基于规则的语义流程异常处理机制

赵 楷[1,2] 应 时[1] 张琳琳[2] 胡罗凯[1] 贾向阳[1] 王权于[1]
(武汉大学软件工程国家重点实验室 武汉 430072)[1] (新疆大学信息科学与工程学院 乌鲁木齐 830046)[2]

摘 要 在语义编程语言SPL的基础上,提出一种基于语义Web服务的语义流程异常处理机制。首先,重点讨论了业务流程运行时调用语义Web服务失败和流程内部逻辑失败的情况,并给出相应的语义流程异常本体。在此基础上,提出了5种异常处理动作,制定了异常处理ECA规则。然后给出了相应的原型系统。最后结合案例讨论了该机制的有效性和可行性。

关键词 SPL语言,异常处理,语义Web服务,业务流程
中图法分类号 TP312 **文献标识码** A

Exception Handling Mechanism for Semantic Process Based on Rules

ZHAO Kai[1,2] YING Shi[1] ZHANG Lin-lin[2] HU Luo-kai[1] JIA Xiang-yang[1] WANG Quan-yu[1]
(State Key Laboratory of Software Engineering, Wuhan University, Wuhan 430072, China)[1]
(School of Information Science and Engineering, Xinjiang University, Urumqi 830046, China)[2]

Abstract Based on the Semantic Programming Language(SPL), a mechanism was proposed for handling exception of semantic process. Firstly, focused on the external services invocation failure and processes internal logic failure during the execution of semantic process, the corresponding exception ontology for semantic process was given. On the basis of exception ontology, five exception handling operations were given to construct ECA rules for exception handling. Then, a prototype implementation based on ECA rules for handling exception of semantic process was given. Finally, a case study demonstrates the effectiveness and feasibility of the mechanism.

Keywords SPL, Exception handling, Semantic Web service, Business process

1 引言

面向服务的软件体系结构(SOA, Service-Oriented Architecture)是一种快速发展的新型网络化软件开发和应用模式,Web服务是SOA最重要的实现技术。在实际应用中,采用流程方式来组织服务和描述业务逻辑,将功能相对独立和单一的Web服务组合成可以满足服务请求者需求的、大粒度的增值服务,已经成为企业用来开发网络化软件的主要方法。然而,基于Web服务的业务流程标准大多假设网络环境稳定、可靠,未能充分考虑软件运行环境的动态性、开放性和难控性[1]以及Web服务的自治性、异构性和随机性等特点。特别是现有的Web服务容错机制缺乏灵活性,导致面向服务的网络化软件系统在执行过程中容易产生异常和失败,因此必须研究一种能够动态地保障流程可靠性的方法。利用该方法,在维持流程原始设计思路的同时,使流程能够动态适应复杂、易变、难控的网络运行环境。

异常处理是容错性服务组合中常用的技术手段之一[2]。随着软件系统规模的不断扩大,在以WS-BPEL[3]语言为典型代表而开发的系统中,大量的异常处理代码与正常的业务逻辑代码交织、混杂在一起,极大地增加了开发的难度和维护更新的成本,降低了代码可读性。另一方面,WS-BPEL主要基于统一的Web服务标准来弥合服务间语法交互的异构性,在运行时缺乏对语义等价而语法异构的服务的动态发现和替换支持。一旦出现服务失效或接口和数据语法不一致等故障,绑定该服务的程序都需要重新开发和部署,造成了编写的程序缺乏一定的灵活性。本体论为刻画Web服务及服务之间的关系提供了强有力的语义标注手段[4];语义Web服务与业务流程管理技术的结合——语义业务流程管理技术[5],也为增强基于流程的服务组合容错机制的灵活性和可靠性提供了新的思路。以此为支撑,本文给出了以语义编程语言SPL[6](Semantic Programming Language)为基础的语义流程异常处理机制。机制引入了面向语义流程的异常本体、异常处理的ECA规则和基于规则的异常处理框架3部分。其中,异常本体建模了基于语义Web服务的语义流程在运行时刻可能出现的异常信息;异常处理的ECA规则描述了建立在一组具有明确语义的异常处理动作之上的异常处理逻辑;基于规则的异常处理框架为语义流程执行时及时发现、定位和处理异常提供了支持。

到稿日期:2010-08-25 返修日期:2010-11-13 本文受国家高技术研究发展计划(863)项目(2006AA01Z168)和国家自然科学基金(61070012/F020202)资助。

赵 楷(1976—),男,博士生,讲师,主要研究方向为语义Web技术、SOA,E-mail:zhawkk@163.com;应 时(1965—),男,博士,教授,主要研究方向为面向对象软件工程方法、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、语义Web技术。

本文第2节阐述了相关研究工作；第3节概述了SPL语言及其异常处理机制；第4节定义了语义流程执行过程中可能发生的异常，构造了面向语义流程的异常本体和异常处理ECA规则，提出了基于规则的语义流程异常处理框架；第5节给出了框架的原型实现；第6节给出了案例应用；第7节对机制进行了讨论和说明。最后是结论和下一步工作。

2 相关工作

为了解决面向Web服务的业务流程在执行过程中所发生的故障，国内外的研究机构提出了许多不同的处理技术。这些技术涉及如何构造具有容错能力的业务流程以及如何管理伙伴服务，提供服务动态绑定和选择的能力，使业务流程的可靠性得到提高。相关的研究工作主要围绕容错性服务组合系统、服务组合故障处理模型及流程的自愈性等问题展开。文献[7,8]总结了服务组件运行时刻可能发生的异常情况。文献[9]阐述了由HP公司开发的、具有一定异常处理能力的E-flow系统。该系统提供了服务流程的组合描述语言、异常处理机制、ACID事务支持以及安全管理，支持动态服务发现和动态对话选择。文献[10,11]提出了一个具有前向恢复能力的、支持QoS管理和服务组合的工作流管理平台METEOR，它利用QoS模型和DAML-S来描述工作流和Web服务，以支持服务的动态发现和选择；使用基于案例的推理技术来处理服务流程的运行时异常问题。文献[12]对BPEL语言进行扩展，利用BPEL的故障处理机制来解决异常问题。文献[13]提出一种策略驱动的异常管理框架，该框架定义了几种常见的异常处理策略来处理WS-BPEL业务流程中的异常。文献[14]提出对WS-BPEL引擎进行修改，从而实现了一种基于策略的异常处理方法。文献[15]提出使用ECA规则来定义工作流系统中异常处理行为的方法。文献[2]提出了一种基于ECA规则的事务化Web服务的容错性组合框架(FACTS)，给出了8种异常处理策略及它们在ECA规则中的使用方法，并对所形成的业务流程的容错能力进行了验证。文献[16-18]讨论了语义Web技术和本体论在Web服务和业务流程故障处理方面的应用和研究。在前期工作中，依托863计划课题，我们提出了基于语义Web服务的编程语言SPL。借鉴上述异常处理的研究成果，本文介绍了SPL语言的异常处理思想，提出了异常处理ECA规则，给出异常处理框架。该框架能够支持SPL语言描述的语义业务流程的执行，及时捕获流程运行时发生的异常，根据ECA规则处理异常，能够动态发现合适的Web服务并替换发生故障的Web服务。

3 语义编程语言SPL

3.1 概述

SPL语言是在WS-BPEL语言的基础上扩展而成的，增加了语义服务、语义类型、语义变量、语义规则和语义流程等语言成分，提供了在语义层组装Web服务，构建大规模复杂业务流程的能力。SPL语言设计和运行时支撑环境[19]提供了动态搜索、组合、替换和绑定语义Web服务资源的功能，可实现SPL程序的动态变换和执行。

一个SPL程序即构成了一个基于语义Web服务的语义化业务流程模型，由定义部分、程序体部分和程序的语义描述3个部分组成。数据及流程本身所蕴含的语义信息(如功能语义和接口语义)直接利用本体库中的相关概念标注和定义。此外，作为SPL设计和运行时支撑环境的基础组成部分，Web服务语义描述框架RDF4S[20]提供了Web服务的各种语义信息和使用规范，语义Web服务搜索引擎[21]能够充分利用语义Web服务中蕴含的语义信息完成服务的搜索，它们为SPL程序的开发和运行提供了全面的基础性支持。

3.2 SPL语言中的异常处理

类似于WS-BPEL采用的Fault Handler活动，SPL语言将异常处理逻辑构建在类JAVA语言的try-catch活动的层次上。但与Fault Handler活动不同的是，try-catch活动提供了一种融合异常处理规则的系统级异常处理机制，目的是在支持常规设计方法的同时，也提供一种可将正常流程逻辑和异常处理逻辑直观地分离开的可选方法，以提高设计异常处理逻辑时的灵活性。本文主要讨论基于规则独立构建异常处理逻辑的方法。try-catch语法结构如图1所示。

```
<try-catch name="NCName">
        <try>   Activity +   </try>
        <catch faultName="QName"? faultVariable ="VariableName"?
                ( faultType ="QName" | faultSType ="QName" )? >
                invoke EH_Rule Service | Activity +
        </catch>
        <finally >? invoke EH_Rule Service | Activity +  </finally >
</try-catch>
```

图1 SPL语言的异常处理活动语法结构

其中，try构造元素限定在运行时可能产生异常的活动集。catch构造元素负责捕获异常和执行处理异常的活动集。属性faultName和faultVariable分别指定异常名和参数；属性faultType和faultSType分别指定参数的数据类型为XML Schema类型或语义数据类型，两者选一。Finally构造元素说明在try-catch活动退出之前必须执行的活动，该构造元素可选。利用try-catch活动，流程开发人员能够根据特定应用需求设计出具有异常处理能力的语义流程模型。具体地，开发人员独立设计异常处理规则；在流程逻辑中定义异常处理规则服务EH_Rule Service，在catch构造元素中利用invoke调用异常处理规则服务。捕获到异常时，通过调用对应被捕获异常的异常处理规则服务触发异常处理模块中的相应规则，并指示流程引擎采取相应的措施处理异常。

4 基于规则的语义流程异常处理机制

基于规则的语义流程异常处理机制如图2所示。它实现了异常处理逻辑与语义流程逻辑的分离，在运行时根据异常触发的规则来动态调整流程的行为。整个过程可分为规则设计和规则处理两个阶段。

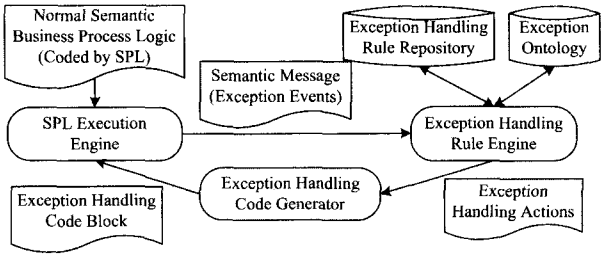

图2 基于规则的异常处理机制

第 1 阶段为异常处理规则的设计阶段。此段采用 ECA 规则作为异常处理逻辑的描述工具。根据语义流程的需求和面向语义流程的异常本体对流程在执行过程中的异常情况和异常处理动作进行分析和设计。最后获得语义流程的异常处理规则集及相应的异常处理服务集。

第 2 阶段为异常处理阶段。当流程执行过程中产生异常时，流程引擎将调用与该异常所对应的异常处理服务，并将异常事件转发给异常处理规则引擎。异常处理规则引擎根据接收到的异常事件，依据异常本体获取被捕获异常的详细信息来触发相应规则的执行，进而通知流程引擎如何处理所捕获的异常。异常处理代码生成工具根据异常处理规则引擎提供的异常处理动作产生异常处理 SPL 语言代码块，并植入到系统中，使输入的 SPL 程序具有异常处理能力，确保程序在 SPL 执行引擎中顺利运行。

4.1 面向语义流程的异常本体

面向语义流程的异常本体描述语义流程在执行过程中可能发生的异常类型及相关信息，为异常的定位、推理和异常处理规则的选择提供支持。对异常的分类包括：(1)服务调用异常。语义流程执行环境中的服务运行失败所引起的故障使得服务不可调用。服务在运行中因无法正常访问所需的数据资源和计算资源而不能够完成其对外申明的业务功能，或所完成的业务功能违背其所申明的 QoS 承诺，以及服务内部逻辑出现错误，都有可能引发服务调用故障。(2)语义流程处理异常。语义流程在执行过程中调用其他语义 Web 服务，该服务对应的描述文档在基于本体库进行解析时，发生语法解析问题，如本体解析出错或文档格式不完整等。(3)流程级执行异常，是语义流程执行过程中，由于流程模型本身的缺陷或不一致性所导致的故障，如提供了错误的输入类型，流程执行的前置条件未得到满足而无法执行，所需的输入并非由预期的一方提供等。基于上述分析，给出面向语义流程的异常本体，为语义流程的异常处理提供一个可共同理解的知识库。该本体定义了特定错误类型及相关的语义信息，如图 3 所示。

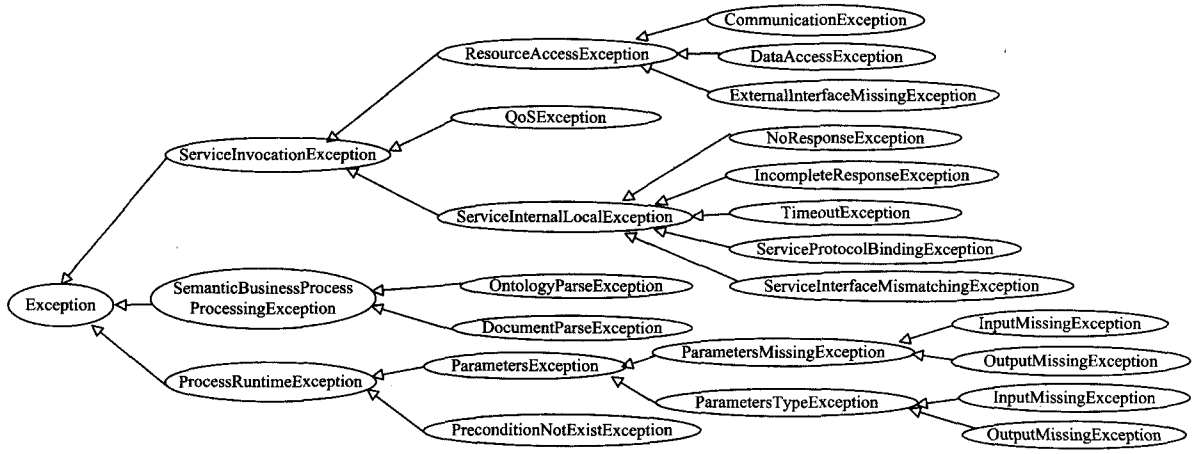

图 3 面向语义业务流程的异常本体

4.2 面向语义流程的异常处理规则

本文采用 ECA 规则来单独构建语义流程中的异常处理逻辑。异常处理规则精确地描述了异常何时发生以及发生异常时的上下文环境，并规定了语义流程应该采取的动作。规则的语法格式如图 4 所示。

```
RULE 规则名 (
        ON 异常名
        [ WHEN 布尔表达式 ]
        DO 异常处理动作
        DOMAIN 异常处理规则的作用领域
)
```

图 4 异常处理规则的语法格式

异常处理规则由事件、条件、动作和规则所作用的领域 4 个部分组成。其中，事件部分(ON)描述了一个具有限定名称的异常名，该异常所属的异常类型由本体中的一个概念指定。条件部分(WHEN)描述了发生异常时业务流程的上下文信息。当条件部分指定的布尔表达式为真时，如果事件部分的定义发生异常，则执行动作部分指定的异常处理动作，否则不需要采取任何动作。条件部分可选，缺省表示条件为真，也可以由规则设计人员自定义布尔表达式。动作部分(DO)指定了当异常发生时流程应该执行的异常处理动作。异常处理动作是语义业务流程执行过程中针对所发生的异常而采取的异常处理手段的一种抽象，包括(1)不干预(Tolerate(E))。该操作的语义是当 try-catch 中发生异常 E 时，流程可以安全地忽略异常对流程状态的影响，或者什么都不做，或者由执行引擎执行某些记账操作(如记录日志等)，并强制停止 try-catch 中所有未执行或正在执行的活动，控制流跳出当前的 try-catch 构件块。(2)重试(Retry(E,n))。该操作的语义是发生异常 E 时，流程重复执行触发该异常的业务逻辑代码块或重新调用相同的服务，直到其执行成功结束，控制流恢复正常。参数 n 表示重试次数或时间阈值。如果重试的次数或时间达到界限值时，可能需要执行 Compensate 动作以恢复某些任务的执行。(3)调整(Regulate(original,new))。该操作的语义是发生异常 E 时，调整流程的控制流，动态替换其中的一些服务或子流程，使其满足某些特定的需求。参数 original 表示需要被调整的服务或子流程，参数 new 表示可满足新要求的服务或子流程。(4)终止(Terminate(instance))。该操作的语义是发生异常 E 时，强制终止流程实例或子流程的执行。参数 instance 表示发生异常的流程实例或子流程。(5)补偿(Compensate())。该操作的语义是发生异常时，对触发异常之前已经执行完毕的服务以及嵌套在该 try-catch 中的所有内层 try-catch 进行补偿操作。

5 原型实现

语义业务流程异常处理框架结构如图5所示。基于SPL设计和运行时支撑环境进行了原型实现。

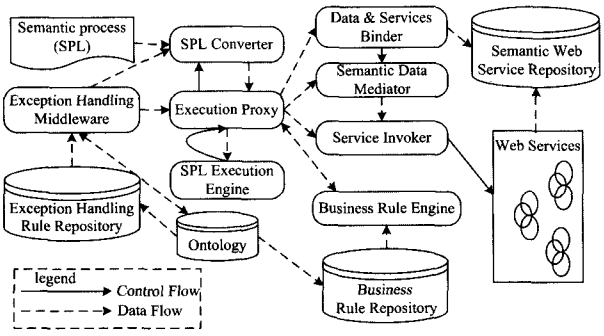

图5 原型的架构

在原型实现中，正常的语义流程逻辑使用SPL语言进行设计，异常处理逻辑则使用ECA规则描述，从而实现了关注点的分离。设计好的异常处理规则存储在异常处理规则库中。SPL Converter将SPL语言描述的语义流程逻辑变换为可在SPL Execution Engine执行的软件实体，然后利用规则变换算法将异常处理规则变换为执行引擎可执行的异常处理操作集合，并封装为对应的异常处理规则服务。Execution Proxy负责实现异常处理中间件、执行引擎和规则引擎的解耦。如果在流程执行过程中发生某种异常，则该异常将由Exception Handling Middleware解析后形成相应的异常处理动作并以消息的方式发送给Execution Proxy。Execution Proxy根据接收到的消息来协调执行引擎进行处理。若发生异常的被调用服务无法继续执行，则调用Data & Services Binder从互联网中搜索语义和功能等价的替代服务进行动态替换；Semantic Data Mediator实现语义数据到XML语法数据的适配处理过程后，最终由Service Invoker来调用替换后的具体Web服务。

6 案例应用

旅游公司利用SPL图形化编辑工具[19]设计了一个基于语义Web服务的旅游预订系统，如图6所示。其中，MemberRankIdentify服务判断客户等级，若是VIP客户则调用VIP服务供客户选择该等级相应的旅游产品并给予7折优惠；否则，就按照一般客户标准供客户选择旅游产品并给予9.5折优惠。最后在给定期限内使用银行提供的结算服务完成费用结算。这些语义Web服务所蕴含的语义信息确保了系统执行时能够准确发现满足要求的、语义和功能等价而语法异构的伙伴服务，并在执行过程中按照需要动态发现替代服务，完成动态绑定和替换。

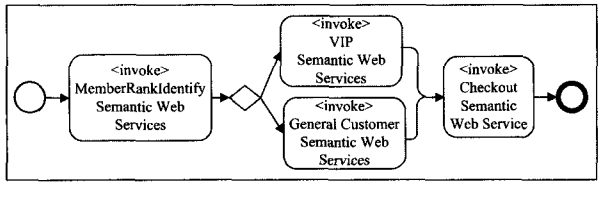

图6 旅游预定系统及相关的语义Web服务

案例服务通过Eclipse WTP工具实现，使用SPL设计和运行时支撑环境完成服务的语义标注、发布和部署。基于上述服务，我们利用本文提出的框架生成了旅游预订系统的一个容错版本。为简便起见，图7给出处理银行结算服务Checkout响应超时异常的ECA规则和部分SPL代码。

```
RULE e_Timeout {
        ON ServiceinvocationTimeout ;
        WHEN (SDuration > 60 sec);
        DO Regulate(Checkout,Payment );
        DOMAIN Financia Application ;
}
<EHruleservice name="timeoutRuleService"
            Rule="e_Timeout"
        faultStype="TimeoutException" />
<try-catch name="Trypayment">
        <try> <invoke name="Invokepayment"
            semanticService="Checkout"
            Function="pay" inputVariable="PayReq"
            outputVariable="PRes" />
        </try>
        <catch faultName="Timeout"
            faultSType="TimeoutException" >
            invoke timeoutRuleService
        </catch>
</try-catch>
```

图7 处理银行结算服务调用超时异常的规则及程序部分代码

首先定义好处理结算服务调用超时异常的规则e_Timeout及规则服务timeoutRuleService。在try-catch活动中，若调用Checkout服务60s后仍未得到响应，则所抛出的超时异常被捕获，并调用timeoutRuleService规则服务触发e_Timeout规则执行。异常处理中间件根据预定义的异常处理动作regulate，将处理超时异常的具体操作信息发送给流程引擎，由流程引擎动态搜索语义和功能等价的服务，并替换发生超时的当前服务，从而完成超时异常的处理。

7 讨论和分析

在现阶段的原型实现中，在SPL程序的变换阶段，利用规则变换算法将预定义的异常处理规则变换为包含在异常处理规则服务中的异常处理逻辑，这是一个静态的变换过程。相应的转换算法设计思想主要参考了文献[2]中所述的方法，但区别主要在于：(1)本文所述方法首先判断并确定所捕获的异常是否是异常本体中某异常类型的实例，然后才会匹配并触发具体的异常处理程序(或调用异常处理规则服务)。如果没有匹配的异常处理程序，则该异常会向上层结构传播。采用这种处理方式的原因在于，变换后的SPL程序已经具备了完整的异常处理逻辑，从而能较好地降低对SPL执行引擎进行结构调整的影响。(2)由于SPL语言有自己独特的异常处理语义活动，具体异常处理规则向SPL异常处理逻辑的转换涉及到对本体中相关概念的引用。这样的概念引用目前尚不能自动、平滑地进行，需要设计人员预先进行一定工作量的手工操作。此外，为了便于异常处理设计人员的使用，目前的异常处理ECA规则是一种语法层规则。在语义流程变换过程中，通过对规则进行预处理，根据异常名与本体中的异常类型进行匹配和推理，从而将异常类型与异常处理规则对应起来。

结束语 对于实现语义Web服务环境下业务流程执行的可靠性和一致性，异常处理方法是一种行之有效的有力手段。基于语义Web技术和WS-BPEL扩展而来的SPL语言提供了灵活而精炼的异常处理机制，却又将如何高效开发异常处理逻辑的难题留给了业务流程开发人员。而在语义流程

(下转第125页)

Design Language(AADL): An Introduction[M]. Carnegie Mellon University, 2006

[2] Flower M. 重构——改善既有代码的设计[M]. 北京: 中国电力出版社, 2003

[3] 肖汉. 基于 Java 平台的软件重用技术的研究与应用[D]. 北京: 北京邮电大学, 2005

[4] Whisnant K, Kalbarczyk Z T, Iyer R K. A system model for dynamically reconfigurable software[J]. IBM Systems Journal, 2003, 42(1)

[5] 张鹏, 姜昊, 许力, 等. Eclipse 插件开发[M]. 北京: 电子工业出版, 2008: 183-212

[6] Perry D E. Software engineering and software architecture [C]// Feng Yu-lin, ed. Proceedings of the International Conference on Software: Theory and Practice. Beijing: Electronic Industry

[7] Ronsse M, Maebe J, Bosschere K D. Software instrumentation using dynamic techniques[J]. Program Acceleration through Application and Architecture, 2002: 63-65

[8] 罗国庆, 等. VxWorks 与嵌入式软件开发[M]. 北京: 机械工业出版, 2003: 15-25

[9] 伽玛, 等. 设计模式——可复用面向对象软件的基础[M]. 李英军, 等译. 北京: 机械工业出版, 2005

(上接第 120 页)

中, 正常业务逻辑与异常处理逻辑相互混杂、交织, 给开发和维护工作带来了诸多不便。本文提出了基于 ECA 规则的语义流程异常处理机制, 构建了异常本体, 提出了 5 类异常处理动作和异常处理 ECA 规则。业务流程开发人员可以使用这些具有高层语义的 ECA 规则来定义异常处理逻辑。在运行时, 由异常处理中间件对截获的异常进行解析并指示流程执行引擎采取对应异常处理措施, 从而实现运行时异常处理。

本文方法没有考虑某些复杂的应用场景, 仅仅考虑了异常处理中间件对单一异常发生时的处理机制。进一步的工作是将研究重点放在发生异常时, 异常处理中间件生成动态植入流程执行引擎的可执行异常处理动作的生成算法上, 进一步提高应用的动态适应能力。

参 考 文 献

[1] Yang F Q, Lü J, Mei H. Technical framework for Internetware: An architecture centric approach[J]. Science in China Series F: Information Sciences, 2008, 51(6): 610-622

[2] Liu A, Li Q, Huang L S, et al. FACTS: A Framework for Fault-Tolerant Composition of Transactional Web Services[J]. IEEE Transactions on Services Computing, 2010, 3(1): 46-59

[3] Alves A, Arkin A, Askary S, et al. Web Services Business Process Execution Language Version 2.0[R]. OASIS Web Services Business Process Execution Language (WS-BPEL) TC, 2007

[4] 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

[5] Hepp M, Leymann F, Domingue J, et al. Semantic business process management: a vision towards using semantic Web services for business process management[C]// Proc. of IEEE International Conf. on e-Business Engineering (ICEBE 2005). Beijing: IEEE Computer Society Press, 2005: 535-540

[6] 曹虹华, 应时, 杜德慧, 等. 一种面向语义 Web 服务的软件设计语言和设计方法[J]. 电子学报, 2007, 35(12A): 129-135

[7] Chan K S M, Bishop J, Steyn J, et al. A fault taxonomy for Web service composition[C]// Proc. of the Int'l Workshops on Service-oriented Computing 2007. Heidelberg: Springer-Verlag, 2009: 363-375

[8] Pleisch S, Schiper A. Approaches to fault-tolerant and transactional mobile agent execution-An algorithmic view[J]. ACM Computing Surveys, 2004, 36(3): 219-262

[9] Casati F, Sayal M, Shan M-C. Developing E-Services for Composing E-Services[C]// Proc. of International Conf. on Advanced Information Systems Engineering. Heidelberg: Springer-Verlag, 2001: 171-186

[10] Luo Z, Sheth A P, Kochut K, et al. Exception Handling for Conflict Resolution in Cross-organizational Workflows[J]. Distributed and Parallel Databases, 2003, 13(3): 271-306

[11] Cardoso J, Sheth A, Miller J, et al. Quality of Service for Workflows and Web Service Process[J]. Journal of Web Semantics, 2004, 1(3): 281-308

[12] Dobson G. Using WS-BPEL to implement software fault tolerance for Web services[C]// Proc. of the 32nd EUROMICRO Conf. on Software Engineering and Advanced Applications. Los Alamitos: IEEE Computer Society Press, 2006: 126-133

[13] Zeng L, Lei H, Jeng J-J, et al. Policy-driven Exception-management for Composite Web Services[C]// Proc. 7th IEEE International Conf. on E-Commerce Technology (CEC2005). Munich: IEEE Computer Society Press, 2005: 355-363

[14] Erradi A, Maheshwari P, Tosic V. Recovery Policies for Enhancing Web Services Reliability[C]// Proc. 4th International Conference on Web Services (ICWS'06). Chicago: IEEE Computer Society Press, 2006: 189-196

[15] Chiu D K W, Li Q, Karlapalem K. A Meta Modeling Approach for Workflow Management System Supporting Exception Handling[J]. Information Systems, 1999, 24(2): 159-184

[16] Wiesner K, Vaculin R, Kollingbaum M, et al. Recovery Mechanisms for Semantic Web Services[C]// Proc. 8th IFIP WG 6.1 International Conf. on Distributed Applications and Interoperable Systems (DAIS2008). Oslo: Springer-Verlag, 2008: 100-105

[17] Vaculin R, Wiesner K, Sycara K. Exception Handling and Recovery of Semantic Web Services[C]// Proc. of 4th IEEE International Conf. on Networking and Services (ICNS 2008). Gosier: IEEE Computer Society Press, 2008: 217-222

[18] Men P, Duan Z H, Yu B. Utilizing Fuzzy Petri Net for Choreography Based Semantic Web Services Discovery[J]. Lecture Notes in Computer Science, 2007, 4546: 362-380

[19] Cao H H, Ying S, Du DH, et al. Orchestrating Semantic Web Services with Semantic Programming Language[C]// Ceballos S, ed. Proc. IEEE International Workshop on Semantic Computing and Systems (WSCS'08). 2008: 101-106

[20] 解丹, 应时, 曹虹华, 等. 基于语义的服务资源描述模型 RDF4S[J]. 武汉大学学报: 理学版, 2008, 54(1): 71-76

[21] 曾志浩, 应时, 曹虹华. 基于 RDF4S 语义服务描述模型的服务资源搜索框架[J]. 计算机科学, 2008, 41(11): 258-262