BPEL中基于异步模式的人工任务执行系统研究

柳 玲 余 港 文俊浩
(重庆大学软件学院 重庆 400030)

摘 要 BPEL(Business Process Execution Language)是一种编写基于Web服务的自动化业务流程的语言,但不支持用户交互,用户交互可以通过人工任务实现。为此,提出一种在BPEL中支持人工任务执行的系统架构。在引擎之外定义一个人工任务管理器来维护人工任务,流程与人工任务管理器采用异步消息模式进行交互,以便更适应人工任务执行时间不确定的特点。使用BPEL提供的消息相关集关联异步交互过程中调用人工任务管理器的活动与接收人工任务管理器执行结果的活动。

关键词 BPEL,人工任务,异步,业务流程
中图法分类号 TP311 **文献标识码** A

Research of Human Task Performing System Based on Asynchronous Pattern in BPEL

LIU Ling YU Gang WEN Jun-hao
(College of Software Engineering, Chongqing University, Chongqing 400030, China)

Abstract BPEL(Business Process Execution Language) is a kind of language which can be used to write automated business process, but it doesn't support user interactions. User interactions can be achieved by human tasks. Therefore, an architecture which supports the performing of the human task in BPEL was proposed. There is a human task manager outside of the engine to maintain human activities. BPEL processes interact with the human task manager with asynchronous pattern, which adapts to the feature of time uncertainty of the human task. The message correlation set in BPEL is used to correlate the invoking activity and the waiting activity during the period of asynchronous invocation.

Keywords BPEL, Human task, Asynchronization, Business process

1 引言

BPEL(Business Process Execution Language)是一种用于Web服务组合、编制和协作的语言,目前已经被广泛应用于Web服务相关的项目中,成为了Web服务组合事实上的标准。BPEL虽然提供了丰富的功能描述业务流程,但缺乏对用户交互的支持。在实际情况中,业务流程往往不能完全由机器自动运行完成,在流程运行过程中经常需要用户参与,而BPEL规范中并未涉及对用户参与流程的描述。用户与流程的交互可通过人工活动来完成。人工活动是流程中一种特殊的活动,它不是由机器自动完成,而是由相关人员来完成的。在一般的工作流系统中,流程运行到人工活动的步骤时,系统会创建一项工作任务,由用户去执行这项任务,用户通过工作任务实现与流程的交互。因此,人工活动也可以称为人工任务。

2007年6月,IBM,BEA等6家厂商共同发布了BPEL4People1.0[1]规范和WS-HumanTask1.0[2]规范。BPEL4People规范中定义了一种新的基本活动来表示用户与流程的交互,新的活动使用人工任务来实现。WS-HumanTask规范负责定义人工任务,包括人工任务的生成、状态管理、与流程的交互等。BPEL4People和WS-HumanTask规范填补了BPEL关于人工任务的空白。但是,已有的支持标准BPEL的引擎并不支持这两个规范。要支持新规范就需要重新设计引擎或对现有引擎进行较多的扩展。

一些文献中提出了扩展一个外部模块的办法,即在执行到人工活动的步骤时,调用外部模块并传递人工活动的相关数据,由外部模块来实现与用户的交互。文献[3]提出了一种名为VieBOP的架构,实现了BPEL4People规范。VieBOP作为一个BPEL引擎之外的系统管理人工活动。BPEL4People中定义人工活动的元素被转换为对VieBOP系统的异步调用,这样就可以使用已有的支持标准BPEL的引擎支持人工活动。但是,由于BPEL4People和WS-HumanTask规范对BPEL进行了比较复杂的扩展,仅仅通过将人工活动转换成标准的BPEL元素,很难完全实现这两个规范。文献[3]也指出,要求流程中定义的人工活动比较简单。

文献[4]提出了一种使BPEL流程支持人工交互的系统架构,设计了一个独立于引擎之外的人工活动管理器,充当用户与流程交互的桥梁,维护人工活动。文献[4]还提出了一种通过扩展BPEL增加新元素定义人工活动的语言,新增的元素被嵌入到一个合法的〈invoke〉活动之中,执行时引擎可以

到稿日期:2011-01-22 返修日期:2011-05-12 本文受中央高校基本科研业务费(CDJZR10090001)资助。
柳 玲(1970-),女,博士,副教授,主要研究方向为服务计算、面向服务的软件工程;余 港(1984-),男,硕士生,主要研究方向为服务计算、面向服务的软件工程;文俊浩(1969-),男,教授,博士生导师,CCF高级会员,主要研究方向为服务计算、面向服务的软件工程。

忽略这些内容。文献[4]还使用 XSLT 对人工活动进行转换，根据人工活动的定义，在人工活动前加入封装输入数据的代码，在人工活动之后加入提取输出数据的代码，以实现流程与人工活动管理器的交互。但是，文献[4]提出的架构中对人工活动管理器的调用采用的是同步调用的方式，而人工交互的时间具有很大的不确定性，有可能几个小时就完成，也有可能会长达几天甚至几个月，因此对人工活动管理器的调用采用异步方式应更为合理。

本文通过研究 BPEL 的特性，借鉴文献[4]的思路，提出基于异步模式的人工任务执行系统架构，设计一个独立于引擎之外的人工任务管理器充当用户与流程交互的桥梁。引擎与人工任务管理器之间采用异步消息模式进行交互，更能适应人工任务执行时间不确定的特点。同时，本文讨论了在异步交互过程中如何使用 BPEL 提供的消息相关集关联〈invoke〉和〈receive〉活动。

2 BPEL 介绍

一个 BPEL 流程由若干个活动组成。BPEL 中的活动可分成两种：一种是基本活动，描述流程中最基本的执行步骤，是不可再分的执行单元，用来完成一些最基本的功能，主要包括：

〈assign〉：赋值活动，用于在变量间传递数据；

〈invoke〉：用于调用由伙伴链接提供的 Web 服务；

〈receive〉：用于等待到达流程的消息；

〈reply〉：用于对客户端的调用进行回复。

另一种是由基本活动组成的复合活动，是对流程控制逻辑的描述，在流程执行过程中控制基本活动的执行顺序。主要包括：

〈sequence〉：顺序活动，嵌套在它里面的基本活动按先后次序顺序执行；

〈if〉/〈switch〉：分支活动，实现流程根据不同的条件执行不同的路径，类似于传统编程语言中的选择结构语句；

〈while〉/〈repeatUntil〉：循环活动，实现反复执行其内部基本活动的功能，类似于传统编程语言中的循环结构语句；

〈pick〉：事件选择处理活动，用于等待一组相互排斥事件中的一个发生，然后执行与发生的事件相关联的活动；

〈flow〉：其内部的基本活动并发执行。

BPEL 流程既可以是 Web 服务的提供者，响应来自客户端的一个调用请求；也可以作为服务的请求者，调用外界的 Web 服务。BPEL 把所有与流程进行交互的其他 Web 服务称为伙伴(Partner)，通过伙伴链接(Partner Link)与外界交互。伙伴链接通过引用伙伴链接类型(Patner Link Type)来定义流程与外界的服务之间的通信接口，伙伴链接通过定义一组角色来声明两个服务之间的关系。其中每个角色指明一组端口类型(Port Type)，亦即明确该角色所提供的服务接口。伙伴链接类型被定义在 WSDL 文档中，由在 BPEL 流程中定义的伙伴链接引用。

在 BPEL 流程运行过程中，经常需要将流程中的一些数据保存下来，比如调用 Web 服务返回的结果、接收客户端输入的数据等。BPEL 提供了变量(Variable)来记录和维护这些数据。变量的值可以通过〈assign〉活动来相互传递，也可以将变量指定为〈invoke〉，〈receive〉等活动的输入、输出变量，从而保存流程与伙伴之间交互的数据。

3 基于异步模式的人工任务执行系统

3.1 系统架构

基于异步模式的人工任务执行系统的架构如图 1 所示。BPEL 引擎负责流程定义文件的部署、流程实例的创建和运行、调用 Web 服务等工作。人工任务管理器充当用户与流程交互的桥梁。数据库负责持久存储任务的信息。当流程运行到需要用户参与的步骤时，流程调用人工任务管理器提供的创建任务的 Web 服务，将执行任务需要的数据、任务的授权规则等信息传递给人工任务管理器，由人工任务管理器根据这些信息创建人工任务，并将任务授权给合适的用户。用户通过客户端调用人工任务管理器提供的 Web 服务认领、查看或执行人工任务。任务执行完毕以后，由人工任务管理器调用流程提供的接收任务执行结果的回调服务，将任务执行结果传递给流程，这样就完成了用户与流程的一次交互。

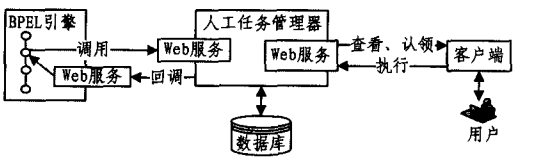


图 1 系统架构

3.2 人工任务异步调用方案设计

BPEL 对 Web 服务的调用有同步和异步两种方式。同步方式是一种请求/响应方式，在同步方式下，流程向 Web 服务发出调用请求，随后阻塞等待 Web 服务传递执行结果；异步方式是一种请求/回调方式，在异步方式下，流程向 Web 服务发出调用请求，得到一个简单的响应，流程得到响应后不必阻塞等待 Web 服务的执行结果，而是可以继续执行其他的操作，或者流程停止运行。Web 服务在完成操作以后，会调用流程提供的回调服务传递执行结果。同步方式比较适合调用执行时间短的 Web 服务，而异步方式更适合调用执行时间长的 Web 服务。

人工任务的异步调用方案设计如图 2 所示。人工任务被设计成包含一个异步调用的〈sequence〉活动，即顺序执行的活动。第一个〈assign〉活动负责将执行任务需要的数据封装为一个 XML 字符串格式的输入参数，〈invoke〉活动负责调用人工任务管理器创建任务，〈pick〉活动用于等待一组相互排斥事件中的一个事件的发生，然后执行与发生的事件相关联的活动，比如等待收到一个合适的消息或超时警报触发。〈pick〉活动里面包含了一个〈onMessage〉活动和一个〈onAlarm〉活动。〈onMessage〉活动的作用与〈receive〉活动类似，用于等待一个特定端口上的消息，这里即是等待人工任务的执行结果，其内部的〈assign〉活动用于将执行结果中的输出数据提取出来赋给流程中的变量。〈onAlarm〉活动定义了一个时间报警器，当到达了指定的时间点或经过了指定的时间段后会执行它里面定义的活动。整个〈pick〉活动可以将流程等待任务执行结果的时间控制在一定范围以内，超过这段时间，流程将抛出异常，由引擎根据流程定义的异常处理逻辑或者由外部的异常处理模块来处理。

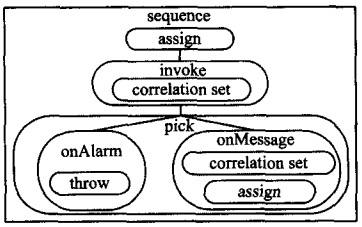


图 2　人工任务异步调用方案

3.3　异步消息关联

在同步调用过程中，流程对服务的调用和服务对流程的响应属于同一次会话，请求与响应存在一一对应关系。然而，在异步调用过程中并不存在这种明显的对应关系。BPEL 引擎在接收到回调消息后，不知道该消息对应于哪一次调用请求，也就不知道将该消息分发给哪一个流程实例(对于一个流程定义，通常有多个流程实例同时运行)，因此需要在调用与回调之间建立一一对应的会话关系，使得引擎知道将消息分发给哪一个流程实例。BPEL 采用与 Web 服务之间所交换的消息中的一个或多个具有唯一性的数据字段集合来标识会话关系，这些数据字段的集合被称作相关集(CorrelationSet)。人工任务管理器创建任务后，会生成一个唯一的任务 ID，这个任务 ID 可以在相关集中使用。使用相关集关联调用与回调可以分为如下 3 步。

① 在流程定义文件对应的 WSDL 文件中定义属性和属性别名。定义属性即创建一个流程全局范围内唯一的名称。通过属性别名可将属性映射到消息的某个字段上。属性和属性别名的定义如下面的代码所示：

```
〈prop:property name="taskID" type="xsd:int"/〉
〈prop:propertyAlias propertyName="tns:taskID"
      messageType="task:createTaskResponse"
      part="parameters"〉
      〈prop:query 〉
            taskID
      〈/prop:query〉
〈/prop:propertyAlias〉
〈prop:propertyAlias propertyName="tns:taskID"
      messageType="tns:TaskResultMessage"
      part="parameters"〉
      〈prop:query〉
            taskID
      〈/prop:query〉
〈/prop:propertyAlias〉
```

上述代码创建了一个名为 taskID 的属性以及两个属性别名。第一个别名把属性 taskID 映射到调用人工任务管理器创建人工任务的响应消息(也就是〈invoke〉活动的响应消息)上，第二个别名把属性 taskID 映射到人工任务执行结果的消息(也就是〈onMessage〉活动所接收的消息)上。

② 在流程定义文件中创建消息相关集，相关集引用 WSDL 文件中定义的属性：

```
〈correlationSets〉
    〈correlationSet name="taskIDCorr"
        properties="wns:taskID"/〉
〈/correlationSets〉
```

③ 分别在〈invoke〉活动和〈onMessage〉活动中引用同一个相关集。

```
〈invoke name="…" …〉
    …
    〈correlations〉
        〈correlation set="taskIDCorr" initiate="yes"
            pattern="response"/〉
    〈/correlations〉
〈/invoke〉
…
〈onMessage name="…" …〉
    〈correlation set="taskIDCorr"/〉
〈onMessage〉
```

〈invoke〉活动调用人工任务管理器创建任务以后，得到任务 ID，引擎用这个 ID 初始化相关集(initiate="yes")。在人工任务管理器回调流程提供的 Web 服务传递执行结果时，引擎会根据回调消息中的任务 ID 查找对应的消息相关集，从而找到相关集对应的〈onMessage〉活动。这样，任务的执行结果就可以分发给正确的流程实例的〈onMessage〉活动。

3.4　人工任务管理器

人工任务管理器充当用户与流程交互的桥梁，它一方面要提供流程创建人工任务的功能，另一方面要提供用户认领、查看、执行任务的功能。人工任务管理器的功能被封装为 Web 服务，其既可以被 BPEL 流程方便地调用，也使得客户端可以灵活地采用不同的技术实现。人工任务管理器需要提供的 Web 服务有：

创建任务服务：创建一个任务并授权给合适的用户；

认领任务服务：用户认领某个授权给他的任务；

任务列表服务：用户获取它可认领/可执行/已执行的任务的列表；

查看任务服务：用户查看某个任务以及任务的数据；

执行任务服务：用户执行某个任务。

结束语　本文针对 BPEL 不支持人工任务这一问题，在借鉴现有研究成果的基础上提出基于异步模式的人工任务执行系统架构，设计了一个独立于引擎之外的人工任务管理器维护人工任务，充当用户与流程交互的桥梁。人工任务被设计成一个含有对人工任务管理器的异步调用的〈sequence〉活动。流程与人工任务管理器之间采用异步消息模式进行交互，更能适应人工任务执行时间不确定的特点。本文还研究了异步调用过程中调用与回调的消息关联，使用 BPEL 提供的消息相关集关联〈invoke〉活动和〈onMessage〉活动。

参 考 文 献

[1] Web Services Human Task(WS-HumanTask). Version 1.0[S]. June 2007

[2] WS-BPEL Extension for People(BPEL4People). Version 1.0 [S]. June 2007

[3] Holmes T, Vasko M, Dustdar S. VieBOP: Extending BPEL Engines with BPEL4People[C]// 16th Euromicro Conference on Parallel, Distributed and Network-based Processing. 2008: 547-555

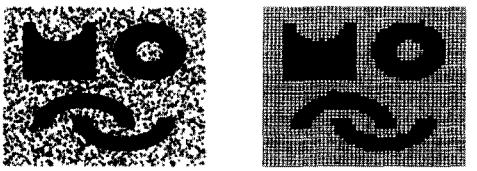
[4] Thomas J, Paci F, Bertino E, et al. User Tasks and Access Control over Web Services[C]// 2007 IEEE International Conference on Web Services(ICWS 2007). 2007: 60-69

[5] 雷金繁. 基于服务的工作流系统架构的研究[D]. 广州：中山大学，2008

(下转第 181 页)

算法参数设置为：$p=50$，$\lambda=0.95$，$Clu_{max}=50$，$Clu_{min}=5$，密度阈值比 $D_m/D_l\in[4,6]$，数据流速 $v=500$。

首先为检测算法对任意形状数据的处理能力，对仿真数据进行处理。实验中对每条数据按其到达时间赋予相应的时间戳，设置 $\lambda=0.998$，结果如图 2(b)所示。实验结果显示算法能够完好识别出数据中 4 个非凸形状的类。



(a) 原始数据　　(b) 聚类结果

图 2　任意形状聚类

对真实数据 KDDCUP99 不同时刻的聚类结果如图 3 所示，聚类结果与 D-Stream 算法比较，采用平均聚类纯度评价聚类质量，聚类纯度即类中主要类标号数据在类中所占的百分比，聚类纯度越高，表示数据属于同一类的比例越大，聚类质量越好。从结果中看出，在不同时刻的聚类结果中 NDD-Stream 取得了较高的类纯度，这是由于 NDD-Stream 在输出聚类结果时采用不均匀的网格划分提高聚类精度，因此具有较好的聚类质量。

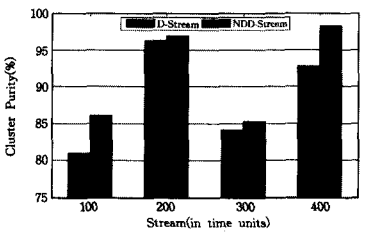


图 3　聚类纯度比较

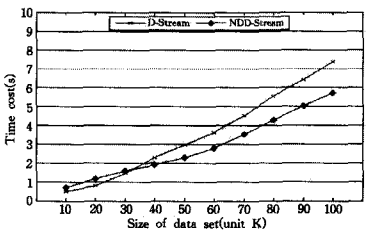


图 4　聚类时间比较

图 4 为 NDD-Stream 在线处理算法与 D-Stream 算法的处理速度比较。结果显示随着数据量的增长，NDD-Stream 所用时间的增长速率低于 D-Stream，因为 NDD-Stream 在运行过程中动态调整 gap 值，使其适应当前数据空间的密度分布情况，避免了过于频繁的格簇调整，提高了算法效率。

结束语　NDD-Stream 算法实现了密度阈值参数的自适应设置，并通过对簇边缘采用不均匀的网格划分提高了簇的精度。虽然在线处理过程中需要消耗部分时间来保存数据，但由于算法动态调整了密度阈值，在一定程度上减少了不必要的簇调整工作，节约了运算时间。实验证明，算法能够取得较高的聚类质量以及较快的运行速度。

参 考 文 献

[1] Guha S, Mishra N, motwani R, et al. Clustering Data Streams [C]// Proc. of the 41st Annual Symposium on Foundations of Computer Science. 2000:359-366

[2] Chen Y, Tu L. Density-Based Clustering for Real-Time Stream Data[C]// Proc, of the International Conference on Knowledge Discovery and DataMining. August 2007:12-15

[3] Aggarwal C, Han J, Wang J, et al. A Framework for Clustering Evolving Data Streams[C]// Proc. of the 29th VLDB Conference. 2003:81-92

[4] 朱蔚恒，印鉴，谢益煌. 基于数据流的任意形状聚类算法[J]. 软件学报，2006，17(3)：379-386

[5] 刘青宝，戴超凡，邓苏，等. 基于网格的数据流聚类算法[J]. 计算机科学，2007，34(3)：159-161

[6] 郑盈盈，倪志伟，吴姗，等. 基于移动网格和密度的数据流聚类算法[J]. 计算机工程与应用，2009，45(8)：129-131

[7] Hinneburg A, Keim D A. An Efficient Approach to Clustering in Large Multimedia Databases with Noise[C]// Proc of the International Conference on Knowledge Discovery and DataMining. 1998:58-65

[8] 单世明. 基于网格和密度的数据流聚类方法研究[D]. 大连：大连理工大学，2006

[9] 何勇，刘青宝. 基于动态网格的数据流聚类分析[J]. 计算机应用研究，2008，25(11)：3281-3284

[10] Sun Y, Lu Y. A Scalable Grid-based Clustering Algorithm for Very Large Spatial Databases[C]// Proc. of the International Conference on Computational Intelligence and Security. 2006: 763-768

(上接第 146 页)

[6] Jia Chun-xin, Lu Chao-jun. Extending BPEL to Model Business Processes Involving Human Activities[C]// 2009 International Conference on Frontier of Computer Science and Technology. 2009:524-529

[7] OASIS. Web Services Business Process Execution Language. Version 2.0[S]. April 2007

[8] Paci E, Ferrini R, Bertino E. Identity Attribute-based Role Provisioning for Human WS-BPEL processes[C]// 2009 IEEE International Conference on Web Services(ICWS2009). 2009:535-542

[9] Juric M. Buisiness Process Execution Language for Web Services [M]. Second Edition. Birmingham: Packt Publishing Ltd, 2006: 145-150

[10] Wang Xin, Zhang Yan-chun, Shi Hao. Access Control for Human Tasks in Service Oriented Architecture[C]// 2008 IEEE International Conference on e-Business Engineering (ICEBE'08). 2008:455-460

[11] 倪晚成，刘连臣，吴澄. Web 服务组合方法综述[J]. 计算机工程，2008，34(4)：79-81