL 业务过程动态演化方法，该方法从用户和系统两个层面支持 BPEL 的动态演化。然而上述研究大多是通用性模型，关注的是服务组合演化机制的各个属性，如可信性、可迁移性、一致性等。在对这些性质进行验证时，均是人为地将服务流程进行了优化、调整，并未提出服务流程演化的具体实现。随着 BPEL 成为服务编制事实上的标准，越来越多的服务组合使用 BPEL 来描述。BPEL 中并发同步的使用使得 BPEL 更加高效，但也带来一些问题。当同步失败时，流程容易发生阻塞。针对同步

收稿日期：2015-09-08；修回日期：2016-01-14 基金项目：国家自然科学基金资助项目(61370051)

作者简介：孙晶(1968-)，女，辽宁沈阳人，副教授，硕士，主要研究方向为软件工程、软件测试；李东方(1988-)，男(通信作者)，硕士，主要研究方向为软件测试(253379833@qq.com)。

失败引起的 BPEL 流程失效，本文提出一种 BPEL 结构演化算法，意在调整引起 BPEL 流程失效的并发任务，保证 BPEL 顺利运行。

本文主要介绍 LOTOS 以及 BPEL 与 LOTOS 之间的映射关系，提出了并发 BPEL 流程的演化算法，并对采购订单流程 BPEL 进行演化，验证该 BPEL 流程演化算法的可靠性，将演化前后的订单流程 BPEL 翻译为 LOTOS 语言，并对其进行模型检测。

## 1 LOTOS 简介

形式化描述语言 LOTOS( language of temporal ordering specification) [11,12] 是国际标准化组织 ISO 为开放分布式系统而建立的两种形式描述技术 FDT 之一。它能提供抽象程度较高的形式描述，它使用进程的形式来描述一个系统，可以准确、无二义性和完整地形式化描述出 Web 服务。使用 LOTOS 对 BPEL 流程进行建模，并且通过 CADP [13] 工具集中的模型检测器 evaluator 对建立的模型进行验证，可以判断 BPEL 的正确性 [14]。本文使用实验室自主开发的工具实现 BPEL 到 LOTOS 的转换 [15]。

LOTOS 由两部分组成: 第一部分用来描述进程行为以及进程间协作，这部分基于进程代数的原理; 第二部分用来描述 LOTOS 的数据结构和数据类型，基于抽象数据类型理论。其行为包括接收、发送消息、顺序、选择、并行、循环等。可以看到，在 BPEL 活动与 LOTOS 行为之间存在着一一映射的关系 [14]。

表 1 BPEL 活动与 LOTOS 行为映射表

| BPEL 活动代码 | LOTOS 行为描述 |
|---|---|
| 〈receive. . . variable = “m” 〉〈 /receive〉 | G? m: Nat; |
| 〈reply. . . variable = “m” 〉〈 /reply〉 | G! m: Nat; |
| 〈invoke. . . invar = “mS” ,outvar = “mR” 〉〈 /invoke〉 | G1mS: Nat; G? mR: Nat; |
| 〈sequence. . . 〉 | . . . act1. . . ; |
| 〈. . . act1. . . 〉 | . . . act2. . . |
| 〈. . . act2. . . 〉 | |
| 〈/sequence〉 | |
| 〈flow. . . 〉 | . . act1. . ; |
| 〈. . . act1. . . 〉 | ( [cond1] →link1 ! 1; ‖ |
| 〈source linkname = “link1 ” | [not( cond1) ] →link1 ! 0; ) |
| condition = “cond1 ”/〉 | ‖ |
| 〈/act1〉 | ( link1 ? x: Bool; |
| 〈… act2 …〉 | ( [x = 1] →. . act2. . ‖ |
| 〈target linkname = “link1 ”/〉 | [x = 0] →i; ) ) |
| 〈/act2〉 | |
| 〈/flow〉 | |
| 〈if condition = “x = 0 ” 〉 | [x = 0] →act1; |
| 〈. . . act1. . . 〉 | ‖ |
| 〈else〉〈. . . act2. . . 〉〈 /else〉 | [x < > 0] →act2; |
| 〈/if〉 | |

CADP( construction and analysis of distributed processes) 是一个对异步系统进行详述、快速建模、验证、测试和性能评估的工具集。它的模型检测器 evaluator，将 LOTOS 规范和待验证的时序逻辑属性作为输入，验证系统是否满足该属性，并且生成完整的诊断( 包括正例与反例) 和更优化的内存消耗。

## 2 BPEL 流程演化算法

BPEL 是一门基于 XML 的用于自动化业务流程的形式规约语言 [10]。它通过〈flow〉活动来描述一个或多个并行执行的活动，并通过 link( 链接) 来实现并发活动间的同步。因此本文可以使用 Dom4J 这一 XML 解析工具来对 BPEL 文档进行演化。

〈process〉是 BPEL 文档的根节点，遍历其子节点，若子节点为〈flow〉节点，则扫描〈flow〉节点是否包含〈target〉或〈source〉节点; 若子节点不为〈flow〉节点，则遍历其孙子节点是否含有〈flow〉节点，若有则扫描〈〈flow〉节点是否包含〈target〉或〈source〉节点〉。找到〈target〉或〈source〉节点后按照要求将此节点移动到指定的节点下面。

根据此过程得到 BPEL 流程的演化算法如下:

算法 1 BPEL 流程的演化算法

```
输入: BPEL 文档
输出: 演化后的 BPEL 文档
Evolved BPEL( String fileName) {
获取 BPEL 文档的根节点 process
//获取 process 的子节点
nodelist = document. getChildNodes( ) ;
for ( int i = 0; i < nodelist. getLength( ) ; i ++ )
{   Node childnode = nodelist. item( i)
if childnode = " flow"
{   NodeList
     childnodeSequ = childnode. getChildNodes( )
     for ( int j = 0; j < childnodeSequ. getLength( ) ; j ++ )
     {   Node grandsonnode = childnodeSequ. item( j) ;
          if grandsonnode = " source" or " target"
              if childnodeSequ. get( i) = " sequence"
                  move( childnodeSequ. get( i) ) ;
              else
                  //移动此节点到指定节点
                  move( grandsonnode) ;
}   }
else
{   NodeList
   childnodeSequ = childnode. getChildNodes( )
     for ( int j = 0; j < childnodeSequ. getLength( ) ; j ++ )
     {   Node grandsonnode = childnodeSequ. item( j)
        if grandsonnode = " flow"
        {   NodeList grandsonSequ = childnode. getChildNodes( ) ;
            for ( int k = 0;
                k < grandsonSequ. getLength( ) ; k ++ )
            {   Node ggrandsonnode = childnodeInfo. item( j) ;
            if ggrandsonnode = " source" or " target"
              if childnodeSequ. get( i) = " sequence"
                  move( childnodeSequ. get( i) ) ;
              else
                  //移动此节点到指定节点
          move( grandsonnode) ;
     } } }
} }
```

依据上述算法，本文对 BPEL 规范 [10] 中提到的采购订单流程作相应的演化。图 1 是该 BPEL 的流程图，当收到客户发出的购买订单后，BPEL 流程同时启动价格计算、物流安排、生产调度三个任务。显然一些进程可以并发地进行，在三任务之间存在控制和数据之间的依赖关系。具体地说，运输价格用来计算最终的价格，运输日期用来安排生产计划。在全部完成这三个任务后才可以开始处理发票并把发票交给客户。

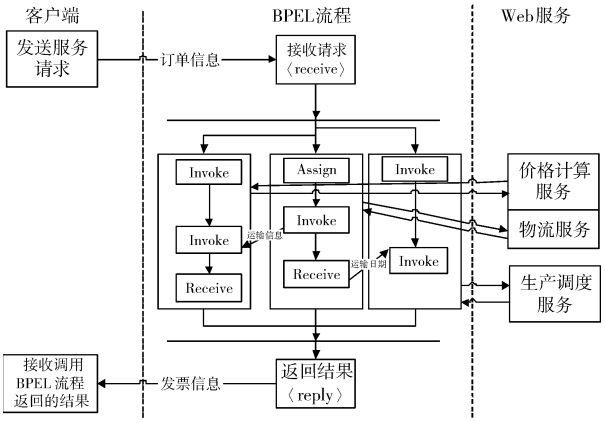



图 1　采购订单 BPEL 流程图

三个任务之间的并发使得整个流程更加高效、快捷。然而,并发任务之间存在着相互制约关系(体现为一个任务需要另一个任务的计算结果)有时又会阻碍流程的顺利执行。例如,在采购订单流程处理大批量的订单时,某一个订单中的计算价格任务无法获取到物流安排任务发送的物流方式,则整个流程就会阻塞,从而影响后续订单的顺利执行,造成经济损失。因此,在遇到上述情况时,需要对采购订单流程进行结构上的调整,如图 2 所示。对于每一个订单请求,最终返回的是最终的价格,而最终价格又取决于运输的价格。因此,需要打破物流服务和价格计算服务之间的同步关系,即当收到客户发出的购买订单后,BPEL 流程首先执行价格计算任务得到最终的价格,之后处理发票并把发票交给客户。然后再执行物流安排和生产调度任务(由于这两个服务并不影响最终获取发票,所以可以保留它们之间的同步关系),从而解除因并发的制约关系造成的流程阻塞问题。

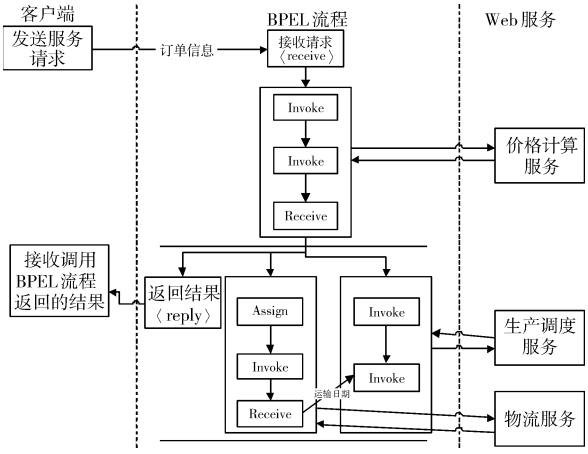



图 2　演化后采购订单 BPEL 流程

## 3　算法可靠性验证

本章采用 LOTOS 语言分别模拟实现了演化前订单流程和演化后的订单流程,并将两者进行比较。

### 3.1　演化前订单流程 LOTOS 描述

```
specification PurchaseOrderProcess [SPO,RS,SS,IPC,SSP,SI,RPS,SSS,S_T_I,S_T_S,O_T_M]: exit
  library(* 引入所需要的数据类型*)
    DataType,BOOLEAN,NATURAL
  endlib
  behavior
  Client [SPO]
    | [SPO] |
  PurchaseOrderProcess [SPO,RS,SS,IPC,SSP,SI,RPS,SSS,S_T_I,S_T_S,O_T_M]
    | [RS,SS,IPC,SSP,SI,RPS,SSS] |
  (   Shipping [RS,SS]
      |||
      Invoicing [IPC,SI,SSP]
      |||
      Scheduling [RPS,SSS]
  ) where
  process PurchaseOrderProcess
   [SPO,RS,SS,IPC,SSP,SI,RPS,SSS,S_T_I,S_T_S,O_T_M]: exit: = SPO ? PO: PM; (* 主进程接收客户端发来的数据* )
          ( ( ( RS! consSRM( extractPM( PO) ) ;
              RS ? shippingInfo: SIM;
            ( [extractSIM( shippingInfo) = = 1] →
              S_T_I ! 1 ! shippingInfo;
              SS ? shippingSchedule: SM;
              ( [extractSM( shippingSchedule ) = = 3] →
                  S_T_S ! 1 ! shippingSchedule; exit
              )
            ▮
              [NOT ( extractSIM( shippingInfo) = = 1) ] →i; exit
        ) )
        | [S_T_I] |
        (   IPC ! PO;
        ( S_T_I ? x: Nat ? shippingInfo: SIM;
        (  [x = 1] →
        ( SSP ! shippingInfo;
        SI ? Invoice: IM;
        O_T_M ! 1 ! Invoice ; exit
        ) ) ) ) )
        | [S_T_S] |
      (   RPS ! PO;
        ( S_T_S ? x: Nat ? shippingSchedule: SM;
        (  [x = 1] →
        (   SSS ! shippingSchedule ; exit
        ) ▮
          [x = 0] →i ; exit
          ) ) ) )
      | [O_T_M] |
        O_T_M ? x: Nat ? Invoice: IM;
    (  [x = 1] →
      ( SPO !    consIM( extractIM( Invoice ) ) ; exit
        (* 主进程向客户端发送数据* )
      endproc
  ...
  endspec
```

上述文件只列出了主进程的内容,并对门名和自定义数据类型名作了简化,以便显示。其中 SPO 是客户端与主进程间交互的门,RS、SS、IPC、SSP、SI、RPS、SSS 分别为 BPEL 与调用服务之间交互的门,DataType 为自定义数据类型。演化后订单流程 LOTOS 描述也是这样。

### 3.2　演化后订单流程 LOTOS 描述

```
specification specification PurchaseOrderProcessLater
  [SPO,RS,SS,M_T_R,S_T_S,IPC,SSP,SI,RPS,SSS]: exit
library
    DateType,BOOLEAN,NATURAL
endlib
behavior
... /* 主结构与 PurchaseOrderProcess 一致
where
process
```

```
PurchaseOrderProcessLater [SPO,RS,SS,M_T_R,S_T_S,IPC,SSP,
SI,RPS,SSS]: exit: =
        SPO ? PO: PM;
        IPC ! PO;
        SSP ! consSIM(4);
        SI ? Invoice: IM;
        SPO !     consIM(extractIM(Invoice));
        (* 主进程向客户端发送数据* )
          ( [extractIM(Invoice) = =2] →
          M_T_R ! 1 ! PO ! Invoice; exit
           I
           [NOT(extractIM(Invoice) = =2)] →
          M_T_R ! 0 ! PO ! Invoice; exit
        ) | [M_T_R] |
        ( M_T_R ? x: Nat ? PO: PM ? Invoice: IM;
        ( [x = 1] →
        ( ( RS ! consSRM(extractPM(PO));
      RS ? shippingInfo: SIM;
        SS ? shippingSchedule: SM;
      S_T_S ! 1 ! shippingSchedule; exit
        ) | [S_T_S] |
        ( RPS ! PO;
          S_T_S ? x: Nat ? shippingSchedule: SM;
        ( [x = 1] →
          SSS ! shippingSchedule; exit
    ) ) ) I
          [x = 0] →i; exit
        ) )
      endproc
      …
  endspec
```

### 3.3 演化前后状态转换图

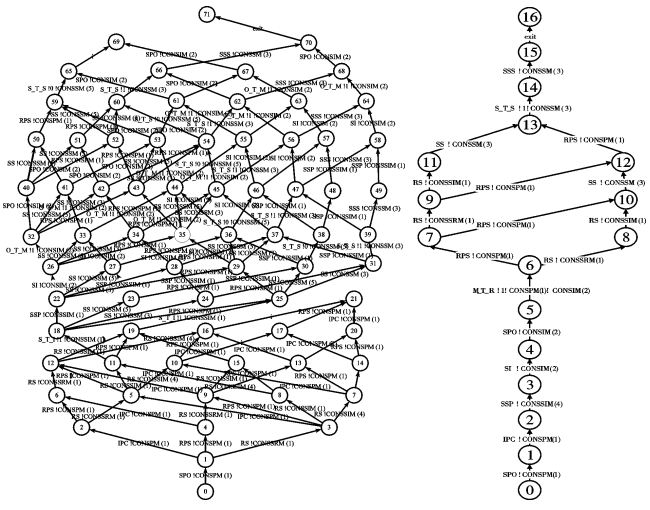
使用 CADP 工具集对演化前后订单 BPEL 转换而成的 LOTOS 文件进行检测，分别得到状态转换图，如图 3 所示。

### 3.4 模型检测

本文将 $\mu$-演算作为成功结束条件规约，对两个演化前后的订单流程进行仿真测试，通过其结果来验证演化的可靠性。

$\mu$-演算表达式如下：[true* ."SPO ! CONSPM(1)".(not "SPO ! CONSIM(2)") * ] ⟨not "SPO ! CONSIM(2)") * ."SPO ! CONSIM(2)"⟩true。它的语义是每一个客户订单申请都可以得到相应的发票信息。

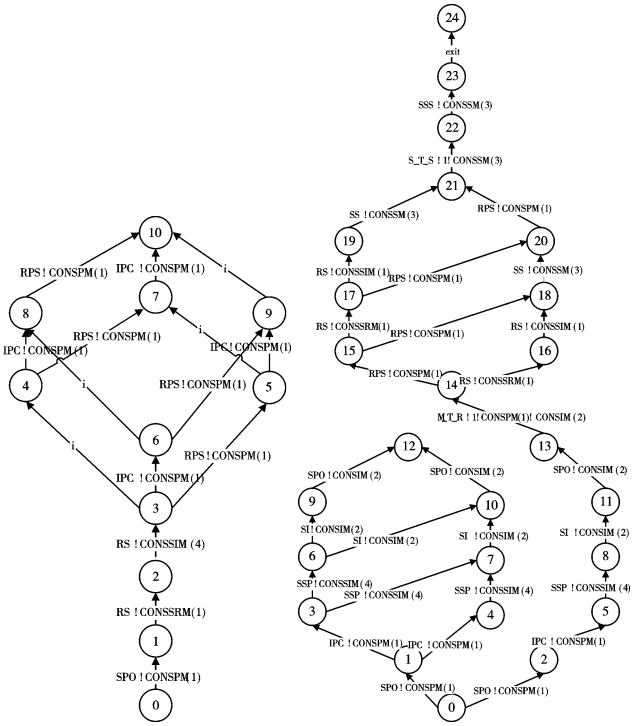
使用上述规约对两个演化前后的订单流程作模型检查后，得到相应的结果状态空间图，如图 4(a) 所示。



(a) 演化前订单流程状态转换　　(b) 演化后订单流程状态转换

图 3　演化流程状态转换图

检测结果如图 4(a) 所示，演化前订单流程有反例出现，无法得到最终的发票信息。原因是物流安排任务无法将数据同步给生产调度任务，导致生产调度任务阻塞，流程无法顺利执行。而需要的结果是每一个递交订单申请的用户都可以得到相应的发票信息。而演化后订单流程如图 4(b) 所示，用户可以后去到最终的发票信息。



(a) 演化前订单流程失败反例图　　(b) 演化后订单服务流程成功状态空间

图 4　订单服务流程状态空间

## 4 结束语

本文针对同步失败引起的 BPEL 流程失效问题，提出一种 BPEL 结构演化算法。该算法通过 BPEL 文档中 ⟨flow⟩⟨target⟩⟨source⟩ 标记定位并发同步节点，之后通过 Dom4J 工具对这些节点进行结构上的调整，达到规避失败的同步过程，删除冗余 link 结构的目的。

本算法仍存在一些缺陷，例如需要人为参与 BPEL 结构的调整过程。在对第 2 章中提到的采购订单流程作演化前，需要人为规定将价格计算服务提前到其他两个服务之前进行，而不是程序自主地对 BPEL 结构进行演化，这就降低了该算法的普适性。下一步工作中，需要对算法进行进一步的完善，争取最大限度地实现 BPEL 结构调整的自动化，即实现在 BPEL 流程阻塞时 BPEL 结构的自适应调整，并引入更多实例验证算法的可行性。

**参考文献：**


[1] Peltz C. Web services orchestration and choreography [J]. IEEE Computer, 2003, 36(10): 46-52.

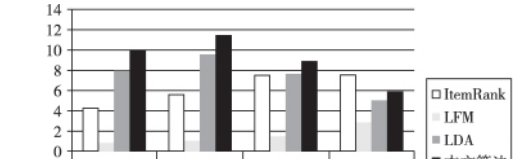
[2] Papazoglou M P. The challenges of service evolution: advanced information systems engineering [C]//Lecture Notes in Computer Science, Vol 5074. 2008: 1-15.

[3] 曾晋，孙海龙，刘旭东，等. 基于服务组合的可信软件动态演化机制 [J]. 软件学报，2010，21(2)：261-276.

[4] 王晓璇，鲍爱华，缪嘉嘉，等. 一种面向语义 Web 的组合服务演化方法研究 [J]. 计算机科学，2011，38(2)：138-143.


(下转第 2668 页)

模式挖掘的研究也有很大的意义，如文献[20]基于 Web 访问信息运用 GSP 算法对用户兴趣序列模式进行分析。




| | top-1 | top-3 | top-5 | top-10 |
|---|---|---|---|---|
| ItemRank | 4.4 | 5.76 | 7.657 | 7.617 |
| LFM | 0.867 | 1.12 | 1.543 | 2.933 |
| LDA | 7.933 | 9.67 | 7.754 | 5.083 |
| 本文算法 | 10.07 | 11.56 | 8.989 | 5.9 |

F-measure/%

图 6 不同算法的 F 值对比

## 5 结束语

本文主要从潜在狄利克雷分配和隐马尔可夫模型的理论基础出发，针对高时效业务推荐场景中用户的潜在兴趣在提高推荐性能上的作用，以及时间迁移性对用户兴趣稳定性的影响展开研究，提出基于用户潜在时效偏好的推荐方法。该方法在有效挖掘用户潜在兴趣的基础上，考虑了高时效业务中，用户兴趣的变动周期短，长期处于不稳定状态的客观事实，并通过实际的新闻数据验证了其方法的可行性。当然，出于用户兴趣的时效性考虑，本文方法在应用场景上存在一定局限性，对于一些用户兴趣较稳定的业务，该方法可能并不能得到很好的推荐性能。后期工作中，将对如何有效结合用户的长期兴趣和时效兴趣进行研究，针对推荐方法在应用业务的可扩展性上做进一步工作。

参考文献：


[1] Vozalis E, Margaritis K G. Analysis of recommender systems algorithms [C] //Proc of the 6th Hellenic European Conference on Computer Mathematics & Its Applications. 2003: 732-745.

[2] Tan Hengsong, Ye Hongwu. A collaborative filtering recommendation algorithm based on item classification [C] //Proc of Pacific-Asia Conference on Circuits, Communications and Systems. 2009: 694-697.

[3] Billsus D, Pazzani M J. User modeling for adaptive news access [J]. User Modeling and User-Adapted Interaction, 2000, 10(2-3): 147-180.

[4] Gori M, Pucci A, Roma V. ItemRank: a random-walk based scoring algorithm for recommender engines [C] //Proc of IJCAI. 2007: 2766-2771.

[5] Dumais S T, Furnas G W, Landauer T K, *et al*. Using latent semantic analysis to improve access to textual information [C] //Proc of the SIGCHI Conference on Human Factors in Computing Systems. 1988: 281-285.

[6] Koren Y, Bell R, Volinsky C. Matrix factorization techniques for recommender systems [J]. Computer, 2009(8): 30-37.

[7] Hofmann T. Probabilistic latent semantic indexing [C] // Proc of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. 1999: 50-57.

[8] Andrzejewski D, Buttler D. Latent topic feedback for information retrieval [C] //Proc of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2011: 600-608.

[9] Gao H, Tang J, Hu X, *et al*. Exploring temporal effects for location recommendation on location-based social networks [C] //RecSys. 2013: 93-100.

[10] Zhang Wancai, Sun Hailong, Liu Xudong, *et al*. Temporal QoS-aware Web service recommendation via non-negative tensor factorization [C] //Proc of the 23rd International Conference on World Wide Web. New York: ACM Press, 2014: 585-596.

[11] Rubin T N, Chambers A, Smyth P, *et al*. Statistical topic models for multi-label document classification [J]. Machine Learning, 2012, 88(1-2): 157-208.

[12] Wang Xuerui, McCallum A, Wei Xing. Topical n-grams: phrase and topic discovery, with an application to information retrieval [C] // Proc of the 7th IEEE International Conference on Data Mining. 2007: 697-702.

[13] Blei D M, Jordan M I. Variational inference for Dirichlet process mixtures [J]. Bayesian Analysis, 2006, 1(1): 121-143.

[14] Moon T K. The expectation-maximization algorithm [J]. Signal Processing Magazine, 1996, 13(6): 47-60.

[15] Griffiths T. Gibbs sampling in the generative model of latent Dirichlet allocation [R]. Stanford: Stanford University, 2002.

[16] Wei Xing, Croft W B. LDA-based document models for Ad hoc retrieval [C] //Proc of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. 2006: 178-185.

[17] Chen Wenyen, Chu J C, Luan Junyi, *et al*. Collaborative filtering for orkut communities: discovery of user latent behavior [C] //Proc of the 18th International Conference on World Wide Web. 2009: 681-690.

[18] Steyvers M, Griffiths T. Probabilistic topic models [J]. Handbook of Latent Semantic Analysis, 2007, 427(7): 424-440.

[19] Wallach H M. Structured topic models for language [D]. Cambridge: University of Cambridge, 2008.

[20] 马力，谭薇，李培. 基于 Web 访问信息的用户兴趣迁移模式的研究 [J]. 计算机科学，2011，38(5)：175-177.


(上接第 2640 页)


[5] 赵会群，孙晶，魏莹，等. 服务体系结构的动态演化方法研究 [J]. 计算机科学，2011，38(8)：116-123.

[6] 宋巍，马晓星，吕建. Web 服务组合动态演化的实例可迁移性 [J]. 计算机学报，2009，32(9)：1816-1831.

[7] 李冰，高岩，王斌，等. 基于变化的服务演化一致性判定 [J]. 计算机工程与科学，2014，36(12)：2258-2266.

[8] 张忠宇，黄光奇. 信息系统服务演化透明性研究 [J]. 计算机工程与设计，2011，32(7)：2340-2343.

[9] 黄波，应时，张韬，等. 一种基于反射的 BPEL 业务过程动态演化方法 [J]. 计算机应用研究，2007，24(5)：31-34.

[10] Alves A, Arkin A, Askary S, *et al*. Web services business process execution language version 2.0 [EB/OL]. (2007-04-11) [2012-08-19]. http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html.

[11] Logrippo L, Faci M, Haj-hussein M. An introduction to LOTOS: learning by examples [D]. Ottawa: University of Ottawa, 1999.

[12] 李昌益. 形式化描述技术 LOTOS 及其在实时系统的扩展和应用 [D]. 广州：广东工业大学，2004.

[13] Salaün G, Ferrara A, Chirichiello A. Negotiation among Web services using LOTOS/CADP [C] //Lecture Notes in Computer Science, Vol 3250. 2004: 198-212.

[14] Ferrara A. Web services: a process algebra approach [C] //Proc of ICSOC. New York: ACM Press, 2004: 242-251.

[15] 赵会群，何霞. 基于翻译模式的 BPEL 到 LOTOS 映射方法研究 [J]. 计算机应用研究，2013，30(6)：1751-1755.