一种策略驱动的BPEL流程异常处理描述方法

王权于[1] 应 时[2] 吕国斌[1] 文 静[2] 程银海[1] 陈 莹[1]
(中国地质大学(武汉)网络教育学院 武汉 430074)[1]
(武汉大学软件工程国家重点实验室 武汉 430072)[2]

摘 要 针对如何提高BPEL流程的异常处理描述能力的问题,提出了一种策略驱动的BPEL流程异常处理描述方法。首先设计了一种新型的BPEL流程异常处理策略描述语言BPEH/PDL,并基于着色Petri网提出了BPEH/PDL异常处理策略的形式化描述方法。最后结合制造执行系统领域的汽车装配流水线管理系统,讨论了基于BPEH/PDL的BPEL流程异常处理策略的应用过程,以供参考。

关键词 BPEL流程,异常处理,策略描述语言,着色Petri网
中图法分类号 TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2014.11.043

Policy Driven Exception Handling Description Approach for BPEL Processes

WANG Quan-yu[1] YING Shi[2] LV Guo-bin[1] WEN Jing[2] CHENG Yin-hai[1] CHEN Ying[1]
(Distance Education College, China University of Geoscience, Wuhan 430074, China)[1]
(State Key Lab of Software Engineering, Wuhan University, Wuhan 430072, China)[2]

Abstract In order to improve the ability of exception handling description for BPEL processes, this paper presented a policy driven exception handling description approach for BPEL processes. Firstly, the paper designed BPEH/PDL language, a new Policy Description language(PDL) for exception handling of BPEL processes, and based on Color Petri Net, proposed a formal description approach for exception handling of BPEL processes by the mean of BPEH/PDL. Summarily, the whole description processes of BPEH/PDL were discussed systematically through a case study in automotive manufacturing exeception system domain.

Keywords BPEL processes, Exception handling, PDL, CPN

1 引言

基于WS-BPEL的面向服务软件开发方法由于Web服务的分布自治性、BPEL流程结构的松散耦合性和运行环境的动态开放性,使得BPEL流程的容错能力正成为面向服务软件工程亟待解决的科学问题之一。BPEL流程运行过程发生异常是不可避免的[1],异常如得不到及时和有效处理,可能导致软件系统失效、崩溃,甚至灾难性后果。而且,随着面向服务技术的不断发展和应用的逐步深入,BPEL流程的应用逻辑越来越复杂,异常发生的频率和异常处理的复杂程度也越来越高。因此,BPEL流程异常处理逻辑的开发正日益成为面向服务软件开发过程中一项十分困难的工作。

在BPEL流程异常处理方面,尽管WS-BPEL语言[2]提供了内置的异常处理机制[3]来支持BPEL流程的异常处理逻辑开发,但仍存在着许多值得研究的问题:①如何面向BPEL流程异常处理开发人员,清晰地描述BPEL流程的异常处理逻辑,有效地分离BPEL流程的异常处理逻辑与正常业务逻辑,提高BPEL流程的可复用性和灵活性? ②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,从而提高BPEL流程异常处理的可配置性和可扩展性? ③如何分析和验证BPEL流程的异常处理逻辑的正确性,从而正确地指导和控制BPEL流程异常处理逻辑的行为?

基于策略的管理是一种分离系统管理行为与系统应用功能的管理范型[4]。将策略技术应用到BPEL流程的异常处理,用策略来定义BPEL流程的异常处理逻辑,从而实现正常业务逻辑和异常处理逻辑的有效分离;通过建立BPEL流程异常处理策略的形式化语义模型,进而开展BPEL流程异常处理策略的分析和验证工作,来提高BPEL流程的容错能力,改善BPEL流程的可信性。

本文设计了一种新型的BPEL流程异常处理策略描述语言BPEH/PDL,提出了一种基于策略的BPEL流程异常处理描述方法。本文第2节介绍相关工作;第3节介绍BPEH/PDL语言;第4节介绍BPEH/PDL语言的静态语义;第5节通过一个"汽车装配流水线管理系统"案例,介绍了BPEH/PDL语言的应用;最后总结全文。

2 相关工作

BPEL流程异常处理方法的研究工作如下:Zeng等人提

到稿日期:2014-01-10 返修日期:2014-03-17 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(2013245080)资助。
王权于(1971—),男,博士,副教授,CCF会员,主要研究方向为SOA、语义Web服务,E-mail:wangqy@cug.eud.cn;应 时 教授,博士生导师,主要研究方向为SOA、语义Web服务;吕国斌 教授,主要研究方向为网络安全和数据库;文 静 博士后,主要研究方向为AOP、语义Web服务;程银海 硕士生,主要研究方向为软件工程和网络信息技术;陈 莹 工程师,主要研究方向为网络安全和通信工程。

出了一种策略驱动的 Web 服务组合异常管理框架，通过策略表示 BPEL 流程异常处理知识[5]，实现了异常处理逻辑和正常业务处理逻辑的有效分离，提高了 BPEL 流程异常处理逻辑的开发效率和 BPEL 流程的可重用性。刘安等人提出的方法[6]与文献[5]基本相似，定义了多种异常处理模式，如 Ignore，Retry，Skip 等，在异常处理规则中异常事件和策略是紧耦合方式。A. Erradi 等人提出了一种基于策略的提高 BPEL 流程可信性的方法[7]，设计了一种 BPEL 流程异常恢复策略语言，通过增加配置(Configuration)、策略参数(如策略优先级)和后置条件(Post-Condition)等策略元素实现了 BPEL 流程异常事件和策略的松耦合。异常处理动作除了文献[5]的 6 种动作之外，还定义了调用 BPEL 流程引擎的实例管理动作，如实例的挂起、恢复和终止等。L. Baresi 等人提出了一种 BPEL 流程自我监管的方法来解决 BPEL 流程的自愈能力问题[8]，其中自我监管包括 BPEL 流程的监控和异常处理两个环节，并设计了 WSCol 和 WSRel 语言用于 BPEL 流程的异常监控和异常恢复。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[9]，设计了一种异常处理规则语言用以支持基于规则的 BPEL 流程异常处理。

策略形式化描述方法的研究工作包括：J. Chomicki 等人提出了一种基于霍尔逻辑的 PDL 策略形式化描述方法[10]，采用霍尔逻辑将 Πp 策略定义为策略规则事件集($Epochs(P)$)到策略动作集的映射：$\Pi p: Epochs(P) \rightarrow ActionSets(P)$，其中：$Epochs(P)$、$ActionSet(P)$分别表示策略 P 的事件实例集和动作集，策略事件和动作是策略常量和变量的谓词公式。C. Montangero 等人提出了一种基于时态逻辑 $\Delta DSTL(x)$的 APPEL 策略建模方法[11]，$\Delta DSTL(x)$是对时态逻辑的扩展，将策略规则映射为 $\Delta DSTL(x)$公式，并利用 $\Delta DSTL(x)$函数定义策略规则操作运算来表示策略规则组的 $\Delta DSTL(x)$语义。V. Kolovski 等人提出了一种基于描述逻辑的 WS-Policy 策略描述方法[12,13]，给出了 WS-Policy 策略运算符、策略断言的转换规则。刘海等人提出了一种基于描述逻辑的 BPEL 流程异常处理规则的描述方法[14]，提出了 ECA 规则中事件、条件和动作与描述逻辑 $ALCQO(Q^*)$之间的映射机制，条件映射基于条件映射规则，其映射结果是产生一个类似于"$condition \equiv def$"的表达式，def 是一个 $ALCQO(Q^*)$的非原子概念，将异常处理模式映射为 $ALCQO(Q^*)$表达式。另外，G. Hughes 等人应用一阶逻辑分析工具 Alloy 对 XACML 策略进行了建模[15]。黄荷娇等人提出了一种基于着色 Petri 网的安全策略形式化描述和验证方法[16]和基于着色 Petri 网的策略属性，即完备性、终止性、一致性和合流性的描述方法，并通过着色 Petri 网过程模型(CPNP)提出了安全策略的 CPN 描述和策略验证方法。孙瑞志等人基于 Petri 网的图形化表示，对异常处理过程中的元素，特别是对不同异常处理策略下的处理过程进行了描述[17]。R. Hamadi 等人提出了一种自适应恢复网 SARN 用于设计时描述工作流的异常行为，SARN 运行时通过自我调整底层的 Petri 网结构实现工作流的异常处理[18]。在此研究工作的基础上，文献[19,20]提出了一种基于自适应恢复网的 BPEL 流程异常处理策略描述方法。

在 BPEL 流程异常处理方法方面，已有工作只支持简单的异常处理逻辑的描述，在复杂 BPEL 流程异常处理逻辑表达方面还存在不足；在 BPEL 流程异常处理策略的形式化描述方法方面，尽管刘海等人提出了一种基于描述逻辑的 BPEL 流程异常处理策略形式化描述方法，但大部分的研究工作还没有给出异常处理策略的形式化描述方法，不支持异常处理策略的形式化分析和验证工作。我们的工作采用基于层次着色 Petri 网的 BPEL 流程异常处理描述方法，它具有较好的图形表达能力，便于 BPEL 流程异常处理策略的分析和验证工作。

3 BPEH/PDL 语言

为了支持策略驱动的 BPEL 流程异常处理方法，本文设计了一种新型的 BPEL 流程异常处理策略描述语言 BPEH/PDL(BPEL Process Exception Handling Policy Description Language)。BPEL 流程异常处理设计人员通过 BPEH/PDL 语言，声明式地定义 BPEL 流程异常处理策略，在较高抽象层次开发 BPEL 流程异常处理逻辑，实现了 BPEL 流程异常处理逻辑和正常业务逻辑的有效分离。BPEL 流程异常管理人员在 BPEL 流程部署和运行时，能静态或动态地配置 BPEL 流程异常处理策略，从而提高 BPEL 流程异常处理的灵活性和适应性。

3.1 BPEH/PDL 的设计原则

G. Tonti 认为策略描述语言(Policy Description Language，PDL)应在描述能力、易用性、扩展性和策略分析等方面满足应用领域策略表达需要[22]。为支持 BPEL 流程复杂的异常处理逻辑描述需求，BPEH/PDL 语言设计遵循以下原则：

①支持 BPEL 流程复杂异常处理逻辑的描述：BPEH/PDL 是面向 BPEL 流程异常处理的策略描述语言，必须明确体现异常开发人员决策过程的复杂性，清晰地表示 BPEL 流程的异常处理过程。

②支持形式化分析和验证：BPEH/PDL 语言的各种语法成分必须具有清晰明确的语义，支持 PEL 流程异常处理策略的分析和验证，如策略冲突和规则冲突的检测和消解。

③可扩展性：BPEH/PDL 语言允许 BPEL 流程异常逻辑开发人员定义新的异常类型、异常处理动作、异常处理规则和异常处理策略，以满足不同应用场景下开发 BPEL 流程异常处理逻辑的需要。

④易用性：BPEH/PDL 语言应采用简洁、易理解的语法和描述方式，同时提供图形表示法，以方便 BPEL 流程异常处理设计人员开发异常处理策略。

3.2 BPEH/PDL 的概念模型

BPEH/PDL 语言是面向 BPEL 流程异常处理的策略描述语言，既具有通用策略描述语言的基本元素和属性，又充分考虑 BPEL 流程异常处理策略描述特点，支持从异常处理动作、规则、策略到策略集等不同抽象层次的描述。一个 BPEL 流程绑定一个 BPEH/PDL 策略文件，策略文件至少包含一个策略集(EHPolicySet)，策略集是非空策略(EHPolicy)的集合；策略是非空的规则(EHRule)集，规则按顺序执行；异常处理规则扩展 ECA 规则范型，当 BPEL 流程发生异常时，若 BPEL 流程状态满足规则使能约束条件(Condition)和动作执行前置条件(Precondtion)，则执行规则异常处理动作(EHAction)，并评估动作效果。若所有规则执行失败，则按返回方式控制 BPEL 流程异常处理行为。

通过分析和研究 BPEL 流程异常处理机制，并在借鉴和参考目前已有工作的基础上，提出了 BPEH/PDL 语言的概念模型(见图 1)。BPEH/PDL 语言概念模型包括异常处理策略集(EHPolicySet)、异常处理策略(EHPolicy)、异常处理规则(EHRule)和异常处理动作(EHAction)等核心元素。策略集(EHPolicySet)是 BPEL 流程异常处理策略的集合。策略集在最高抽象层次描述了指导和控制 BPEL 流程异常处理的行为。策略集的策略变量定义 BPEL 流程异常处理的流程状态和系统环境。策略组合算法元素(PolicyCombiningAlg)定义策略集内策略冲突时的消解决策。策略(EHPolicy)是规则的集合。BPEH/PDL 语言中策略是策略集和规则的中介，既是策略集的基本构成单位，又是异常处理规则容器。策略目标元素定义了策略所能处理的异常类型。策略是规则的顺序组合，即如果前继规则执行失败，则执行后续规则，以此类推，直至执行成功或规则集执行完毕。若策略执行失败，则按照策略返回方式控制 BPEL 流程异常处理的行为。规则(EHRule)扩展 ECA 规则范型，BPEH/PDL 规则定义了对 BPEL 流程异常的反应：当 BPEL 流程发生异常时，如果 BPEL 流程运行时状态满足规则使能条件，且同时满足动作前置条件，则执行规则的异常处理动作；执行完成后，计算规则后置条件是否满足异常处理成功的约束。

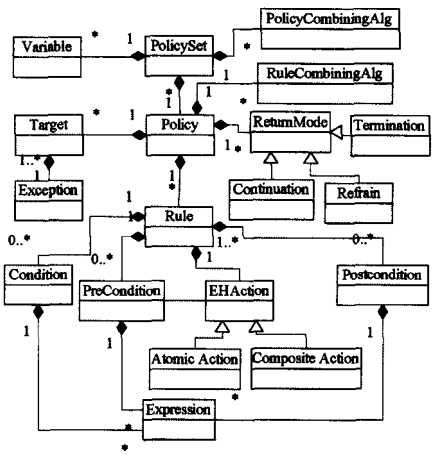

图 1 BPEH/PDL 语言元模型

动作(EHAction)定义规则的异常处理行为，异常处理动作(见图 2)包括原子动作(Atomic Action)和组合动作(Composite Action)。目前定义原子动作包括：忽略(Ignore)、重试(Retry)、替换(Replace)、跳过(Skip)、等价服务调用(Alternate)、补偿(Compensate)、取消(Cancel)、外部服务调用(Call)、报警(Alert)；组合动作由原子动作组成，包括顺序动作(EHSequence)、并行动作(EHFlow)和选择动作(EHSwitch)。

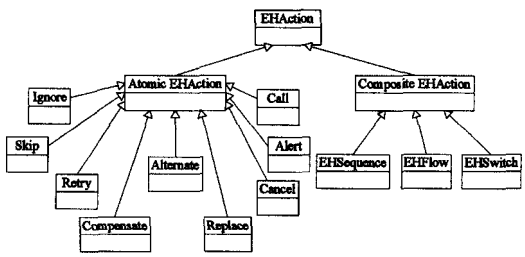

图 2 BPEH/PDL 语言动作模型

3.3 BPEH/PDL 的语法结构

BPEH/PDL 使用 XML 作为其元语言。BPEH/PDL 语言的总体语法结构如下：

```
〈xs:schema targetName sapce=R2E〉
〈PolicySet PolicySetId=NCName PolicySetDefault=NCName?〉
〈Descriptions/〉?
〈Variables/〉?
〈PolicyCombingAlg/〉
〈Policy PolicyId=NCName Version=NCName? Priority=XSInteger?〉
  〈Description/〉?
  〈RuleCombiningAlg/〉
    〈Target〉
      〈Exception〉
        〈/Target〉
〈Rule RuleId=NCName DateTime=XS:DataTime? Priority=XS:Integer?〉
      〈Condition/〉
        〈PreCondition/〉
        〈EHAction〉
        〈PostCondition〉
      〈/Rule〉+
    〈ReturnMode/〉
  〈/Policy〉+
〈/PolicySet〉+
〈/xs:schema〉
```

3.3.1 BPEH/PDL 策略集的描述

EHPolicySet 是 BPEL 流程异常处理策略的集合，EHPolicySet=(PolicySetId, Description, Variables, PolicyCombiningAlg, EHPolicies)。PolicySetId 在策略文件中唯一地定义了异常处理策略集；Description 是描述说明策略集的可选元素；Variables 定义了策略集的全局变量，策略集内部的元素和属性(如策略、规则等)可以直接使用策略集变量；Variables 可以是 XML Schema 的基本数据类型或用户自定义的数据类型，变量分为 BPEL 流程内部变量和外部变量，内部变量引用 BPEL 流程模型内定义的流程变量，外部变量定义 BPEL 流程运行环境的属性(如流程实例状态、活动状态等)，Variables 用指示符 $ 引导；PolicyCombiningAlg 定义了策略集的策略间发生冲突时的消解决策；Policies 是策略集内的异常处理策略的集合。

```
〈PolicySet PolicySetId=NCName PolicySetDefault=NCName?〉
  〈Descriptions/〉?
  〈Variables/〉?
  〈PolicyCombingAlg/〉
  〈/Policy〉+
〈/PolicySet〉
```

3.3.2 BPEH/PDL 策略的描述

异常处理策略 EHPolicy 是 BPEL 流程异常处理规则的序列，策略既是策略集的基本构成单元和策略决策和执行的最小分配单元，也是异常处理规则的容器。策略定义为：EHPolicy=(PolicyId, Description, Target, RuleCombiningAlg, EHRules, ReturnMode)，PolicyId 唯一地定义了策略集的异常处理策略；Version 为可选属性，描述了策略的版本信息；策略目标 Target 定义了策略所能处理的 BPEL 流程异常类型，

刻画了策略的异常处理能力，Target 是策略所能处理的 BPEL 流程异常的集合；为了兼容 WS-BPEL 规范，异常类型与 WS-BEPL 异常类型一致；RuleCombiningAlg 元素定义了策略内的异常处理规则之间发生冲突时的消解决策；ReturnMode 定义了策略执行后 BPEL 流程的行为方式，包括继续模式、抑制模式和终止模式。①继续模式：重新抛出策略捕获的原始异常，BPEL 流程回到异常发生时的初始状态，将异常抛给故障域的上层作用域异常处理，抛出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使用场景。②抑制模式：记录策略捕获的异常，忽略该 BPEL 异常，BPEL 流程继续执行其故障点的后继活动。其语义等价于原子动作 Alert 和 Ignore 的顺序组合。③终止模式：立即终止 BPEL 流程实例的执行。

```
〈Policy PolicyId=NCName Version=NCName? Priority=XS:Integer?〉
  〈Description/〉
    〈RuleCombiningAlg/〉
      〈Target〉
        〈Exception/〉
      〈/Target〉
    〈Rule/〉
  〈ReturnMode/〉
〈/Policy〉
```

3.3.3 BPEH/PDL 规则的描述

异常处理规则 EHRule 扩展了 ECA 规则范型，定义了 BPEL 流程异常发生时，如果 BPEL 流程状态使规则使能条件(C_{enable})为真，则执行规则异常处理动作，并计算规则的执行对 BPEL 流程的影响。其定义为 EHRule=(RuleId,Description,$Condition_{enable}$,$Condition_{pre}$,EHAction,$Condition_{post}$)，异常处理规则标识符属性 RuleId 在策略的范围内唯一地定义了异常处理规则。$Condition_{enable}$ 描述了执行规则必须满足的约束条件或遵守的限制，只有当规则使能条件计算为“真”时，规则才被激活使能；$Condition_{pre}$ 描述了异常处理动作执行时必须满足的约束或限制，异常处理动作设计人员定义动作前置条件；将规则使能条件和动作前置条件分开，使在相同的异常上下文环境下，异常设计人员可以通过定义不同的异常处理动作来选择控制不同的 BPEL 流程异常处理行为，这为异常处理开发人员提供了更为灵活的描述能力。规则使能条件和动作前置条件默认都为“真”值，表示在默认情况下，异常处理规则处于激活就绪状态。异常处理动作 EHAaction 描述了对 BPEL 流程异常的反应或响应，异常处理动作不能为空，动作必须是原子动作或复合动作，BPEH/PDL 语言支持动作类型的扩展。$Condition_{post}$ 是计算异常处理动作执行成功必须满足的约束，默认情况下为真。

```
〈Rule RuleId=NCName DateTime=XS:DateTime? Priority=XS:Integer?〉
    〈Description/〉?
     〈Condition/〉?
      〈PreCondition/〉?
      〈EHAction〉
      〈PostCondition〉?
〈/Rule〉
```

3.3.4 BPEH/PDL 策略绑定的描述

策略绑定描述 BPEL 流程异常处理策略对 BPEL 流程异常行为的指导和控制。BPEH/PDL 语言借鉴了 WS-Policy 框架的绑定机制，通过 BPEL 流程异常处理策略配置文件定义 BPEL 流程和 BPEH/PDL 策略的绑定。BPEH/PDL 绑定机制包括策略集和 BPEL 流程的绑定、策略和故障作用域的绑定两个层次。

```
〈R2E:Attachment〉
  〈R2E:PolicySetAttachment〉
   〈R2E:AppliesTo〉
     〈R2E:BP/〉+
   〈/R2E:AppliesTo〉
   (
    〈R2E:PolicySet〉%PolicySetId%〈/R2E:PolicySet〉|
    〈R2E:PolicySetReference〉%PolicySetURI%
    〈/R2E:PolicySetReference〉
     )+
  〈/R2E:PolicySetAttachment〉
  〈R2E:PolicyAttachment〉
     〈R2E:AppliesTo〉
        〈R2E:Scope〉+
        〈/R2E:AppliesTo〉
        (
          〈R2E:Policy〉%PolicyId〈/R2E:Policy〉|
          〈R2E:PolicyReference〉%PolicyURI%
          〈/R2E:PolicyReference〉
        )
        〈/R2E:PolicyAttachment〉
〈/R2E:Attachment〉
```

4 BPEH/PDL 的 CPN 模型

通过建立 BPEH/PDL 语言的 CPN 模型，来支持 BPEH/PDL 策略的形式化分析工作。

4.1 CPN 基本概念

Petri 网是一种图形化的并发系统建模和分析语言[20]。着色 Petri 网(Color Petri Net,CPN)是一种融合了 Petri 网和结构化程序设计语言的高级 Petri 网，通过扩展数据类型和数据操作，增强了 Petri 网的表达能力[21-24]。本文采用 K. Jensen提出的 CPN[25]相关概念和形式化定义。

4.1.1 非层次着色 Petri 网

定义 1(非层次着色 Petri 网) 非层次着色 Petri 网是一个九元组 $CPN=(P,T,A,\Sigma,V,C,G,E,I)$：

①P 是一个有限库所集合；

②T 是一个有限变迁集合，$P\cap T=\emptyset$；

③A 是一个有向弧集合；

④Σ 是一个非空的颜色集的有限集合；

⑤V 是一个类型化变量有限集合；

⑥C 是一个颜色集函数 $C:P\rightarrow\Sigma$，为每个库所令牌赋予颜色集；

⑦G 是一个卫式函数 $G:T\rightarrow EXPR_V$，为变迁 t 赋值逻辑表达式，即 $Type[G(t)]=Bool$；

⑧E 是一个弧表达式函数 $E:A\rightarrow EXPR_V$，对每个弧 a 赋

值一个弧表达式，满足 $Type[E(a)]=C(p)MS$，p 是连接到弧 a 的库所；

⑨I 一个是初始函数 $I:P\rightarrow EXPR_{\emptyset}$，对每个库所进行初始化，满足 $Type[I(p)]=C(p)_{MS}$。

4.1.2 层次着色 Petri 网

层次着色 Petri 网(Hierarchical CPN，HCPN)融合了模块化程序设计思想，是对着色 Petri 网的扩展。一个 HCPN 由多个子网(SubNet)构成。HCPN 通过模块(Module)表示子网(Subnet)，模块分为主模块(Prime Module)和子模块(SubModule)。主模块包含子模块，主模块用替代变迁(Substitution Transition)表示子模块。子模块与主模块的接口称为槽库所(Socket place)，替代变迁的输入库所称为输入槽(input sockets)，输出库所称为输出槽(output socket)，既是输入又是输出库所称为输入输出槽(input/output socket)。子模块与外部环境的接口称为端口库所(Port place)，端口库所包括输入、输出和输入/出 3 种类型。与模块化程序设计类似，子模块可以被主模块多次重复调用，且子模块也可嵌套子模块，形成模块层次关系。

定义 2(CPN 模块) $CPNM=(CPN,T_{sub},P_{port},PT)$：

①$CPN=(P,T,A,\Sigma,V,C,G,E,I)$ 是一个非层次的 CPN；

②T_{sub} 是一个替代变迁的集合；

③P_{port} 是一个端口库所的集合；

④$PT:P_{port}\rightarrow\{IN,OUT,I/O\}$ 是端口类型函数，为每个端口库所赋值端口类型。

定义 3(层次着色 Petri 网) $CPNH=(S,SM,PS,FS)$：

①S 是 CPN 模块的有限集，对于 $\forall s\in S$

$s=((P^s,T^s,A^s,\Sigma^s,V^s,C^s,G^s,E^s,I^s),T^s_{sub},P^s_{port},PT^s)$

且任意两个模块没有相同的库所和变迁元素，即

$\forall s1,s2\in S:(P^{s1}\cup T^{s1})\cap(P^{s2}\cup T^{s2})=\emptyset$

②子模块函数 $SM:T_{sub}\rightarrow S$ 将每个替代变迁映射到某个子模块，注意模块层次结构必须是非环形的；

③端口-槽关系函数 $PS:P_{sock}(t)\rightarrow P^{SM(t)}_{port}$，为每个替代变迁 t 指派端口-槽关系；要求所有的端口库所和槽库所有相同的端口类型、颜色集和初始值，即：

$\forall(p,p')\in PS(t):(ST(p)=ST(p'))\wedge(C(p)=C(p'))\wedge(I(p)<>=I(p')<>)$

④$FS\subseteq 2^P$ 是非空融合集的幂集，即对于任何融合集，其颜色集和初始值相同，亦即：

$\forall(p,p')\in fs,\forall fs\in FS:C(p)=C(p')\wedge I(p)<>=I(p')<>$

定义 4(模块层次) $CPNH$ 的模块层次是一个有向图 $MH=(NMH,AMH)$，其中：

①NMH 是节点集合；

②$A_{MH}=\{(s1,t,s2)\in N_{MH}\times T_{sub}\times N_{MH}\mid t\in T^1_{sub}\wedge s2=SM(t)\}$ 是弧的集合。模块层次(MH)的根称为主模块(Prime Modules)，所有主模块集合用 S_{PM} 来表示，一个层次着色 Petri 网模型可以有多个主模块。

4.2 BPEH/PDL 的静态语义

本节根据 BPEH/PDL 的语法结构，自底向上地定义了 BPEH/PDL 语言各核心元素的 CPN 模型。

4.2.1 BPEH/PDL 动作的 CPN 模型

定义 5 BPEH/PDL 动作定义为：

$EHAction::=EHX\mid EHSequence\mid EHFlow\mid EHSwitch$

①EHX 代表原子动作，其定义为：

$EHX::=Ignore\mid Skip\mid Retry\mid Alternate\mid Compensate\mid Call\mid Alert\mid Cancel$

②$EHSequence$ 为顺序动作，定义为：$EHSequence::=A1\gg A2$，$\gg$表示动作 $A1$ 和 $A2$ 顺序执行，即如果 $A1$ 执行失败，则执行 $A2$；如果 $A1$ 执行成功，则 $A2$ 不再被执行。

③$EHFlow$ 代表并行动作，定义为：$EHFlow::=A1\parallel A2$，$\parallel$ 表示动作 $A1$ 和 $A2$ 并行执行，即任何一个动作执行成功则终止其他动作。

④$EHSwitch$ 为选择动作，表示按照动作条件 c 执行其中的某个异常处理动作($A1$ 或 $A2$)。

(1)原子动作 CPN 模型

原子动作是不能再分的基本动作，因此，将原子动作建模为一个非层次 Petri 网。由于动作参数、策略变量由多个部分组成，因此将动作参数和策略变量映射为消息类型颜色集，消息类型颜色集定义为记录颜色集类型；同样，相关集是多个 BPEL 流程属性的集合，也采用记录颜色集类型来定义相关集颜色集。动作的状态映射为动作状态颜色集，动作状态颜色集定义为枚举颜色集类型。不同的原子动作有着不同的动作参数，因此映射为不同的消息类型颜色集。异常处理策略激活前，异常处理动作处于非激活状态；当异常处理策略激活后，如果匹配到合适的异常处理规则，则激活异常处理规则；当异常处理规则被激活后，同时激活规则的异常处理动作，其状态从非激活状态变迁到就绪状态，动作接受输入信息，准备执行异常处理行为；异常处理动作就绪后，会根据前提条件的不同，决定是否执行。异常处理动作的执行过程会产生不同的结束状态，在组合动作的执行过程中，可能由于前继动作的返回方式而导致直接跳过后续动作的执行；按照动作后置条件，动作要么执行成功，要么失败。

定义 6 BPEH/PDL 原子动作 EHX 是一个非层次着色 Petri 网，$EHX=(P,T,A,\Sigma,V,C,G,E,I)$

$$P^{S0}=\left\{\begin{matrix}Requeste_{EHX},Exception,Variable,CS,\\Input_{EHService},Output_{EHService},Input_{EHX},Response_{EHX}\end{matrix}\right\}$$

$$T^{S0}=\{Receive_{EHX},EHService,Evaluate_{EHX}\}$$

$$A^{S0}=\left\{\begin{matrix}(Request_{EHX},Receive_{EHX}),(Receive_{EHX},Exception),\\(Receive_{EHX},Variable),(Receive_{EHX},CS),\\(Receive_{EHX},Input_{EHService}),(Input_{EHX},EHService),\\(Input_{EHService},EHService),(EHService,Output_{EHService}),\\(Output_{EHService},Evaluate_{EHX}),(Evaluate_{EHX},Output_{EHX})\end{matrix}\right\}$$

$$\Sigma^{S0}=\{EHRequest,Variable,Variables,Exception,CS,EHResponse\}$$

$$V^{S0}=\{ex:Exceptoin,vx:Variables,x:Variable,cs:CS\}$$

$$C^{S0}=\left\{\begin{matrix}(ex,vx,cs), & \text{if } (p\in\{Request_{EHX},Input_{Service}\})\\x, & \text{if } (p\in\{Input_{EHX}\})\\ex, & \text{if } (p\in\{Output_{Service},Output_{EHX}\})\end{matrix}\right.$$

$G^{S0}(t)=\emptyset$

$E^{S0}(a)=$

$$\begin{cases}(ex,vx,cs),\text{if}(a\in\{(Request_{EHX},Receive_{EHX})(Receive_{EHX},Input_{EHService}),(Input_{EHService},EHService)\})\\ ex,\text{if}(a\in\{(Receive_{EHX},Exception),(Exception,EHService)\})\\ vx,\text{if}\ (a\in\{(Receive_{EHX},Variable),(Varaible,EHService)\})\\ cs,\text{if}\ (a\in\{(\text{Receive}_{EHX},CS),(CS,EHService)\})\\ x;\text{if}\ (a\in\{(Input_{EHX},EHService)\})\end{cases}$$

$I^{S0}(p)=\emptyset$

(2)组合动作 CPN 模型

组合动作是将原子动作或者组合动作按顺序、并行或者选择控制逻辑组合而成的复合动作。组合动作的组合过程具有嵌套包含的特性，即外层组合动作可以包含另一个组合动作，以此类推，最内层的组合动作只由原子动作组合而成。这种嵌套组合方式符合层次着色 Petri 网在层次划分上的特点。因此，基于上面的原子动作非层次 Petri 网模型，使用层次 CPN 对组合动作中的组合关系建模。

A. 顺序动作 CPN 模型

顺序动作是将多个原子动作或者组合动作按照执行顺序组合而成的活动，定义为：

$$Action=(A\triangleright B)\equiv(A_{succeed}\rightarrow B_{skip})\vee(A_{skip}\rightarrow B_{skip})\vee(A_{failed}\rightarrow B_{ready})$$

即：如果动作 A 执行成功，则动作 B 被跳过；或者，如果动作 A 被跳过，则动作 B 也被跳过；或者，如果动作 A 执行失败，则开始执行动作 B。顺序动作的子动作可以是原子动作或组合动作，利用替代变迁来表示顺序动作的子动作。因此，将顺序动作建模为一个层次着色 Petri 网，每个子动作采用层次着色 Petri 网的替代变迁来描述。顺序动作和子动作之间的接口采用端口-槽库所来表示。顺序动作的层次着色 Petri 网模型如图 2 所示。在此说明的是，并行动作和选择动作的 CPN 模型采用同样的思路，因此不再作详细描述。

定义 7 顺序动作 *EHSequence* 是一个层次着色 *Petri* 网，$EHSequence=(CPNsequence,SM,PS,FS)$：

①$CPNsequence=\{S0,EHX\}$，EHX 是异常处理原子动作模块，S0 表示顶层模块，定义为：

$S0=(P^{S0},T^{S0},A^{S0},\sum^{S0},V^{S0},C^{S0},G^{S0},E^{S0},I^{S0})$，其中：

$P^{S0}=\{Request_{EHSequence},Request_{EHXA},Response_{EHXA},Request_{EHXB},Response_{EHXB},Response_{EHSequence}\}$

$T^{S0}=\{Receive_{EHSequence},EHXA,EHXB,Check_{EHXA},Check_{EHXB}\}$

$A^{S0}=$

$$\begin{cases}(Request_{EHSequence},Receive_{EHSequence}),(Receive_{EHSequence},Request_{EHXA}),\\ (Request_{EHXA},EHXA),(EHXA,Response_{EHXA}),(Response_{EHXA},Check_{EHXA}),(Check_{EHXA},Response_{EHSequence}),(Check_{EHXA},Request_{EHXB}),(Request_{EHXB},EHXB),(EHXB,Response_{EHXB}),(Response_{EHXB},Check_{EHXB}),(Check_{EHXB},Response_{EHSequence})\end{cases}$$

$\sum^{S0}=\{EHRequest,EHResponse\}$

$V^{S0}=\{ex:Excetpion,vx:Variables,cs:CS,(ex,vx,cs):EHRequest\}$

$C^{S0}=$

$$\begin{cases}EHRequest:\text{if}(p\in\{Request_{EHSequence},Request_{EHXA},Request_{EHXB}\})\\ EHResponse:\text{if}(p\in\{Response_{EHSequence},Response_{EHXA},Response_{EHXB}\})\end{cases}$$

$G^{S0}=\emptyset$

$E^{S0}(a)=$

$$\begin{cases}(ex,vx,cs),\text{if}\ (a\in\{(Request_{EHSequence},Receive_{EHSequence}),(Receive_{EHSequence},Request_{EHXA}),(Response_{EHXA},Requet_{EHXB}),(Check_{EHXA},Response_{EHSquence}),(Check_{EHXB},Response_{EHSquence})\end{cases}$$

$I^{S0}(p)=\emptyset_{MS}$

$T_{sub}=\{EHXA,EHXB\}$

$$SM=\begin{Bmatrix}EHXA\vdash\!\!\rightarrow EHX\\ EHXB\vdash\!\!\rightarrow EHX\end{Bmatrix}$$

$$PS=\begin{Bmatrix}(Request^{S0}_{EHXA},Request^{S1}_{EHX}),(Request^{S0}_{EHXB},\\ Re\text{-}quest^{S1}_{EHX}),(Response^{S0}_{EHXA},Response^{S1}_{EHX}),\\ (Re\text{-}sponse^{S0}_{EHXB},Response^{S1}_{EHX})\end{Bmatrix}$$

$FS=\emptyset$

B. 并行动作的 CPN 模型

并行动作的多个异常处理动作并发地执行，任何一个异常处理动作执行成功后，将停止其他并发动作的执行，表示并行组合动作成功结束。

$$Action=A\parallel B\equiv(A_{skip}\wedge B_{skip})\vee(A_{ready}\wedge B_{ready})\vee((A_{succeed}\wedge B_{skip})\vee(B_{succeed}\wedge A_{skip})\vee(A_{failed}\wedge B_{failed}))$$

①如果并行动作 EHFlow 跳过，则 A 和 B 都跳过；

②如并行动作 EHFlow 就绪，则 A 和 B 动作都就绪；

③如果并行动作 EHFlow 执行，且 A 成功执行，则终止 B 的执行，反之亦然；

④如 A 和 B 都失败，则并行动作 EHFlow 执行失败。

定义 8 并行动作 *EHFlow* 是一个层次着色 *Petri* 网，$EHFlow=(CPNflow,SM,PS,FS)$，其中 $CPNflow=\{S0,EHX\}$，EHX 是异常处理原子动作模块，S0 表示并行动作顶层模块，定义为：$S0=(P^{S0},T^{S0},A^{S0},\sum^{S0},V^{S0},C^{S0},G^{S0},E^{S0},I^{S0})$。

C. 选择动作的 CPN 模型

选择动作(EHSwitch)是按照选择控制逻辑组合异常处理动作的组合动作，定义为：

$$Action=A\Theta B\equiv(A_{ready}\rightarrow B_{skip})\vee(A_{skip}\rightarrow B_{ready})$$

选择动作根据动作条件，选择其中某一动作执行，该动作的执行结果则表示了选择动作的执行结果。即如果 A 就绪，则 B 被跳过；否则，如果 A 跳过，则 B 执行。选择动作建模为一个层次着色 Petri 网。

定义 9 选择动作 *EHSwitch* 是一个层次着色 Petri 网，$EHSwitch=(CPNswitch,SM,PS,FS)$：

①$CPNswitch=\{S0,EHX\}$，EHX 表示异常处理原子动作模块，S0 表示选择动作顶层模块，定义为：$S0=(P^{S0},T^{S0},A^{S0},\sum^{S0},V^{S0},C^{S0},G^{S0},E^{S0},I^{S0})$。

4.2.2 BPEH/PDL 规则的 CPN 模型

A. BPEH/PDL 规则 CPN 模型的颜色集定义

//规则属性、异常请求、使能条件颜色集

colset RuleProperty=product RuleId * Priority * ActionType;

colset RULExEHRequest = product RULE * EHRequest;

colset Condition=BOOL;

colset RULExEnable=product RULE * BOOL;

B. BPEH/PDL 规则定义

定义 10 BPEH/PDL 规则 $EHRule$ 是一个层次着色 Petri 网，$EHRule=(CPNrule, SM, PS, FS)$，$CPNrule=\{S0, EHAction\}$，$EHAction$ 表示异常处理动作模块

$$P^{S0}=\left\{\begin{array}{l}Request_{Rule}, Condition, Property_{Rule}, Enabled_{Rule},\\ Rule, Input_{Rule}, Output_{Rule}, Input_{EHX}, Output_{EHX},\\ Response_{Rule}\end{array}\right\}$$

$$T^{S0}=\{Enable_{Rule}, Receive_{Rule}, EHX, Evaluate_{Rule}\}$$

$$A^{S0}=\left\{\begin{array}{l}(Request_{Rule}, Enable_{Rule}), (Condition, Enable_{Rule}), (Property_{Rule},\\ Enable_{Rule}), (Enable_{Rule}, Enabled_{Rule}), (Input_{Rule}, Receive_{Rule}),\\ (Enabled_{Rule}, Receive_{Rule}), (Receive_{Rule}, Rule), (Receive_{Rule},\\ Input_{EHX}), (Input_{EHX}, EHX), (EHX, Output_{EHX}),\\ (Output_{EHX}, Evaluate_{Rule}), (Rule, Evaluate_{Rule}),\\ (Evaluate_{Rule}, Output_{Rule})\end{array}\right\}$$

$$\sum^{S0}=\{RULE, EHRequest, Variables, Exception, CS, RULExEHRequest, RuleProperty, RULExEnabled\}$$

$$V^{S0}=\{ex: Exceptoin, gx, vx: Variables, cs: CS, b: BOOL, rpx: RuleProperty\}$$

$$C^{S0}=\left\{\begin{array}{l}RULExEHRequest: \text{if } (p\in\{Request_{Rule}, Input_{Rule}\})\\ RULExEHResponse: \text{if } (p\in\{Output_{Rule}\})\\ BOOL: \text{if } (p\in\{Condition\})\\ RuleProperty: \text{if } (p\in\{Property_{Rule}\})\\ RULExEnabled: \text{if } (p\in\{Enabled_{Rule}\}\\ RULE: \text{if } (p\in\{Rule\})\\ EHRequest: \text{if } (p\in\{Input_{EHX}\})\\ EHResponse: \text{if } (p\in\{Output_{EHX}\})\end{array}\right.$$

$$G^{S0}(t)=\left\{\begin{array}{l}b: \text{if } (t=Enable_{Rule})\\ (bandalso(rule=rulex)): \text{if } (t=Receive_{Rule})\\ (not(rm=Continuation)): \text{if } (t=Resume)\end{array}\right.$$

$$E^{S0}(a)=\left\{\begin{array}{l}(rule,(ex,vx,cs)): \text{if } (a\in\{(Request_{Rule}, Enable_{Rule})\\ \quad (Request_{Rule}, Input_{Rule}), (Evaluate_{Rule}, Output_{Rule})\})\\ b, \text{if } (a\in\{(Condition, Enable_{Rule})\})\\ rpx, \text{if } (a\in\{(Property_{Rule}, Enable_{Rule})\})\\ (rule, b), \text{if } (a\in\{(Enable_{Rule}, Enabled_{Rule}), (Enabled_{Rule}\\ \quad Receive_{Rule})\})\\ (ex, vx, cs), \text{if } (a\in\{(Receive_{Rule}, Input_{EHX}), (Output_{EHX}\\ \quad Evaluate_{Rule})\}),\\ rule, \text{if } (a\in\{(Rule, Evaluate_{Rule})\})\end{array}\right.$$

$$I^{S0}(p)=\emptyset$$

$$T_{sub}=\{EHX\}$$

$$SM=\{EHX\mapsto EHX\}$$

$$PS=\{(Input_{EHX}^{S0}, Input_{EHX}^{S1}), (Output_{EHX}^{S0}, Output_{EHX}^{S1})\}$$

$$FS=\emptyset$$

4.2.3 BPEH/PDL 策略的 CPN 模型

A. BPEH/PDL 策略 CPN 模型的颜色集定义

//策略规则集合，策略属性，策略目标，返回方式

colset RULE=index Rule with 1..RSIZE;

colset PolicyProperty=product PolicyId * Priority;

colset Target=list Exception;

colset ReturnMode=with Continuation | Refrain | Termination;

//规则属性队列，异常请求，异常响应

colset RuleSequence=list RuleProperty;

colset POLICYxEHRequest = product POLICY * EHRequest;

colset POLICYxEHResponse = product POLICY * EHResponse;

colset POLICYxProperty = product POLICY * Policy Property;

B. BPEH/PDL 策略 CPN 模型函数定义

列表成员判定函数 member(element, list);

异常请求函数 allRulesEHRequest(EHRequest);

规则遍历函数 fetch(RuleSequence, k);

规则继续执行函数 continue(RULE, Exception);

C. BPEH/PDL 策略定义。

定义 11 BPEH/PDL 策略 EHPolicy 是一个层次着色 Petri 网，$EHPolicy=(CPNpolicy, SM, PS, FS)$：

① $CPNpolicy=\{S0, EHRule, EHRuleConflict\}$，$EHRule, EHRuleConflict$ 见定义 10。

$$P^{S0}=\left\{\begin{array}{l}Request_{Policy}, Target, Property_{Policy}, Next_{Policy}, Response_{Policy},\\ Input_{Policy}, Input, Count, TargetRuleSet, Continue, Iterator,\\ Next_{Rule}, Request_{Rule}, Response_{Rule}, Input_{Rule}, Output_{Rule},\\ ReturnMode\end{array}\right\}$$

$$T^{S0}=\left\{\begin{array}{l}Enable, Receive, SendRequest, SendInput,\\ EHRule, Conflict_{EHRule}, Iterator, Evaluate_{Rule}\end{array}\right\}$$

$$A^{S0}=\left\{\begin{array}{l}(Request_{Policy}, Enable), (Target, Enable), (Property_{Policy},\\ Enable), (Enable, Next_{Policy}), (Enable, Response_{Policy}),\\ (Input_{Policy}, Receive), (Receive, Input), (Input, Send\ Request),\\ (Input, Send), (Send\ Request, Count), (Count, Send\ Request),\\ (Continue, Send), (Next_{Rule}, Send), (Send\ Input_{Rule}),\\ (Send\ Request, Request_{Rule}), (Request_{Rule}, EHRule),\\ (EHRule, Response_{Rule}), (Input_{Rule}, EHRule), (EHRule, Output_{Rule}),\\ (Output_{Rule}, Evaluate_{Rule}), (ReturnMode, Evaluate_{Rule}),\\ (Target\ RuleSet, Interate), (Iterate, Iterator)(Response, Conflict_{Rule}),\\ (Conflict_{Rule}, TargetRuleSet), (Evaluate_{Rule}, Continue),\\ (Evalaute_{Rule}, Output_{Policy})\end{array}\right\}$$

$\Sigma^{S0}=$

$\{RULE, Targets, PolicyProperty, Variables, INT, BOOL, Exception, CS, POLICYxEHRequest, POLICYxProperty, POLICYxEHResponse, ReturnModeRULExEHRequest, RULExEHResponse, RuleSequence, RuleProperty, POLICYxEHResponse\}$

$V^{S0}=\{ex: Exceptoin, vx: Variables, cs: CS, policy: POLICY, m: INT, b: BOOL, tl: Targets, rule: RULE, rp: RuleProperty, rs: RuleSequence, rm: ReturnMode, ptx: PolicyType, prx: Priority\}$

$T_{sub}=\{EHRule, Conflict_{Rule}\}$

$SM=\begin{cases} S1 \vdash\!\!\rightarrow EHRule \\ S2 \vdash\!\!\rightarrow RuleConflict \end{cases}$

$PS=$

$\{(Request_{Rule}^{S0}, Request_{Rule}^{S1}), (Response_{Rule}^{S0}, Response_{Rule}^{S1}), (Input_{Rule}^{S0}, Input_{Rule}^{S1}), (Output_{Rule}^{S0}, Output_{Rule}^{S1}), (Response_{Rule}^{S0}, Response_{Rule}^{S2}), (TargetRuleSet^{S0}, TargetRuleSet^{S2})\}$

$FS=\emptyset$

4.2.4 BPEH/PDL 策略集的 CPN 模型

策略集(PolicySet)是 BPEH/PDL 策略的集合，$PolicySet=\{Policy_1, Policy_2, \cdots, Policy_n\}$。BPEH/PDL 策略集捕获到 BPEL 流程发生异常，根据目标策略元素生成目标策略队列；如果目标策略队列长度大于 1(即目标策略库所上限大于 1)，则表示策略集发生了策略冲突，通过策略决策算法计算执行策略；调用执行策略；输出执行策略结果。BPEH/PDL 策略集 CPN 模型采用模块参数化技术，将策略集内的所有策略建模为策略数组颜色集合，好处之一是减小了策略集 CPN 规模；其二是通过配置策略集策略规模常量 PSIZE，可以灵活地配置策略集的规模，提高建模的灵活性。

A. BPEH/PDL 策略集模型颜色集定义

```
//策略规模、变量、变量容器、相关集颜色集
colset POLICY=index Policy with 1..PSIZE;
colset Variable=product VarName * VarType;
colset Varialbes=list Variable;
colset CS=product CSName * properties;//
colset Exception=product ExcetionType * FaultVariable;
colset EHRequest=product Exception * Variables * CS;
```

B. BPEH/PDL 策略集模型函数定义

异常请求函数 allPoliciesEHRequest(EHRequest)

```
{List.map (fn req => (req,v)) (POLICY.all());}
```

C. BPEH/PDL 策略集定义

定义 12 策略集 $EHPolicySet$ 是一个层次 CPN，

$EHPolicySet=(CPNpolicyset, SM, PS, FS)$

$CPNpolicyset=\{S0, EHPolicy, EHPolicyConflict\}$

$P^{S0}=$

$\{Request, Variables_{PolicySet}, EHRequest, Exception, Variable, CS, Request_{Policy}, Response_{Policy}, Input_{Policy}, Output_{Policy}, Target_{Policy}, Next_{Policy}, Response\}$

$T^{S0}=\{Initialize, SendRequest, Trigger, Resume, Throw, Policy, Conflict_{Policy}\}$

$A^{S0}=$

$\{(Request, Initialize), (Variables_{PolicySet}, Initialize), (Initialize, Exception), (Initialize, Variable), (Initialize, CS), (Initialize, EHRequest), (EHRequest, SendRequest), (SendRequest, Request_{Policy}), (Policy, Response_{Policy}), (Input_{Policy}, Policy), (Policy, Response_{Policy}), (Policy, Next_{Policy}), (Next_{Policy}, SendEHRequst), (Response_{Policy}, Conflict_{Policy}), (Conflict_{Policy}, Target_{Policy}), (Target_{Policy}, Trigger), (Trigger, Input_{Policy}), (Output_{Policy}, Throw), (Throw, EHRequest), (Output_{Policy}, Resume), (Resume, Response)\}$

$\Sigma^{S0}=\{POLICY, EHRequest, Variables, Exception, CS, PolicyxEHRequest, PolicyxProperty, PolicyxEHResponse, ReturnMode\}$

$C^{S0}=$

$\{EHRequest: \text{if } (p\in\{Request\})$
$Variables: \text{if } (p\in\{Variables_{PolicySet}, Variable\})$
$Exception: \text{if } (p\in\{Exception)$
$CS: \text{if}(p\in\{CS\})$
$EHRequest_{Policy}: \text{if } (p\in\{EHRequest, EHRequest_{Policy} Input_{Policy}\})$
$POLICYxProperty: \text{if}(p\in\{EHResponse_{Policy}, Target_{Policy}\})$
$POLICYxResponse: \text{if}(p\in\{Output_{Policy}\}$
$INT: \text{if}(p\in\{Next_{Policy}\})$
$ReturnMode: \text{if}(p\in\{Response\})\}$

$V^{S0}=\{ex: Exceptoin, gx, vx: Variables, cs: CS, policy: POLICY, n: INT, rm: ReturnMode, ptx: PolicyType, prx: Priority\}$

$T_{sub}=\{EHPolicy, Conflict_{Policy}\}$

$SM=\begin{cases} EHPolicy \vdash\!\!\rightarrow EHPolicy \\ PolicyConflict \vdash\!\!\rightarrow Conflict_{Policy} \end{cases}$

$PS=$

$\{(Request_{Policy}^{S0}, Request_{Policy}^{S1}), (Response_{Policy}^{S0}, Response_{Policy}^{S1}), (Input_{Policy}^{S0}, Input_{Policy}^{S1}), (Output_{Policy}^{S0}, Output_{Policy}^{S1}), (Response_{Policy}^{S0}, Response_{Policy}^{S2}), (Target_{Policy}^{S0}, Target_{Policy}^{S2})\}$

$FS=\{FSException, FSVariable, FSCS\}$

5 案例应用

“汽车装配流水线管理系统”是我们与某公司合作研发的“面向服务的制造执行系统平台软件”项目的示范性应用系统。系统基于 R2E 平台，使用 BPEL 流程驱动整个汽车的装配过程，各工序具体装配操作采用 Web 服务技术实现。本节以“汽车装配流水线管理系统”的“整车装配工位检测流程”作为研究应用案例，介绍基于 BPEH/PDL 的 BPEL 流程异常处理策略的描述方法。

5.1 BPEH/PDL 策略实例描述

“整车装配工位数据采集流程”是“R2E 关键件追溯系

统”的重要数据采集业务流程之一，通过采集整车装配过程中各工位数据，实时地检测整车装配情况，其对应的 BPEL 流程如图 3 所示。

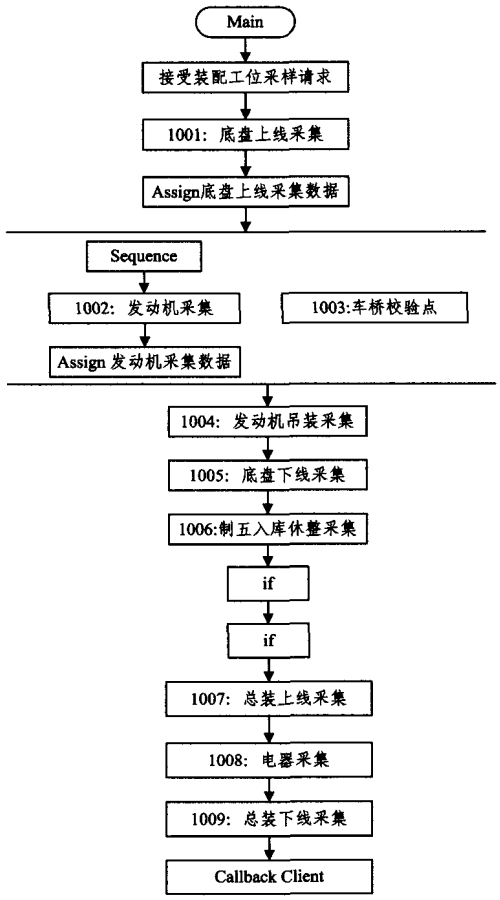

图 3　整车装配工位数据采集 BPEL 流程

5.1.1　异常类型

“整车装配工位数据采集”流程执行过程由于运行环境、伙伴服务和业务流程逻辑设计等故障会引发多种异常。主要异常场景包括：①采集点未启用；②不正确的 SN；③不正确的作业调度信息；④错误的采集基本信息；⑤采集设备故障；⑥数据采集故障；⑦数据检查或数据校验不合格。分析上述场景得到实例异常类型，如表 1 所列。

表 1　实例异常类型

异常类型	异常说明
UnRegistered	采集点未注册异常
InvalidSN	SN 输入异常
InValidSchem	作业调度异常
InCompleteBasic	采集基本信息不完备异常
UnavailService	采集服务不可用
InVerification	完工时数据校验异常
InvalidSystem	完工时系统异常
TimeOut	采集超时
SLA	采集数据不满足 QOS 要求
MissingPieces	缺件异常
ThaningPieces	多件异常

5.1.2　异常处理动作定义

根据表 1 的异常类型信息，定义了每类异常所适用的异常处理动作，生成了异常-动作关系矩阵（见表 2），用以指导异常处理规则的设计。

表 2　异常-动作关系矩阵

异常类型	重试	忽略	替换	补偿	Call	取消	报警
UnRegistered						√	√
InvalidSN	√					√	√
InCompleteBasic	√					√	√
UnavailService	√		√		√	√	√
TimeOut		√				√	√
SLA	√	√	√			√	√
InvalidLable					√		√
MissingPieces	√	√		√	√	√	√

5.1.3　异常处理规则描述

针对“采集服务不可用异常”，定义其异常处理规则如下。

```
〈Rule RuleId="R2E4MESP01R01"〉
    〈Condition〉$ SchemaStatus="Planning"〈/Condition〉
        〈PreCondition〉$ DeviceStatus="Ready"〈/PreCondition〉
        〈Retry Count=3 Interval=300/〉
        〈PostCondition〉$ DeviceStatus="Active"〈/PostCondition〉
〈/Rule〉
〈Rule RuleId="R2E4MESP01R02"〉
    〈Condition〉$ SchemaStatus="Planning"〈/Condition〉
      〈PreCondition〉$ DeviceStatus="Ready"and $ ExceptionType
      =Collecting::UnavailService
      〈/PreCondition〉
        〈Alternate
        wsdl="http://r2e4mes. com/CollectingService"
        operation="Collecting"/〉
        〈PostCondition〉$ DeviceStatus="Active"〈/PostCondition〉
〈/Rule〉
〈Rule RuleId="R2E4MESP01R03"〉
    〈Condition〉$ SchemaStatus="Planning"〈/Condition〉
      〈PreCondition〉$ ExceptionType=Collecting::MissingPiecies
      〈/PreCondition〉
    〈Alternate
    wsdl=http://r2e4mes. com/CollectingService
    operation="CheckMissingPiecies"/〉
    〈PostCondition〉$ DeviceStatus="Active"〈/PostCondition〉
〈/Rule〉
〈Rule RuleId="R2E4MESP01R04"〉
    〈Condition〉$ SchemaStatus="Planning"〈/Condition〉
        〈PreCondition〉$ DeviceStatus="Ready"〈/PreCondition〉
    〈Ignore〉
        〈PostCondition〉True〈/PostCondition〉
〈/Rule〉
〈Rule RuleId="R2E4MESP01R05"〉
    〈Condition〉$ SchemaStatus="Planning"〈/Condition〉
    〈PreCondition〉$ DeviceStatus="Ready"and $ ExceptionType=
     Collecting::UnavailService
  〈/PreCondition〉
    〈EHSequence〉
      〈Retry Count=3 Interval=100/〉
      〈Alternate wsdl=http://r2e4mes. com/CollectingService
       operation="Collecting"/〉
      〈/EHSequence〉
      〈PostCondition〉$ DeviceStatus="Active"〈/PostCondition〉
〈/Rule〉
```

5.1.4 策略及策略集描述

根据上述对异常类型、动作和规则的定义，“采集服务不可用”(UnavailService)异常处理策略及策略集定义如下：

```
〈Policy PolicyId="R2E4MESP01"
RuleCombingAlg="R_FIFO"Priority=10〉
    〈Target〉
        〈Exception ExceptionType="Collection::UnavailService"/〉
        〈Exception ExceptionType="Collection::MissingPicies"/〉
    〈/Target〉
    〈Rule RuleId="R2E4MESP01R01"〉
    〈Rule RuleId="R2E4MESP01R02"〉
    〈Rule RuleId="R2E4MESP01R03"〉
    〈Rule RuleId="R2E4MESP01R04"〉
    〈Rule RuleId="R2E4MESP01R05"〉
〈/Policy〉
〈Policy PolicySetId="R2E4MES" PolicyCombiningAlg="P_FIFO"?〉
    〈Variables〉
        〈Variable name="SchemStatus" type="QName"/〉
        〈Variable name="DeviceStatus" type="QName"/〉
    〈/Variables〉
    〈Policy PolicyId="R2E4MESP01"〉
    〈Policy PolicyId="R2E4MESP02"〉
〈/PolicySet〉
```

5.2 BPEH/PDL 策略实例 CPN 模型

以“采集服务不可用”异常为例，基于 BPEH/PDL 的 CPN 模型构建方法，我们设计了案例应用的 CPN 模型。

5.2.1 BPEH/PDL 策略集的 CPN 模型

异常处理策略集(R2E4MES)(见图 4)由 R2E4MESP01 和 R2E4MESP02 策略组成。

colset POLICY=index Policy with 1..PSIZE;

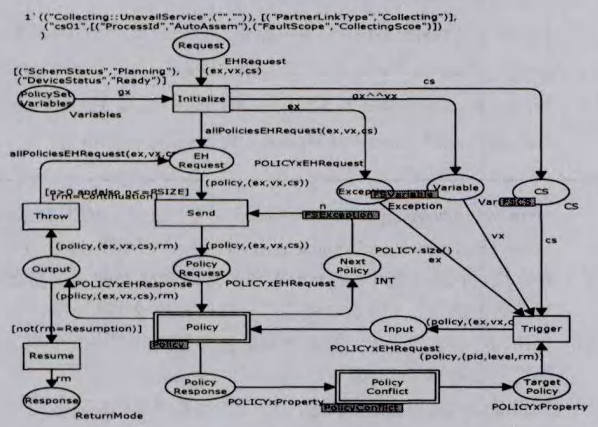

图 4 实例的策略集 CPN 模型

5.2.2 BPEH/PDL 策略的 CPN 模型

策略 R2E4MESP01 的策略目标目标 Target={UnavailService,MissingPiecies}，R2E4MESP01 的规则规模为 5，其 CPN 模型如图 5 所示。

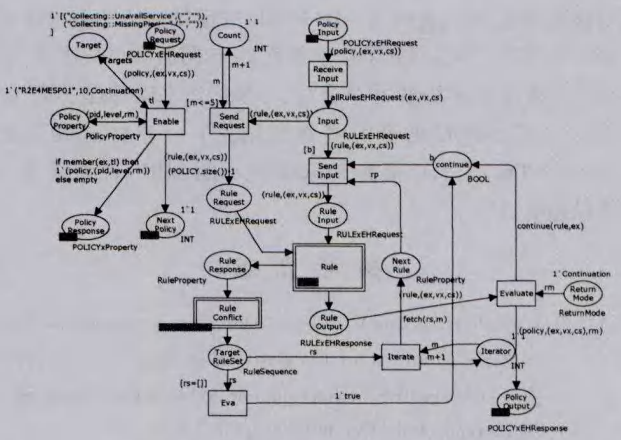

图 5 实例的 R2E4MESP01 策略的 CPN 模型

5.2.3 BPEH/PDL 规则的 CPN 模型

以策略 R2E4MESP01 的规则 R2EMESP01R01 为例，给出异常处理规则的 CPN 模型，如图 6 所示。

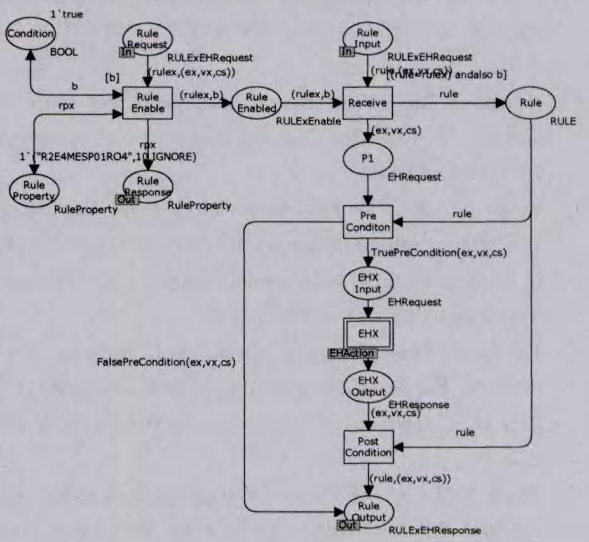

图 6 规则 R2E4MESP01R01 的 CPN 模型

5.2.4 BPEH/PDL 动作的 CPN 模型

以规则 R2E4MESP01R05 为例，给出 BPEH/PDL 规则动作的 CPN 模型，如图 7 所示。

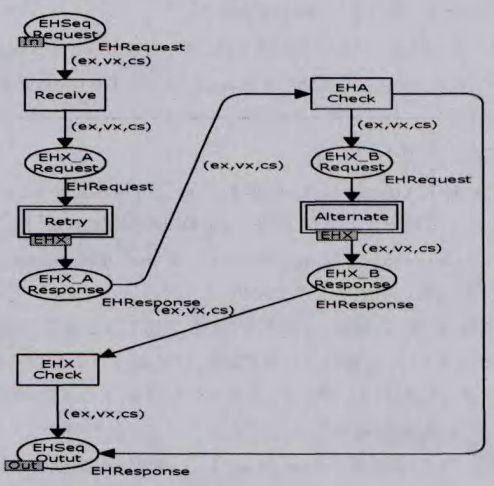

图 7 规则 R2E4MESP01R05 动作的 CPN 模型

结束语 针对策略驱动的 BPEL 流程异常处理方法的策

略描述问题，本文提出了一种 BPEL 流程异常处理策略描述语言 BPEH/PDL，研究了基于着色 Petri 网的 BPEH/PDL 语言的形式化描述方法，并结合“汽车装配流水线管理系统”给出了 BPEH/PDL 语言的实例应用。下一步将继续完善 BPHE/PDL 语言，以提高 BPEH/PDL 的 BPEL 流程异常处理描述能力。

参 考 文 献

[1] Tartanoglu F, Issarny V, Romanovsky A, et al. Coordinated Forward Error Recovery for Composite Web Services[C]// Proceedings of 22nd IEEE Symposium on Reliable Distributed Systems. Florence, Italy, October 2003: 167-176

[2] OASIS. BPEL2.0. Web Services Business Process Execution Language Version 2.0[EB/OL]. http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html

[3] Curbera F, Khalaf R, Leymann F, et al. Exception Handling in the BPEL4WS Language[C]// International Conference on Business Process Management. EINDHOVEN, NETHERLAND, 2003: 26-27

[4] Boutaba R, Aib I. Policy-Based Management: A Historical Perspective[J]. Journal of Network and Systems Management, 2007, 15(4): 447-480

[5] Zeng Liang-zhao, Lei Hui, Benatallah B. Policy-Driven Exception-Management for Composite Web Services[C]// Proceedings of the Seventh IEEE International Conference on E-Commerce Technology(CEC'05). 2005: 355-363

[6] Liu An, Li Qing, Huang Liu-sheng, et al. A Declarative Approach to Enhancing the Reliability of BPEL Processes[C]// 2007 IEEE International Conference on Web Services. 2007: 272-279

[7] Erradi A, Maheshwari P, Tosic V. Recovery Policies for Enhancing Web Services Reliability[C]// IEEE International Conference on Web Services(ICWS'06). 2006: 189-196

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

[9] Kim K, Choi I, Park C. A Rule-based Approach to Proactive Exception Handling in Business Process[J]. Expert Systems with Applications, 2010, 38(1): 394-409

[10] Chomicki J, Lobo J, Naqvi S. Conflict Resolution Using Logic Programming[J]. IEEE Transactions on Knowledge and Data Engineering, 2003, 15(1): 244-249

[11] Montangero C, Marganiec S R, Semini L. Logic-based conflict detection for distributed Policies [J]. Fundam. Inform. (FUIN), 2008, 89(4): 511-538

[12] Kolovski V, Parsia B, Katz Y. Representing WEB Service Policies in OWL-DL [C]// Proc. of the 4th International Semantic Web Conference(ISWC'05). Galway, Ireland, 2005: 461-475

[13] Kolovski V, Parsia B, Katz Y, et al. Expressing WS policies in OWL [C]// Proc. of WWW 2005 Workshop on Policy Management for the Web. Chiba, Japan, 2005: 29-36

[14] 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

[15] Hughes G, Bultan T. Automated Verification of XACML Policies Using a SAT Solver[J]. International Journal on Software Tools for Technology Transfer, 2008, 10(6): 503-520

[16] Huang He-jiao, Kirchner H. Formal Specification and Verification of Modular Security Policy based on Colored Petri Nets [J]. IEEE Transactions on Dependable and Security Computing, 2011, 8(6): 852-865

[17] 孙瑞志，史美林. 工作流异常处理的形式描述[J]. 计算机研究与发展，2003，40(3)：393-397

[18] Hamadi R, Benatallah B. A Petri Net-Based Model for Web Service Composition [C]// Proc. 14th Australasian Database Conf. Database Technologies. ACM Press, 2003: 191-200

[19] Hamadi R, Benatallah B, Mejahed B. Self-adapting recovery nets for policy-driven exception handling in business processes[J]. Distributed and Parallel Databases, 2008, 23(1): 1-44

[20] Peterson J L. Petri Net Theory and the Modeling of Systems [M]. Prentice Hall, 1981

[21] Jensen K. Coloured Petri nets and the invariant method [J]. Theoretical Computer Science, 1981, 14(3): 317-336

[22] Jensen K. Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use[M]. Springer, 1992

[23] Jensen K. Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use[M]. Springer, 1994

[24] Jensen K. Condensed state spaces for symmetrical coloured Petri nets[J]. Formal Methods in System Design, 1997, 9(1/2): 7-40

[25] Jensen K, Kristensen L M. Coloured Petri Nets Modelling and Validation of Concurrent Systems[M]. Springer, July 2009

(上接第 207 页)

[3] Ak M I, George L, Govind K, et al. Threshold Based Kernel Level HTTP Filter (TBHF) for DDoS Mitigation[J]. International Journal of Computer Network and Information Security (IJCNIS), 2012, 4(12): 31-39

[4] 刘陶，何炎祥，熊琦. 一种基于 Q 学习的 LDoS 攻击实时防御机制及其 CPN 实现[J]. 计算机研究与发展，2011，48(3)：432-439

[5] 黄亮，冯登国，连一峰，等. 基于神经网络的 DDoS 防护绩效评估[J]. 计算机研究与发展，2013，50(10)：2100-2108

[6] Bao N, Kreidl O P, Musacchio J. A network security classification game[M]// Game Theory for Networks. Springer Berlin Heidelberg, 2012: 265-280

[7] Bommannavar P, Alpcan T, Bambos N. Security risk management via dynamic games with learning[C]// 2011 IEEE International Conference on Communications (ICC). IEEE, 2011: 1-6

[8] 陈永强，付钰，吴晓平. 基于非零和攻防博弈模型的主动防御策略选取方法[J]. 计算机应用，2013，33(5)：1347-1349，1352

[9] 姜伟，方滨兴，田志宏，等. 基于攻防博弈模型的网络安全测评和最优主动防御[J]. 计算机学报，2009，32(4)：817-827

[10] 林旺群，王慧，刘家红，等. 基于非合作动态博弈的网络安全主动防御技术研究[J]. 计算机研究与发展，2011，48(2)：306-316

[11] 张义荣，鲜明，王国玉. 一种基于网络熵的计算机网络攻击效果定量评估方法[J]. 通信学报，2004，25(11)：158-165

[12] Watkins C J C H, Dayan P. Q-learning[J]. Machine learning, 1992, 8(3/4): 279-292

[13] Littman M L. Markov games as a framework for multi-agent reinforcement learning[C]// ICML. 1994, 94: 157-163