一种基于异常分类的面向服务异常处理方法

丁月华[1,2],应 时[1],贾向阳[1],王一兵[3]
(1. 武汉大学 软件工程国家重点实验室,湖北 武汉 430072;
2. 武汉工业学院 数学与计算机学院,湖北 武汉 430023;
3. 第二炮兵指挥学院 三系,湖北 武汉 430012)

摘 要:异常处理本就是一件复杂耗时且容易出错的任务,尤其在面向服务体系结构中,异常抛出的行为变得更加复杂,因此对面向服务的异常捕获和异常处理提出了更高的要求。文中首先对面向服务的异常产生的源头进行了分析,提出了面向服务的异常分类方法,将面向服务的异常分为服务层异常和流程层异常。在此基础上,服务层异常和流程层异常再各自细分为系统异常、资源异常和应用异常。最后针对每类异常,提出了对应的异常处理方法,能够分别在服务层和流程层上为面向服务的系统提供更好的健壮性。

关键词:面向服务;异常分类;异常处理;服务层;流程层
中图分类号:TP311 **文献标识码**:A **文章编号**:1673-629X(2012)12-0015-06

A Service-oriented Exception Handling Method Based on Exception Classification

DING Yue-hua[1,2], YING Shi[1], JIA Xiang-yang[1], WANG Yi-bing[3]
(1. The State Key Lab of Software Engineering, Wuhan University, Wuhan 430072, China;
2. School of Mathematics and Computer Science, Wuhan Polytechnic University, Wuhan 430023, China;
3. Third Department, The Second Artillery Command College, Wuhan 430012, China)

Abstract: Exception handling is a time-consuming and easy to make error task. Especially in service-oriented architecture, exception throw chain becomes complicated particularly, so it improves requirements of exception capture and exception handling. It analyzes service-oriented exception produce source and proposes service-oriented exception classification. The exception classification divides exception into service layer exception and process exception. The two kind of exception is subdivided into system exception, resource exception and application exception. It presents a corresponding exception handling method for all kinds of exceptions. The method can enhance robustness of service layer and process layer for service-oriented system.

Key words: service-oriented; exception classification; exception handling; service layer; process layer

0 引 言

面向服务的技术体系架构作为一种新兴及蓬勃发展的软件架构理念有着良好的可重用性、松耦合性、平台无关性和互操作性。然而,Web服务所处的环境是一个异构的、分布式的自治和快速变化发展的动态环境,网络环境的异构性、分布式等特点决定了服务组合在执行的过程中会遭遇各种异常,如通讯模式的变化、网络失效、服务拒绝攻击、基础设施失效、应用逻辑约束失效等,这些问题都将导致服务组合的非正常运行,此时服务组合的异常处理方法就显得尤为重要。BPEL作为服务组合事实的工业标准,对其异常处理程序的开发展开深入研究对指导工业化面向服务软件的开发有着重要意义。

目前,BPEL中提供了处理异常的系统级别机制,业务流程开发人员可以使用<Scope>、<FaultHandler>、<Compensation>等标记处理异常,利用BPEL提供的机制完成具有容错能力的业务流程定制,但仍然存在以下问题:

①正常业务逻辑和异常业务逻辑交织在一起,异常流和正常流程紧密耦合,流程的可读性较差,开发、维护的难度都很难进行。

②需要流程设计人员全面考虑各种类型的异常,

收稿日期:2012-05-15;**修回日期**:2012-08-22
基金项目:国家自然科学基金资助项目(60773006)
作者简介:丁月华(1979-),女,博士生,CCF会员,研究方向为面向服务的软件开发、形式化方法;应 时,博士,教授,博士生导师,研究方向为面向服务的软件工程方法、基于组件的软件工程方法、软件体系结构和模式、软件的可重用性与互操作性等。

手工完成异常的处理费时费力,工程实施难度较大。

③异常发生时,正常流程直接终止后续的执行,BPEL 没有提供 retry、resume 等机制。需要提供更好的方法完成面向服务的异常处理开发。

文中在全面分析面向服务故障发生原因的基础上,提出了面向服务的异常行为分类方法,将面向服务的异常首先分为了服务层异常和流程层异常两大类型。根据异常行为产生的源头,进一步将面向服务的异常细分为服务层系统异常、服务层资源异常、服务层应用异常、流程层系统异常、流程层资源异常、流程层应用异常。在此基础上,针对以上的异常类型针对性地提出了异常处理方法,为规范化、简单化面向服务的异常处理提供了值得借鉴的方案。

1 相关研究

1.1 面向服务的异常分类

由于 Web 服务具有松耦合、自治和异构等特点,导致抛出的异常情况复杂。而且,随着面向服务技术的进步和应用程度的深入,人们根据不同的企业需求,将多个现有的面向服务软件集成,形成更大规模、更复杂的大型软件系统。这类软件系统应当处理的异常及其组合情况也更加复杂,需要结合面向服务软件及 Web 服务环境的特点对异常进行分类,继而国内外学者主要展开了以下研究。

国内学者在异常分类方法的代表性工作有:华东理工大学的范贵生等人将 BPEL 流程异常分类为行为异常、运行异常和 SLA 异常[1]。中国科技大学的刘安等人将 BPEL 业务流程可能遇到的异常可以分成三类[2]:伙伴级别异常、流程级别异常和系统级别异常。中国科学技术大学的黄涛和夏永霖以减少 QoS 非功能异常为出发点,深入研究了在 Web 服务环境下复合服务的自恢复技术,并将非功能异常划分为基础设施异常、基本服务异常和复合服务异常三大类[3]。

国外学者关于异常分类的主要代表性工作有:意大利米兰理工大学的 ArdagnaD. 等人根据异常影响到的系统组件的级别,将异常划分为三大类[4]:基础设施及中间件级异常、Web 服务级异常和 Web 应用级异常。三大类包含六个小类:Web 服务运行异常、Web 服务协调异常、内部数据异常、应用协调异常、角色异常、QoS 冲突异常。奥地利克拉根福大学的 G. Friedrich 等学者则将 Web 服务调用时发生的异常分为:被调用 Web 服务返回错误信息和无法返回任何信息两种异常[5]。德国基尔大学计算机学院 Germany Jensen 等人提出了在 Web 服务组合的执行故障的传播以及处理方法,提出了一些服务分类准则,在此基础上提出了一种故障传播处理的方案。但是分析的不够全面仔细,仅仅探讨了异常发生可能之后的一种处理补偿[6]。奥地利维也纳科技大学的 O. Moser 等人从异常监控角度将异常分类为系统异常和应用异常[7]。希腊雅典大学的 K. Christos 等人根据异常的来源将异常分类为服务失效、组件运行故障和网络发生故障[8]。

在上述研究中,可以看出每位研究者按照不同的标准对面向服务的异常进行了分类,有一定的借鉴意义。但以上面向服务的异常分类中,均没有区分服务层异常和流程层异常,从而无法对异常产生的源头进行全面的分析。

1.2 面向服务的异常处理

对于面向服务的异常处理,可以借鉴部分事务处理技术来展开研究。然而,在传统事务模型中"全有或全无"的原子性质很难适用于长运行、松耦合和自治的服务组成的分布式应用环境,因此面向服务的异常处理对于提高系统的健壮性尤为重要,国内外学者同样针对面向服务的异常处理提供了一些可供参考的解决方案。

目前,国内关于面向服务的异常处理方法的代表性工作有:中国科学技术大学的刘安等人提出了一种基于规则的异常处理机制[2]。该机制使用 ECA 规则来描述异常处理逻辑,这些 ECA 规则建立在一套具有明确语义的异常处理模式之上。在业务流程部署前,这些 ECA 规则自动转化为标准的 BPEL 代码,并集成到业务正常逻辑中,形成具有容错能力的业务流程。中国科学院计算技术研究所软件研究室的李东来等人提出了一种基于案例匹配的异常处理方法[9],从服务最终呈现的功能和非功能属性方面进行分析和处理,并重点探讨服务不可用引发的应用异常处理。IBM 的 ZengLiangzhao 等人提出了一种策略驱动的 Web 服务组合异常管理框架,采用策略以表示 BPEL 流程异常处理知识[10]。

国外关于面向服务的异常处理方法的代表性工作有:意大利米兰理工大学的 BaresiL 等人提出了一套完整的自愈框架,分为异常监控部分和异常恢复 2 个部分[11]。在异常监控部分,监控逻辑通过由 WSCoL 规约语言定义,标注在 BPEL 流程中程。异常恢复部分提供了异常恢复语言 WSReL 支持服务异常恢复策略的定义,该语言使用基于 ECA 规则的范氏。奥地利克拉根福大学的 G. Friedrich 等人提出的异常处理自愈方法支持基于服务的过程的异常处理和基于模型的故障活动修复[5],其基本思路通过故障诊断,发现引发流程异常的原因,执行 BPEL 流程异常处理策略,从而修复 BPEL 流程异常。意大利米兰理工大学的 S. Modafferi 等人提出的一种自愈能力的 BPEL 引擎 SH-BPEL,该引擎检测到异常后自动执行恢复动作集

合[12]。澳大利亚新南威尔士大学的 Rachid Hamadi 提出了 BPEL 流程自恢复网 SARN[13],SARN 是一种基于 Petri 网的 BPEL 流程恢复策略描述语言。SARN 通过恢复变迁和恢复令牌来建模 BPEL 流程的异常处理。

在上述研究中,以上方案都存在一定的局限性。例如,刘安提出的方案仅仅采用 ECA 规则定义异常处理,提供了 ECA 规则到带容错能力的 BPEL 流程转换思路,由于缺乏规范的策略语言,工程化实施难度较大;在 BaresiL 等人的基于断言语言的自监管方法,语言自身可读性较差,从某种程度上加大了开发人员的工作难度; G. Friedrich 等人提出的异常处理自愈方法的修复计划生成算法其复杂度会随着 BPEL 流程复杂度的增加而急剧增长;其他解决方案仅仅对服务的某类异常进行专门处理。

2 案　例

为了更好地分析面向服务的异常处理方法,文中使用 BPMN 描述一个电子商务案例。该应用模型描述的是一家网上售卖食品商店订单流程编制为 SHOP 服务,如图 1 所示。

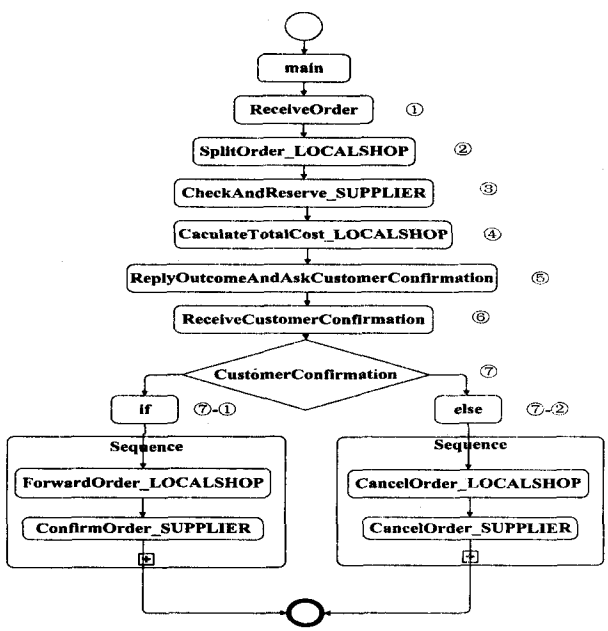

图 1　食品商店案例

通过面向服务的技术连接了 3 个服务:

· LOCALSHOP 服务:本地商店服务,对每个订单项获得 IDcode 和商品目录,属于食品公司内部服务。

· SUPPLIER 服务:供货商服务,货品请求被分发给 SUPPLIER 服务,属于食品公司外部服务。

· SHOP 服务:是主要的服务,由 LOCALSHOP、SUPPLIER 服务编制完成,提供售卖商品功能,属于食品公司内部服务,图 1 展示了它的完整流程。

在本流程中,主要描述了客户完成订单的流程。该流程如下:

1. 首先通过 ReceiveOrder 活动接收来自客户的订单请求;

2. 然后通过 LOCALSHOP 服务的 StoreOrder、SplitOrder 完成订单的存储以及订单的分离,将订单分解成若干个不同的商品项;

3. 将分解出来的商品项发送到 SUPPLIER 服务的 CheckAndReserve 端口;

4. 调用 LOCALSHOP 服务的 CaculateTotalCost 完成商品项总额的计算;

5. 接下来返回计算计算金额并且要求用户确认;

6. 接收来自用户的确认消息 ReceiveCustomer-Confirmation;

7. 根据用户发送的确认信息 CustomerConfirmation,形成 2 个新的分支流:确定订单信息流和取消订单信息流。其中,确定订单信息流顺序调用 LOCALSHOP 服务的 ForwardOrder 接口和 SUPPLIER 的 ConfirmOrder 接口,而取消订单信息流调用 LOCALSHOP 服务的 CancelOrder 接口和 SUPPLIER 的 CancelOrder 接口;

8. 最后,回复给用户确定的消息 ReplyToCustomer。

3 面向服务的异常分类

由于面向服务系统在松散网络中的特性,面向服务的软件异常产生的原因较为复杂。文中针对面向服务软件产生异常所处的层次,将面向服务的异常类型分为服务层异常和流程层异常。在此基础上,每类异常根据异常产生的源头,进一步细分为系统异常、资源异常、应用异常。下文具体说明每种异常类型的特征,并确定了它们之间的层次关系。

3.1 服务层的异常分类

根据服务层异常的产生源头,文中将服务层异常划分为系统异常、资源异常和应用异常,并且按照自底向上的顺序形成了服务层异常层次,图 2 展示了它们之间的层次关系。

①系统异常:是指服务运行时因软硬件基础设施的错误而引发的异常。例如网络、CPU、硬盘等硬件故障,以及操作系统、应用服务器、服务引擎等软件故障,该类异常的发生无法预知。

②资源异常:是指服务执行中所需的资源发生错误而引发的异常。包括数据库、文件系统、LDAP、FTP、Email 等数据资源和 Web 服务、外部系统接口等计算资源引发的异常。

③应用异常:是指服务被调用时因不满足服务业务协议而引发的异常,往往应用异常需要在 WSDL 文件中描述其异常类型,根据下层封装的组件层完成应

用异常的定制。当应用异常发生时,服务本身并不能做任何处理,只能将异常发生的信息通过 SOAP 协议的方式发送给服务调用者。

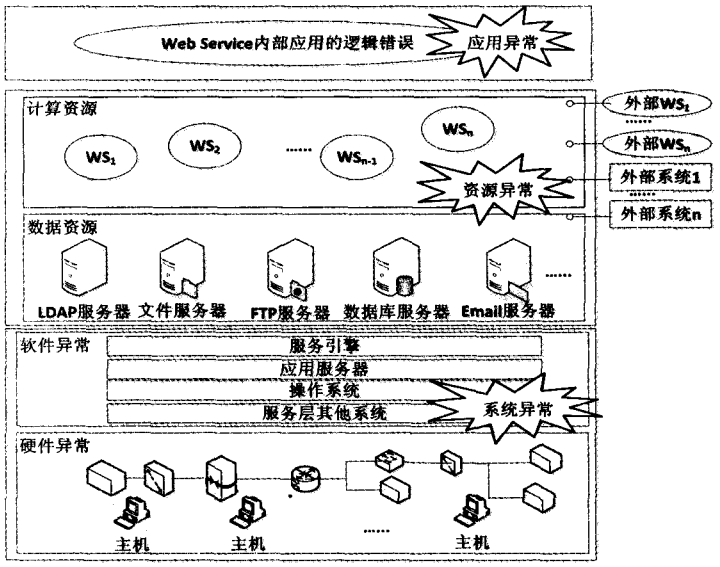

图2 面向服务软件的服务层异常行为

3.2 流程层的异常分类

根据流程层异常的产生源头,文中将流程层异常同样地划分为系统异常、资源异常和应用异常。与服务层异常类似,按照自底向上的顺序形成流程层异常层次,图3展示了它们之间的层次关系。

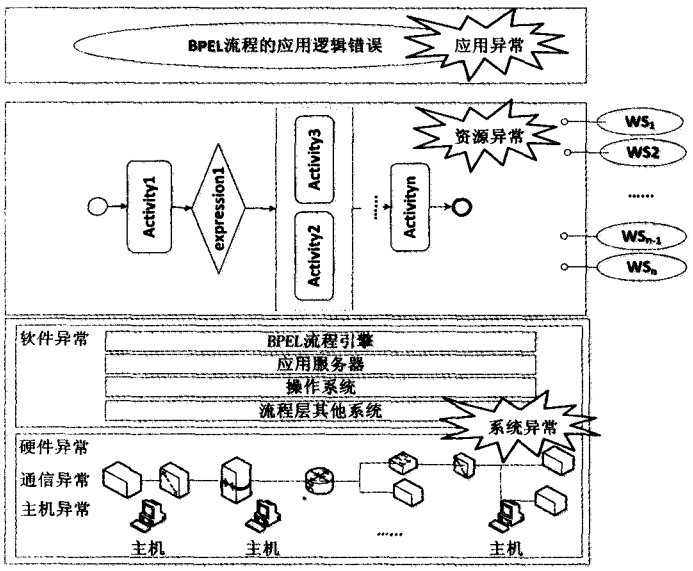

图3 面向服务软件的流程层异常行为

①系统异常:是指流程执行时软硬件基础设施的错误而引发的异常。包括流程所处的计算机硬件故障引发的异常,以及流程所在的 BPEL 引擎、应用服务器、操作系统故障引发的异常。

②资源异常:是指流程执行中所需的资源发生错误而引发的异常。在 BPEL 流程中,由于外部资源是通过服务的方式进行访问的,因此,将主要关注服务资源错误而引发的异常,如服务不可用、服务协议绑定错误、服务接口不匹配、SLA 冲突等。

③应用异常:是软件的应用逻辑发生错误而引发的异常,流程执行过程中的上下文信息不满足应用逻辑或约束而引发的异常,此类异常需要由 Web 服务和 BPEL 流程设计人员在设计阶段定义。

流程层应用异常又分为:

1 交互异常:流程与伙伴服务交互过程中引发的异常,伙伴服务交互异常可进一步细分为:交互内容错误、交互顺序紊乱等异常;

2 业务约束异常:流程执行过程中由于业务约束不满足而引发的异常,该类异常在 BPEL 流程中由 Throw 或 Rethrow 活动抛出。

4 面向服务的异常处理

在面向服务机制下,如果服务异常不进行处理,那么将会终止服务的执行,仅仅向服务调用者返回带有异常类型的 SOAP 消息,而使得服务异常的发生后不能得到有效的处理。类似地,如果流程异常不进行处理,流程直接终止后续活动的执行,而使得流程实例的执行很容易由于异常导致流程实例的执行中断。

下文将会按照前述的分类,给出各类异常的处理方法。

4.1 服务层的异常处理

在文中提出的服务层异常处理方法中,当服务调用发生异常后,不是简单的向服务调用者返回 SOAP 类型的错误信息,而是会调用发生异常的被调用服务的 FaultHandler 服务完成异常处理,例如通过补偿服务完成状态回滚操作。这样,原本不带容错能力的 Web 服务成为一个带容错能力的 Web 服务,其原理如图4所示。并且,本方案提供了一些基本的 FaultHandler 服务,包括重试(retry)、取消(cancel)、人工处理(human)、警告(alert)、等待(wait)、替换(replace)完成常见的异常处理。当基本的 FaultHander 服务不能满足异常处理要求的时候,需要定制 FaultHandler 服务完成异常处理。

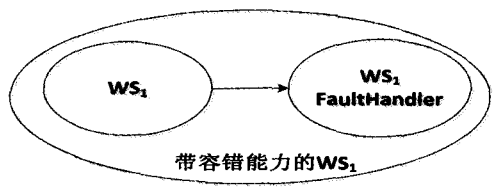

图4 服务层异常处理的原理图

当服务调用过程中发生了系统异常,那么该类异常可以通过人工处理、等待的基本 FaultHandler 进行处理。在文中的案例中,假定 LOCALSHOP 服务调用过程中,当对其接口 StoreOrder 进行调用时,发生了网络连接断开该类系统异常,可以通过等待 FaultHandler 服务(直到中断网络自动恢复)或者人工处理 FaultHandler 服务(人工干预使得网络恢复)完成使网络重新恢复连接才能正常调用 LOCALSHOP 服务。

当服务调用过程中发生了资源异常,那么该类异常可以通过重试、替换的基本 FaultHander 进行处理。在文中的案例中,假定 SUPPLIER 服务调用过程中,当对其接口 CheckAndReserve 进行调用时,进行数据库访问的时数据库连接中断,发生了数据资源异常,可以通过重试 FaultHandler 服务(重新访问数据库)完成数据访问。

当服务调用过程中发生了应用异常,那么该类异常可以通过重试、替换、人工处理、警告的基本 FaultHander 进行处理。如果基本 FaultHandler 无法有效处理,往往需要定制 FaultHandler。根据服务调用对持久层的修改与否,将服务分为可持久性服务和非持久性服务两种类型。其中,可持久性服务表示服务的调用会对持久层的数据造成影响,修改了持久层的数据,而非持久性服务的调用不会修改持久层的数据。由于可持久性服务的调用会修改持久层数据,非持久性服务的调用不会修改持久层的数据,因此该类服务的调用需要定制异常处理的服务,而非持久性服务的调用不需要定制异常处理的服务。在文中的案例中,假定在 SUPPLIER 服务的调用过程中,当对其接口 CheckAndReserve 进行调用时抛出了 UnavailableItem 应用异常,当基本的异常处理服务无法有效处理应用异常,需要为 CheckAndReserve 定制一个专门的 FaultHandler 服务对 UnavailableItem 应用异常进行处理,该定制的 FaultHandler 的逻辑如图 5 所示。

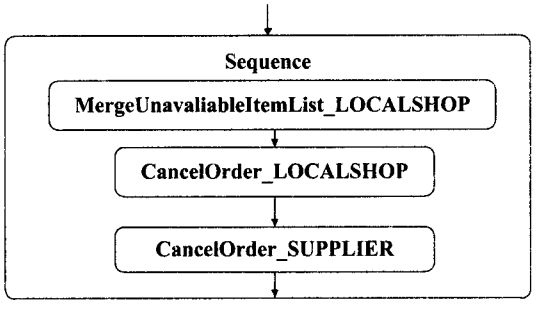

图 5　SUPPLIER 的定制 FaultHandler 服务

首先调用 LOCALSHOP 的 MergeUnavaliableItemList 完成无效商品项的合并。接着调用 LOCALSHOP 服务的 CancelOrder 接口,取消本地订单。最后调用 SUPPLIER 服务的 CancelOrder,接着取消供货商订单。对于应用异常而言,存在大量的定制 FaultHandler 服务。

4.2　流程层的异常处理

流程层将服务层提供的多个服务编制完成流程的定制。因此,其复杂度较服务层更大,当碰到异常在某个活动发生的时候,需要进行处理的工作不仅仅涉及到当前发生异常的活动,而且会对该流程中已经调用的可持久性服务进行异常处理。处理流程层异常,同样也需要流程层的 FaultHandler 服务,其基本原理和服务层类似,将不带容错能力的 BPEL 流程通过匹配 FaultHandler 服务封装成带容错能力的 BPEL 流程,如图 6 所示。

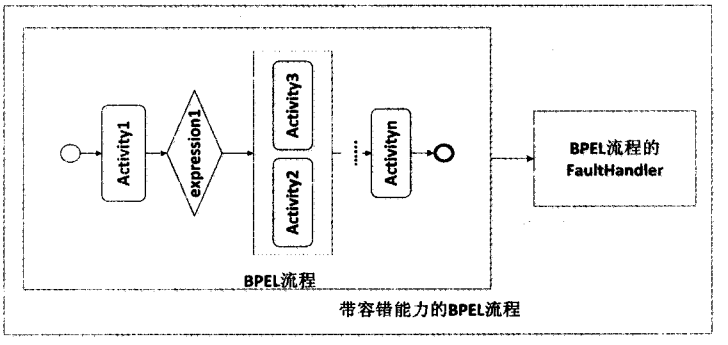

图 6　流程层异常处理的原理图

BPEL 流程作为由多个服务组合而成的服务更加复杂,因此在进行异常处理的时候需要考虑的问题更多。例如,流程层中需要考虑到流程中一系列活动的异常处理调用。按照服务层异常处理的原则,其中的可持久性服务需要调用各自对应的 FaultHandler 完成异常的处理。那么,一个流程中发生异常的活动位置决定了需要进行恢复的服务范围,在发生异常的活动位置之前的所有可持久性服务都需要调用对应的 FaultHandler 服务。文中提出的方案,按照流程中定制的活动顺序,从发生异常的活动位置逆向调用每个可持久性服务的 FaultHandler。本方案提供了面向流程层的基本 FaultHandler 服务,包括忽略(Ignor)、跳过(Skip)、依次代替(Alternate)、重试(Retry)、补偿(Compensate)、取消(Cancel)、人工处理(Human)、警告(Alert)、调用外部服务(Call)、替换(Replace)、等待(Wait)。

当流程实例执行过程中发生了系统异常,可以使用人工处理、等待的基本 FaultHandler 服务进行处理。在文中案例中,假定在流程实例 shop_instance 执行过程中发生了 BPEL 引擎 ODE 出现故障无法响应服务请求,导致流程实例执行过程中的强行中断,可以通过等待(直到服务器自行恢复电力)或人工处理(人工干预使得服务器电力恢复)完成使网络重新恢复连接才

能正常调用服务。

当流程实例执行过程中发生了资源异常,可以使用重试、替换、调用外部服务的基本 FaultHandler 服务进行处理。在文中案例中,假定在流程实例 shop_instance 执行过程中发生了服务资源不可用的异常,SUPPLIER 的服务提供者更新了发布服务的 URI,使得服务调用者无法获取其服务描述信息,可以通过重试(再次调用 SUPPLIER 服务)、或替换(使用功能与 SUPPLIER 相同的服务来代替 SUPPLIER 的执行)完成异常的处理。

当流程实例执行过程中发生了应用异常,可以使用重试、替换、补偿、报警、调用外部服务的基本 FaultHandler 服务进行处理。如果以上基本 FaultHandler 不能正确处理异常,需要定制流程 FaultHandler 服务。在文中案例中,假定在流程实例 shop_instance 执行过程中发生了应用异常,SUPPLIER 服务的 CheckAndReserve 和 LOCALSHOP 服务的 CaculateTotalCost 服务在交互的过程中发生了交互参数错误,在使用重试、替换的基本 FaultHandler 无法正确处理流程实例异常的情况下,对该流程的此类参数错误应用异常定制 FaultHandler。

5 结束语

文中按照服务层、流程层对面向服务的异常行为进行分类,并根据异常源头进行进一步细分,对每种异常类型进行特征描述,并通过案例进行了说明。在此基础上,针对每种异常类型分别提出了处理方法。这种分类的异常处理方法,能够分别在服务层和流程层提供更好的健壮性。下一步的工作,主要从框架技术支撑和形式化验证两方面展开:

①根据文中的异常行为的分类及其处理的基本思想,设计细化分层的异常处理框架对面向服务的异常进行处理,实现异常处理框架中的基本异常处理动作算法,并提供支撑面向服务的异常处理开发的可视化工具,为用户自定义异常处理服务提供平台支撑。

②通过形式化的方法验证文中提出的异常处理方法能够保证每类异常都能够得到处理,从而保证每个异常都能够得到处理或者终止流程。

参考文献:

[1] 范贵生,虞慧群,陈丽琼,等. 基于 Petri 网的服务组合故障诊断与处理[J]. 软件学报,2010,21(2):231-247.

[2] An L, Qing L, Liusheng H. FACTS: A Framework for Fault-tolerant Composition of Transactional Web Services[J]. IEEE Transactions on Services Computing, 2010, 3(1): 46-59.

[3] 夏永霖. 复合服务自恢复关键技术研究[D]. 合肥:中国科学技术大学,2010.

[4] Ardagna D, Cappiello C, Fugini M, et al. Faults and Recovery Actions for Self-healing Web Services[C]//Proceedings of the 15th International Conference on World Wide Web. New York, NY, USA: ACM, 2006.

[5] Friedrich G, Fugini M, Mussi E, et al. Exception Handling for Repair in Service-based Processes[J]. IEEE Transactions on Software Engineering, 2010, 36(2): 198-215.

[6] Jensen M. A Fault Propagation Approach for Highly Distributed Service Compositions[C]//SCC '08. Honolulu, HI: IEEE, 2008: 507-510.

[7] Moser O, Rosenberg F, Dusdar S. Non-Intrusive Monitoring and Service Adaptation for WS-BPEL[C]//Proceedings of the 17th International Conference on World Wide Web. New York, NY, USA: ACM, 2008: 815-824.

[8] Christos K, Costas V, Panayiotis G. Enhancing BPEL Scenarios with Dynamic Relevance-based Exception Handling[C]//ICWS 2007. Salt Lake City, UT: IEEE, 2007: 751-758.

[9] 李东来,韩燕波,王建武,等. 面向服务应用中服务可用性及其引发的异常处理研究[J]. 计算机研究与发展,2004,41(12):2104-2107.

[10] Zeng L, Lei H, Jeng J, et al. Policy-driven exception-management for composite Web services[C]//CEC 2005. [s. l.]: IEEE, 2005: 355-363.

[11] Baresi L, Guinea S. Self-supervising BPEL Processes[J]. IEEE Transaction on Software Engineering, 2011, 37(2): 247-263.

[12] Modafferi S, Mussi E, Pernici B. SH-BPEL-A Self-healing Plug-in for WS-BPEL Engines[C]//Proceedings of the 1st Workshop on Middleware for Service Oriented Computing. New York, NY, USA: ACM, 2006: 48-53.

[13] Hamadi R, Benatallah B. Recovery nets: towards self-adaptive workflow systems[C]//Proceedings of the 5th International Conference onWeb Information Systems Engineering (WISE'04). Berlin Heidelberg: Springer, 2004: 439-453.

(上接第14页)

[7] Wichman I. 4D-Trajectory Enabled Continuous Descent Approaches[R]. Arlington Virginia: Smiths Aerospace FAA New Technologies Workshop I, 2007.

[8] 王大海,苏 彬,杨 俊. 终端区域4D导引的高度剖面与速度剖面研究[J]. 飞行力学,2000,18(1):14-18.

[9] 吴 鹏,潘 薇. 基于数据挖掘的四维飞行轨迹预测模型[J]. 计算机应用,2007,27(11):2637-2639.

[10] 陈巧雅. 北京管制区域飞行流量仿真原型系统研究与开发[D]. 北京:清华大学,2004.

[11] 中国民用航空总局. MH4007-2006 民用航空飞行动态固定电报格式[S]. 北京:中国标准出版社,2006.

[12] 张 荣,祁 伟,许 坚,等. 高空风GRIB报文解析及精度分析[J]. 空中交通管理,2010(4):17-20.