基于BPEL和WS-TX的Web服务组合事务协调框架研究

蔡正平 黄志球 王 进 王珊珊 徐丙凤
(南京航空航天大学计算机科学与技术学院 南京 210016)

摘 要 针对当前Web服务事务协调框架中缺乏对服务组合流程定义的支持而需要在协调过程中手动定义业务流程的问题,提出了一种支持流程和事务语义自动抽取的服务组合事务协调框架。该框架从扩展了事务语义的业务流程执行语言(Business Process Execution Language,BPEL)中抽取协调信息,并结合Web服务事务规范(Web Services Transaction,WS-TX)所定义的协调器实现了服务组合的全局事务协调。通过将WS-TX在事务协调和BPEL在业务流程设计方面的优势相结合,该框架将业务逻辑与事务处理逻辑有效地分离。最后通过实例分析说明了该框架的可行性。

关键词 Web服务,事务,协调框架
中图法分类号 TP301 **文献标识码** A

Research of Web Services Composition Transaction Coordination Framework Based on BPEL and WS-TX

CAI Zheng-ping HUANG Zhi-qiu WANG Jin WANG Shan-shan XU Bing-feng
(Department of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China)

Abstract As composite service process must be manually defined when using Web service transaction coordination framework because of lack of process definition support, a transaction coordination framework which supports process and transaction semantic information automatic extracting was proposed. It coordinates the composite service by using transaction coordinator defined by Web services transaction (WS-TX) and extracting coordination information from extended business process execution language (BPEL). The proposed framework combines the advantages of both WS-TX in the transaction coordination and BPEL in business process design to separate transaction logic from business logic effectively. Finally the effectiveness of the framework was validated through a case study.

Keywords Web service, Transaction, Coordination framework

1 引言

Web服务是一种崭新的分布式计算模型,并已成为Internet上数据和信息集成的有效机制[1]。Web服务的跨平台、松耦合、易于互操作等特点,使之为不同实现标准和不同通信机制下软件系统之间的交互与集成提供了有效的支持。单一Web服务往往功能有限,无法满足B2B环境下的复杂业务需求,因此将多个现有的、可能是异构的单一服务组合在一起,来为用户提供功能复杂的Web服务成为必然。

Web服务组合协作中,保持服务组合后应用一致性是一个关键的问题。对于这类问题,可以借助事务机制来解决。事务"非全则无"的语义可保证多个服务交互与协作的可靠一致,但是由于Web服务自身松散耦合、运行时间长等特点,传统的对资源进行严格锁定的事务处理机制不再适用。目前主要采用异常处理这一柔性方式结合适当放松的ACID属性实现Web服务事务处理。

目前工业界已有一系列针对Web服务事务的标准与协议。如Web服务组合应用框架(Web Services Composite Application Framework,WS-CAF)[3];Web服务事务处理规范(Web Services Transactions,WS-TX)等。2009年OASIS新版本标准WS-TX 1.2中包括WS-Coordination(WS-C)[4]、WS-AtomicTransaction(WS-AT)[5]和WS-BusinessActivity(WS-BA)[6]3个协议,其描述了一个为协调分布式应用程序行为提供协议支持的可扩展框架,这种协调协议可用于支持多个业务流程实例的运行,它们在分布式事务的输出上有一致性的要求。WS-TX标准被建议用来实现对业务的参与者提供注册或撤销等分布式协调的通知。它描述了相应的同意协议(Agreement Protocol)支持由多个不同流程的活动所组成的分布式原子事务。其中WS-C描述了一个可扩展的框架,其用于提供协调分布式应用程序的操作的协议,支持多种

到稿日期:2011-08-07 返修日期:2011-10-17 本文受国家高技术研究发展计划(863计划)项目(2009AA010307),江苏省研究生培养创新工程基金(CXZZ11_0218)资助。

蔡正平(1987-),男,硕士生,主要研究方向为事务处理、面向服务计算,E-mail:nuaa_czp@nuaa.edu.cn;黄志球(1965-),男,博士,教授,主要研究方向为软件工程、形式化方法、面向服务计算、知识工程;王 进(1980-),男,博士生,主要研究方向为形式化方法、面向服务计算;王珊珊(1963-),女,硕士,副教授,主要研究方向为软件工程、形式化方法、面向服务计算;徐丙凤(1986-),女,博士生,CCF学生会员,主要研究方向为软件工程、嵌入式软件建模与分析。

协调类型的协议；WS-AT 协议主要支持类似原来传统事务的短周期原子事务，而 WS-BA 则专门为长周期事务而设计。但目前的这些协调框架中都缺乏对服务组合流程语义的刻画机制，用户在考虑事务协调时，需要手动编写复杂代码界定业务流程，这样既不利于复用修改，也难以与业务模型间进行有效地对应。

另一方面，针对 Web 服务组合流程的描述，为业界广泛认可和接受的 OASIS 标准业务流程执行语言（Business Process Execution Language，BPEL）[2] 是用来描写业务流程的编程语言的，其中业务流程的每一个操作步骤则是由 Web 服务来实现。基于工作流的 BPEL 虽然能够灵活定义业务流程，支持以作用域（scope）为基本单元的补偿机制，但是在事务方面其缺乏灵活性，如 BPEL 并不支持原子事务和跨流程事务，也不支持重试等向前恢复策略。考虑到 BPEL 和 WS-TX 的优势与不足，将两者结合开展研究已成为目前 Web 服务组合和事务协调方向的研究热点。

文献[11]仅给出了基于 SOA 的 WS-C 和 WS-BA 的实现，并没有涉及服务组合环境下的使用。开源组织 Apache 的 Kandula[8] 的项目正在对 WS-TX 进行实现，但是它并没有结合 BPEL，因而缺乏对不同组合服务的灵活支持。文献[12]将 WS-TX 的事务管理集成到 BPEL 框架，由于业务逻辑和事务逻辑的混合，使得要重复定义事务协调行为。文献[9]基于 WS-TX 提出了一个事务中间件框架 OpenWS-Transaction，并通过将 BPEL 和 WS-TX 结合实现了原型系统，但它没有给出参与者间事务性依赖的描述方法，需要针对特定的服务组合需求在协调器中给定提交方法。文献[10]基于 WS-TX 提出了一种由 BPEL 自动生成 Web 服务多参与者协调事务模型的方法，并通过 Pi-演算建模验证了方法的正确性。

基于上述问题，本文通过对 BPEL 进行事务描述能力扩展，并将其与 WS-TX 规范相结合，给出一种可靠的 Web 服务组合协调框架；第 2 节对 BPEL 和 WS-TX 的规范进行了介绍，给出了协调框架的体系结构并对其协调流程进行了说明；第 3 节提出了 BPEL 事务语义的扩展机制，以及协调框架的自动抽取方法和完整的协调执行过程；第 4 节通过一个实例分析说明协调框架的可行性；最后对全文进行总结。

2 协调框架体系结构

2.1 BPEL 和 WS-TX

Business Process Execution Language（BPEL）是 OASIS 标准组织定义的一种标准执行语言，使用该语言可以编排由服务构成的业务流程，它定义了如何表示业务流程中的活动，以及流控制逻辑、数据、消息相关性和异常处理等。

BPEL 支持顺序（sequence）、分支（if）、并行（flow）、选择（pick）、循环（while）等丰富的结构化活动定义，并且已有图形化开发插件的支持，能够满足简单、快速、灵活的业务流程定义的需求。

BPEL 长期运行的事务活动被集中在作用域（scope）上。其作用域是允许活动分组的结构化活动，并为相应的一组活动定义公共的执行上下文。

通过作用域机制，在错误情况下，BPEL 允许定义可一起被撤销的活动集。撤销已完成活动的所需操作被称为补偿处理程序。作用域的故障处理程序可以使用补偿处理程序来撤销这个作用域中进行的操作。在作用域中进行的活动要么全部完成，要么全部被补偿。而实际组合服务中，可能允许部分参与者完成、部分参与者被补偿，并且很多业务操作是可以重试的，因而 BPEL 缺乏灵活的事务协调能力。

Web Services Transactions（WS-TX）协议作为 OASIS 标准，定义 Web 服务域间的事务互操作性机制，由 3 个具体部分组成，分别为 Web 服务协调（WS-C）、Web 服务原子事务（WS-AT）以及 Web 服务业务活动（WS-BA）。

WS-C 描述了一个可扩展框架，支持以下服务。

1）激活服务（Activation service）：能够使应用程序创建协调实例或者事务上下文。一个协调者可以提供激活服务，也可以不提供。

2）注册服务（Registration service）：允许一个或多个 Web 服务在此注册协调协议。其用来协调协议选择和注册参与者，协调者必须提供注册服务。

3）协调协议服务（Coordination protocol service）：一组针对于每个支持的协调类型的协调协议集合。根据协调协议创建的服务，用于协调消息的传递，以 Web 服务的形式存在，服务中定义了协调消息交换的具体格式。

WS-AT 和 WS-BA 将与 WS-C 描述的协调框架一起使用，前者定义了原子事务类型，包含了 3 个具体的协定协调协议：Completion、Volatile Two-Phase Commit 和 Durable Two-Phase Commit。后者定义了支持业务事务的业务活动协调类型，包含 BusinessAgreementWithParticipantCompletion（参与者主动完成）和 BusinessAgreementWithCoordinatorCompletion（协调者主动完成）两个协调协议，并支持原子输出（AtomicOutcome）和混合输出（MixedOutcome）。但 WS-TX 规范并未描述业务活动的具体协调和管理方式，同时缺乏对业务流程的刻画，业务活动流程的定义只能采用硬编码的方式实现。

基于上述的讨论，结合 BPEL 和 WS-TX 的服务组合定义成为一个很好的选择。

2.2 体系结构

本文结合 BPEL 在业务流程定义方面和 WS-TX 在 Web 服务事务协调方面的优势，提出了一种基于两者的 Web 服务组合协调框架，如图 1 所示。

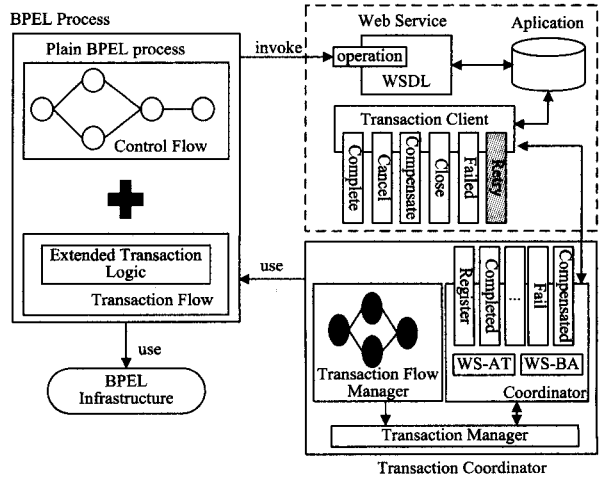

图 1 Web 服务组合协调框架体系结构

框架包括 3 个主要部分，即事务协调器、单个的 Web 服

务参与者、扩展的 BPEL。对各部分介绍如下。

1)事务协调器。事务协调器负责组合服务整个过程中的事务协调和控制,其包含 3 个组成单元:事务管理单元、事务流管理单元和协调单元。事务管理单元根据每个组合服务的事务逻辑使用协调单元与 Web 服务参与者进行交互,控制参与者的行为。事务流管理单元通过抽取扩展的 BPEL 来获取每一个组合服务特定的事务逻辑,为事务管理单元提供服务。协调单元实现了 WS-TX 规范中的 WS-C 协调框架,并集成协调协议 WS-AT 和 WS-BA,用来与参与者服务进行事务协调的交互。

2)Web 服务参与者。参与组合服务某项业务工作的完成,如图 1 虚线框部分所示。用 WSDL 描述的服务接口将被 BPEL 执行时调用,以完成特定的业务活动。为了使企业原有应用只需要做很小的改变就能实现 Web 服务环境下的事务协调,需要在原有应用前端安装用来进行事务协调的客户端组件。该组件的功能是使得服务参与者与协调器之间可以进行交互,接收来自协调器的指令和向协调器发送自己的状态更新等。

3)扩展的 BPEL。组合服务的执行文件。使用 BPEL 语言定义出组合服务的业务流程,并基于 scope 对业务流程进行事务逻辑扩展,详细说明见本文第 3 节。由 BPEL 执行引擎负责组合服务业务流程的执行。

2.3 执行和恢复策略

基于上述框架,一个服务组合从定义到运行的过程如图 2 所示。由 Web 服务组合设计人员根据具体的业务流程设计出组合服务的工作流程,并根据业务特性划定组合服务执行过程中的事务范围,同时还需要根据组合服务的事务特性定义出所有参与者之间的事务依赖关系,这些信息都以 BPEL 文件保存。图 2 中①表示协调器需要根据定义好的 BPEL 组合服务抽取整个服务的执行流程和各服务间的事务依赖信息,用于事务的协调;②表示 BPEL 执行引擎执行定义的服务组合流程,协调器监控各参与者的执行情况;③表示组合服务顺利执行完毕;④表示异常或错误发生时,协调器负责异常和错误的处理。

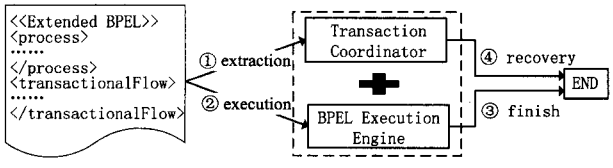

图 2 Web 服务组合协调框架执行流程

当某个参与者服务发生错误,不能完成业务活动或因为某些原因需要退出时,参与者通过客户端组件向协调器报告状态;协调器再根据该参与者的事务属性和整个业务活动的状态进行协调处理;参与者根据协调结果向 BPEL 执行引擎返回执行结果,并影响整个组合服务的继续运行。

本文提出的协调框架,其同时支持向前和向后两种恢复策略。如果该参与者服务的业务操作可以重试,则向参与者发送重试消息,参与者进行重试操作;如果参与者的业务动作不能完成或者必须退出,则协调器根据各参与者的状态和事务依赖关系向其它参与者发送事务协调控制消息,如补偿、取消操作等。

3 BPEL 的扩展和协调过程

3.1 BPEL 的扩展

第 2 节中的协调框架要正确运行,必须解决两个问题:一是获取组合服务的执行流程,这样协调器才能掌控整个服务的执行状态;二是获取各服务间的事务依赖信息,这里的事务依赖信息指的是组合服务中某一个服务的执行情况对其他合作服务执行的影响。如一个服务的开始执行导致对另一个服务的取消等。对于前者,BPEL 可以满足组合服务灵活的流程定义,已经有很多关于从 BPEL 到执行流程自动机的转换等工作[13,14],这为我们获取组合服务的执行流程提供了理论和实现基础。但对于后者,BPEL 原有的机制是不能提供的,因此需要对 BPEL 进行事务语义描述能力的扩展。

WS-AT 协议定义了 Web 服务原子事务的协调类型。WS-AT 类似于传统的分布式或者数据库事务,一般使用两阶段提交协议,参与事务的 Web 服务的执行结果或者全部成功或者由于一个失败其它执行全部取消。这样的事务一般持续时间比较短。WS-AT 事务可以保证短时间的分布式活动获得一致的结果,从而具有 all-or-nothing 的特性。

使用 WS-AT 协议的业务流程要么所有执行都成功,要么所有结果都被取消,事务语义很明确,所以本文主要针对 Web 服务长事务活动的事务语义进行扩展。而 WS-TX 中的 WS-BA 协议是用来描述长事务活动的协议,因而我们考虑的是基于 WS-BA 协议的事务扩展。

文献[7]中指出,一个可靠的 Web 服务需要兼顾业务逻辑控制流(control flow)和异常处理逻辑事务流(transaction flow),并给出了一个 Web 服务事务流的形式化定义。其基本思想是通过定义服务在执行期间自身状态迁移发生的前置条件来描述 Web 服务组合事务流逻辑,并以此作为事务协调的基础。

在 WS-BA 协议中描述了 Web 服务执行期间的所有状态迁移,如图 3 所示。状态迁移分为两种:外部迁移和内部迁移。其中,外部迁移是指由于外部实体的变化而触发了本服务的状态迁移,内部迁移是服务内部业务逻辑执行情况使得状态发生迁移。为了表述方便,在此仅考虑部分迁移:activate(),completed(),exit(),fail(),cancel(),compensate()。对于其他迁移如 failed()等不会引起其他服务的状态变化,CannotComplete()等则可以参照已有的迁移做类似处理。其中外部迁移有 activate(),cancel()和 compensate()。内部迁移为 complete(),exit()和 fail()。

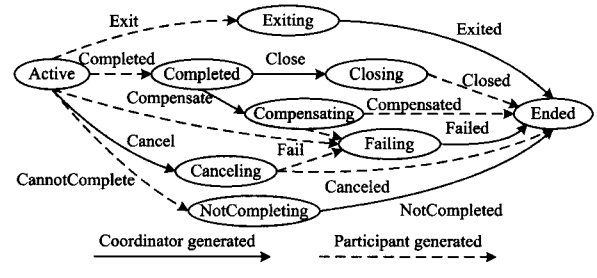

图 3 BusinessAgreementWithParticipantCompletion 状态转换图

基于以上的考虑,一个组合服务的业务逻辑可以通过定义其迁移 activate()的前置条件(CondAct(s))来描述;异常处理逻辑可以通过定义迁移 cancel()和 compensate()的前置条

件(CondCnl(s)和 CondCps(s))来刻画,而两个逻辑的条件是相互独立的,本文的协调框架仅需关注其事务流。对于一个 Web 服务,CondCnl(s)和 CondCps(s)是唯一的,当前置条件被满足时,协调器将向参与者服务发送对应的协调消息。因此下面结合 WS-BA 给出一个服务组合的事务流定义。

定义 1 一个事务流是一个二元组:$TF=(ES, Prec)$。其中 ES 是一个 Web 服务的集合,$Prec$ 是服务发生外部迁移的前置条件的集合,即:

$$\forall s,s' \in ES, Prec(s)=\{Prec(s.cancel())=CondCnl(s)\} \cup \{Prec(s.compensate())=CondCps(s)\}$$

$$CondCnl(s), CondCps(s) \subset \{s'.activate, s'.exit, s'.fail, s'.cancel, s'copmesate\}$$

$CondCps(s)$ 隐含服务 s 已经完成。在以上定义的基础上,为了同时可以实现向前和向后事务恢复,在 BPEL 框架的基础上对 BPEL 语言进行了事务语义的扩展定义如下:

```
〈process name="NCName"〉
  〈partnerLinks〉
    〈partnerLink name="NCName"…retriable="true|false"〉
    〈/partnerLink〉+
  〈/partnerLinks〉
  ……
  〈scope coordinationType="WS-AT|WS-BA"〉
    activity
  〈/scope〉+
  //Transactional Flow Extention
  〈transactionalFlow〉
  〈service name = "NCName"〉
  〈cancelCondition〉〈/cancelCondition〉
    〈compensationCondition〉〈/compensationCondition〉
  〈/service〉+
  〈/transactionalFlow〉
〈/process〉
```

〈partnerLink〉中声明了所有的参与者服务,如果该参与者提供的服务操作具有可重试性,则将属性 retriable 设置为 true,否则设置为 false。〈scope〉标签表明了一个事务的范围,coordinationType 属性表明该事务协调的协议类型。〈transactionFlow〉中详细描述了每个服务的异常状态迁移的前置条件,〈cancelCondition〉定义了服务的操作取消执行的前置条件,〈compensationCondition〉定义了服务操作完成后需要进行补偿操作的前置条件,基于每个服务的异常状态迁移前置条件形成了整个服务组合的事务流。对于前置条件的定义,文献[7]针对各种事务模式给出了定义方法,本文仅在最后的实例分析中给出一个例子来说明。

3.2 协调过程

组合服务执行过程中,流程参与者之间的交互关系如图 4 所示。下面仅对使用 WS-BA 协议协调的流程执行和协调过程进行说明。

1)事务流获取。事务协调器通过读取 BPEL 流程获取业务流程和事务语义,其中,事务流管理单元以此作为对业务流程执行状态维护的基础。协调器读入 BPEL 文件,读取〈partnerLinks〉标签中的描述服务参与者操作可重试性的属性值;读取〈scope〉标签中事务协调协议类型属性值;扫描〈transactionFlow〉标签下的每个参与者服务的异常状态迁移前置条件并加以记录。同时,协调器将〈scope〉中描述业务流程转化为自动机用来跟踪组合服务的执行状态。

2)协调上下文的创建和参与者注册。在进入 scope 标识的事务范围之前,需通过 invoke 操作在事务协调器上创建协调上下文:

〈invoke partnerLink = "TransactionCoordinator" portType = "ws:createContextPT" operation = "createContext"〉

〈/invoke〉

获得协调上下文之后将其在参与者之间进行传播。当参与者接收到协调上下文后,通过协调客户端来向协调器进行注册。

3)业务流程的执行。待参与者注册完毕后,BPEL 执行引擎接收到协调器注册完成消息后进入 scope,BPEL 引擎将开始调用参与者的服务来完成业务活动。参与者在完成业务活动的过程中,通过协调客户端根据事务协调协议来向协调器报告自身的执行情况。

4)错误和异常的处理。当某服务的执行发生异常时,则向协调器发送协调消息,报告自身的业务执行情况。事务管理器根据事务流管理单元提供的流程执行状态和事务流通过协调单元发送协调协议消息。当接收到参与者报告的执行异常或错误消息时,如果该服务操作可以重试,则发送 retry() 消息,并重置该参与者服务的执行状态;如果该服务操作不能完成,而根据每个服务的 Prec()得知该服务结果不影响整个组合服务的执行结果,则根据协调协议发送相应消息让其退出组合服务并协调部分受其影响的服务;如果该服务的失败使得整个服务失败,则协调器要根据流程的执行状态和每个服务的 Prec()来取消执行或者补偿执行结果,并终止整个组合服务的执行。对每个服务的协调动作按照其状态迁移前置条件成立的先后顺序来处理。

5)执行结果的提交。若所有必要服务操作都顺利完成,BPEL 调用协调器的 commit()操作来完成所有执行结果的最后提交。

〈invoke partnerLink = "TransactionCoordinator" portType = "ws:commitPT" operation = "commit"〉〈/invoke〉

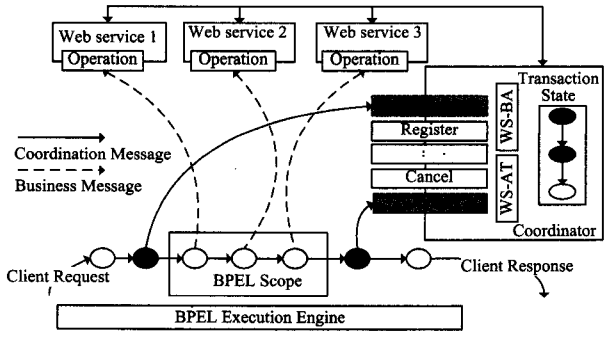

图 4 业务流程的执行和协调交互过程

4 实例分析

考虑一个在线旅行代理服务组合应用,如图 5 所示。旅客需要在线订购到某地的飞机票(Flight Booking,FB),并且在一家当地的旅馆预定客房(Hotel Booking,HB)。如果需求能够被满足,旅客将进行在线付款(Online Payment,OP)。成功付款后,整个服务结束。如图 5 所示,同时假定机票和旅馆

的预定操作不能重试但可以补偿，付款操作可以重试但不能被补偿的。旅客允许客房预订不成功，但是在客房预订成功的情况下飞机票也必须能够被预订。

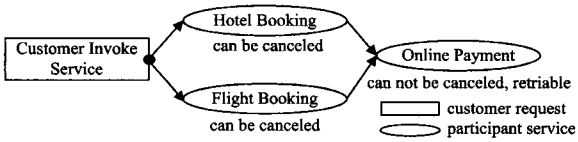

图 5　在线旅行代理组合服务

基于上述业务逻辑和事务语义，给出各服务的前置条件定义如下：

Hotel Booking：

CondCnl(HB)={FB. fail, FB. exit, FB. cancel}

CondCps(HB)={FB. fail, FB. exit, FB. cancel | HB. completed}

Flight Booking：

CondCnl(FB)={}

CondCps(FB)={}

Online Payment：

CondCnl(OP)={FB. fail, FB. exit, FB. cancel}

CondCps(OP)={}

由于 HB 的失败不会影响 FB 的执行，而 OP 经过若干次重试是可以完成的，因此 CondCnl(FB)、CondCps(FB) 和 CondCps(OP) 均为空。

该组合服务采用 WS-BA 协议来进行事务协调。假设上述组合服务执行过程中，除 Hotel Booking 服务未能完成之外，其他服务都正确完成。各参与者、BPEL 流程和协调器之间的交互如图 6 所示。

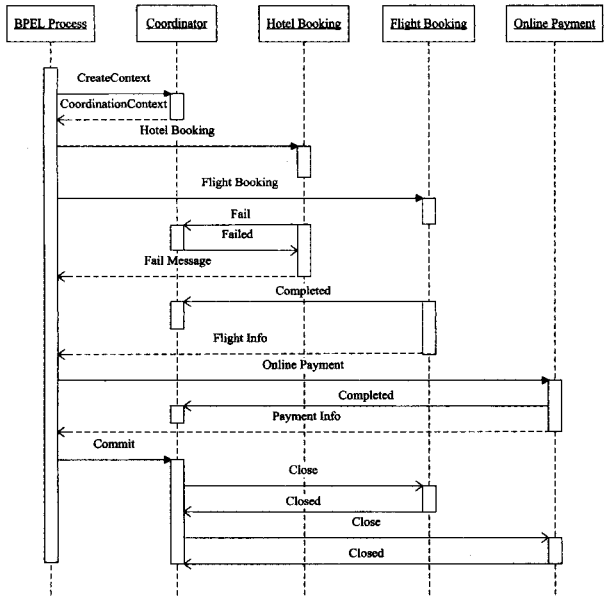

图 6　部分完成的旅行代理组合服务

在图 6 中，省略了参与者在协调器上注册的过程。BPEL 执行引擎并行调用了 Hotel Booking 和 Flight Booking 两个服务。Hotel Booking 服务执行失败，其向协调器发送了 Fail 消息，协调器检查整个组合服务的状态和其他服务的前置条件后，由于其执行结果不影响整个服务的执行，因而向其发送 failed 消息，让其退出组合服务的执行。Flight Booking 服务执行完成之后，向协调器报告执行状态后将结果返回给 BPEL，BPEL 继续调用 Online Payment 服务。Online Payment 执行完成并报告给协调器后将结果返回给 BPEL。最后 BPEL 执行引擎向协调器发送 commit 消息，协调器随之向各完成的参与者发送 Close 消息，结束协调。

结束语　结合 BPEL 和 WS-TX 的优势，给出了一个 Web 服务组合协调框架并通过一个实例分析说明其可行性。该协调框架支持混合业务结果的提交，采用了向前和向后两种恢复策略，并将组合服务的业务流程和事务协调逻辑进行了有效分离，以支持灵活的业务流程设计和事务协调逻辑定义。

本文尚未对事务的嵌套实现很好的支持，但 BPEL 支持嵌套的 scope 为此提供了便利，同时 WS-TX 也具有嵌套事务的协调能力。这也是下一步工作的重点。

参 考 文 献

[1] 岳昆，王晓玲，周傲英. Web 服务核心支撑技术：研究综述[J]. 软件学报，2004，15(3)：428-442

[2] OASIS. Web Services Business Process Execution Language (BPEL4WS/BPEL)[EB/OL]. http://docs. oasis-open. org/wsbpel/2. 0/OS/wsbpel-v2. 0-OS. pdf, 2007

[3] Bunting D, et al. Web Services Composite Application Framework (WS-CAF)[EB/OL]. http://xml. coverpages. org/WS-CAF-Primer200310. pdf, 2003

[4] OASIS. Web Services Coordination (WS-C)[EB/OL]. http://docsoasis-open. org/ws-tx/wstx-wscoor-1. 2-spec-os. pdf, 2009

[5] OASIS. Web Services Atomic Transaction(WS-AT) [EB/OL]. http://docs. oasis-open. org/ws-tx/wstx-wsat-1. 2-spec-os. pdf, 2009

[6] OASIS. Web Services Business Activity(WS-BA) [EB/OL]. http://docs. oasis-open. org/ws-tx/wstx-wsba-1. 2-spec-os. pdf, 2009

[7] Bhiri S, Godart C, Perrin O. Transactional Patterns for Reliable Web Services Compositions[C]// Proceedings of the 6th international conference on Web engineering. ICWE'06. USA: ACM 2006：137-144

[8] Apache. Kandula. http://ws. apache. org/kandula/

[9] Vasquez I, Miller J, Verma K, et al. OpenWS-Transaction: Enabling Reliable Web Service Transactions[C]// Proceedings of the third international conference on Service-Oriented Computing. ICSOC'05. Springer Berlin, 2005：490-494

[10] 闫艳，黄志球，袁敏，等. 面向服务的多参与者协调事务建模方法[J]. 计算机科学与探索，2011，5(4)：356-366

[11] Vogt F H, Zambrovski S. Implementing Web Service Protocols in SOA: WS-Coordination and WS-BusinessActivity[C]// Proceedings of the Seventh IEEE International Conference on E-Commerce Technology Workshops, CECW'05. IEEE, 2005：21-26

[12] Sun Chang-ai. Towards Transaction-based Reliable Service Compositions[C]// Proceedings of the 33rd Annual IEEE International Computer Software and Applications Conference. IEEE, 2009：216-221

[13] Wombacher A, Fankhauser P, Neuhold E. Transforming BPEL into annotated deterministic_nite state automata for service discovery[C]// Proceedings of the 2nd IEEE International Conference on Web Services. IEEE, 2004：316-323

[14] Fu X, Bultan T, Su J. Analysis of interacting BPEL web services [C]// Proceedings of 13th International Conference on World Wide Web. ACM, 2004：621-630