EZMS:支持EZ-Flow的工作流管理系统

陈彦光[1],徐 玮[1],张 亮[1],苏建文[2]
(1.复旦大学 计算机科学技术学院,上海 200433;
2.加州大学圣芭芭拉分校 计算机科学系,美国 加州 93106-5110)

摘 要:业务流程与服务的结合为企业实现跨部门的业务自动化和业务目标的达成奠定了基础。在竞争日益激烈,服务计算和业务流程迅速普及的应用环境中,企业不仅要实施业务工作流的流程定义、建模、测试及执行,还需要完善的业务流程管理(BPM)功能:在运行时对业务数据及运行状态进行查询监测,并能根据业务变化对工作流做出及时响应和调整。近年来广泛关注的业务物件(business artifacts或artifacts)同时包含业务数据和流程执行状态,自然地反应出业务数据语义及运行逻辑。以artifacts为核心概念的EZ-Flow工作流模型不仅易于业务专家理解建模,并且支持执行时的即席查询(ad-hoc query)。文中实现了支持EZ-Flow的工作流管理系统(EZMS),以XML表示EZ-Flow的模型定义,执行引擎以单一任务调度器完成基于事件的任务调度,任务执行器则以多线程方式并行处理人工或自动任务(Web服务),业务流程及其运行状态基于Hibernate和DB2实现关系数据和XML文档的存储及访问,并在运行时允许用户借助SQL和Xpath完成即席查询和监测。通过实现EZ-Flow的执行语义,EZMS使得流程动态改变易于实现。

关键词:工作流;业务流程管理;业务物件;EZ-Flow;业务流程管理系统
中图分类号:TP31 **文献标识码**:A **文章编号**:1673-629X(2012)12-0001-06

EZMS:A Workflow Management System for EZ-Flow

CHEN Yan-guang[1],XU Wei[1],ZHANG Liang[1],SU Jian-wen[2]
(1. School of Computer Science, Fudan University,Shanghai 200433,China;
2. Department of Computer Science, University of California at Santa Barbara, California 93106-5110,USA)

Abstract:The combination of workflow techniques and Web service serves as a basis for automation of business processes that span multiple departments for specific business goals. With the prevalence of SOC (Service Oriented Computing) and business processes,organizations need not only defining,modeling,testing and executing business process,but also complete functions of BPM (Business Process Management) including,in particular,querying business data and process status at runtime and rapidly responding to process changes. In recent years,artifact-centric workflow that naturally reflects the core business data and process logic has emerged as a prominent approach to modeling and development of business workflow. As an artifact-centric workflow model,EZ-Flow is easily understood by business stakeholders and enables support for ad-hoc query during execution. It reports an implementation of a workflow management system named EZMS (EZ-Flow Management System). The execution engine of EZMS includes a scheduler to handle events and dispatch to task performer to execute human tasks or Web service tasks in a multi-threaded fashion. EZMS uses Hibernate with DB2 to store artifact instances and workflow execution states using relation as well as XML store;it supports ad-hoc query on these data at runtime using SQL and Xpath. Through supporting EZ-Flow workflow schema creation,execution,and execution monitoring,EZMS also makes dynamic modification of workflows easy.

Key words:workflow;business process management;business artifact;EZ-Flow;workflow management system

0 引 言

随着企业流程的日趋复杂化,迫切需要对业务流程进行快速灵活建模和综合管理。传统工作流方法难以满足这些需求。近年来,以artifact为中心的工作流模型方法,将掩盖在活动之下的业务数据提升到同样高度,围绕业务数据生命周期建模[1],进一步将数据与流程运行逻辑结合,给企业业务流程管理提供有力保障,帮助企业回避错误风险[2],成为研究热点。

然而迄今为止国内外的研究成果主要集中在建模方法和原理上,尚未形成完整的知识体系,缺乏一套切实可用的工作流管理系统。

收稿日期:2012-05-15;**修回日期**:2012-08-21
作者简介:陈彦光(1988-),男,硕士研究生,CCF会员,研究方向为以数据为中心的业务流程管理;张 亮,博士,教授,博士生导师,IEEE学会会员,ACM会员,研究方向为Web与服务计算、以数据为中心的业务流程管理;苏建文,博士,教授,博士生导师,IEEE学会会员,ACM会员,SIGMOD会员,Computer Society会员,研究方向为以数据为中心的业务流程管理。

文中扩展了前期工作 ArtiFlow 引擎[3]，设计并实现了一个新的支持 EZ-Flow[4] 的工作流管理系统 EZMS，具备以下新特性：

1）以事件为主要流程控制手段；

2）流程控制和任务执行分离；

3）数据和流程状态统一管理，支持复杂数据类型；

4）实现动态业务数据和状态的即席查询；

5）可扩充支持运行时工作流动态变化。

1 相关工作

IBM 的研究人员提出了 business artifact 的概念以及围绕生命周期建模的方法[2]。随后 artifact 的理论模型被提出[5-7]，这些工作在流程持久性、独特性、可达性和可验证性等方面进行了深入的挖掘。严志民等将 artifact 与房管实践结合，提出了一套自底向上的申明式建模方法 Dart[8]。Xu 等提出的 DEZ-Flow[4] 在 EZ-Flow 的基础上把动态变化与流程基础定义分离，以申明性规则管理变化，以过程性描述定义流程，支持工作流执行时的动态变化。Hull 等提出了另一种 artifact 建模方法 GSM[9]，采用 Guard，Stage 及 Milestone 来定义流程生命周期，并在文献[10]中研究了流程交互和数据监控的问题。此外刘国华等研究了 artifact 工作流执行检测的问题[11]。Artifact 工作流系统实现的前期工作还包括基于消息中间件 MoM 的实现方案 ArtiMT[12]，ArtiFlow 引擎设计与实现[3]。

文中工作 EZMS 在模型理论上基于 EZ-Flow 设计了 XML 定义，执行时基于事件驱动，支持并发的交错语义对并行任务和人工任务提出了全新的设计。并提供运行时管理的有效手段。

2 事例分析

本节以某市住房保障和房产管理局的商品房预售许可证行政审批流程 CAPA（Certificate Approval for the Preselling of Apartments）为例，说明研究动机。

CAPA 包含 7 个主要活动：受理窗口“收件”，启动流程；在“初审”、“复审”、“终审”三个环节中，审核项目的预售面积是否超过规划面积、是否配置了物业用房和拆迁安置用房等；批准之后，开发商“付费”和房管局“制证”同时进行；完成之后，房管局向申请者“发证”并对所有审批内容进行归档。

传统方法中，流程执行状态和业务数据区隔在不同层次上，缺乏统一模型，使得业务数据与执行状态的综合查询难以实现，流程监管与生产系统环境紧耦合。例如，在 CAPA 审批环节中，业务逻辑规定房屋的预售面积不能超过规划面积。实际中，一次规划的土地常常分期建设和申报预售。假如一块土地的总体规划面积为 10000 平方米，而开发商对这块土地上的房屋进行两次 6000 平方米的房产预售申报，若房管工作流系统不能跨实例查询，则在审批环节将会批准这两个申请，违背了业务逻辑。另一个典型的情况是业务人员需要查询一次性预售申报面积超过 10000 平方米的业务审批状况。此类既包含业务数据又含有流程状态的综合（即席）查询和管理在现有流程控制层面或者数据层面下都难以独立完成。

EZ-Flow 方法[4] 解决了这种不一致的问题，为数据与流程提供统一模型，易于理解、便于监控管理，完善了工作流建模的语义。

3 EZ-Flow 工作流模型

CAPA 的 EZ-Flow 工作流模型描述如图 1 所示。图中，流程围绕核心 artifact—PAF（Presale Application Form）的生命周期展开。BL（Building List）是一个外部 artifact，被 CAPA 流程中的一部分任务（task，矩形表示）访问。流程中任务由事件（event，如 E1-E5）驱动并在完成时又可能产生新事件。通过任务处理，artifact 从一个仓库（repository，圆形表示）被运送到另一个仓库，代表了流程执行状态的变化。

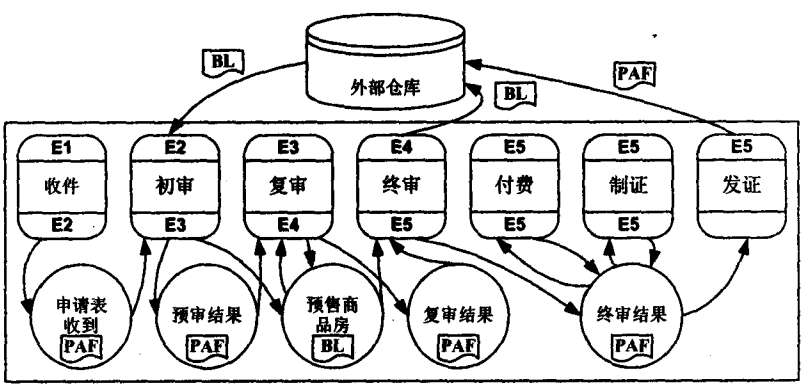

图 1 CAPA 的 EZ-Flow 流程图

下面将具体描述 EZ-Flow 模型各个部分的形式化定义及其据此设计的 XML 格式定义。

定义 1 Artifact 类型是一个元组 $c=(n,A,R)$，其中 n 表示 artifact 的名称，每个名称都是唯一的。A 是一个有限二元组 (a,t) 的集合，其中 a 为属性名称，t 定义属性类型。R 是一个有限状态集合，每个 R 中的元素表示一个 artifact 的状态，映射到工作流中一个 artifact 所能到达的任务或仓库。

根据定义 1，用 XML 语言设计定义 artifact 包含在 <artifactClasses> 标签中。支持复杂数据类型，以嵌套

结构表示。例如 CAPA 中 PAF 的定义框架如下:

```
< PAF >
<ProjectType type = "String" ></ProjectType>
…
<Comments>
<PreComment type = "String" ></PreComment>
</Comments>
</PAF>
```

定义 2 任务(task)是一个元组 $t = (n, p, a, I, O)$,其中 n 表示任务名。p 表示任务类型,分为人工任务、自动任务两类。a 表示任务执行角色。I 和 O 是两个有限的 artifact 集合,分别表示任务的输入输出集合。

每个任务定义都包含在<tasks>标签下。人工任务包含执行角色,用于控制资源的访问权限。自动任务采用 Web 服务来实现,包含 wsdl 地址、service、port、operation 名称和 I/O 与 artifact 域的映射关系。为提高效率,任务定义还需明确 artifact 的具体属性域。如 CAPA 中一个 Task 的例子是:

```
<task name = "Approval" type = "HT" actor = "manager" >
<inputset>
<artifact type = "PAF" repository = "AppReviewed" guard = "id
=COREID" >…</artifact>
</inputset>
<outputset>
<artifact type = "PAF" repository = "FinalApproved" >
…</artifact>
</outputset>
</task>
```

定义 3 事件(Event)是一个元组 $e = (t, P, C)$,其中 t 表示事件的类型。P 为一组 artifact 属性的集合,表示事件参数。C 为事件的关联定义,将一个事件映射到一个 EZ - Flow 的流程实例。

事件的 XML 定义包含在<events>标签下,包括类型、参数和关联关系:流程名、流程 id、核心 artifact 的 id。CAPA 中一个 Event 的例子是:

```
<event type = "E5" >
<parameter>PAF/Fee,PAF/Presale PermitNo</parameter>
<correlation></correlation>
</event>
```

设计中以<triggers>标签标示事件的触发,每个元素(trigger)包含一个事件及所触发的任务和条件。用<producers>标签表示事件的产生,每个元素(producer)为一个任务及其产生的事件类型。例如 CAPA 中的一个事件触发和产生定义如下:

```
<trigger>
<event>E5</event>
<condition>|PAF/Fee| = [ ]</condition>
<action>PaymentProcessing</action>
</trigger>
<producer>
<task>FinalApproval</task>
<event>E5</event>
</producer>
```

定义 4 仓库(Repository)是一个三元组 $r = (n, p, c)$,其中 n 表示仓库的名字。p 表示仓库类型,分内部或外部两类。c 映射为一个 artifact 类型。表示该仓库所存放的 artifact 类型。

根据定义 4,仓库在 XML 中定义在<repositories>标签下。如 CAPA 中一个仓库可如下定义:

```
<repository name = "AppFormReceived" type = "internal" >
<artifact type = "PAF" />
</repository>
```

定义 5 EZ-Flow 的一个 artifact 工作流模型(workflow schema)是一个元组 $(C, \Gamma, R, E, \Sigma, T, P)$,其中 C 是工作流的核心 artifact。Γ 是外部 artifact 的集合。R 是仓库的集合。E 是事件的集合。Σ 是任务的集合。T 和 P 为事件的触发和产生集合。T 是一系列三元组 (e, t, g) 的集合。其中,$e \in E$ 同时 $t \in \Sigma$,而 g 表示由 e 到 t 的条件,这个条件以布尔表达式的形式出现。P 是一系列二元组 (t, e) 的集合。其中,$e \in E$ 同时 $t \in \Sigma$。

根据定义 5,以 CAPA 为例一个完整的 EZ-Flow 工作流的 XML 定义框架如下:

```
<? xml version = "1.0" encoding = "UTF-8" ? >
<ezflow name = "CAPA" >
<artifactClasses></artifactClasses>
<repositories></repositories>
< tasks></tasks>
<events></events>
< triggers></triggers>
< producers></producers>
</ezflow>
```

定义 6 一个 artifact 工作流系统(workflow system)是集合 S,包含一组 artifact 工作流模型。并且所有工作流模型中所涉及到的 artifact 也必定包含在 S 中,即所有 artifact 工作流模型在 S 上封闭。

4　EZMS 执行引擎的设计与实现

EZMS 的执行语义遵循 EZ-Flow 工作流模型[4],流程以事件为驱动,流程的执行可以看做一系列快照(snapshot)的变迁(transition)。在 EZMS 执行引擎的设计中,见图 2,采用单一调度器来管理事件队列,执行时每处理一个事件就分派一个任务执行器线程来处理实际的任务(人工任务或自动任务)。任务处理器自动地将快照的变迁映射为数据库的更新。任务完成

后又产生新的事件回到事件队列中,如此循环。

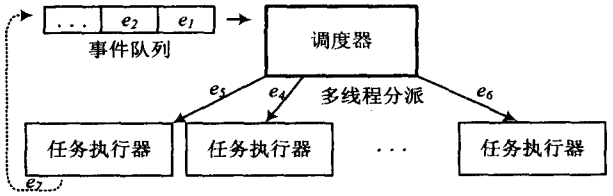

图2 EZMS执行引擎的执行语义

EZMS的实现总体采用Eclipse作为开发工具,以Java作为开发语言,使用Spring框架作为基础配置、反向注入和模块整合工具,采用流行的ORM(Object-Relation Mapping)框架Hibernate对数据库进行各项操作,数据库存储则采用DB2 Express。引擎架构如图3所示,主要分为调度器、任务执行器、数据控制器以及数据存储几个主要部分。

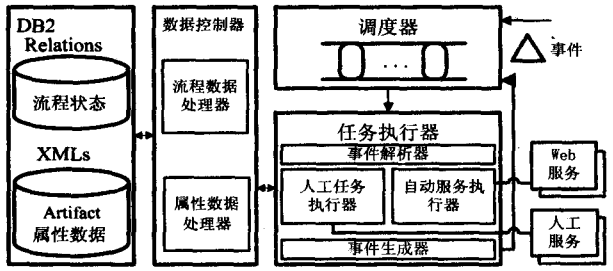

图3 EZ-Flow引擎架构

调度器是引擎的运转核心,维护一个事件队列,并产生事件对应的任务执行器线程。这种流程控制调度与任务执行分离的方式使得调度核心的设计维持轻量级,保证流程顺畅进行而不会受单个任务执行的影响整体性能。作为引擎的运转核心,事件的存储和分发需要稳定并且快速,同时要能在多线程环境下正常运作,并且需要顾及在分布式环境中运行的情况。综合这些因素,调度器的实现选择了消息中间件ActiveMQ。实现中,EZMS将ActiveMQ的管理通过Spring来设置,启动引擎时在内部开启独立的broker完成消息的存储和转发,避免ActiveMQ的独立启动。此外EZ-Flow引擎包装事件序列时采用ActiveMQ的队列机制,保证存储的消息只被消耗一次,而不会出现重复执行的状况。

任务执行器是引擎的执行核心,包含事件的生成、解析器,人工、自动任务的执行器几个部分。事件解析器负责解析事件,根据事件关联的artifact的内容确定对应任务的前置条件是否满足,触发满足条件的人工任务或Web服务。这部分设计利用Scala语言中的解析器组合子(parser combinator)来实现。特别的是该设计允许在EZ-Flow定义中,根据不同的条件触发不同任务。在任务执行完后根据任务定义将事件生成器产生的内部事件交给调度器。

数据库存储选用DB2 Express,不仅因为它的高效性,更重要的是它对关系型数据和XML文档提供了统一的支持。EZ-Flow模型中,artifact包含业务数据和流程状态两部分:业务数据部分以XML形式来存储,XML的好处是其树状结构对于表示包含多层嵌套关系的复杂数据结构很有帮助。XML的使用使得EZ-Flow引擎得以很好的支持复杂数据类型;而流程状态则需要关系数据结构存储,更易于理解。因此,引擎中存储、获取和查询数据库时,需要涉及大量的XML与SQL的互操作。DB2对复杂的XML的各种操作提供了完善的支持,有良好的SQL/XML混合支持,包括XQuery查询。

流程与数据库的交互都设计为通过数据控制器来完成。属性数据控制器提供XML结构的业务数据接口,而流程状态由流程数据处理器来完成。

下面以CAPA流程为例具体描述EZ-Flow引擎的工作过程和细节设计。任务的调度需要事件触发。假设事件队列中有一个E2类型的事件,内容如下:

```
<event type="E2">
  <parameter></parameter>
<content>PNAME=CAPA&PID=120&
          COREID=235</content>
</event>
```

这一事件包含了关联的流程实例的名称、流程实例的id以及核心artifact的id。E2首先会被调度器分派给一个任务执行器线程,根据相关Trigger(如下所示):

```
<trigger>
    <event>E2</event>
    <condition></condition>
    <action>PreliminaryDecision</action>
</trigger>
```

可知E2对应的任务活动为PreliminaryDecision初审任务。当任务执行器找到关联流程实例后便触发任务的执行。EZMS将流程执行的快照(snapshot)[4]映射到数据库,如表1所示,将在任务实例表中产生新的任务记录(id=348),同时保存任务开始时间,记录关联流程实例(pins_id=120)。接着,流程实例表也更新当前的流程状态为初审任务调度中。随后任务执行器通过数据控制器需要获取所需artifact。首先根据核心artifact的id在artifact数据表中找到PAF实例(id=235),同时根据PAF的内容在外部仓库中取出所需的外部artifact BL实例(id为167)。

在获取了所有artifact之后,任务执行器就根据不同的任务类型分别调用人工服务或Web服务。实际中人工任务常常持续几个小时甚至几天,而EZ-Flow引擎所设计采用的多线程技术允许多任务同时执行。为避免过多人工任务长期占有系统资源影响性能。

EZMS 设计中引入人工服务"中断"处理机制。人工服务启动时,引擎先将所需的数据保存到人工任务数据表中,而后便中断该线程。人工任务在用户实际调用的时候可根据存储的数据独立完成。完成后再发送一个内部事件到引擎,恢复流程执行。如上述初审例子中,人工任务启动后保存的数据如表 2 所示。

表 1　EZMS 流程执行快照

任务实例表

id	name	start	end	pins_id
348	Preliminary Decision	2011-11-10 15:34 924000	null	120

流程实例表

id	current_task_id	current_task_name
120	348	PreliminaryDecision

start	end	process_name
2011-11-10 15:32 837000	null	CAPA

Artifact 数据表

id	type	content	pins_id	repository
235	PAF	xml	120	ApplicationReceived
167	BL	xml	null	Archived

表 2　人工任务数据表

id	name	value	actor	task_id
687	PAF/ProjectType	affordable	staff	348
688	PAF/ProjectName	Sunny Apt	staff	348
…	…	…	…	…
708	BL/Area	8000	staff	348

中断恢复的内部事件包含人工任务的 id,其 parameter 字段上会带上人工服务执行后的业务数据。例如初审人工服务完成后的内部事件如下:

```
<event>
<parameter>PAF/Comments/PreComment=
OK</parameter>
<content>PNAME=CAPA&PID=120&
COREID=235&TASKID=348</content>
</event>
```

当调度器收到该内部事件并分派给一个新的任务执行器时,将会根据事件中的 TASKID 载入任务数据更新 EZMS 执行快照中的 artifact 数据表。

若为自动任务,任务执行器会调用 Web 服务,由于引擎设计时无法预知用户需要哪些 Web 服务,所以设计采用动态调用方式来实现,即由用户在定义流程时给出 Web 服务的基本信息,在运行时去解析 wsdl 来调用。EZ-Flow 引擎实现中选择了 JBoss Wise 这一框架来完成动态调用的任务。先由 Wise 来动态生成服务的 Stubs,然后通过反射技术解析函数参数,最后根据定义文件将 artifact 的对应属性的具体值带入函数参数中并完成调用。Wise 支持 RPC 和 Document 的 Web 服务形式,并且支持由 Axis2,JAX-WS 等多种方法编写的服务,对 wsdl 2.0 也有较好的支持。这样广泛的支持能保证引擎的适应性更强,在不同的生产应用环境中都可以发挥效力。

无论何种任务,任务的服务调用完成并存储 artifact 后,任务执行器都将根据任务定义产生新事件,如上述例子中的初审任务将产生一个 E3 事件发送到事件队列中:

```
<producer>
<task>PreliminaryDecision</task>
<event>E3</event>
</producer>
```

当完成所有这些之后,标注该任务已经完成并记录任务的完成时间。至此,一个任务调度执行周期结束。

5　EZMS 运行时管理:查询与动态改变

EZMS 运行时管理区别与以往工作流引擎的两个主要特性在于支持运行时对流程状态及业务数据的混合即席查询(ad-hoc query),并能够扩充支持运行时工作流执行的动态改变。

首先举例说明 EZMS 的查询特色。如房产预售中常需要查询房屋面积超过 1000 平方米的预售流程所处的审批状况。由于 EZMS 中业务数据与流程状态统一于 artifact 模型中,并实现为执行快照的三张数据表(如表 3 所示)中,因此此类查询有明确语义而独立于系统实现。

设计综合 SQL 和 XML 的混合脚本进行查询:

```
SELECT current_task_id, current_task_name, ARTIFACTS.
content
FROM PROCESS_INSTANCE JOIN ARTIFACTS
ON ARTIFACTS.pins_id = PROCESS_INSTANCE.id
WHERE ARTIFACTS.type = 'PAF' AND
XMLEXISTS('$ c/PAF[Area > 1000]' PASSING ARTI-
FACTS.content AS "c")
```

便可以得到面积超过 1000 平方米的流程实例所处的流程状态,结果如表 4 所示。

EZMS 也支持跨流程实例的查询。例如,下述查询即可统计所有处于付费环节的预售房屋面积:

```
SELECT SUM(XMLQUERY('$ c/PAF/Area' PASSING AR-
TIFACTS.content AS "c"))) AS totalArea
FROM PROCESS_INSTANCE JOIN ARTIFACTS
ON ARTIFACTS.pins_id = PROCESS_INSTANCE.id
WHERE ARTIFACTS.type = 'PAF' AND
current_task_name = 'PaymentProcessing'
```

表3 查询EZMS运行时快照

任务实例表

id	name	start	end	pins_id
348	Preliminary Decision	2011-11-10 15:34 924000	null	120

流程实例表

Id	current_task_id	current_task_name	…
120	348	PreliminaryDecision	…
123	357	PaymentProcessing	…
127	382	PaymentProcessing	…

Artifact数据表

Id	type	content	pins_id	…
235	PAF	<PAF> … <Area type = " Integer" > 1500 </Area >… </PAF>	120	…
238	PAF	<PAF> … <Area type = " Integer" > 800 </Area>… </PAF>	123	…
242	PAF	<PAF> … <Area type = " Integer" > 900 </Area >… </PAF>	127	…

表4 查询结果

current_task_id	current_task_name
348	PreliminaryDecision
Content	
<PAF> … <Area type = " Integer" >15000</Area> </PAF>	

可得查询结果为id分别为123和127的两个流程实例(表3:流程实例表)所对应的id分别为238和242的两个PAF(表3:artifact数据表)的面积之和为1700。

总之,以往工作流引擎难以完成的跨流程实例的数据查询或混合了流程状态的查询均可在EZMS中通过混合SQL/XML查询得以实现。

此外,业务工作流系统需要适应频繁的业务变化。对工作流动态改变的支持能够显著地提高企业的竞争力,节约资源和劳动力成本[4]。

EZMS实现了EZ-Flow的执行语义,任务的调度与任务的执行,数据访问基于独立的数据控制器使得工作流执行时的动态改变易于实现。在之前的工作[4]中所提出的add(新增)、replace(替换)、skip(跳过)、retract(驳回)四种运行时动态改变工作流执行的变化规则可以很方便地基于EZMS扩展规则库和规则检验得以实施。

6 结束语

根据工作流研究和实际应用的现状,针对传统工作流方法所存在的问题,文中以支持artifact为中心的业务流程管理为主要研究内容。设计并实现了支持EZ-Flow的EZMS工作流管理系统。

该系统的创新理念在于:在工作流管理中将业务数据和流程控制结合在artifact这个统一模型中,在运行时可以统一管理、即席查询,并且能够快速适应业务变化。流程的执行机制由事件驱动,将流程控制和任务执行分离的设计降低了耦合,加强了系统的稳定性和灵活性。

参考文献:

[1] Hull R. Artifact-centric Business Process Models: Brief Survey of Research Results and Challenges [C]//Proc. of OTM 2008. [s. l.]: Springer-Verlag, 2008: 1152-1163.

[2] Nigam A, Caswell N S. Business artifacts: An approach to operational specification[J]. IBM Syst. J., 2003, 42(3): 428-445.

[3] Lv Y. Design, Implementation and Applicaton of Artifact-centric Workflow Management System[D]. Shanghai: Fudan University, 2011.

[4] Xu W, Su J, Yan Z, et al. An Artifact-centric Approach to Dynamic Modification of Workflow Execution [C]//Proc. of CoopIS. [s. l.]: Spinger-Verlag, 2011: 256-273.

[5] Gerede C, Su J. Specification and Verification of Artifact Behaviors in Business Process Models[C]//Proc. of ICSOC'07. [s. l.]: Springer-Verlag, 2007: 181-192.

[6] Bhattacharya K, Gerede C, Hull R, et al. Towards Formal Analysis of Artifact-centric Business Process Models [C]// Proc. of BPM'07. [s. l.]: Springer-Verlag, 2007: 288-304.

[7] Frit C, Hull R, Su J. Automatic Construction of Simple Artifact-based Business Processes[C]//Proc. of ICDT'09. [s. l.]: [s. n.], 2009: 225-238.

[8] Yan Z, Xu W. Bottom-up workflow modeling approach for business changes[J]. Computer Integrated Manufacturing Systems, 2011, 17(8): 1595-1602.

[9] Hull R. Introducing the guard-stage-milestone approach for

(下转第10页)

中使用一个隐半马尔科夫模型来描述攻击行为的状态化的协议过程,对6LowPAN网络安全问题起到一定的积极意义。

除此以外,类似互联网中,物联网同样存在地址配置和管理,以及网络管理[19]等问题也需要实际解决。

4 结束语

随着物联网应用的普及和无处不在,寻址和标识的唯一性问题越来越成为物联网数据交换中的核心问题,受到国内外学者的广泛关注。文中从物联网寻址概念着手,在综述物联网寻址关键问题和最新研究进展的基础上,探讨了未来需要进一步研究的课题,为加快物联网的应用带来一定的研究意义。

参考文献:

[1] EPCglobal Inc. EPCgloba Object Name Service(ONS),v 1.0.1[EB/OL]. 2008. http://www.gsl.org/gsmp/kc/epcglobal/ons.

[2] 张捍东,朱 林. 物联网中的RFID技术及物联网的构建[J]. 计算机技术与发展,2011,21(5):56-59.

[3] 孔 宁. 物联网资源寻址关键技术研究[D]. 北京:中国科学院,2008.

[4] Kong Ning, Li Xiaodong, Yan Baoping. A Model Supporting Any Product Code Standard for the Resource Addressing in the Internet of Things[C]//First International Conference on Intelligent Network and Intelligent Systems. [s. l.]:[s. n.], 2008:233-238.

[5] 孔 宁,李晓东,罗万明,等. 物联网资源寻址模型[J]. 软件学报,2010,21(7):1657-1666.

[6] Piero B, Daniel O. Driving and monitoring provisional trust negotiation with metapolicies[C]//Proceedings of the 6th IEEE International Workshop on Policies for Distributed Systems and Networks. Washington, DC: IEEE Computer Society, 2005:14-23.

[7] 王守信,张 莉,李鹤松. 一种基于云模型的主观信任评价方法[J]. 软件学报,2010,21(6):1341-1351.

[8] 万年红,王雪蓉. 云信任驱动的物联网信息资源寻址模型[J]. 计算机应用,2011,31(5):1184-1188.

[9] 王淑惠,谭清中,唐 彦,等. Ipv6是物联网最佳的寻址技术[J]. 数字通信,2011(3):28-31.

[10] Durvy M, Abeille J, Wetterwald P. Maing sensor networks IPv6 ready[C]//Proceedings of the 6th ACM Conference on Embedded Networkd Sensor Systems. [s. l.]:[s. n.],2008:421-422.

[11] Bag G, Shams S M S, Akhbar A H. Network assisted mobility support for 6LowPAN[C]//Proceedings of the 6th IEEE Conference on Consumer Communications and Networking. [s. l.]:[s. n.],2009:383-387.

[12] 王忠敏. EPC与物联网[M]. 北京:中国标准出版社,2004.

[13] Shelby Z, Bormann C. 6LowPAN: The Wireless Embedded Internet[M]. [s. l.]: Wiley Publishing, 2010.

[14] Cao Zhongyu, Lu Gang. S-AODV: Sink Routing Table over AODV Routing Protocol for 6LowPAN[C]//Proceedings of the 2010 Second International Conference on Networks Security, Wireless Communications and Trusted Computing. [s. l.]:[s. n.],2010:340-343.

[15] Kim J H, Hong C S. A scheme for supporting optimal path in 6LowPAN based MANEMO networks[C]//Proceedings of the 12th Asia-Pacific network operations and management conference on management enabling the future internet for changing business and new computing services. [s. l.]:[s. n.],2009:161-170.

[16] Herber U, Clausen T. A comparative performance study of the routing protocols LOAD and RPL with bi-directional traffic in low-power and lossy networks (LLN)[C]//Proceedings of the 8th ACM symposium on performances evaluation of wireless and hoc, sensor, and ubiquitous networks. [s. l.]:[s. n.],2011:73-80.

[17] Jung W, Hong S, Ha M. SSL-based lightweight security of IP-based wirless sensor networks[C]//Proceedings of the 2009 International Conference on Advanced Information Networking and Applications. [s. l.]:[s. n.],2009:1112-1117.

[18] 刘外喜,唐 冬,胡 晓,等. 6LowPAN网络安全问题的分析[J]. 电信科学,2010(4):66-70.

[19] Chos H, Kim N, Cha H. 6LowPAN-SNMP: Simple Network Management Protocol for 6LowPAN[C]//Proceedings of the 2009 11th IEEE International Conference on High Performance Computing and Communications. [s. l.]:[s. n.],2009:305-313.

(上接第6页)

specifying business entity lifecycles[C]//Proc. of 7th Intl. Workshop on Web Services and Formal Methods (WS-FM 2010). [s. l.]:Springer,2010.

[10] Hull R, Narendra N C, Nigam A. Facilitating Workflow Interoperation Using Artifact-centric Hub[C]//Proc. of ICSOC-Service Wave'09. [s. l.]:Springer-Verlag,2009:1-18.

[11] Zhao D, Liu G, Jiang Y, et al. The execution and detection of artifact-centric business process[C]//Proc. of CSAE'11. [s. l.]:[s. n.],2011:491-495.

[12] Liu G, Liu X, Qin H, et al. Automated realization of business workflow specification[C]//Proc. of ICSOC/Service Wave 2009. [s. l.]:Springer-Verlag,2010:96-108.