面向服务软件异常处理研究综述

管 华[1,2] 应 时[1] 贾向阳[1] 蒋曹清[1] 王一兵[3]
(武汉大学软件工程国家重点实验室 武汉 430072)[1] (武汉工业学院网络中心 武汉 430023)[2]
(第二炮兵指挥学院三系 武汉 430012)[3]

摘 要 随着面向服务软件逐渐成为软件工程领域的研究热点,面向服务软件异常处理的研究日益受到关注,其重要性越来越突出。首先给出了面向服务软件异常的相关概念,阐述了当前面向服务软件异常的几种典型分类,介绍了几个面向服务软件的异常处理中使用的方法,并指出它们的局限性。通过分析和总结当前面向服务软件异常处理的研究方向和研究现状,提出了要解决的关键问题,探讨了当前面向服务软件异常处理研究的不足之处,并预测了下一步面向服务软件异常处理的主要研究方向。

关键词 面向服务软件,异常处理,综述
中图法分类号 TP311 **文献标识码** A

Overview on Exception Handling in Service Oriented Software

GUAN Hua[1,2] YING Shi[1] JIA Xiang-yang[1] WANG Yi-bing[3]
(The State Key Lab of Software Engineering, Wuhan University, Wuhan 430072, China)[1]
(Network Center, Wuhan Polytechnic University, Wuhan 430023, China)[2]
(No. 3 Dept., The Second Artillery Command Coll, Wuhan 430012, China)[3]

Abstract As service oriented software is becoming a hot research field of software engineering, the reseach of exception handling in service oriented software (EHSOS) gets more and more attentions and its importance is becoming more prominent. We first defined some EHSOS related concepts, described several typical classifications, introduced several methods used in EHSOS and pointed out their limitations. Through the analysis and summary of current research fields and status quo of EHSOS, we proposed the key issues to be addressed, discussed the inadequacies of current research, and predicted the next main research tendency of EHSOS.

Keywords Service oriented software, Exception handling, Overview

1 引言

面向服务是一种新兴的软件开发范型,使用这种范型开发的软件系统以因特网上分布的各种服务为基本构成单元,通过静态或者动态地发现并组合服务来完成系统的业务功能。Web服务所处的网络环境是一个异构的、分布式自治和快速变化发展的动态环境。网络环境的异构性、分布式自治等特性决定了服务组合在执行过程中可能会受到通信模式的变化、网络失效、服务拒绝攻击、基础设施失效等问题的影响[1]。组件服务的自治性、平台独立性以及自身的更新与升级都可能导致原有服务操作失效,甚至出现服务不可用,这都将影响服务组合的正常运行,还可能导致不良甚至是灾难性的后果,软件的可用性和可信性面临着更严峻的挑战。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系统集成,形成更大规模、更复杂的大型软件系统,面向服务软件的处理逻辑会越来越复杂,异常发生的频率和异常处理的复杂程度也会越来越高,这类软件系统应当处理的异常及其组合情况也更加复杂。

自John. Goodenough在1975年第一次提出异常处理的概念以来[2],异常处理作为一种有效的软件容错机制一直得到学术界和工业界的广泛研究和应用。面向服务的异常处理研究逐渐成为研究工作的热点。异常处理是软件工程研究领域的重要研究内容。对于一个高可信的面向服务软件系统来讲,健全的异常处理机制必不可少。近年来异常处理得到了国内外学术界和工业界的广泛关注,IEEE Transaction早在2000年就将异常处理专题连续刊登在第9期和第10期专刊上,2009年再次就此专题重新征稿,FSE在2008针对该主题

到稿日期:2012-06-29 返修日期:2012-10-18 本文受国家自然科学基金面上项目(61070012,61170022),国家自然科学基金重点项目(61272113,61272108,91118003)资助。

管 华(1978—),男,博士生,主要研究方向为面向服务的软件开发、形式化方法,E-mail:gh@whpu.edu.cn;应 时(1965—),男,博士,教授,博士生导师,主要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;贾向阳(1972—),男,博士,讲师,主要研究方向为面向服务的软件开发、面向服务的中间件;蒋曹清(1973—),男,博士生,讲师,主要研究方向为形式化方法、软件程序分析、软件体系结构。

举办了一期国际会议。ICSE(International Conference on Software Engineering)等顶级会议和 TSE(IEEE Transactions on Software Engineering)等顶级期刊出现了越来越多的关于异常处理的优秀研究成果。TSE 2010 年第 2 期出版了一期异常处理专刊,表明异常处理是目前软件工程的研究热点。目前大部分的研究成果仍然集中在程序设计语言、工作流的异常处理等方面。

目前的面向服务软件开发语言和相关规范,都在异常处理方面给予了很多关注。如在 WSDL 协议中,允许开发人员定义服务接口的错误消息及其格式;在 SOAP 协议中,定义了异常消息的语法规范;在 OASIS 制定的 WS-BPEL 语言中定义了 FaultHandler、Throw 等一系列支持异常处理的语言成份等;WS-CDL 提供了〈informationType〉、〈variable〉、〈WorkUnit〉、〈exceptionBlock〉和〈FinalizerBlock〉等处理服务编排实例运行时异常的语言成分。这些语言设施可以有效地支持面向服务软件异常处理的编码和测试,以提高软件可信性。

本文第 2 节给出了面向服务软件的异常处理的相关概念;第 3 节结合面向服务软件及 Web 服务环境的特点对异常进行分类;第 4 节对当前面向服务软件的异常处理中使用的方法学进行分类,并指出它们的局限性;第 5 节针对面向服务软件的异常处理的研究方向和研究现状进行分析;最后展望未来的工作,分析了当前面向服务软件的异常处理下一步研究的主要方向。

2 面向服务软件异常处理的几个基本概念

从宏观角度来看,影响一个面向服务的软件系统正常执行的因素包括缺陷(Defect)、故障(Fault)、错误(Error)和失效(Failure)[3]。本文所用术语借鉴并遵循国标 GB/T11457-2006(信息技术软件工程术语)[4]和 IEEE Std 610.12-1990 (IEEE Standard Glossary of Software Engineering Terminology)[5]。其中,缺陷是指静态地隐藏在软件系统内部的那些不被期望出现或不可接受的偏差,缺陷的存在将导致软件系统运行到某个特定阶段或条件时出现故障。故障是指软件系统在运行过程中未遵循预期的正确规约而导致的一种不被期望出现或不可接受的内部状态,是软件系统的一种运行时行为。错误是指软件系统的外部状态与假定的或理论上正确的外部状态产生偏差,是偏离了可接受的代码行为的一个动作或一个实例。失效是指软件系统未按预定的性能要求和服务规约执行其被要求的功能,因而产生了不被用户期望或接受的外部行为结果和错误服务。软件的失效是因为软件执行了含有缺陷的程序而引起的软件的故障模式/原因。容错(Fault Tolerance)是系统发生故障(Fault)时,避免系统失效(Failure)的有效机制。用"异常"表示由"故障"、"错误"、"失效"及特殊情况所引发的不正确或不寻常的事件,且仅在容易引起混淆的情况下对这些术语进行严格区分。Avizienis 分别对异常(Exception)给出了多种不同的定义[6]。目前为大多数人所接受的定义是:异常是在程序的执行过程中检测到的不正常事件,它使程序不能继续沿着正常的路径执行。作者认为异常是软件执行过程中遇到的错误状况和未预料到的行为。

面向服务软件的异常是指由于基础运行平台异常、服务异常以及流程异常等引起的服务执行失败或者流程执行失败,导致面向服务软件无法正常运行完成预定义的业务流程的情况。在面向服务软件中要区分流程层和服务层的异常。其中服务层异常是指服务协议绑定错误、服务描述错误、服务接口不匹配、服务输入输出消息违反约束条件等造成服务资源不可用的异常。流程异常是指流程实例中的相关数据违反了约束条件、无法获取资源以及活动执行时发生的执行失败、操作错误等异常。由于 BPEL 语言在服务组合使用的广泛性,因此经常用到 BPEL 流程异常的概念,BPEL 流程异常是由于 BPEL 流程、伙伴服务和运行环境等故障而导致 BPEL 流程行为偏离了正常执行路径,是 BPEL 流程运行时引发的故障消息,也是 BPEL 流程运行时行为与设计时的正常行为的偏移。

3 面向服务软件中的异常分类

传统的异常处理技术无法处理由于 Web 服务具有松耦合、自治和异构等特点抛出的各类异常。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。在 Web 服务故障分类体系方面,具有代表性的工作如下。

德国洪得堡大学的 Stefan Bruning 等人从体系结构角度将 SOA 异常分类为:服务发布异常、服务发现异常、服务组合异常、服务绑定异常和服务执行异常[7]。Lau 等人提出 Web 服务异常,包括基础设施异常、流程异常、流程定义异常和应用异常 4 类[8]。澳大利亚阿德莱德大学的 Quan Z. Sheng 等人将异常分为用户异常、组件服务异常和社区异常 3 类[9]。Nazaraf Shah 等人在其提出的 Web 服务异常诊断框架中,从多 Agent 系统的角度,将异常分为环境层异常、知识层异常和社会层异常;又从 Web 服务栈的角度,将异常分为连接栈异常、描述栈异常和发现栈异常[10]。Fugini 等人根据异常影响到的系统组件的级别,将异常划分为 3 大类:基础设施及中间件级异常、Web 服务级异常和 Web 应用级异常。3 大类包含 6 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[11]。

Maamar 和 Benslimane 认为 Web 服务组装时,可能在资源级、语义级、组合级和组件级 4 个层次抛出 3 类异常:工作项失效、时限终止和资源不可用[12]。Zeng 等人将组合 Web 服务执行期间发生的异常划分为应用异常和流程定义异常[13]。Steyn 依据引发失效的原因,将服务组合中的失效划分为以下几类:不正确组合引发的失效、不正确的并行执行引发的部分失效、不一致性引发的失效、歧义性输出引发的失效、依赖性引发的失效、并发性引发的失效、可用性引发的失效[14]。Wang 和 Bandara 等人通过对动态服务组合过程中产生的业务约束冲突和运行时环境异常的研究,将异常划分为 4 类:QoS 故障、功能上下文故障、领域上下文异常和平台上下文故障[15]。在语义 Web 服务方面,Vaculin 和 Sycaya 等学者在研究 OWL-S 描述的语义 Web 服务中的异常处理问题时,将异常划分为服务调用错误、OWL-S 文档处理错误、流程级执行错误、应用级错误和约束冲突 5 类异常[16]。刘安将伙

伴服务异常划分为逻辑异常、系统异常、内容异常和服务层协议异常4类[17]。

从以上综述可发现，当前对异常进行分类的依据和分类角度多样化，缺乏统一的分类标准和表示方式，无法呈现较为完整的异常知识体系。不同分类之间或多或少存在概念及术语的混乱和重叠现象，模糊或混淆了异常之间的关联关系，不利于异常知识的共享、重用和推理。而且对异常及其属性缺乏深层次的表示和描述。但总的来说，面向服务软件的异常可以从服务层和流程层两个层次进行分类，每个层次从系统异常、资源异常和应用异常3个维度进行划分。

4 面向服务软件中的异常处理方法分类

根据异常处理技术特点的演化和面向服务软件的应用领域，可以简单地将当前主流的异常处理方法划分为基于注释的异常处理方法、基于运行时异常处理方法、事务型异常处理方法，ECA(Event-Condition-Action)规则法、基于人工智能理论的异常处理方法(案例推理法)和其它异常处理法6种。下面分别介绍这6种处理方法。

4.1 基于注释的异常处理方法(静态方法)

基于注释的异常处理方法中具有代表性的方法有S. Modaferi和Conforti等人提出的一种基于注释的WS-BPEL服务流程异常处理方法。该方法首先在正常业务流程的WS-BPEL代码中加入预定义的注释来声明式地描述服务流程的异常处理逻辑，接着在业务流程被执行前对这些注释进行解释[18]。基于注释的异常处理方法的优点是不需要对WS-BPEL引擎做任何修改，这种方法目前研究比较少，缺乏实用的成果。

4.2 基于运行时异常处理方法(动态方法)

在应用设计时预见所有的异常是很困难的，因此在应用的运行时处理发生的异常就显得非常重要。Simmonds建立了一个工业级的在线监控框架。在运行时监控Web服务会话的执行，使用状态机进行形式化描述[19]。Barbon F.等提出对BPEL执行引擎进行扩展，添加了一般形式的Java代码监控模块来捕获服务执行信息[20]。Raimondi提出了如何应用时间连续时序逻辑(TCTL)和时间自动机的形式化来支持对Web服务的SLA描述的时效性约束的有效监控[21]。Hai L提出了一种捕获Web服务会话过程中的外层异常的方法，并提出了一个事件驱动机制来将捕获的异常上下文合并到组合Web服务异常处理中[22]。

基于运行时异常处理方法主要是基于监控实施方法，需要对原有的组合服务描述语言(如BPEL，OWL-S等)进行扩展，需要定义新的扩展标记，并把新标记映射到新定义的监控服务上，通过直接添加程序代码形成监控引擎，这虽然实现了服务核心逻辑与监控逻辑的分离，但需要一个中间者来协调。在这些服务监控体制中，监控方面往往比较固定。

4.3 面向事务型异常处理方法

事务型异常处理方法根据面向服务软件系统和执行环境的特点，对传统的事务模型进行扩展，建立了高级事务模型(ATM)，放宽了事务的原子性、孤立性，以保证应用的一致性。学者们在ACID的基础上，通过其可补偿特性和可重试特性使系统退回到初始状态或得出符合逻辑的结果。代表性研究工作有：Dayal提出了基于ECA规则来摆脱严格的补偿策略问题，实现比较灵活的异常处理机制[23]。Ardagna等人提出了流程和自适应Web服务的PAWS框架，其通过简单的恢复动作集，来支持运行时选择最合适的伙伴服务，恢复设计时定义，实现了关注点分离，但不支持运行时恢复，也不支持运行时根据上下文选择恢复策略[24]。Charfi等人提出了一种基于AOP的Web事务的向后恢复解决方案，即采用故障处理器来回滚无法关闭的事务，将策略静态编码恢复到BPEL流程中[25]。Rachid Hamadi提出了BPEL流程自恢复网SARN，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[26]。意大利米兰理工大学L. Baresi等人提出了一种BPEL流程自愈解决方案Dynamo。Dynamo是基于断言的中间件，采用WSCol语言和WSRel语言分别定义了BPEL流程的监控和恢复能力[27]。国内王海洋提出了一种基于补偿业务生成图的组合服务异常处理方法[28]，其基于业务流程中任务间的补偿依赖关系，设计了补偿业务生成图的自动生成算法和组合服务执行过程中的异常处理算法，并通过仿真实验验证了异常发生时组合服务执行过程的可靠性和一致性。

基于事务法实际上是把异常作为一个事务处理，当异常发生时，异常处理程序与流程引擎顺利交接，从而保证了流程发生变化时执行的持续性；事务型异常处理方法在解决异常处理问题的同时，也存在一定的不足。如服务补偿的方法虽然充分考虑了业务逻辑和业务规则的完整性，较好地保持了业务的完备性，但服务替换基于静态语义，对运行时的动态语义支持不足，事务化技术的负载型过高等。

4.4 基于规则的异常处理方法

为了克服事务处理可能引发的设计僵化的问题，研究者提出了通过配置和重配置来提高面向服务软件系统适应性从而完成异常处理的方法。这个方面的研究热点集中在如何采用ECA规则来分离和配置异常处理过程和应用逻辑，以提高面向服务软件系统的适应性和异常处理过程的适应性。代表性工作有：K. Kim等人提出了一种基于规则的BPEL流程主动异常处理方法[29]。D. Karastoyanova等人基于BPEL流程事件模型、发布/订阅模式、AOP技术和WS-Policy框架，提出了一种面向方面的BPEL流程异常处理框架[30]。Casati提出chrimera-Exe语言描述异常规则，以提高WfMS检测和处理可预见异常的能力[31]。Francisco Javier Díez提出了一种结合使用OPC XML-DA和规则引擎的Web服务动态异常处理方法，并将该方法成功地应用于基于Web服务的工业自动监控软件中[32]。Sheng Quan和Boualem Benatallah等人提出的SELF-SERV平台中也采用了可配置的异常处理策略，并基于预定义的异常处理策略处理对等网络环境下的运行时Web服务异常[9]。国内刘海等人提出了一种基于描述逻辑的BPEL流程异常处理规则描述方法[33]。

ECA规则通过“计算与配置分离”的方式处理问题，通过配置/重配置来提高应用的适应性从而完成异常的处理是一个很好的方法，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。

4.5 基于人工智能理论的异常处理方法

为了增强异常处理过程的自动化程度，基于人工智能理

论的异常处理方法也受到研究者的密切关注。这一方面的代表性方法包括基于案例推理的异常处理方法、基于知识的异常处理方法等。通过计算发生的异常和知识库中的异常处理模式之间的相似度来决定处理发生的异常的动作，重点强调了通过用户的参与来定义新的异常处理方法，以达到扩充知识库的目的。国内韩燕波研究员提出了一种基于案例匹配的异常处理方法 ASEED。该方法对已处理完毕的异常进行了信息抽取，将抽取出的异常上下文及相应处理方法等信息作为案例形成案例库；依据服务自身描述与用户需求的比较结果，动态检测服务可用性所引发的异常，并通过案例匹配辅助处理服务不可用异常[34]。此类研究在实际应用中较难得到推广，因为相关案例和知识的积累、分析和抽取往往耗时较长且非常困难。

4.6 其它异常处理方法

Web 服务容错技术方面的代表性工作还有：Jorge Salas 等人提出的 WS-Replication 框架，该框架用于解决 Web 服务可用性问题。WS-Replication 允许一个 Web 服务在一个站点集上进行多次部署，并允许使用组播技术与 Web 服务的多个副本交互，增强了该 Web 服务的可用性[35]。Lingesh Sagara 等人给出了 Web 服务编排的语义模型，并在此基础上提出了一种基于 UPPAAL 语言的 Web 服务交互时异常处理器的转换规则[36]。O. Ezenwoye 等人针对 BPEL 流程的伙伴服务异常，提出了一种基于代理的 BPEL 流程异常处理方法，即利用等价服务来取代故障服务，从而实现由于伙伴服务故障而引发的 BPEL 流程异常[37]。Giuliana Teixeira Santos 等人提出了一种 Web 服务容错基础设施。该设施提供了一个用于客户端和服务方交互的代理，代理采用主动容错技术来实现客户端的透明容错[38]。徐伟等人提出了一种基于移动 Agent 的复合 Web 服务容错模型——MAFTM 模型，它从系统层次而非标准层来解决复合 Web 服务的容错问题。但是该模型没有提及如何有效地将 MAFTM 与复杂控制结构相结合[39]。Fu-Chiung Cheng 等人在分析 Web 服务环境下异常产生原因的基础上，提出了一种三层（节点层、流程层和全局层）Web 服务异常处理机制，该机制在服务集成工具中得到了成功实现，提高了服务执行的可靠性和准确性[40]。北京大学裘宗燕教授提出了适用于分布式环境和 Web 服务环境的协同异常处理方法 CEH。通过定义异常处理的规则给出了服务协同完成异常处理的过程，并基于 PPL 语言给出了该过程的操作语义[41]。

4.7 面向服务的异常处理方法小结

上述的几种面向服务的异常处理方法中，都存在一定的片面性。基于注释的异常处理方法在流程中增加大量的注释也是一件比较繁杂并且容易引发异常的工作。基于运行时异常处理方法改变了服务流程，对其执行效率影响较大，对监控性能也有所影响，在检测服务属性时，算法又比较依赖于被检测的监控方面。事务型异常处理方法的研究主要集中在业务流程的设计阶段，没有解决异常如何分析和监测的问题，并不能保证执行阶段的一致性。ECA 规则法包括异常的分析和监测和处理，适用范围最广，伸缩性也最强。该方法是基于 ECA 规则工作流建模方法的推广，可以对不同的异常提供不同的灵活处理方法，但未考虑异常传播、流程活动的事务特征和组织层次协调等问题。而采用预定义的补偿方法可能造成业务逻辑的变更。案例匹配法要求建立一套异常知识库，通过知识库能够比较精确地定位异常，但实现的难度较大，对于不可预测异常，则无法处理。基于 Web 服务规范的方法目前只是提供了事务处理和协调机制的协议，可用性和实践性尚在测试阶段。

但是以上几种处理方法面对不可预测异常则显得力不从心，几乎不可能预测和处理各种各样的处理层异常，而且无法广泛应用于面向服务的软件系统，因为越来越多的处理层次使 SOA 变得更加复杂。

5 面向服务软件异常处理的主要研究方向和研究现状分析

5.1 面向服务软件异常处理的框架和工具

近年来国外许多研究机构提出了面向服务软件系统的异常处理框架，取得了不少有价值的研究成果。Kareliotis Christos 提出了一种面向 Web 服务的自动异常处理框架（SRRF）及其实现方案，即通过简单地替换失效 Web 服务来处理部分异常[42]。Abdelkarim Erradi 基于可管理和可适应的服务组合策略语言 WS-Policy4MASC，提出了一种策略驱动的可管理和自适应的服务组合中间件 MASC，MASC 利用 SOAP 消息层和业务流程编制层的协同管理，实现了跨层的服务流程异常处理[43]。Oliver Moser 提出一种对 BPEL 过程的非侵入的监控与服务调整的框架（VieDAME），它允许根据 QoS 属性和现存基于各种置换策略的合作服务的置换来监控 BPEL 流程。通过截取 SOAP 消息，服务能在运行时自动地被置换，而不需要整个系统停机[44]。Zeng 提出了一个策略驱动的异常管理框架来处理 WS-BPEL 业务流程中的异常。该框架定义了几种常见的异常处理策略，允许开发者声明式地指定异常处理策略[13]。国内学术界也开始涉足面向服务软件系统异常处理的研究工作。李青等人提出了一种事务性 Web 服务的容错组合框架 FACTS，该框架提供了一种融合异常处理和事务技术的混合型容错机制 EXTRA，并给出了异常处理逻辑正确性的验证算法[45]。陈俊亮针对 BPEL 语言提出了一种多策略的异常处理系统（MPEHS）[46]。该方法通过多种策略在不同的 BPEL 过程中的重用，减小了开发的工作量。

国内外对异常及异常处理机制的研究大多集中于某种特定的异常处理策略，可配置、可扩展的异常处理框架的研究比较少。需要开发支持多层次和多级别面向服务软件异常处理的开发框架、异常处理策略及其配置语言，但基于框架和策略的面向服务软件异常处理逻辑的开发方法较少。如何全面、快速、高效地为新开发的或者已有的面向服务软件添加异常处理逻辑，以提高面向服务软件系统的可靠性，依然是面向服务软件系统的异常处理框架和工具开发所面临的主要问题。

5.2 面向服务软件异常处理的形式化建模与验证

学术界在研究面向服务软件的异常处理时，希望能充分利用形式化描述模型和描述语言提供形式化语义，通过形式化的推理实现对面向服务软件的异常处理的性质验证，用形式化模型指导实际应用。研究面向服务软件中的异常处理逻辑的建模和形式化分析验证方法，构造具有容错能力的面向服务软件设计模型，并在设计阶段保障面向服务软件的可信性，以提高面向服务软件的异常处理的有效性和可靠性，具有

重要的学术意义和应用价值。学术界提出了部分的形式化描述和分析验证方法，根据形式化描述技术的不同分为：基于自动机的建模与验证、基于进程代数的建模与验证、基于 Petri 网的建模与验证、基于 CSP 的建模与验证 4 种。

采用自动机描述面向服务软件的异常处理机制的代表性研究有：Yuhong Yan 等人提出了一种 Web 服务编制的建模和诊断方法，该方法将 BPEL 流程转换成同步自动机模型，当异常抛出时，基于该形式化模型计算流程执行的轨迹，并通过轨迹上的变量依赖性诊断异常的发生[47]。基于进程代数的异常处理方法的代表性研究有：Marco Carbone 提出了 Web 服务编排时交互异常的处理方法[48]，其通过协同多个参与交互的 Web 服务来处理交互过程中产生的异常，并且基于 global 演算给出了该异常处理机制的形式化语义。Ferrara 利用进程代数 LOTOS 及其支撑工具，提出了一个设计和验证 Web 服务组合的框架。该框架规定了如何在基于进程代数的抽象规约和可执行的 BPEL4WS 服务组合规约之间进行双向映射，并考虑到补偿机制、事件和错误处理机制[49]。基于 Petri 网的方法的代表性研究有：Karolina Zurowska 等针对服务组合中的错误和失效问题，提出了一个新的使用着色 Petri 网的模型驱动的方法。它显示如何建模 CWS 和与其它服务的交互，模型如何分析使得 CWS 更健壮[50]。国内范贵生针对服务组合的不同故障情况给出了相应的故障处理方法[51]。段振华提出了扩展着色 Petri 网的模型检测方法，并将扩展后的着色 Petri 网模型检测方法及工具应用在 Web 服务组合的验证问题中，它能够验证 Web 服务组合是否存在逻辑错误[52]。基于 CSP 方法的代表性研究有：北京大学的裘宗燕提出了一种保障协同异常处理行为正确性的系统开发方法。该方法使用基于 CSP 的 PPL 语言形式化协同异常处理算法。还基于 Web 服务编排的思路，提出了一种“全局对局部”的开发方法[41]。

综合当前的研究情况来看，从异常处理逻辑的形式化描述方法来看，自动机方法描述面向服务软件的异常处理的能力有限，Petri 网主要用于验证组合方案的有效性和可行性，可以直观准确地描述各种异常处理行为，但难以处理面向服务软件的异常处理的动态特性。进程代数对动态演化系统的灵活描述使它适合刻画体系结构的异常处理的动态行为，但缺乏直观的图形化支持。基于 CSP 描述面向服务软件中的异常处理行为时，其形式化模型较难理解，建模过程较为复杂。

5.3 面向服务软件体系结构的异常处理研究

围绕面向服务软件体系结构的异常处理研究主要涉及容错软件体系结构模型和支持容错机制建模的软件体系结构描述语言两个方面的工作。支持异常处理的软件体系结构建模方法方面代表性的工作有：Rogério de Lemos 等人提出了在体系结构层改进面向服务的软件系统可靠性的方法，该方法使用一种体系结构模式来提高系统的可靠性[53]。Ferda Tartanoglu 等人通过研究前向错误恢复机制和反向错误恢复机制在 Web 服务组件进程和分布式事务处理协议中的应用，解决了 Web 服务体系结构在支持分布式系统方面不足的问题[54]。Ken Birman 等人扩展了 Web 服务体系结构，保证即使错误中断了系统某些结点，应用程序也能保持可操作性[55]。Diego Zuquim Guimarães Garcia 等人提出的一种面向服务的容错软件体系结构，该体系结构通过扩展 Web 服务标准来实现容错[56]。国内学术界也开展了面向服务软件系统异常处理的研究工作，梅宏等人提出一种基于运行时软件体系结构的容错管理方法，以支持开发者和管理员针对不同中间件服务的失效问题定制合适的故障检测和修复机制[57]。麻志毅等人提出了一种面向服务软件体系结构参考模型和服务执行异常的处理方法[58]。陈振邦等通过对已有 Web 服务接口模型进行扩展，提出了通过错误处理和补偿支持事务信息描述的接口模型[59]。

国内外的许多研究机构虽然在软件体系结构层异常处理方面取得了一些有价值的研究成果，但在面向服务软件体系结构的异常及其处理机制方面的研究工作很少，且支持异常及其处理逻辑建模的软件体系结构描述语言、软件体系结构设计方法以及支撑工具的研究不足。

5.4 WS-Diamond 研究项目及基于服务的过程的自愈方法

WS-Diamond(Web-Service Diagnosability, Monitoring & Diagnosis)项目由欧盟委员会支助并成为 FET(未来新兴技术)的开放的架构框架[60]，其发展目标是自我修复的 Web 服务框架，即服务能够自我监控、检测，自我诊断出现故障的原因，并具有从失败中自我恢复的功能。架构能为每一个故障关联诊断并提供应用服务(例如，数据收集意见、服务之间的交换)，执行恢复和修复操作。该架构还包括监测服务、解决 QoS 问题的修复模块，支持对于服务质量、支付时间和时间约束的监测、异常情况的监测，检测和诊断功能性或非功能性错误(服务质量)，执行修复/重新配置，保证 Web 服务的可靠性和可用性；服务设计的方法和工具保证在执行过程中的有效和高效的诊断性/可修理。另一个目标是设计有效和高效的服务设计方法和工具，其可以在执行过程中用于诊断和恢复，支持复杂的自我愈合过程设计。它于 2005 年 9 月开始，其研究成员包括欧洲的一些知名大学、如法国图卢兹大学、米兰理工大学、奥地利卡拉根福大学等。典型的研究成果有：奥地利卡拉根福大学的 G. Friedrich 等人提出了一种基于服务的过程的自愈方法，该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；基于自愈理论，将服务流程的异常处理问题转化为智能规划问题，根据服务流程的结构约束、流程活动间的数据依赖关系和定义的修复动作，动态生成运行时服务流程修复计划，从而实现对服务流程的异常处理；开发了保证服务在流程运行时可诊断和可修复的一套方法和工具[61]。

5.5 语义 Web 服务的异常处理研究

在面向语义 Web 服务的语言方面，OWL-S, WSMO, SWSF 等语义 Web 服务标准并没有为语义 Web 服务提供充分的异常处理和恢复的相应支持。OWL-S 语言是国内外学者推崇的语言之一，但由于其未定义异常处理相关的操作，而仅仅间接地为异常的捕获提供了一定支持，因此异常处理能力尚待进一步发掘和改进。WSMO 和 SWSF 虽然将异常信息建模到本体中，但同样也未对异常的处理提供直接的支持。相对于语言层次的异常处理问题，面向语义 Web 服务的异常处理机制更多地集中在如何利用本体建模异常知识，以及如何利用语义 Web 服务的推理技术和动态替换技术来实施具

体的异常处理策略。代表性的研究有：Sycara 等人围绕着 OWL-S 的异常处理问题，定义了事件本体，以建模包括异常在内的事件知识，然后对 OWL-S 进行扩展，增加了用于处理异常的原语动作；提出了一种基于事件的异常监控方法，即通过构造可触发相应构造元素的事件表达式来充当构造元素的触发条件，进而实现异常的捕获和处理。Roman Vaculín 等人提出了一种基于 OWL-S 的 Web 服务的异常处理及恢复方法。该方法使用 BPEL 标准的错误处理器及其补偿机制来处理长期运行的业务事务，结合使用两种处理器及语义监控技术来处理各种 Web 服务环境下的异常[62]。LSDIS Lab 的 METERO-S 项目利用 OWL-S 本体描述 Web 服务的语义，使用基于服务动态发现、选择和替换的动态配置方法来处理运行时异常，因而具有前向异常处理能力。但该方法使用预置的异常处理方案，因而灵活性不足[63]。其它代表性的研究有：Jelena Zdravkovic 和 Vandana Kabilan 提出了一种异常处理流程本体(EHPO)来建模业务流程中的各种异常信息和处理模式，以此为基础设计具有异常处理能力的可执行流程[64]。Stein 等人在研究服务的提供方法中，提出一种同时具备主动式和反应式的故障处理方法。该方法利用语义匹配技术识别和选择功能相近的候选服务，为故障频发的服务提供冗余[65]。赵楷提出一种基于语义 Web 服务的语义流程异常处理机制[66]。

总体来看，目前基于语义 Web 技术的异常处理相关研究工作相对缺乏，其影响主要局限于学术界语义 Web 研究群体中，关键性的研究成果主要还停留在学术论证阶段，真正在实际的工程项目中的应用非常少。

5.6 面向服务软件异常处理研究存在的不足

国内外对面向服务软件的异常及异常处理机制的研究主要集中于面向服务软件的容错机制，主要涉及面向服务软件的相关概念、面向服务软件的异常分类体系、面向服务软件的异常处理方法、框架、技术和形式化模型等方面的工作。在构建具有容错能力的面向服务软件的异常处理方面取得了不少有价值的研究成果。但总结起来，目前还存在以下问题：

1)在异常处理方法和支撑工具方面，对于面向服务的大型软件系统而言，仍然缺乏充分的异常处理机制和足够的处理能力，缺乏快速高效开发复杂异常处理逻辑的系统化方法和支撑工具。多数研究从模型定制出发来定义异常及处理方法，如基于规则的方法、恢复补偿方法等。此类方法的前提是需要用户能够清晰定义异常的边界，用户只有在清晰理解整个业务体系的前提下，才有可能完成这项工作，因此，需要解决如何面向业务流程管理人员，灵活地配置 BPEL 流程的异常处理逻辑，如何自动地实现 BPEL 流程的异常处理逻辑，将异常处理从正常的业务流程中分离出来，从而提高 BPEL 流程异常处理的开发效率。提高异常处理逻辑的模块化程度、可配置性、可扩展性和可重用性一直是异常处理研究和开发的重要技术目标。

2)在异常处理方法的动态自适应性方面，已有方法都要求设计人员花费很多时间对每类异常及其处理器进行详细定义。在应用设计阶段预见所有的异常是很困难的，因此在应用的运行阶段采用基于案例及知识的处理方法、流程的动态调整及自适应方法等来处理发生的异常非常重要。多数研究力求异常处理的自动实现，而传统的“先定义、后执行”的应用模式，使得难度很大，无法实现运行时的动态自适应性和动态自修复，应该让系统尽可能地自动处理发生的异常情况，使异常处理过程对于用户和系统管理员来说是透明的。

3) 在异常处理形式化分析验证方面，虽然针对面向服务软件某些特定的异常处理技术的形式化方法已各自取得了不少高水平的研究成果，但描述面向服务大型软件系统开发的异常处理逻辑仍缺乏一个完整的形式化描述方法来有效地描述一个高抽象层次的、易于理解的、易于验证评估的异常处理总体方案。如何面向 BPEL 流程异常开发人员，清晰地描述 BPEL 流程的异常处理逻辑，明确地区分正常处理逻辑和异常处理逻辑？如何建立正常处理逻辑和异常处理逻辑之间有效的协同关系？如何分析和验证异常处理逻辑的正确性、完备性和可终止性？如何验证异常处理逻辑的执行与异常处理逻辑形式化描述的一致性等问题，从而正确地指导和控制流程异常处理逻辑的行为？以上问题还需要进一步深化研究。

4) 面向语义 Web 服务的软件异常处理相关研究工作尚处于起步阶段，已有的研究成果相对分散且体系不够完整，目前在面向语义 Web 服务的异常处理领域，借助本体和语义 Web 服务技术可以有效克服使用补偿和直接等价服务替换方式而产生的缺陷。现主要利用语义技术来对现有技术进行增强，编程语言层面的语义技术应用不够深入，对异常及异常产生时的程序上下文信息的语义描述不充分，需要提供较为丰富的异常语义支持，把大部分可预期的异常及相应的异常处理逻辑显式地建模为语义 Web 服务程序流程定义的一部分。设计通过应用语义技术来有效检测异常的方法，以为未预见异常的自动化处理提供支持。

5)缺乏异常处理结果的分析评估，其中如何确保业务流程异常处理后的语义正确性是关键问题。需要进一步研究异常处理逻辑的仿真分析方法，定量分析和评估异常处理逻辑对系统可靠性和性能的影响，辅助开发人员在异常处理逻辑的设计过程中对设计方案进行权衡和决策。

6)分布式环境下的异常处理研究还比较少，集中式的异常管理并不能满足 Web 服务高度动态和分布式本质，中国科学院孟丹等提出了面向云计算环境的基于失效规则的节点资源动态提供策略，其综合考虑了资源需求和失效规律，以保证动态提供的节点资源的可靠性[67]。英国纽卡斯尔大学的 AlexeiIliasov 等人提出了一种针对移动 Agent 的协同域间异常的传播方法，该方法基于穿越域边界的联系链，显式地建立异常传播的路径，有助于多层协同容错系统的实现[68]。目前仍然需要研究分布式环境和云计算环境下的异常处理和资源失效问题。

7) 面向服务的异常传播目前是研究的热点之一，如纽卡斯尔大学的 Gorbenko A 等人将错误和故障注入不同的 Web 服务平台，通过试验结论分析不同平台中异常的传播行为，以及异常传播对不同平台的性能影响[69]。目前的异常传播模型难以表达隐式的没有检查的异常传播机制，更不能表达异常所引发的不安全状态的传播。很少将面向服务软件中的异常传播和错误传播结合起来进行研究，需要建立异常和错误之间相互伴随、相互激发、相互转化的统一的传播模型，来预测其他节点出现异常的时间和可信度，支持异常的自动传播。

表1从异常分类、异常定义、异常处理结构等11个方面对当前异常处理方法与未来的异常处理趋势进行了比较，通过比较，可以总结未来的异常处理研究发展的主要趋势。

表1 当前异常处理方法与未来的异常处理趋势的比较

项目	现有的异常处理方法	未来的异常处理趋势
异常分类	明确	不明确
异常定义	建模阶段	执行阶段
异常处理结构	分布式，集中式	分布式
处理异常类型	可预测	可预测，不可预测
事务处理技术	不完全支持	完全支持，自愈
人工参与频率	经常	很少
异常传播	人工传播	自动传播
复杂与智能化程度	简单	复杂，智能程度高
严重异常处理	终止执行	继续执行
异常建模和分析	建模，部分性质分析	建模，异常评价与正确性验证
异常处理支持工具	特定机制	灵活地配置，自动化程度高

结束语 面向服务软件通过动态发现和组合网络上分布的标准化的服务，来实现网络资源共享和应用软件协同，从而构造分布式的、松耦合的网络应用软件。由于互联网系统运行环境的不确定性，允许系统在运行时刻动态发现并绑定新服务，从而造成系统的边界在运行时发生变化，表现出其开放性的特点。可以预测下一步面向服务软件异常处理的主要研究方向呈现以下趋势：

1）需要提出动态灵活、自适应强的异常处理机制，常规的异常处理支撑机制过于贴近系统底层，改进或提出适应Web服务环境的支撑机制是当前的研究热点。由于面向服务软件开发对灵活性的要求日益增加，研究方向主要侧重于改善和提高异常处理支撑机制的动态性和灵活性、可配置性和适应性以及主动性和事务化等，所采用的异常处理手段也逐渐从手动、半自动向自动化方向演化。现有的面向服务软件系统运行时异常处理融合了多种技术和方法，在解决一类异常处理问题的同时，也存在一定的不足。为此，在面向服务软件异常处理方法的定义过程中，既需要避免定义过于贴近系统底层的方法而为业务用户带来开发难度高的问题，又需要屏蔽一切都由系统自动完成的方法引发的知识积累难度高的问题。

2）要设计支持异常及其处理逻辑建模的软件体系结构描述语言、体系结构层的异常处理模型的验证和仿真分析方法。构造面向服务软件系统的异常模型，从服务异常（服务被调用时抛出的异常）、编排异常（流程中产生的异常）两个角度，显式地、系统地标识和规约面向服务软件体系结构设计时所关注的各类异常。支持在设计阶段，系统地、全面地识别和标识系统中出现的异常，异常处理建模的主要任务是对异常及其处理过程（异常的发生、捕获、处理、抛出）进行描述。建模时，需要遵循关注点分离原则，明确分离体系结构中的正常业务逻辑和异常处理逻辑。形式化验证包括语法、语义、结构方面的验证。可验证的问题有合理性、正确性、可恢复性问题。验证异常处理逻辑的完备性和可终止性，辅助设计人员检测异常处理逻辑设计模型中的缺陷，改进异常处理逻辑的设计方案。分析异常处理逻辑对系统的影响，以确保所有声明的异常都有正确的异常处理服务，确保异常处理逻辑能够正确终止并返回正常处理逻辑，确保经异常处理逻辑处理后能恢复到发生异常前的正确状态。提供异常影响的性能分析或异常传播分析的方法和工具。

3）需要重点考虑面向服务软件的分布式、并发的特性对异常产生的影响，以及并发的流程运行时异常的补偿、恢复处理研究。考虑并发流程中的异常嵌套、并发子事务和事务的协同处理，保证当流程事务发生异常或失败时系统能够达成可预测的和可靠的结果，解决事务处理的资源冲突问题和复杂的流程事务的可靠性问题，支持灵活的异常恢复策略，包括向前和向后恢复，保证失效恢复的一致性和正确性，并且在系统处理这种异常时避免人手工的介入。

4）要提高面向服务异常处理的智能化程度，面向服务软件的运行环境由封闭、静态、可控向开放、动态、难控的环境进行转移，在确保程序具有一定异常处理能力的同时，如何减低未预见异常带来的威胁则是进一步需要解决的问题。同时让异常处理机制具有可扩展性，支持设计时不可预期的异常的处理，需要开发主动式异常管理框架，实现异常预测，使得业务用户及时采取预防措施；异常的捕捉是否及时和完备，对于不可预期异常处理的实施具有重要影响。需要仔细考虑异常的检测及监控的方式和内容，其中错误的监测与预警方法和工具具有十分重要的意义。系统应能对将要发生的行为进行自动的推理，以预测可能对系统产生的影响，防止系统运行超出它的约束。当自组织行为发生后，系统应能对其产生的效果采用量化的手段来度量和评估系统自身的自适应程度。在对客户透明的情况下自动处理、有效解决不可预测异常不可解的问题。目前对未预见异常处理机制主要有基于智能算法和概率推理、过数据挖掘和机器学习、基于语义推理的和基于策略规则的未预见异常处理机制。

未来要对面向服务软件中异常处理逻辑的建模、验证和仿真分析进行研究，提出相对完整的在面向服务软件阶段对异常处理逻辑进行设计的理论、方法学和技术体系，进行异常处理结果的用户满意度评价与语义约束正确性验证研究。帮助软件开发人员在软件开发早期识别和控制面向服务软件系统中可能出现的各种异常，为构造可信的面向服务软件系统提供支持。

参 考 文 献

[1] Papazoglou M P, Heuvel W-J. Service oriented architectures: approaches, technologies and research issues[J]. The VLDB Journal, 2007, 16(3): 389-415

[2] Goodenough J B. Exception Handling: issues and a Proposed notion[J]. Communications of the ACM, 1975, 18(12): 683-696

[3] 单锦辉，徐克俊，王戟. 一种软件故障诊断过程框架[J]. 计算机学报，2011，34(2)：371-382

[4] 中国国家标准化管理委员会. GB/T11457-2006 信息技术软件工程术语[S]. 北京：中国标准出版社，2006

[5] IEEE Software & Systems Engineering Standards Committee. IEEE Std 610.12-1990: 1. IEEE Standard Glossary of Software Engineering Terminology[S]. IEEE, 1990

[6] Avizienis A, Laprie J, Randell B, et al. Basic concepts and taxonomy of dependable and secure computing [J]. IEEE Transactions on Dependable and Secure Computing, 2004, 1(1): 11-33

[7] Bruning S, Weißleder S, Malek M. A fault taxonomy for service-oriented architecture[R]. Technical reports. Humboldt-Universitat zu Berlin, 2007: 1-215

[8] Lau K-K, Tran C M. Server-side Exception Handling by Composite Web Services[C]// Binder W, Dustdar S, eds. Emerging Web Services Technology. Volume III. Birkhäuser Basel, 2010: 37-54

[9] Quan Z S, 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

[10] Shah N, Iqbal R, Iqbal K, et al. A QoS Perspective on Exception Diagnosis in Service-Oriented Computing[J]. Journal of Universal Computer Science, 2009, 15(9): 1871-1885

[11] Fugini M, Mussi E. Recovery of Faulty Web Applications through Service Discovery[C]// Proceedings of the 1st Semantic Matchmaking and Resource Retrieval (SMR) Workshop, 32nd International Conference on Very Large Databases. 2006: 67-80

[12] Maamar Z, Benslimane D. Exceptions in a Web services environment[C]// IEEE/ACS International Conference on Computer Systems and Applications, AICCSA. 2008: 770-777

[13] Zeng Liang-zhao, Lei Hui, Jeng Jun-jang. Policy-Driven Exception-Management for composite Web services[C]// CEC. 2005: 355-363

[14] Steyn P J. Approaches to Failure and Recovery in Service Composition[R]. University of Pretoria, South Africa, 2006

[15] Ming X W, Bandara K Y, Pahl C. Integrated Constraint Violation Handling for Dynamic Service Composition[C]// Proceedings of IEEE International Conference on Services Computing, SCC '09. 2009: 168-175

[16] Vaculín R, Wiesner K, Sycara K. Exception handling and recovery of semantic web services [C]// Proceedings 4th International Conference on Networking and Services (ICNS 2008). 2008: 217-222

[17] 刘安. Web服务驱动的业务流程的容错性研究[D]. 合肥: 中国科学技术大学, 2008

[18] Modafferi S, Conforti E. Methods for Enabling Recovery Actions in WS-BPEL[C]// OTM 2006, LNCS 4275. 2006: 219-236

[19] Simmonds J, et al. Runtime Monitoring of Web Service Conversations[J]. IEEE Transactions on Services Computing, 2009, 2(3): 223-244

[20] Barbon F, Traverso P, Pistore M, et al. Run-time monitoring of instances and classes of Web service compositions [C]// The 2006 IEEE International Conference on Web Services. Washington DC, USA, 2006: 63-71

[21] Raimondi F, et al. Efficient Monitoring of Web Service SLAs[C]// Proceedings of the 16th ACM SIGSOFT. 2008

[22] Hai L, et al. Enhancing Web Services Conversation with Exception Contexts for Handling Exceptions of Composite Services [C]// the 4th IEEE International Conference on Enterprise Computing. E-Commerce, and E-Services, 2007

[23] Dayal U, Hsu M, Ladin R. Organizing long-running activities with triggers and transactions[J]. SIG MOD Rec., 1990, 19(2): 204-214

[24] 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. 2006

[25] Charfi A, Mezini M. AO4BPEL: An Aspect-oriented Extension to BPEL[J]. World Wide Web, 2007, 10(3): 309-344

[26] Hamadi R, Benatallah B. Recovery nets: towards self-adaptive - workfow systems [C] // Proceedings of the 5th International Conference on Web Information Systems Engineering (WISE' 04). LNCS3306, 2004: 439-453

[27] Baresi L, Guinea S. A Dynamic and Reactive Approach to the Supervision of BPEL Processes[C]// ISEC'08. 2008: 39-48

[28] 尚宗敏, 崔立真, 王海洋, 等. 基于补偿业务生成图的组合服务异常处理方法研究[J]. 计算机学报, 2008, 31(8): 1478-1490

[29] Kim K, Choi I, Park C. A Rule-based Approach to Proactive Exception Handling in Business Process[J]. Expert Systems with Applications, 2011, 38(1): 394-409

[30] Karastoyanova D, Khalaf R, Schroth R, et al. BPEL Event Model [Z]. University Stuttgart, 2006

[31] Casati F, Castano S, Fugini M. Using Patterns to Design Rules in Workflows [J]. IEEE Transactions on Software Engineering, 2000, 26(8): 760-785

[32] Díez F J, Maurtua I. Dynamic Exception Handling Based on Web Services and OPC XML-DA[C]// Proceeding of IEEE International Conference on Web Services (ICWS08). IEEE Computer Society, 2008: 593-599

[33] 刘海, 刘安, 李青, 等. 一种 ECA 规则驱动的 BPEL 流程异常处理和分析机制[J]. 小型微型计算机系统, 2010, 31(7): 1363-1370

[34] 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): 2101-2107

[35] Salas J, Perez-Sorrosal F, Patiño-Martínez M, et al. WS-Replication: a framework forhighly available Web services[C]// Proceedings of the 15th International Confe-rence on the World Wide Web. 2006: 357-366

[36] Sagara L, Chaudhary B D. A model of scope for verification of implementation of choreography with exception handler using UPPAAL[C]// Proceeding of The Second International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies. IEEE Computer Society, 2008: 510-519

[37] Ezenwoye O, Sadjadi S M. RobustBPEL2: Transparent Autonomization in Business Processes throught Dynamic Proxies[C]// ISADS. 2007: 17-24

[38] Santos G T, Lung L C, Montez C. FTWeb: A Fault Tolerant Infrastructure for Web Services[C]// Proceeding of 9th IEEE International Enterprise Computing Conference (EDOC 2005). Enschede, The Netherlands, 2005: 95-105

[39] 徐伟, 金蓓弘, 李京, 等. 一种基于移动 Agent 的复合 Web 服务容错模型[J]. 计算机学报, 2005, 28(4): 558-567

[40] Cheng Fu-chiung, Hung Tai-chang, Chiou Young-jang. Analysis, Design and Implementation of Exception Handling in WWW Services[C] // Proceeding of the Second IEEE International Symposium on Service-Oriented System Engineering. IEEE Computer Society, 2006: 102-109

(下转第 21 页)

Research,2006(7):31-54
[51] Esuli A,Fagni T,Sebastiani F. Boosting multi-label hierarchical text categorization[J]. Information Retrieval,2008(11):287-313
[52] Tsoumakas G,Katakis I,Vlahavas I. Effective and efficient multilabel classification in domains with large number of labels[C]// Proceeding of ECML/PKDD 2008 Workshop on Mining Multidimensional Data (08' MMD). 2008:30-44
[53] 苏金树,张博峰,徐昕. 基于机器学习的文本分类进展[J]. 软件学报,2006,17(9):1848-1859
[54] Chawla N V,Sylvester J C. Exploiting diversity in ensembles. Improving the performance on unbalanced datasets[C]// Proceedings of the 7th International Conference on Multiple Classifier Systems. 2007:397-406
[55] Tahir M A,Kittler J,Ahmed,et al. Multilabel classification using heterogeneous ensemble of multi-label classifiers[J]. Pattern Recognition Letters,2012,33(5):513-523
[56] Shi Chuan,Kong Xiang-nan,Yu P S,et al. Multi-label ensemble learning[C]// Proceedings of the 2011 European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD'11). Heidelberg,2011(3):223-239

(上接第 8 页)

[41] Cai Chao, Qiu Zong-yan, Yang Hong-li, et al. Global-to-Local Approach to Rigorously Developing Distributed System with Exception Handling[J]. Journal of Computer Science and Technology,2009,24(2):238-249
[42] Christos K,Costas V,Panayiotis G. Enhancing BPEL scenarios with Dynamic Relevance-Based Exception Handling[C]// Proceeding of IEEE International Conference on Web Services (ICWS07). IEEE Computer Society,2007:751-758
[43] Erradi A,Maheshwari P,Tosic V. Recovery Policies for Enhancing Web Services Reliability[C]// IEEE International Conference on Web Services(ICWS'06). 2006
[44] Moser O, Rosenberg F, Dustdar S. Non-Intrusive Monitoring and Service Adaptation for WS-BPEL[C]// World Wide Web. Beijing China,2008:815-824
[45] Liu An,Li Qing,Huang Liu-sheng,et al. FACTS:A Framework for Fault-Tolerant Composition of Transactional Web Services [J]. IEEE Transactions on Services Computing,2010,3(1):46-59
[46] Wen Jia-jia, Chen Jun-liang, Peng Yong, et al. A Multi Policy Exception Handling System for BPEL Process[C]// First International Conference on Communications and Networking in China,ChinaCom '06. 2007:1-5
[47] Yan Yu-hong, Dague P, Pencole Y, et al. A Model-Based Approach for Diagnosing Fault in Web Service Processes [J]. IJWSR,2009,6(1):87-110
[48] Carbone M. Session-based Choreography with Exceptions[J]. Electronic Notes in Theoretical Computer Science,2009,241(C):5-55
[49] Ferrara A. Web Services: a Process Algebra Approach [C]// Proceedings of the 2nd International Conference on Service-Oriented Computing(ICSOC'04). New Y,Drk NY9,ACM Press,2004:242-251
[50] Zurowska K,Deters R. Overcoming Failures in Composite Web Services by Analysing Colored Petri Nets[C]// Eighth Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools. 2007:87-106
[51] 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
[52] 门鹏,段振华. 着色 Petri 网模型检测工具的扩展及其在 Web 服务组合中的应用[J]. 计算机研究与发展,2009,46(8):1294-1303
[53] de Lemos R,Romanovsky A. Exception Handling in the Software Lifecycle[J]. Int. J. Computer Systems Science and Engineering,2001,16(2):167-181
[54] Tartanoglu F,Issarny V,Romanovsky A,et al. Dependability in the Web Service Architecture[C]// 25th International Conference on software engineering(ICSE 2003) Workshop on Software Architectures for Dependable Systems(WADS 2003). Architecting Dependable Systems. Springer,2003:89-108
[55] Birman K,Van R R,Vogels W. Adding high availability and autonomic behavior to Web services[C]// Proceeding of 26th International Conference on Software Engineering (ICSE 2004). IEEE Computer Society,2004:17-26
[56] Garcia D Z G,de Toledo M B F. A fault tolerant web service architecture[C]// Latin American Web Conference. Santiage,Chile,2007:42-49
[57] 李军国,黄罡,邹键,等. 一种中间件服务容错配置管理方法[J]. 计算机学报,2007,30(10):1696-1704
[58] 麻志毅,陈泓婕. 一种面向服务的体系结构参考模型[J]. 计算机学报,2006,29(7):1011-1019
[59] 陈振邦,王戟,董威. 面向服务软件体系结构的接口模型[J]. 软件学报,2006,17(6):1459-1469
[60] http://wsdiamond.di.unito.it/
[61] 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
[62] Vaculín R, Sycara K. Monitoring execution of OWL-S Web services[C]// Proceedings of the European Semantic Web Conference,OWLS:Experiences and Directions Workshop. 2007:32-45
[63] Verma K,Gomadam K,Sheth A P,et al. The Meteor-S approach for Configuring and Executing Dynamic Web Processes[R]. Athens,LSDIS Lab,University of Georgia,2005
[64] Zdravkovic J, Kabilan V. Contract-Based Exception Handling Process Patterns[C]// Proceeding of 2nd International United Information Systems Conference. Klagenfurt,Austria,Springer,2008:531-543
[65] Stein S,Payne T R,Jennings N R. Robust Execution of Service Workflows Using Redundancy and Advance Reservations[J]. IEEE Transactions on Services Computing,2011,4(2):125-139
[66] 赵楷,应时,张琳琳,等. 基于规则的语义流程异常处理机制[J]. 计算机科学,2011,38(7):117-121
[67] 田冠华,孟丹,詹剑锋. 云计算环境下基于失效规则的资源动态提供策略[J]. 计算机学报,2010,33(10):1859-1871
[68] Iliasov A, Romanovsky A. CAMA: Structured Coordination Space and Exception Propagation Mechanism for Mobile Agents [R]. Technical report series. University of Newcastle Upon Tyne Computing Science,2005
[69] Gorbenko A,Romanovsky A,Kharchenko V,et al. Experimenting with exception propagation mechanisms in service-oriented architecture[C]// Proceedings of the 4th International Workshop on Exception Handling. ACM,2008:1-7