一种面向服务流程异常处理的策略描述语言

王权于[1,2] 应 时[1] 吕国斌[2] 罗俊沣[3] 文 静[1]
(武汉大学软件工程国家重点实验室 武汉 430072)[1] (中国地质大学网络教育学院 武汉 430074)[2]
(武汉大学计算机学院 武汉 430072)[3]

摘 要 策略描述语言是策略驱动的面向服务流程异常处理方法的基础和前提。针对目前已有策略语言在描述面向服务流程异常处理逻辑方面的不足,提出了一种新的面向服务流程异常处理的策略描述语言WS-Policy4BPEH。它在Web服务策略框架的基础上,扩展了ECA对规则执行影响的描述,定义了多种面向服务流程的异常处理动作模式,能够详细、准确地描述异常及异常处理方式、返回方式和传播方式。WS-Policy4BPEH采用XML作为元语言,因此具有良好的可扩展性。

关键词 面向服务流程,异常处理,策略描述语言

Policy Description Language for Exception Handling in Service-oriented Processes

WANG Quan-yu[1,2] YING Shi[1] LV Guo-bin[2] LUO Jun-feng[3] WEN Jing[1]
(State Key Lab of Software Engineering, Wuhan University, Wuhan 430072, China)[1]
(Distance Education College, China University of Geoscience, Wuhan 430074, China)[2]
(School of Computer, Wuhan University, Wuhan 430072, China)[3]

Abstract The policy description languages is the basis and premise of policy-driven exception handling for service-oriented processes. Aiming at the lack of existing policy languages representing capacity for exception handling, this paper presented WS-Policy4BPEH, a new policy description language for exception handling in service-oriented processes. This language, based on the Web services policy framework, extends the specifications of effects in ECA rules and defines a variety of action patterns supporting exception handling, which can describe in detail the exception and exception handling, and flexibly specifies the return modes and the exception propagation after exception handling. The paper adopted XML as a meta-language for specifying WS-Policy4BPEH, with a good scalability.

Keywords Service-oriented processes, Exception handling, Policy description language

1 引言

面向服务的计算(SOC)是一种以服务及服务组合为基础来构造分布式应用的新型开发模式。WS-BPEL[1]已成为服务组合描述语言的事实标准,它能够将分布在网络上的服务按照特定的业务逻辑组合在一起形成面向服务流程(BPEL流程),并在重用现有服务资源的同时,满足复杂应用的需求。BPEL流程内在的松耦合性、运行环境的动态复杂性以及Web服务的自治性,使得BPEL流程的可靠性面临严峻的挑战。针对如何提高BPEL流程可靠性这一科学问题,学术界开展了广泛而深入的研究,并且取得了诸多研究成果[2-4]。但当前在BPEL流程异常处理方面,仍存在许多亟待解决的问题,如缺乏有效的异常处理逻辑与正常业务逻辑的分离技术、缺乏异常处理逻辑的描述方法等。

策略作为控制系统行为的规则集[5],已广泛应用于安全、网络管理和分布式系统等领域。为了克服WS-BPEL内在异常处理机制的缺陷,提出了一种基于策略的面向服务流程异常处理方法,使用WS-BPEL来表示正常业务逻辑,采用策略来描述异常处理逻辑,有效地实现了关注点的分离;面向服务流程异常处理策略可以动态调整和控制流程运行发生异常时的系统行为,而不用修改代码,提高了异常处理逻辑的可重用性和可配置性,进一步增强了面向服务流程的容错能力。本方法实现的基础和前提是如何灵活地定义和配置面向服务流程的异常处理策略,也就是说,应该定义哪些语言元素来描述面向服务流程的异常处理策略。

通过设计一种新的面向服务流程异常处理的策略描述语言WS-Policy4BPEH,为基于策略的面向服务流程的异常处理逻辑的开发和维护提供灵活的语言机制。本文第2节介绍相关工作;第3节介绍面向服务流程异常处理策略模型;第4节介绍WS-Policy4BPEH语言;第5节通过一个"毕业审核流程"案例,介绍WS-Policy4BPEH语言的应用;最后总结全文。

2 相关工作

基于策略的面向服务流程异常处理正成为学术界的研究

到稿日期:2011-03-09 返修日期:2011-07-21 本文受国家自然科学基金项目(61070012),湖北省自然科学基金项目(2009CDB307)资助。
王权于(1971—),男,博士生,副教授,CCF会员,主要研究方向为SOA、语义Web服务,E-mail:wangqy@cug.edu.cn;应 时 教授,博士生导师;吕国斌 教授。

热点之一，许多研究者对此提出了相关的框架和策略描述语言。本文的工作建立在这些研究的基础之上。

Ponder 语言[6]中的职责策略定义了当指定事件(Event)发生时主体(Subject)对目标对象(Target)的反应(Action)，并能够将简单事件通过事件合成运算组合成复合事件。Q. Z. Sheng 等人[7]在 Ponder 职责策略的基础上，扩展了策略时态约束元素有效时间间隔(Effective Interval)，描述面向服务流程的异常处理策略。其直观语义为：当流程运行发生异常事件(On)时，在策略有效期内(Effective Interval)，且满足策略触发条件(When)时策略主体(Subject)对目标实体(Target)的恢复动作(Do)；恢复动作的组合可以采用 Ponder 语言提供的串行和并发操作符实现。CHIMERA-EXC 语言[8]是 WIDE 项目组在面向对象数据库语言 CHIMERA 的基础上提出的一种基于分离式 ECA 规则的工作流异常处理策略描述语言。CHIMERA-EXC 策略由事件类(Events)、触发条件类(Conditions)、动作类(Actions)和优先级类(Priorities) 4 个要素组成。事件类又分为数据操作事件类(Data Manipulation Events)、外部事件类(External Events)、工作流事件类(Workflow Events)和时态事件类(Temporal Events)；而时态事件又分为即时事件(Instant Events)、周期事件(Periodic events)和时间间隔事件(Interval events)；触发条件类(Conditions)是原子谓词公式的合取范式，定义为 WIDE 数据库状态和内容的查询表达式，用以验证规则是否需要响应异常事件；动作类(Actions)定义为工作流管理系统调用或数据操作序列；优先级类则定义了规则的执行顺序，具有相同优先级的规则的执行由其创建时间的先后决定。

Web 服务策略框架(WS-Policy)[9]提供了一个通用的描述 Web 服务系统中实体(服务、端口、操作、消息或整个服务流程)策略的模型和语法。WS-Policy 策略模型中，策略(Policy)定义为策略选择(Policy Alternative)集，而每个策略选择又由一组策略断言(Policy Assertion)组成，策略断言定义了 Web 服务实体的能力、需求和一般特性。WS-Policy 是一个轻量级的框架，只定义了描述实体需求和能力的策略语言基本构造元素集(Policy，All，ExactlyOne 和 PolicyReference 元素及 wsp:optional，wsp:ignorable 属性)，并不支持策略断言的定义。特定领域(如安全、可靠消息)定义领域断言，WS-Policy 提供一种类似于“胶水”的方法，将领域断言“粘合”在一起，形成 Web 服务策略，如 WS-SecurityPolicy，WS-ReliableMessaging-Policy 等。WS-Policy 采用 XML 作为元语言，因而具有较强的灵活性、兼容性和互操作性；WS-Policy 与已有的 Web 服务标准(如 WSDL，WS-BPEL 等)兼容，是对已有 Web 服务标准描述能力的补充。WSPL 语言[10]是 OASIS 和 SUN 公司共同制定的一种通用 Web 服务策略描述语言，支持服务双方的策略协商。WSPL 语言的语法是 XACML 标准语言的严格子集。策略集(PolicySet)表示策略的层次化组织结构，定义为一组不同层次目标(Target)的策略(Policy)；策略针对 Web 服务调用的不同目标(如 port，operation，message 等)，定义了策略可接受的规则集，策略方面(Aspect)定义为 Web 服务调用应该满足的某方面目标要求(如安全性、可靠性等)；规则(Rule)定义可接受的选择，规则之间是析取关系；规则由谓词约束(Predicate)组成，谓词约束之间为合取关系，谓词约束定义为返回布尔值的 XACML 函数，这些函数的参数定义为 XACML 的属性(Attribute)和取值(Value)；XACML 的属性定义为领域本体定义的领域词汇，而谓词约束是对属性取值的约束。WSPL 内建的谓词约束函数主要包括数值关系和集合关系运算(如大于、大于等于、小于、小于等)。WSPL 与 WS-Policy 不兼容影响到其广泛的使用，目前它主要用于安全领域。IBM 公司提出的自主计算策略语言(Autonomic Computing Policy Language，ACPL)[11]是一种基于 XML 的强类型化策略描述语言。ACPL 语言的主要元素为范围(Scope)、条件(Condition)、业务价值(Business Value)和决定(Decision)；范围定义了策略的主体(Subject，即应用领域)；条件定义策略触发的约束；业务价值定义策略的相对优先级和重要性；决定定义可观察的策略行为或预期结果。ACPL 语言完全依赖于 IBM 的自主计算策略管理中间件(PMAC)；策略主体与策略紧耦合，影响了策略的可重用性和可维护性。

WS-Policy4MASC[12]是一种基于 WS-Policy 框架的可管理和适应性的服务组合策略描述语言，定义了目标策略断言、动作策略断言、效用策略断言和元策略断言。目标策略断言(Goal Assertion)定义了正常业务操作所需达到的目标；动作策略断言(Action Assertion)定义在条件满足或不满足时需要执行的动作；效用策略断言(Utility Assertion)定义特定情况下的业务价值约束；元策略断言(Meta Assertion)定义了动作策略和效用策略冲突时的消解决策。WS-Policy4MASC 语言的研究工作为本文提供了非常好的研究基础，但 WS-Policy4MASC 只提供了基本的异常处理能力，无法评估规则执行的影响和结果，也不支持异常处理之后流程返回方式和异常传播方式的定义，WS-Policy4MASC 策略定义较为繁琐；WS-Policy4MASC 策略的执行依赖其策略执行引擎(MASC)。WSCoL 语言[13]是一种领域无关的 Web 服务客户端约束描述语言。WSCoL 变量不仅包含流程内部变量(Internal Variable)，还支持流程运行环境外部变量(Ext-ernal Variable)和流程执行的历史变量(Hist-orical Variable)。Web 服务恢复语言(Web Services Recovery Language，WSReL)在 WSCoL 语言基础上，为可编程和可扩展的 Web 服务恢复提供了一种灵活的语言机制。WSReL 恢复策略(Strategy)是一个步(Step)序列，每个步定义为有序的原子动作集；WSReL 语言提供了 4 组原子恢复动作，包括简单异常处理动作(如 Ignore，Notify，Retry，Halt 等)、监管动作(如修改监管参数动作、修改监管规则动作等)、重新绑定动作(等价活动调用和等价服务调用动作)以及其他动作(如调用外部 Web 服务动作)。WSReL 支持步的条件选择。

自适应恢复网(SARN)[14]是一种基于扩展 PETRI 网的 BPEL 流程恢复策略描述语言。SARN 扩展定义恢复变迁和恢复令牌，用以描述 BPEL 流程的异常处理逻辑，定义了更改 SARN 网结构的原子操作(如增加弧和禁用变迁等)，通过组合这些原子操作来支持不同的异常处理活动。SARN 由于采用形式化的方法，因此支持策略的分析和验证，推导能力强；但与声明式语言相比，它无法清晰地面向用户表达异常处理能力，缺乏应用层的支持。

通过对上述工作的分析，发现这些语言都采用了 ECA 规则范型，针对不同的应用领域，具有不同的描述内容。但已有语言还不能完全满足描述面向服务流程异常处理策略的需求，如异常处理之后流程返回方式和异常传播方式的定义等，

描述内容不够丰富。因此，本文提出的 WS-Policy4BPEH 语言应达到如下设计要求：1）能抽象面向服务流程的异常处理行为，语法简洁、直观；2）能充分地描述面向服务流程异常处理机制，支持异常及异常处理方式、流程返回方式及异常传播方式的描述；3）与现有 Web 服务策略标准兼容，便于扩展和容易使用。

3 面向服务流程异常处理策略模型

本节首先给出面向服务流程异常处理策略描述语言 WS-Policy4BPEH 设计时的基本思想，并在此基础之上给出面向服务流程异常处理策略描述语言的策略模型。

3.1 基本思想

基于策略的面向服务流程异常处理，将异常处理逻辑从正常业务逻辑中分离出来，遵循了软件工程中关注点分离原则。正常业务逻辑通过 WS-BPEL 来表示，而异常处理逻辑则使用策略来表达。异常处理策略是对 WS-BPEL 语言在流程异常处理描述能力的补充。

ECA 规则是一种良定义和容易理解的系统行为控制规则。面向服务流程异常处理方法将 ECA 规则与异常处理机制相结合，通过扩展 ECA 规则，将面向服务的异常处理策略定义为一组指导流程异常处理的规则集，描述流程运行时的异常行为，能够表示流程运行时引发的异常信息、异常处理方式、异常抛出方式和流程返回方式，同时提供了将异常处理策略灵活地绑定到面向服务流程不同应用层次的语言机制。

3.2 异常处理策略概念模型 BPEHM

基于上述思想，设计了面向服务流程异常处理策略模型 BPEHM（见图 1）。BPEHM 模型中，异常表示流程运行时的错误状况和非预期的行为；每条策略（Policy）只处理一类异常（Exception），一个策略定义为一个异常处理规则集，它由多个规则（Rule）组成；异常处理规则基于 ECA 规则范型，由基本属性（Attribute）、条件（Condition）、动作（Action）和后置条件（Post Condition）组成，其中基本属性给出了规则的时效范围、执行模式和优先级等基本特征；条件定义了规则执行所要满足的约束；而满足约束后要执行的处理动作由一组异常处理原子动作（Action）序列构成。目前，所支持的异常处理原子动作包括忽略（Ignore）、重试（Retry）、替换重试（Alternative）、Skip 动作（Skip）、替换（Replace）、服务并发调用（Replicate）、补偿（Compensate）、外部服务调用（CallService）。为了描述异常处理规则执行后产生的影响，引入了后置条件。条件和后置条件在面向服务的异常处理策略模型中都定义为逻辑表达式（BooleanExpression）。

为了表示异常处理逻辑的返回方式，模型中定义了流程返回方式元素（ReturnMode），该元素取值范围为{terminate, resume}。“resume”表示异常处理完成后 BPEL 流程继续其策略作用域中故障活动后面活动的执行，而“terminate”表示异常处理完成后退出该策略的作用域（如 BPEL 流程的 Scope）。异常处理结果可能成功，也可能失败。如果处理异常失败，需要重新抛出异常或引发新的异常。新的异常又触发其他异常处理策略，从而构成一条异常传播链，不仅体现了规则系统的链式执行这一特性，而且符合系统运行过程中的实际场景。策略模型提供了异常传播（Rethrows）元素用以定义异常传播方式。

每类异常可能会同时触发策略中的多条异常处理规则，这些规则可能相互冲突。为此，在模型中定义规则优先级元素（Priority）来声明式地定义规则的重要性，为规则冲突的消解提供一定的语言支持。策略绑定提供了策略与其作用范围（Domain）的关联机制，策略和策略绑定分开定义，以提高策略的可重用性和可维护性。

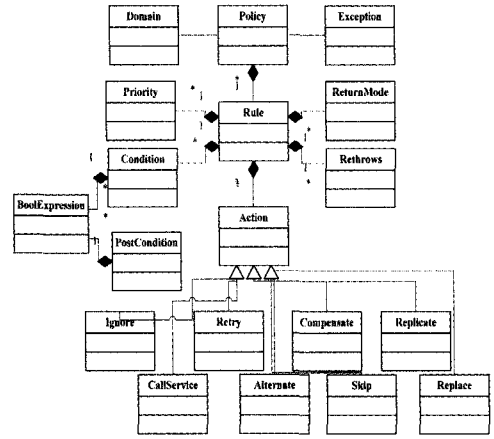

图 1 面向服务流程异常处理策略概念模型

4 WS-Policy4BPEH

本节定义了一种面向服务流程异常处理的策略描述语言 WS-Policy4BPEH。WS-Policy 作为 Web 服务策略的工业标准已经得到了广泛的支持和应用。为了保证与已有 Web 服务标准（特别是 WSDL 和 WS-BPEL）兼容，WS-Policy4BPEH 在 WS-Policy 框架的基础上，通过定义新的面向服务流程异常处理的断言类型来描述面向服务流程异常处理的策略，从而保证 WS-Policy4BPEH 与 WSDL 和 WS-BPEL 兼容，并能支持其他与 WS-Policy 兼容的断言语言（如 WS-SecurityPolicy 等）。

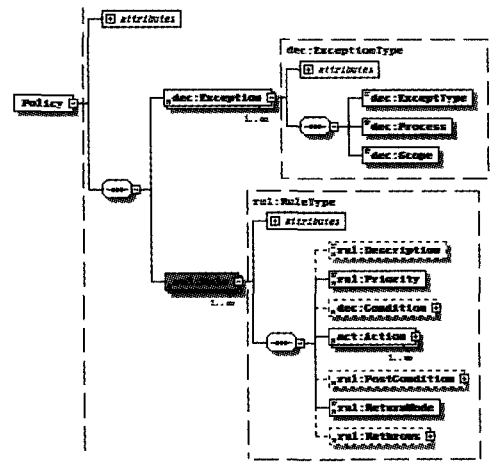

图 2 语言主要元素

根据 BPEHM 模型，WS-Policy4BPEH 语言需要描述的内容有异常、策略、规则、规则优先级、条件、动作、后置条件、流程返回方式、异常传播方式、策略作用域以及用以定义各种断言的表达式。WS-Policy4BPEH 语言以 XML 作为元语言，语法形式简单、容易使用，便于未来异常处理策略的扩展（如支持新的异常处理动作等）。WS-Policy4BPEH 中的各类主要元素（如异常、动作、规则等）都采用 XML 模式，定义为断言类型（见图 2）。下面介绍 WS-Policy4BPEH 语言的主要元素。

4.1 异常及分类

异常(Exception)是软件执行过程中引发的错误状态。根据引发面向服务流程异常的故障来源,可将异常分为资源异常(Resource Exception)、应用异常(Application Exception)和系统异常(System Exception)。资源异常是指流程执行中所需的资源发生错误而引发(raise)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,因此资源异常主要指参与 BPEL 流程的伙伴服务异常,如服务不可用、服务协议绑定错误、服务接口不匹配等,伙伴服务可以在它们的 WSDL 文档中以 fault 消息的形式给出可能发生的异常。当伙伴服务在执行过程中发生异常时,它们就返回相应的 fault 消息,这些消息可以被 WS-BPEL 中的 catch 和 catchAll 活动捕获到。应用异常是指 BPEL 流程的应用逻辑发生错误而引发的异常,WS-BPEL 提供的 throw 和 rethrow 活动可以显式地抛出异常,这些应用异常同样可以被 catch 和 catchAll 活动捕获到。系统异常包括系统内部或外部错误条件自动引发的异常和 BPEL 内在(built-in)标准异常,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处的 BPEL 引擎、中间件系统(middleware)、操作系统(OS)故障引发的异常。系统异常由引擎隐式地抛出,但这些异常能够被 catchAll 活动捕获。异常可表示为一个四元组(Quad)(ExceptionID, ExceptionType, Process, Scope),其分别表示异常标识、异常类型、发生异常的流程和异常保护域标识,定义如下:

⟨Exception⟩::= Exception(⟨ExceptionID⟩
⟨ExceptionType⟩⟨Process⟩⟨Scope⟩)
⟨ExceptionId⟩::=xsd:NCName
⟨ExceptionType⟩::=xsd:QName
⟨Process⟩::=xsd:QName
⟨Scope⟩::=xsd:QName

4.2 动作

动作是异常触发策略之后的异常处理行为,分为原子动作和组合动作。目前,WS-Policy4BPEH 支持的原子动作包括异常忽略(Ignore)、重试(Retry)、替换重试(Alternative)、SKIP、活动替换(Replace)、并发调用(Replicate)、补偿(Compensate)、调用外部服务(CallService)等原子动作。组合动作定义为多个原子动作的序列。WS-Policy4BPEH 支持动作的扩展。

⟨Action⟩::=⟨Ignore⟩|⟨Retry⟩|⟨Alternative⟩|
⟨Skip⟩|⟨Replace⟩| ⟨Replicate⟩|⟨Compensate⟩|
⟨CallService⟩

4.2.1 Ignore

Ignore 动作执行语义:当作用域 s 发生异常时,流程不需要采取任何特殊的动作,忽略该异常,继续流程的执行。从执行流的角度来看,s 之内的所有活动被强制约束,执行流转移到 s 之后的 Scope。该动作可以用来处理所有的服务资源级异常,其语法表示为:

⟨Ignore⟩::=Ignore(⟨Scope⟩)
⟨Scope⟩::=xsd:QName

4.2.2 Retry

Retry 动作执行语义:当 Scope s 发生异常时,流程重复执行 s,直到其成功结束或指定的条件满足。其语法定义如下:

⟨Retry⟩::=Retry(⟨Scope⟩⟨Count⟩⟨InDuration⟩)
⟨Scope⟩::=xsd:QName
⟨Count⟩::=xsd:integer
⟨InDuration⟩::=xsd:float

其中,Scope 表示发生异常的 Scope,Count 表示重复执行该 Scope 的次数,InterDuration 表示每两次执行之间的间隔。该动作可以用来处理服务资源异常。该动作的出发点在于某些异常只是临时性地发生,所以间隔一段时间再继续执行,该异常很有可能就不会再发生。

4.2.3 Alternative

Alternative 动作执行语义与 Retry 类似,其不同之处在于每次调用故障活动时绑定的服务并不是同一服务,而是与源服务等价的服务。其语法定义如下:

⟨Alernative⟩::=Alternative(⟨Scope⟩{Service})
⟨Scope⟩::=xsd:QName
⟨Service⟩::=xsd:QName

4.2.4 Skip

Skip 动执行语义:当某个 Scope s 发生异常时,流程将不再执行该 Scope。从执行流的角度来看,执行流会简单地跳过 s 而转移到 s 之后的 Scope。值得注意的是,s 应该是发生异常的 Scope 的后续 Scope。一般来说,该模式可用来处理服务资源中的 QoS 异常。其语法定义如下:

⟨Skip⟩::=Skip(⟨Scope⟩)
⟨Scope⟩::=xsd:QName

4.2.5 Replace

替换(Replace)动作执行语义:当源 Scope 发生异常时,流程转而调用目标 Scope,其中源 Scope 和目标 Scope 提供了相同的业务功能。该动作可用来处理流程异常,其出发点在于多个活动具有相同的业务功能。所以当一个活动不能正常工作时,流程可以使用具有相同功能的另一个活动。

⟨Replace⟩::=Replace(⟨SourceScope⟩⟨TargetScope⟩)
⟨SourceScope⟩::=xsd:QName
⟨TargetScope⟩::=xsd:QName

4.2.6 Replicate

服务并发活动(Replicate)执行语义:当 Scope 发生异常时,流程并发执行与该调用服务等价的其他多个服务。如果某个等价服务成功完成,终止其他服务调用活动,异常处理结束。该动作可用以处理服务资源异常。该模式的出发点在于某个服务具有等价功能的服务社区,所以当服务调用出现异常,调用服务社区的服务来实现异常处理。

⟨Replicate⟩::=Replicate({RepService})
⟨RepService⟩::=xsd:QName

4.2.7 Compensate

补偿动作的执行语义:当某个 Scope 发生异常时,流程对已执行的一组cope 依次进行补偿。这里发生异常的 Scope 应该嵌套在被补偿的作用域中。该动作可以用来处理所有的服务资源异常。该模式的出发点是流程的执行原子性。这种执行原子性语义在面向服务流程中是通过补偿来实现的,即对于已经执行完毕的伙伴服务,调用其提供的补偿操作在语义级别上来撤销其已做过的工作。

⟨Compensate⟩::=Compensate(⟨Scope⟩)

〈Scope〉::=xsd:QName

4.2.8 CallService

服务调用动作 CallService 执行语义：当某个 Scope 发生异常时，采用上述异常处理动作不合适，需要调用专用的 Web 服务来处理异常（如日志服务、通知服务等）。该动作可用来处理所有异常类型，其出发点在于某些情况下直接调用特殊的服务更便于异常的处理。

〈CallService〉::=CallService(〈Service〉)

〈Service〉::=xsd:QName

4.3 规则

异常处理规则基于 ECA 规则范型，规约了异常触发的策略满足特定条件时必须执行的动作。但传统 ECA 规则在规则的冲突检测和消解以及规则执行方面存在着某些限制[15]，因此扩展定义了规则动作执行的前置条件和后置条件，用以表示执行异常处理动作时需要满足的前提条件和动作对系统所产生的影响。针对目前已有策略语言在表示面向服务流程异常处理机制上的不足，扩展定义了流程返回方式来表示异常处理完成后如何返回到正常流程，扩展定义了异常传播方式来表示如何将规则无法处理的异常变换并抛出给流程客户端。异常可能会同时触发策略的多条规则，因此定义了规则优先级表示规则的重要程度。规则定义为：

〈Rule〉::=Rule(〈Priority〉[Condition]{Action}〈PostCondition〉〈ReturnMode〉〈Rethrows〉)

〈Priority〉::=〈ArithmaticExpression〉

〈Condition〉::=〈BooleanExpression〉

〈PostCondition〉::=〈BooleanExpression〉

如规则中不显示定义条件、前提条件和后置条件，则默认为“true”。

4.4 流程返回方式

异常返回方式取值为“Terminate”，表示异常处理结束后终止该作用域异常发生点的后续活动的执行；“Resume”返回方式表示异常处理结束后继续执行作用域中异常发生点的后续活动，默认为“Resume”异常返回方式。

〈ReturnMode〉::=Terminate|Resume

4.5 异常传播方式

如果异常处理失败，则以两种方式抛出异常：可以直接抛出原异常；根据需要调用异常变换服务，抛出变换后的异常。

〈ReThrows〉::=Rethrows(〈Exception〉|Transfer〈Exception〉)

其中，Transfer 表示异常抛出之前的变换服务。

4.6 策略作用域

策略作用域定义了策略应用的对象。在面向服务流程中，策略作用域主要包括流程活动（如 Invoke 活动）、Scope 和流程 3 类。为了保证策略的可重用性，在策略定义中并不定义策略作用域，而是利用 WS-AttachmentPolicy 类似的关联机制，将策略绑定到合适的作用域上。

〈Domain〉::={Scope}|〈Process〉

〈Scope〉::=xsd:QName

〈Process〉::=xsd:QName

4.7 表达式

WS-Policy4BPEH 语言支持多种数据类型、操作和表达式，提供了丰富的表达能力。数据类型包括布尔类型（如 true，false）、数值类型（如整型、浮点型、Long 型和 Double 型）、字符串类型和时间类型，并支持各种数据类型的相关操作（如数值类型的+，-，×，/等）。每种数据类型包括常量、变量和引用 3 部分，变量分为流程内部变量和外部变量。WS-Policy4BPEH 表达式包括逻辑表达式（BooleanExpression）、算术表达式（ArithmaticExpression）、字符串表达式（StringExpression）和时间表达式（TimeExpression）。逻辑表达式由常量（Constant）、变量（Variable）、逻辑操作符（BooleanOperator）、比较操作符（Comparator）和引用（Reference）按照一定的规则组成，表示策略中的各种约束和需求。算术表达式、字符串表达式和时间表达式由常量、变量、运算符和引用组成，表示策略中的相关属性（如优先级）。

5 案例应用

本节以现代远程教育的毕业审核流程为应用场景，说明本文提出的 WS-Policy4BPEH 语言的使用过程。

5.1 应用场景

毕业审核流程（见图 3）旨在为学生毕业提供方便、快捷的审核服务，以减轻管理人员的工作负担。场景描述如下：学生网上提交毕业申请；毕业审核流程接收学生毕业申请，调用“学籍审核服务”；如果学籍审核合格，则同时调用“教学审核服务”和“学费审核服务”；审核接收后，可选择打印审核结果备案；最后，流程将审核结果返回给学生。

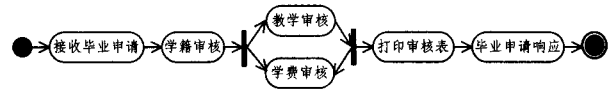

图 3 毕业审核流程

毕业审核流程调用的伙伴服务包括学籍服务（EnrollService）、教学服务（TeachingService）、学费服务（BillingService）和打印服务（PrintService）。为了简化案例应用，假设学籍服务、教学服务、学费服务和打印服务分别只包含学籍审核操作 isEnrollApproval、教学审核操作 isEnrollApproval、学费审核操作 isBillingApproval 以及毕业审核打印操作 printApproval。毕业审核流程运行时可能发生的异常如表 1 所列。

表 1 毕业审核流程异常列表

异常	异常类型	描述
failEnrollServiceFault	服务异常	学籍审核服务执行失败
invalidInputParamFault	服务异常	教学审核操作中输入参数不正确
noneBillingRecord	服务异常	没有找到该学生相应的缴费记录
unvailablePrinterFault	服务异常	打印服务不可用
nullStudentFault	应用异常	学籍审核无法根据学号找到学生信息
invalidTimeFault	应用异常	流程接收的毕业审核申请时间不在规定时间
invalidExpressionValue	系统异常	赋值活动中的非法表达式值

5.2 WS-Policy4BPEH 示例

针对上述应用场景中学籍毕业审核活动的学籍审核服务执行失败异常（failEnrollServiceFault），其相应的异常处理 WS-Policy4BPEH 策略描述如图 4 所示。

其中，第 5－9 行定义了学籍审核服务执行失败异常 failEnrollServiceFault；第 11－26 行分别定义了规则的条件和动作的前提条件和后置条件；第 28－31 行定义了调用学籍审核操作重试动作 enrollApprovalRetry；第 33－41 行定义了处理 failEnrollServiceFault 的规则；第 43 行定义了流程的返回方式“resume”；第 44－47 行定义了如果异常处理失败，则重

新抛出异常 failEnrollServiceFault。

```
<!-- WS-Policy4BPEH language example -->
<wsp:Policy wsu:Id="BPEHPolicySample">
 <!-- failEnrollServiceFault exception definition-->
 <bpeh-dee:Exception ExceptionId="failEnrollServiceException">
    <bpeh-dee:ExceptionType name = "bpeh:failEnrollServiceFault"/>
    <bpeh-dee:ProcessId name = "GraduationApprovalProcess"/>
    <bpel-dee:ScopeId name = "InvokeEnrollApprovalActivity"/>
 </bpeh-cf:Exception>
 <!-- condition definition -->
 <bpeh-dee:Condition ConditionId="InvalidEnrollApprovalDuration">
    <bpeh-exp:Variable>$enrollApprovalTime</bpeh-dee:Variable>
    <bpeh-exp:TimeComparator>Between</bpeh-exp:TimeComparator>
    <bpeh-exp:TimeConstant>2011-01-01</bpeh-exp:TimeConstant>
    <bpeh-exp:TimeConstant>2011-12-30</bpeh-exp:TimeConstant>
 </bpeh-dee:Condition>
 <bpeh-dee:Condition ConditionId="retryExecutionPrecondition">
    <bpeh-exp:Variable>$currentScopeStatus</bpeh-exp:Variable>
    <bpeh-exp:booleanComparatorTypeChoice>==</bpeh-exp:booleanComparatorTypeChoice>
    <bpeh-exp:StringConstant>Faulty</bpeh-exp:StringConstant>
 </bpeh-dee:Condition>
 <bpeh-dee:Condition ConditionId="retryExecutionPostcondition">
    <bpeh-exp:Variable>$currentScopeStatus</bpeh-exp:Variable>
    <bpeh-exp:booleanComparatorTypeChoice>==</bpeh-exp:booleanComparatorTypeChoice>
    <bpeh-exp:StringConstant>Complete</bpeh-exp:StringConstant>
 </bpeh-dee:Condition>
 <!-- primary action definition -->
 <bpeh-act:Action ActionID="enrollApprovalRetry">
    <bpeh-act:Count>2</bpeh-act:Count>
    <bpeh-act:Interval>10</bpeh-act:Interval>
 </bpeh-act:Action>
 <!--rule definition -->
 <bpeh-rul:rule RuleId="RetryRuleOfEnrollApproval">
    <bpeh-rul:priority>1</bpeh-rul:priority>
    <bpeh-rul:condition>InvalidEnrollApprovalDuration</bpeh-rul:condition>
    <bpeh-rul:actionGroup>
       <bpeh-act:Action>enrollApprovalRetry</bpeh-act>
    </bpeh-rul:actionGroup>
    <bpeh-rul:preCondition>retryExecutionPrecondition</bpeh-rul:preCondition>
    <bpeh-rul:postCondition>retryExecutionPostcondition</bpeh-rul:postCondition>
 </bpeh-rul:rule>
 <!--return mode and rethrow definition-->
 <bpeh-rt:Return>Resume</bpeh-rt:Return>
 <bpeh-rt:Rethrows>
    <bpeh-dee:Exception>failEnrollServiceException</bpeh-dee:Exception>
 </bpeh-rt:Rethrows>
</wsp:Policy>
```

图 4 WS-Policy4BPEH 示例

为了实现异常处理策略的重用，采用 WS-AttachmentPolicy 将策略和策略作用域进行绑定，将前面定义的示例策略“BPEHPolicySample”绑定到毕业审核流程的学籍审核活动中。

```
〈wsp:PolicyAttachment xmlns:wsp="…"〉
〈wsp:AppliesTo xmlns:bpeh-dee="…"〉
    〈bpeh-dee:domain〉
        /GraduationApprovalProcess/InvokeEnrollApproval
    〈/bpeh-dee:domain〉
〈/wsp:AppliesTo〉
〈wsp:PolicyReference URI="http://www.sklse.whu.edu.cn/SOAEHPL/policies#BPEHPolicySample"〉
〈/wsp:PolicyAttachment〉
```

结束语 基于策略的面向服务流程异常处理建模方法，将异常处理逻辑从正常业务逻辑中分离出来，简化了异常处理逻辑的开发和维护。本文借鉴已有的研究成果，提出面向服务流程异常处理的策略描述语言 WS-Policy4BPEH，以支持面向服务流程异常及异常处理方式、异常返回方式和异常传播方式的表示。本文介绍了 WS-Policy4BPEH 语言的元模型、语法结构；结合现代远程教育中的“毕业审核流程”应用场景，给出了 WS-Policy4BPEH 语言的策略示例。

策略部署执行之前，分析和验证策略的正确性是十分必要的，而分析验证策略正确性需要策略语言具有良定义的形式化语义。因此，下一步的工作重点是定义WS-Policy4BPEH语言的形式化语义，以支持策略的分析和验证。

参 考 文 献

[1] Business Process Execution Language for Web Services version 2.0 [EB/OL]. http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.pdf

[2] Friedrich G, Fugini M, Mussi E, et al. Exception handling for repair in service-based processes[J]. IEEE Trans. Software Eng., 2010, 36(2), 198-215

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

[4] Lerner B S, Christov S, Osterweil L J. Exception Handling Patterns for Process Modeling[J]. IEEE Trans. on Software Engineering, 2010, 36(2), 162-183

[5] Sloman M. Policy Driven Management for Distributed Systems [J]. Plenum Press Journal of Network and Systems Management, 1994, 2(4): 333-360

[6] Damianou N, Dulay N, Lupu E, et al. The ponder policy specification language [C] // The Policy Workshop. Bristol, U. K: Springer-Verlag, LNCS1995, Jan. 2001

[7] Sheng Q Z, Benatallah B, Maamar Z, et al. Configurable composition and adaptive provisioning of Web services [J]. IEEE Transactions on Services Computing, 2009, 2(1): 34-49

[8] Casati F, Ceri S, Paraboschi S, et al. Specificati-on and implementation of exceptions in workflow management systems[J]. ACM TODS, 1999, 24(3): 405-451

[9] W3C Web Services Policy Working Group. Web Services Policy (WS-Policy)1.5[EB/OL]. www.w3.org/TR/ws-policy/, Nov. 2006

[10] Anderson A H. An Introduction to the Web Services Policy Language(WSPL)[C]//IEEE, 2004: 189-192

[11] Pautasso G A C, Heinis T. Autonomic Execution of Service Compositions[M]. Proc. of ICWS. 2005

[12] Tosic V, Erradi A, Maheshwari P. WS-Policy4MASC-A WS-Policy extension used in the MASC middleware[C]//SCC'07. 2007: 458-465

[13] Baresi L, Guinea S. Self-supervising BPEL Processes[J]. IEEE Transactions on Software Engineering, 2010: 99(PrePrints)

[14] Hamadi R, Benatallah B, Medjahed B. Self-adapting Recovery Nets for Policy-driven Exception Handling in Business Processes [J]. Distributed and Parallel Databases, 2008, 23(1): 1-44

[15] Shankar C S, Ranganathan A, Campbell R. An ECA-P Policy-based Framework for Managing Ubiquitous Computing Environment[C]//Mobiquitous 2005: The Second Annual International Conference on Mobile and Ubiquitous Systems: Networks and Services. San Diego, California, July 2005

(上接第 117 页)

参 考 文 献

[1] Adiga N R, et al. Blue Gene/L Torus Interconnection Network [J]. IBM J. Research and Development, 2005, 49: 265-276

[2] 段新明. Mesh 网络耐故障虫孔路由[J]. 计算机科学, 2007, 34(11): 29-31

[3] Chalasani S, Boppana R V. Fault-tolerant wormhole routing in tori[C]//Proceedings of the 8th International Conference on Supercomputing. July 1994: 146-155

[4] Glass C J, Ni L M. The turn model for adaptive routing[C]// Proceedings of the 19th International Symposium on Computer Architecture. May 1992: 278-287

[5] Duato J. A new theory of deadlock-free adaptive routing in wormhole networks[J]. IEEE Trans. Parallel and Distributed Systems, 1995, 4(12): 1320-1331

[6] Jiang Z, Wu J, Wang D. A New Fault Information Model for Fault-Tolerant Adaptive and Minimal Routing in 3-D Meshes [J]. IEEE Trans. Reliability, 2008, 57(1): 149-162