Drools规则引擎在现代物流信息平台的应用

陆歌皓[1] 李仕金[2] 吴超凡[2]
(云南大学软件学院 昆明 650091)[1] (云南大学信息学院 昆明 650091)[2]

摘 要 规则引擎是一个具有非常广阔的应用前景的技术。首先介绍了系统的应用背景、采用技术以及规则引擎的相关概念。然后重点介绍了规则引擎的工作机制以及与其密切相关的专家系统。最后利用Drools规则引擎实现了实验室中物流平台的权限控制,并对Drools规则引擎的优缺点进行了阐述,对其发展趋势进行了展望。

关键词 规则引擎,Drools,RETE算法,物流信息平台

Drools Rules Engine in the Application of Modern Logistics Information Platform

LU Ge-hao[1] LI Shi-jin[2] WU Chao-fan[2]
(School of Software, Yunnan University, Kunming 650091, China)[1]
(School of Information Science and Engineer, Yunnan University, Kunming 650091, China)[2]

Abstract Rule engine is a technology which owns a very broad application prospects. First introduced its concept and function, also the application background of the system. Then mainly introduced the working mechanism of rules engine and the RBES. Finally, used the drools to solve the realization of the rule engine based on the logistics system, and then discussed the advantage and the disadvantage about the rule engine, also concerned on the future development of this technology.

Keywords Rule engine, Drools, RETE algorithm, Logistics information platform

1 引言

物流(logistics)[1]是指利用现代信息技术和设备,将物品从供应地向接收地准确、及时、安全、保质保量、门到门地传递的合理化服务模式和先进的服务流程。现代物流系统是从供应、采购、生成、运输、存储、销售到消费的供应链。开发物流系统的主要原因是提高业务人员的工作效率,让管理层能够实时掌握企业运营情况,为其决策提供科学依据和参考,最终提高企业的管理水平。

规则引擎是推理引擎发展而来的,它是这样一种组件,通过嵌入在应用程序中,从而实现了将业务决策从应用程序代码中分离,并使用预定义的语义模块编写业务决策。接受数据输入,解释业务规则,从而根据业务规则做出相应决策。由于在现代物流系统中,那些操作员的权限会出现变化,时而出现新的一些角色,因此我们把这些会出现变化的规则给抽取出来,利用规则引擎对其管理,实现业务与逻辑的分离。

在物流信息平台中由于各个角色登录后的功能会出现变化,尤其对于那些业务比较复杂又经常出现变更的,需要将其抽取出来,文章将从分析规则引擎的工作机制出发,并且利用Drools规则引擎在物流信息平台上来实现这一优点。

2 背景

2.1 物流信息平台背景

随着科技的日益进步,近年来物流业处于较快增长阶段,物流基础设施、综合运输网络体系都具有一定的规模,交通运输、仓储等传统物流业蓬勃发展,代表现代物流组织形式的第三方快速兴起,发展现代物流的基础条件已经具备。

但是大多数物流公司都是从传统的运输企业和仓储中发展而来,信息现代化技术比较落后。比如热门的RFID技术、GPS技术。一些常用的物流管理软件在物流领域中的应用水平也比较低。以XML技术、规则引擎、互联网等为基础的物流信息系统还远远没有得到开发。所以建设现代物流信息平台的一个主要目的是将物流、资金流、信息流等整合为一体,提升物流信息平台,推进物流的标准化建设。

2.2 两种主流规则引擎简介

目前主流的两个规则引擎主要是:Sandia National实验室的Jess规则引擎,开源软件的代表Drools规则引擎。

Jess的表达式形式采用CLIPS的语法结构,通过对规则的前件和后件的条件限定,使其支持内容丰富的模式匹配语言。此外,Jess支持面向过程的编程方式,通过提供一些语句来控制规则后件的操作流程,通常使用while…do或者if…then…else形式,它可以充分利用面向过程的优点。总之,

本文受工业与信息产业部电子信息发展资金项目资助。

陆歌皓 男,副教授,主要研究方向为Agent、分布式计算;**李仕金**(1984—),男,硕士,主要研究方向为人工智能、服务计算;**吴超凡**(1987—),男,硕士,主要研究方向为人工智能、服务计算。

Jess 的这些特性使系统拥有很强的知识表示能力。

Drools：Drools [2] 是一个具有易于访问企业策略、易于调整以及易于管理的开源业务规则引擎，符合业内标准，速度快、效率高。业务分析师或审核人员可以利用它轻松查看业务规则，从而检验是否已编码的规则执行了所需的业务规则。Drools 是为 Java 量身定制的基于 Charles Forgy 的 RETE 算法的规则引擎的实现。具有 OO 接口的 RETE，使得商业有了更自然的表达。

Drools[3] 提供了声明式程序设计，并且使用域描述性语言为问题域定义了某种模式的 xml，从而可以描述用户问题域，域描述语言包含的 xml 元素（element）和属性（attribute）代表了问题域中各种元素。例如：

```
〈xs:element name="ruleset"〉
    〈xs:complexType〉
        〈xs:sequence〉
            〈xs:element ref="package"/〉
            〈xs:element ref="global"/〉
            〈xs:element ref="rule" maxOccurs="unbounded"/〉
        〈/xs:sequence〉
        〈xs:attribute name="description" type="xs:string"/〉
        〈xs:attribute name="name" type="xs:string"/〉
    〈/xs:complexType〉
〈/xs:element〉
```

通过上述两种典型规则引擎的比较，由于 Drools 是一个用 Java 语言编写的开放源码规则引擎，使用 Rete 算法对所编写的规则求值。它允许使用声明方式表达业务逻辑。可以使用 Java/XML 语法编写规则，还可以使用 Groovy/XML 语法或 Python/XML 语法在 Drools 中编写规则。此外它还具有易用、快速的执行速度、兼容 JSR94 等优点。主要的一点是它完全免费，基于上述原因我们物流信息平台采用了 Drools 规则引擎。

3 Drools 规则引擎的工作机制

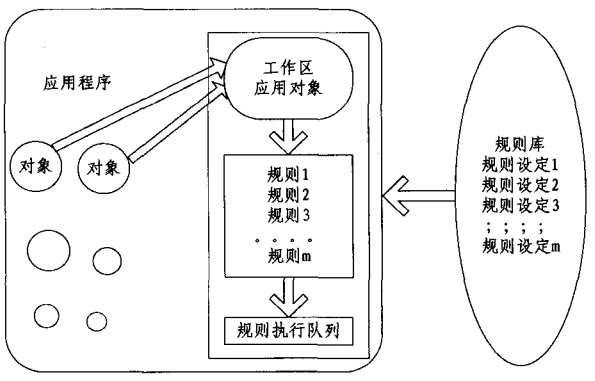

图 1 规则引擎结构示意图

规则引擎的工作原理是对提交给引擎的数据对象进行检索，然后根据这些对象的当前属性值和它们之间的关系，从加载到引擎的规则集中发现符合条件的规则，创建这些规则的执行实例，这些实例将在引擎接到执行指令时，依照某种优先顺序依次执行，一般规则引擎内部由下面几个部分组成：工作内存，用于存放被引擎引用的数据对象集合；规则执行队列，用于存放被激活的规则执行实例；静态规则区，用于存放所有被加载的业务规则，这些规则将按照某种数据结构组织，当工作区中的数据发生改变后，引擎需要迅速根据工作区中的对象现状，调整规则执行队列中的规则执行实例。规则引擎的结构示意图，如图 1 所示[4]。

3.1 基于规则的专家系统简介

Java 规则引擎起源于基于规则的专家系统，专家系统属于人工智能的范畴，它通过模仿人类的推理方式，使用试探性的方法进行推理，并使用人类能理解的术语解释和证明它的推理结论。

规则引擎[5] 其实是人工智能技术的分支，它实现了业务逻辑的分离，主张业务逻辑从我们的应用编码中解脱出来，便于我们业务逻辑的修改。它真正的核心功能是对复杂的信息进行智能分析和判断最后给出决策。它是基于规则的专家系统的一部分，为了更深刻地了解规则引擎，必须对基于规则的专家系统（RBES）作深入的了解。RBES 的结构图如图 2 所示。

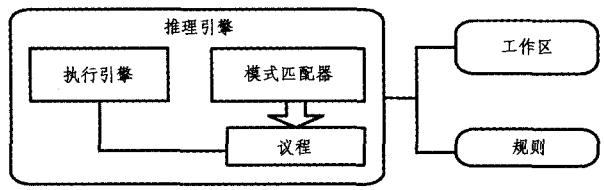

图 2 基于规则的专家系统构成

如图 2 所示，推理引擎包括 3 部分：模式匹配器、执行引擎、议程。它们各自的主要作用如下[6]。

推理引擎（Inference engine）：通过决定哪些规则满足事实或目标，并授予规则优先级，满足事实或目标的规则被加入议程。

模式匹配器（Pattern matcher）：决定执行哪个规则，何时执行规则。

议程（Agent）：由推理引擎创建一个规则优先级表，通过议程管理模式匹配器挑选出来的规则的执行顺序。

执行引擎（Execution engine）：负责执行规则和其他动作。

3.2 RETE 算法简介

RETE 算法是由 Charles Forgy 博士于 1979 年提出的，它是目前效率最高的一个前向链形推理算法，其核心思想是将分离的匹配项根据内容动态构造匹配树，以达到显著降低计算量的目的。它主要采用的是一种基于推理引擎的时间冗余性和规则结构的相似性，通过保存中间去处来提高效率的一种模式匹配算法。

3.3 Drools 规则描述语言

Drools 采用原生的规则语言，是一种非 XML 文本格式，它的规则结构非常简单。规则详解如图 3[7] 所示。

attributes 是简单的、可选的、用以提示规则的行为方式。LHS 是规则的条件部分，需要按照一定的语法来写。RHS 基本上是一个允许执行 Java 语法的代码的块，任何在 LHS

中使用的变量都可以在 RHS 中使用。

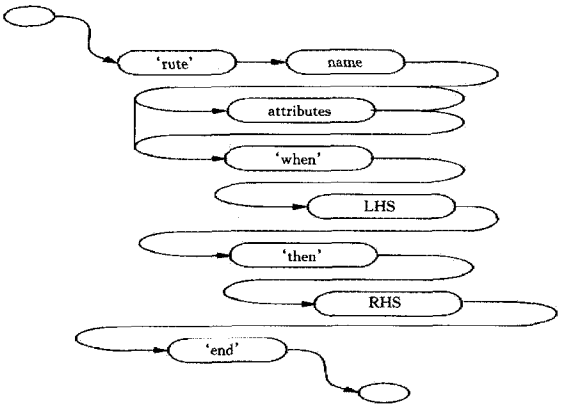

图 3　规则详解

4　物流平台下 Drools 规则引擎应用方案

该物理平台系统要求 Drools 实现类似如下规则组成，如图 4 所示。

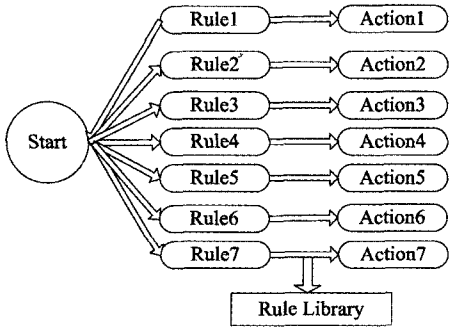

图 4　物流平台规则组件

具体实现功能描述如表 1 所列。

表 1　角色功能划分

Role name	Role Description	Operation
超级管理员	超级管理员	运单、客户信息、驾驶员、派车单、角色、大宗货物等的相关操作
财务	回单结算	对网点、运输线路的查询，以及运单的相关操作
开单员	开单员	对运单、网点、运输、客户信息、客户货运等的相关操作
运单员	运单员	对运单、网点、运输线路、客户信息等的相关操作
派车单员	派车单员	对运单、派车单等的相关操作
普通管理员	普通管理员	对运输工具、驾驶员、运输线路、用户信息、网点等的相关操作
大宗货物管理员	大宗货物管理员	对大宗货物、大宗派车单、大宗货物清单的相关操作

4.1　物流系统的权限组成

该物流信息系统的权限设计如图 5 所示，物流系统的应用服务器采用 Tomcat 应用服务器，数据库采用 MySqL，利用 WebService 思想来实现负载均衡这一目的，我们把这个系统分成 5 层，从上至下分别为：数据库、数据访问层、业务层、表示层、客户端。其中权限控制主要在表示层、业务层、数据访问层上进行。

4.2　设计业务流程

与传统的设计模式相同，基于规则引擎的物流系统的开发模式也包含企业服务的业务流程设计。由于引入了规则引擎，使得业务流程和业务规则的分离，使得业务流程的设计得到大大的简化，业务流程中不再需要设计繁琐和庞大的条件判断，大大减轻了业务程序的负担。业务流程设计应遵从图 6 描述的服务模型[8]。

图 6 中描述的 2 部分功能介绍如下。

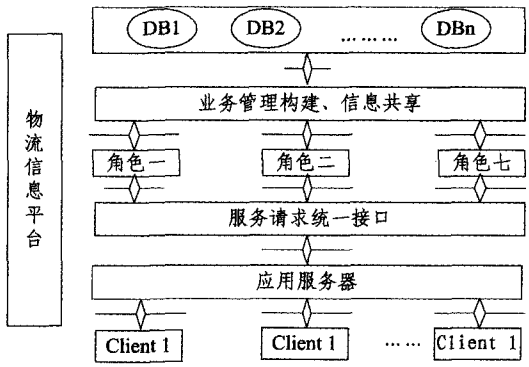

图 5　物流系统的权限设计

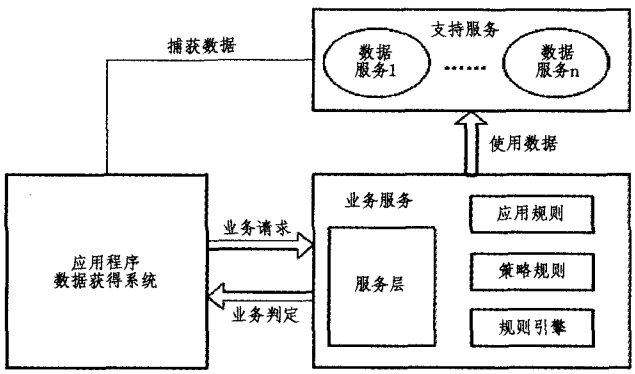

图 6　服务模型结构

应用程序/数据获得系统：捕获和存储应用程序提交的所有数据，是业务服务的使用者。主要功能是提交业务请求和处理业务判定。

业务服务：通过具体实现的可调用的网络服务器，调用设定的规则引擎来执行业务规则逻辑或对业务规则逻辑进行运算，产生反馈信息和数据。同时也提供方便和有效维护业务规则逻辑的功能。系统中采用规则引擎后使得业务流程与业务规则分离，这样更加使得业务流程设计简化，如图 7 所示。

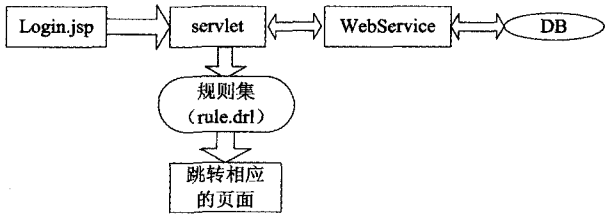

图 7　数据流程图

从图 7 中可以明确地看到该物流系统采用经典的 3 层(MVC)架构，利用传统的 JSP＋Servlet 模型，采用先进的 WebService 服务，通过 DB 与 WebService 之间的数据交换，将数据流程图中登录规则抽取出来，利用 Drools 规则引擎技术，将规则具体化。从图中可以清晰地看到，业务流程已不包

含业务逻辑，当业务逻辑改变时，不需要对业务流程作出改变，只需要修改规则文件。

4.3 业务规则代码化

代码化的过程是将物流服务具体到实现过程，换句话说就是程序代码的实现过程。在这个过程中需要遵从前序步骤的设计和结果，在此以表 1 中要求实现的例子为例，选用 Java 语言和 Drools 规则引擎来实现代码化，本文只给出需要实现的类及其描述，而不给出具体的实现代码。

支持服务部分的实现：

```
UserRoleQueryServiceStub. class
// 用户角色查找
RoleServicePageQueryServiceStub. class
//通过角色编号查找权限列表
UserLoginServiceStub. calss;
//负责用户登录的 WebService 服务
LoginResponse. class;负责用户登录后的跳转
List serviceIdList=newArrayList(Long)();
//用于保存用户的角色编号列表信息
```

规则的代码化：

以表 1 为例，其对应 Drools 规则引擎的代码如下：

```
Rule "rule1"
Salience 1 //优先级设定
When //LHS
$m;setUserLogin(state==1)
then //RHS
UserLoginServiceStub. Login login=new UserLoginServiceStub. Log-
in();
UserLoginServiceStub. UserEntity user=new UserLoginServiceStub.
UserEntity();
user. setUserLogin(state);
user. setUserPassword(pwd);
login. setUser(1);
end;
Rule "rule2"
Salience 2 //优先级设定
When //LHS
$m;setUserLogin(state==2)
then //RHS
UserLoginServiceStub. Login login=new UserLoginServiceStub. Log-
in();
UserLoginServiceStub. UserEntity user=new UserLoginServiceStub.
UserEntity();
user. setUserLogin(state);
user. setUserPassword(pwd);
login. setUser(2);
end;
```

4.4 引入规则引擎后的优势

图 8 给出传统业务逻辑编写方式，图 9 给出引入规则引擎后业务逻辑的编写方式。

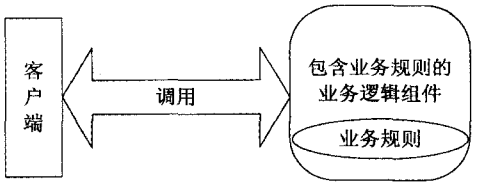

图 8 传统业务逻辑编写方式

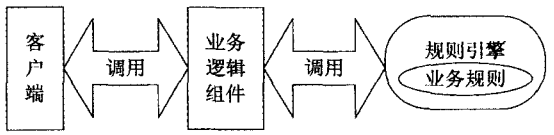

图 9 引入规则引擎后业务逻辑的编写方式

引入规则引擎后[9]，可以很好地实现业务逻辑与业务规则的分离，实现业务规则的集中管理；可以动态修改业务规则，从而快速响应需求变更；可以使业务分析人员也可能参与编辑、维护系统的业务规则；可以使用规则引擎提供的规则编辑工具，使复杂的业务规则实现变得更简单。总之，能有效地提高实现复杂逻辑的代码的可维护性，完全符合组织对敏捷或迭代开发过程的使用。在物流信息平台中可以有效地对角色功能进行管理，比如角色的添加、功能的变更等。

结束语 从前面的讨论中可以看出，采用 Drools 规则引擎，能够有效地将业务规则和技术分离，特别适合业务规则复杂且变动比较频繁的应用。规则引擎不仅灵活，而且在大规模规则集的情况下，能够快速地完成规则匹配，提高系统的性能。

本文提出的基于物流系统下的 Drools 规则引擎的应用，紧密地与规则引擎结合，提出的步骤和方法使得这些规则具有可描述性，实用性强，最终在应用服务中实现了作用。虽然规则引擎的应用还不广泛、还不彻底，Drools 也存在版本不稳定、文档不完善以及缺少可视化等一些缺点。但由于其概念和理论已经非常成熟，针对于企业级的应用，尤其是对规则复杂且经常出现变更的情况，有着非常明显的优势。在实际的企业应用开发中，引入规则引擎的架构模式使得企业具有更好的可维护性和可操作性，成为解决实际工程中复杂业务规则问题的有力工具。

参 考 文 献

[1] http://baike. baidu. com/view/1495. htm 2009

[2] Josuttis N M. SOA in practice[M]. Southeast University, 2007

[3] 张渊，夏清国. 基于 Rete 算法的 java 规则[J]. 2006(06)

[4] http://soft. chinabyte. com/database/351/118948512. shtml. 2011

[5] http://baike. baidu. com/view/1636209. htm

[6] 缴明洋，谭庆平. Java 规则引擎工作原理以及应用[J]. 2006(06)

[7] http://www. blogjava. net/guangnian0412/archive/2006/06/09/51756. html

[8] 陶晓俊，朱敏. 基于规则引擎的企业服务开发模式[J]. 2006

[9] http://wenku. baidu. com/view/719f6a3e0912a216147929fe. html