北京科技大学

硕士学位论文

选 题 报 告

论文题目：基于 Drools 的规则引擎在产品质量在线判定与评级中的应用

学　　号：S20131639

学　　生：姚贵鹏

专　　业：计算机科学与技术

指导教师：吕志民

单　　位：冶金工程研究院

2014 年 07 月 24

目录

目录..... I
1. 课题综述.....1
1.1 课题背景.....1
1.2 课题意义.....2
2. 文献综述.....3
2.1 规则引擎研究现状.....3
2.1.1 基于规则的专家系统.....3
2.1.2 JAVA 规则引擎.....4
2.1.3 商用规则引擎.....5
2.2 Drools 规则引擎.....6
2.2.1 Drools 概念.....6
2.2.2 Drools 总体结构.....7
2.2.3 Drools 工作机制.....7
2.2.4 Drools 规则语言.....10
2.3 Rete 快速匹配算法.....13
2.3.1 Rete 算法简介.....13
2.3.2 Rete 算法优缺点.....14
2.3.3 Rete 算法研究现状.....14
3. 课题内容及进程.....16
3.1 课题内容.....16
3.2 课题难点.....17
3.3 课题进度.....17
4. 参考文献.....19

1. 课题综述

1.1课题背景

钢铁工业是国民经济中重要的基础原材料产业，也是我国经济的重要支柱。随着国民经济的快速发展，我国钢铁工业迅猛发展，规模不断扩大，取得了很大成就。但是，无论从技术水平，还是从管理的层次上看，目前我国仅是钢铁生产大国，而非钢铁强国，与国外的先进企业还有一定的差距，尤其是中国钢铁企业的信息化水平。中国缺乏成规模的信息化软硬件产品服务商支撑并解决钢铁工业信息化关键共性技术、建设钢铁企业信息化集成管理系统以及促进企业信息化向决策支持等深度应用拓展。“十一五”期间钢铁行业工业化和信息化相互促进，不断融合，主要钢铁企业实现了企业管理信息化，逐步形成多层次、多角度的信息化整体解决方案[1]。信息化为中国钢铁工业提供了重要机遇，使用先进的信息技术能够实现优化设计、制造和管理，通过对各种生产和消费过程进行数字化、智能化的实时监控，能够及时发现问题，解决问题，不仅提高了钢铁生产效率，也大大降低各种资源的消耗。

良好的质量控制对于任何一个钢铁生产企业都具有非常重要的意义和价值，产品质量是钢铁企业的生命线，现代化钢铁厂的主要特点是不再单纯依赖某一单一的生产工序控制产品质量，必须从原料开始对每一工序都实现严格的质量控制与管理，才能保证最终产品质量[2]。这就要求钢铁企业应具备进行在线质量监控的功能。

近年来，随着科学技术的迅猛发展和市场竞争的日益激烈，为了保证产品的质量和经济效益，先进控制和优化控制纷纷被应用于工业过程中。然而，不管是在先进控制策略的应用过程中还是对产品质量的直接控制过程中，一个最棘手的问题就是难以对产品的质量变量进行在线实时测量。受工艺、技术或者经济的限制，一些重要的过程参数和质量指标难以甚至无法通过硬件传感器在线检测。目前，生产过程中通常采用定时离线分析的方法，即每几小时采样一次，送化验室进行人工分析，然后根据分析值来指导生产。由于时间滞后大，因此远远不能满足在线控制的要求。

在线检测技术正是为了解决这类变量的实时测量和控制问题而逐渐发展起来的。在线检测技术，根源于推理控制中的推理估计器，即采集某些容易测量的变量（也称二次变量或辅助变量），并构造一个以这些易测变量为输入的数学模型来估计难测的主要变量（也称主导变量）[3]。从而为过程控制、质量控制、过程管理与决策等提供支持，也为进一步实现质量控制和过程优化奠定基础。在线连续检测技术已是现代流程工业和过程控制领域关键技术之一，它的成功应用将极大地推动在线质量控制和各种先进控制策略的实施，使生产过程控制得更加理想。

1.2课题意义

各钢铁企业一直在为提升企业竞争力做努力，不断提升企业生产效率以及不断提高产品质量。产品质量的检测一般是依靠事后检测的方法，有一定的滞后性。虽然企业在信息化自动化的过程中，在产品生产线设置了许多监控点，能够及时采集产品的相关数据信息，但是这些数据信息并没有得到很好的利用。本论文所研究的基于规则引擎的产品过程质量在线判定，以生产线实时数据作为输入，通过规则引擎的匹配算法及时执行预先定义好的策略措施等，对钢铁企业提高产品质量、提高生产效率以及管理效率都有十分重要的意义。具体有如下几方面：

（1） 使用规则引擎，实现了业务逻辑和应用程序的分离，方便管理人员的维护，以及根据生产需要及时添加新的规则组；

（2） 高效利用生产线采集到的实时数据，对数据分析后反馈到生产线，进一步促进了企业的信息化建设；

（3） 通过对实时数据的处理，对生产过程中的产品实现在线评级与判定，实现产品质量在线预警；

（4） 对不合格产品及时报警，及时定位故障点，帮助生产人员及时排除故障，减少不合格产品的数量，提高生产效率，提高产品的合格率。

2. 文献综述

2.1规则引擎研究现状

Drools 是一个使用 Java 语言编写的规则引擎框架。Java 规则引擎是推理引擎的一种，它起源于基于规则的专家系统。专家系统是人工智能的一个分支，它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[4]。介绍 Drools 之前先介绍一下相关知识。

2.1.1 基于规则的专家系统

Java 规则引擎是推理引擎的一种，它起源于基于规则的专家系统[5][6]。专家系统是人工智能的一个分支，它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。专家系统有很多分类：神经网络、基于案例推理和基于规则系统等。

规则引擎[7] 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 包括三部分：Rule Base（knowledge base）、Working Memory（fact base）和 Inference Engine。它们的结构如下系统所示[8]：。

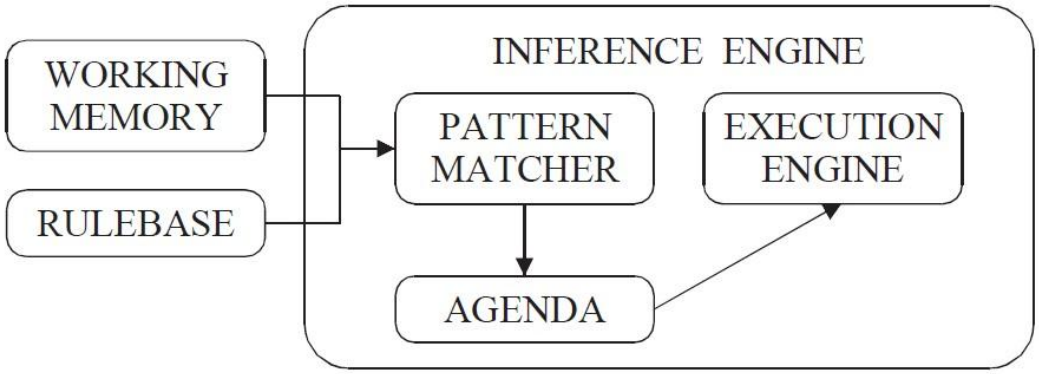

图 2-1 基于规则的专家系统构成

如图 1 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[9]。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。

推理引擎的推理步骤如下[10][11]：

（1）将初始数据（fact）输入 Working Memory。

（2）使用 Pattern Matcher 比较规则库（rule base）中的规则（rule）和数据（fact）。

（3）如果执行规则存在冲突（conflict），即同时激活了多个规则，将冲突的规则放入冲突集合。

（4）解决冲突，将激活的规则按顺序放入 Agenda。

（5）使用执行引擎执行 Agenda 中的规则。重复步骤 2 至 5，直到执行完毕所有 Agenda 中的规则。

上述即是规则引擎的原始架构，Java 规则引擎就是从这一原始架构演变而来的[12]。

2.1.2 JAVA 规则引擎

Java 规则引擎由下面几个部分构成： 工作内存(Working Memory) 即工作区、规则执行队列、静态规则区。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。Java 规则引擎的结构如图 2 所示[13]。

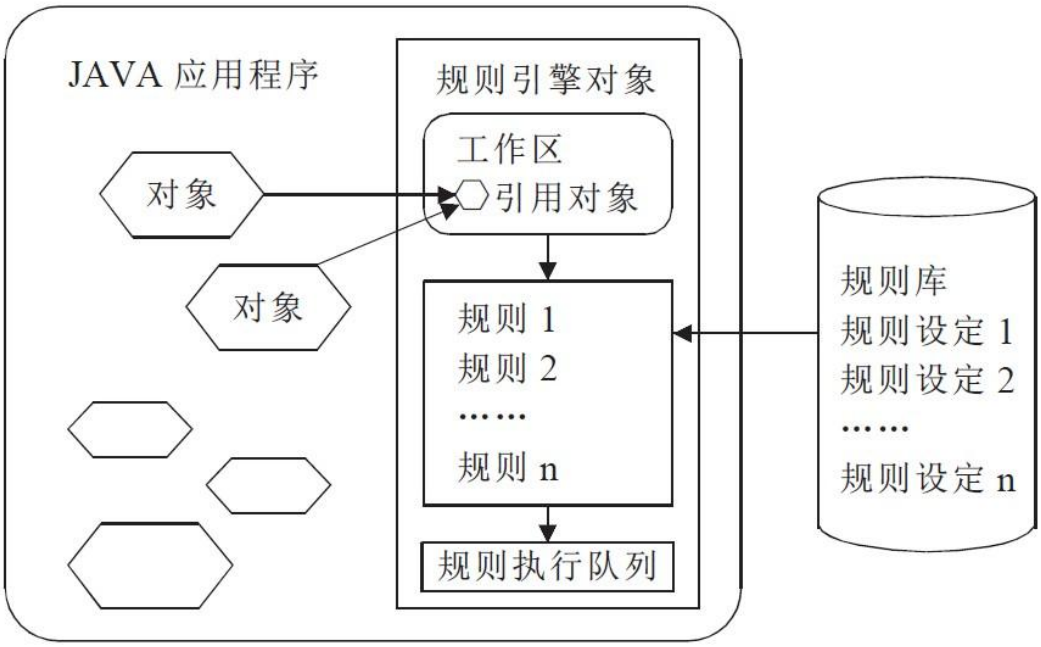

图 2-2　Java 规则引擎工作机制

当引擎执行时，会根据规则执行队列中的优先级顺序逐条执行规则执行实例，由于规则的执行部分可能会改变工作区的数据对象，从而会使队列中的某些规则执行实例因为条件改变而失效，必须从队列中撤销，也可能会激活原来不满足条件的规则，生成新的规则执行实例进入队列。于是就产生了一种动态的规则执行链，形成规则的推理机制。这种规则的“链式”反应完全是由工作区中的数据驱动的。

任何一个规则引擎都需要很好地解决规则的推理机制和规则条件匹配的效率问题。规则条件匹配的效率决定了引擎的性能，引擎需要迅速测试工作区中的数据对象，从加载的规则集中发现符合条件的规则，生成规则执行实例，最后输出匹配结果[14]。1979 年，美国卡耐基 梅隆大学的 Charles L.Forgy 博士在其博士论文[15]中首次提出了一种叫做 Rete 的算法，很好地解决了这方面的问题[16]。目前世界顶尖的商用规则引擎产品基本上都使用 Rete 算法。

2.1.3　商用规则引擎

处于应对当代商业活动前所未有的动态性，近年来计算机应用业界开始致力于将规则系统技术应用于企业服务。由于这类规则系统集成于企业服务的整体解

决方案中，为外部应用程序提供逻辑处理服务，因而被称为规则引擎。主流的企业服务开发平台也提出了相应的规则引擎标准，例如 Java 规则引擎标准 JSR - 94[17]。许多厂商和研究机构或组织也相继推出和不断改进商用规则引擎产品，其中比较有代表性的有 Microsoft 基于.Net 平台的 BizTalk[18]，IBM 的 Web Logic[19]，JBoss 的 Drools（JBoss Rule Engine）[20]，JEss[21]和 ILog JRules[22]等。可以说商用规则引擎技术及其应用正是当前企业服务领域快速发展和推广的实用技术。

2.2Drools 规则引擎

2.2.1 Drools 概念

Drools 规则引擎作为开源规则引擎的代表，它是由 JBoss 开发出来的一款优秀的规则引擎，故也称为 JBoss Rules，它不仅具有易于访问企业策略、易于调整、易于管理等特征，而且标准符合业内需求，运行速度快、执行效率高。一些相关人员可以利用 drools 查看业务规则，从而判断规则是否执行了相应的业务规则。Drools 适用于 java 编程环境下具有 Rete 算法的规则引擎的实现，它具有面向对象的接口，从而使得 Drools 适用于商业上。

此外，Drools 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[23]。

总结一下，Drools 具有以下特点：

（1） 规则语言丰富：Drools 提供了多种规则语言， 可以使用 Jav a/ 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。

（2） 算法效率高：由于 Drools 对 Rete 算法进行了改进，提高了匹配效率，故也称为 ReteOO 算法。

（3） 完整的 API： Drools 规则引擎根据 JSR94 规范，为用户提供众多的 API 接口，同样用户可以根据自己的需要进行选择。

（4） 工具集成：本系统采用的 IDE 为 Eclipse 集成开发环境，该开发环境

可以将 Drools 规则文件导入，方便程序人员的开发。

2.2.2 Drools 总体结构

Drools 是一款开源的 Java 规则引擎，它采用了大众公认高效的模式匹配算法——Rete 算法，实现了面向对象节点规则的引擎执行，实现业务逻辑与业务数据的分离。Drools 基于面向对象思想和网状图搜索原理，将 Rete 算法封装成 Rete-OO，放在一个核心模块(Drools-Core)中，并在这个核心模块的基础上添加了一系列外围模块扩展功能[24]。 Drools 基本结构图如下：

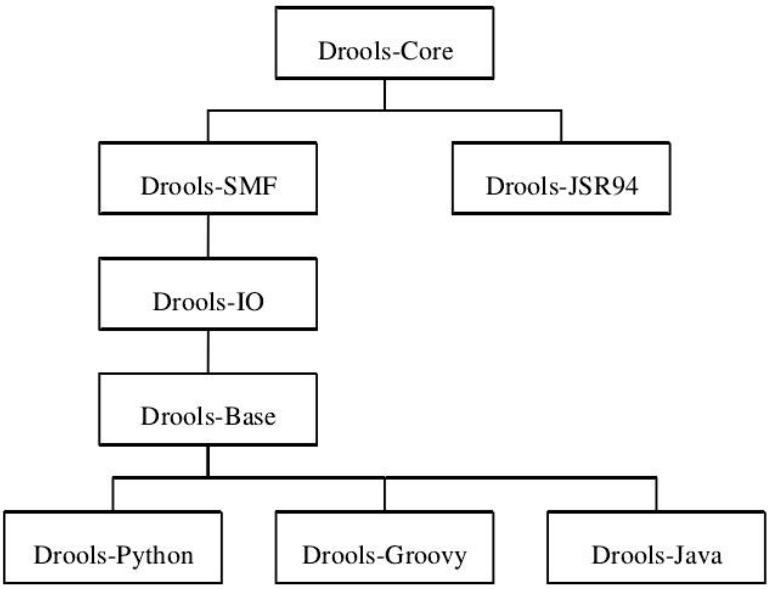

图 2-3 Drools 的结构图

其中 Drools-SMF(语义模块框架)和 Drools-IO 模块用于从当前的文件系统中创建规则集。Drools-Base 模块是一个基于 XML 语言的基础语义理解模块。在 XML 语言的基础上 drools 项目支持 Python、Groovy、Java 三种语言进行规则的定义。Drools 支持 JSR94 规范，提供了相应的兼容模块 Drools-JSR94。

2.2.3 Drools 工作机制

规则引擎的运行状态包括构建和运行两个状态，Drools 根据运行状态分为构建 (Authoring)时组件和运行时(Runtime)组件两个主要部分。

1. 构建时组件主要是根据文件中的规则描述建立规则引擎规则库的组件。

构建时组件包括规则文件、规则文件解析器、规则解释器、规则打包器，其输入的是.xml 或.drl 的规则文件，输出是规则的 package 对象。具体其结构入下图所示：

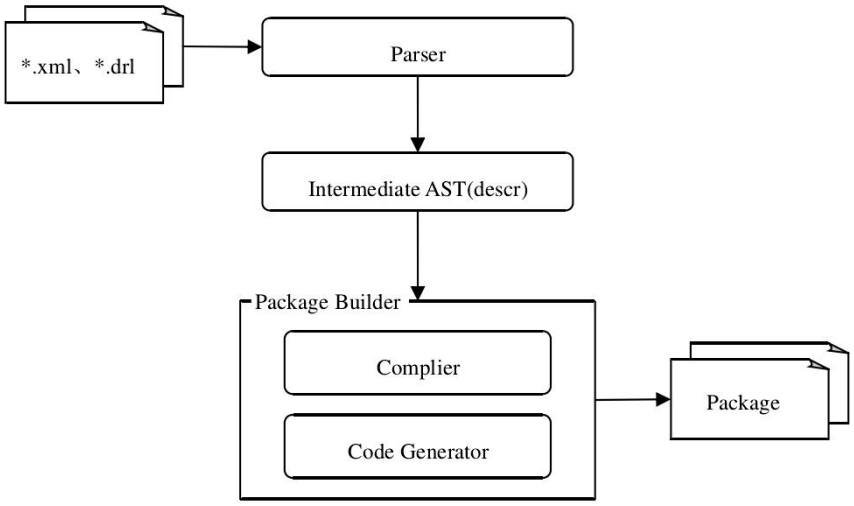

图 2-4　Drools 构建时组建的结构图

构建的过程包括两部分，一是规则文件的创建；二是规则文件的加载。在 Drools 中，规则文件的形式是.drl 或.xml 格式的文件[25]，并有一定的语法定义。而规则文件的加载过程是.drl 或.xml 文件被读入一个解析器(Parser)，该解释器是使用 ANTLR3(Another Tool for Language Reeognition，即语言识别的另一个工具)进行语法解析。解析器在对规则文件作相应的语法正确性检查，之后生成一种由“descr”描述的中间结构，即 AST。AST 生成完，统一后被传到 PackageBuilder 即规则包打包器中，对 AST 描述的规则进行规则重编译和代码重生成，最终将 AST 转换成 Package 类型对象。Package 类型的对象是一个可序列化、可以配置的[26]，由规则集合组成的 Package 类型的对象实例。经过这些过程，Drools 规则引擎将.drl 或.xml 描述的规则最终转换成了 Drools 的规则库的集合元素，供 Drools 引擎运行时使用。

2. 运行时组件主要是驱动事实和规则匹配，运行规则的组件。

运行时组件包括 RuleBase、事实工作空间两部分，其输入是构建时产生的规则 package 对象和插入事实工作空间的事实，输出是根据规则和事实匹配进行规则执行，其具体结构入下图所示：

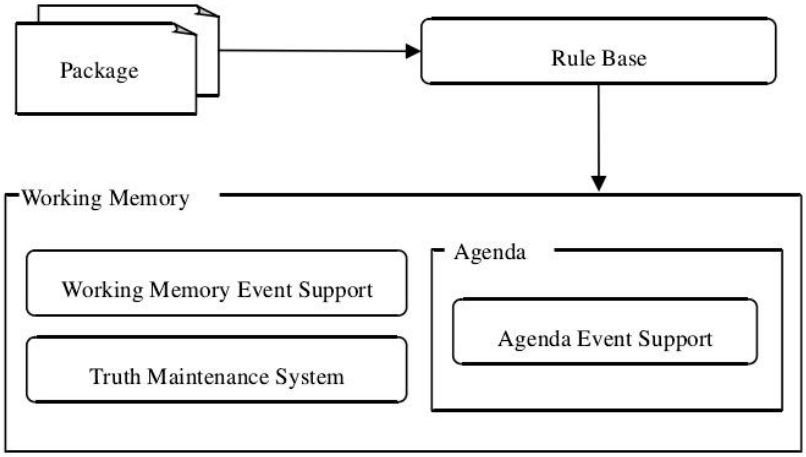

图 2-5　Drools 运行时时组建的结构图

RuleBase 即规则库元件是运行时组件的一个模块，一个 RuleBase 可以包含一个或多个 Package 类型的对象对象。规则引擎在运行过程中，规则库元件可以在任何时刻将一个 Package 类型的对象加入或移出规则库元件容器对象中。规则引擎可以根据规则库元件容器中 Package 类型的对象，在任何时刻下实例化一个或多个 Working Memory 即事实库的对象，而在规则库元件容器中保存着他生成的所有事实库对象的弱引用。事实库元件由一系列子元件组成。规则引擎执行过程中，对象形式的事实数据被 assert 即声明或插入进事实库中，可能会导致一个或多个规则激活的状态产生，然后由 Agenda 负责安排这些冲突规则的执行顺序[27]。

在一个规则集中一次有多条规则符合条件，将被触发，它们相互之间的不同执行次序将会产生不同的执行结果，这称为规则的冲突。Drools 中内建了专门的冲突处理机制，采用冲突处理策略链路的方式来处理冲突。冲突处理策略形成一个队列，前面策略没有解决的冲突规则转到后面策略处理，直到所有冲突解决为止。Drools 的冲突处理共有 7 种冲突处理策略[28]。

（1）　优先级策略。每一条规则赋予一个整数优先级，优先级可正可负，缺省为 0 。拥有较高优先级的规则优先执行。优先级相同的规则将作为一个冲突规则子集传给其他冲突策略解决。

（2）　复杂度优先策略。这种策略判断规则的复杂度，一个规则的条件越多，这个规则的复杂度就越高，其执行的优先级就越高。复杂度相同的规则作为

一个冲突规则子集传给其他冲突策略解决。

（3） 简单性优先策略。与复杂度优先策略相反，这一策略把条件最少，复杂度最低的规则优先执行。

（4） 广度策略。这一策略基于规则加入议程的先后顺序来解决冲突，最近加入到议程中的规则最先执行。由于每个规则加入议程都有一个序号，因此采用这种策略可以解决所有冲突，不会有未解决规则子集需要转发。

（5） 深度策略。这一策略与广度策略相反，最早加入议程的规则优先执行。

（6） 装载序号策略。每条规则加入规则集时都有唯一的装载序号，根据装载序号来处理主策略处理后剩下的未解决规则子集。这样可以保证解决所有冲突。不过由于装载序号是与所选用的具体语义模型相关，因此改变语义模型有可能会改变规则执行次序。

（7） 随机策略。最简单的策略。规则被随机加入议程，顺序执行。它适用于对顺序不敏感的规则。

Drools 使用一个已定义好的规则冲突的解决器(Default Conflict Resolver)解决冲突。Default Conflict Resolver 可以配置相应的冲突解决策略链路。默认的冲突解决策略链路是优先级策略→广度策略→复杂度优先策略→装载序号策略，当然也可以根据实际需要配置合适的策略链路。

2.2.4 Drools 规则语言

业务逻辑（即规则）要按照一定的语法格式写入规则库中，目前还没有对规则语言的标准化定义，国外有代表性的产品纷纷定义了自己的业务规则描述语言和业务规则表达形式。Drools 有自己的规则语言，提供了相关语法和规则编辑器，同时提供了规则验证器用于自动检测编辑器中写入的规则是否有语法错误，并通过 Problem 视图报错。Drools 规则库通常是一个以.drl 为扩展名的文件，在一个.drl 文件中，可以有多条规则，也可以将规则分布在多个.drl 文件中，这有利于管理大量的规则。规则文件的结构如下[29]。

图 2-6　Drools 规则文件结构

（1）　Package：规则的命名空间。用法类似于 java 中的 package 关键字。在同一包名下的规则名必须是唯一的。可以简单的认为 package 就是一组相关的规则集合。

（2）　Imports：规则文件中的 Imports 关键字也类似于 java 中的 Imports 语句。使得在规则文件中直接使用类名而不需要再写包名。

（3）　Expander：可选项。用来支持.dsl 文件(即域描述语言)。

（4）　Globals：定义全局变量，全局变量的使用范围是 session。

（5）　Functions：函数是定义在规则文件当中一代码块，作用是将在规则文件当中若干个规则都会用到的业务操作封装起来，实现业务代码的复用，减少规则编写的工作量[30]。

（6）　Queries：通过它可以方便的从 knowledge session 中查询事实对象（fact）。

（7）　Rules：规则文件的核心--具体的规则。

Drools 规则文件中的每一条规则都遵循如下格式[31]：

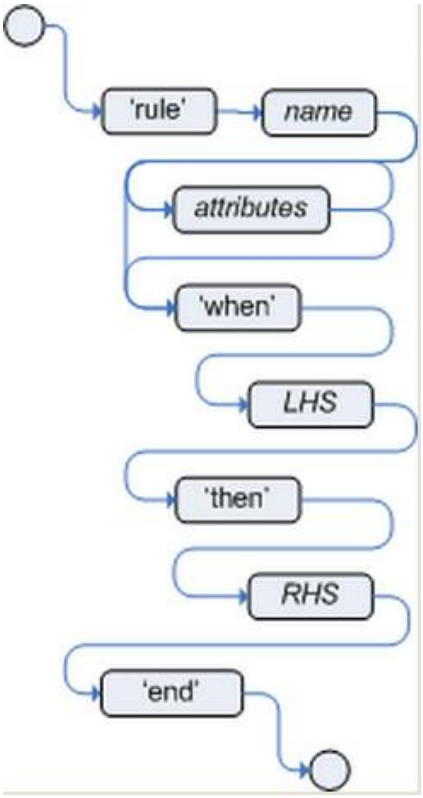

图 2-7　规则结构

Ruel、when、then、end 是 Drools 的关键字，每一条规则以 rule 为起始标识，以 end 为结束标识，when 提示规则的条件部分，then 提示规则的动作部分，其中：

（1） name。规则名，每一条规则必须有自己的名字，同一文件中不同的规则名字不能重复。

（2） Attributes 是可选的，提示规则应该怎样运行。一条规则可能有如下的属性：salience：integer 型数据。agenda-group： string 型数据。auto-focus：boolean 型数据。activation-group：string 型数据。no-loop：boolean 型数据。

（3） LHS (left hand side) 是规则的条件部分，遵守特定的 Drools 规则语言语法。规则引擎就是将工作内存中的事实的条件部分与规则文件中每条规则的 LHS 进行模式匹配，并激活满足条件的规则。

（4） RHS(right hand side)是规则的结果(consequence)或者动作 (action)部分，RHS 基本上是一个允许执行的 Java 语义代码块。

2.3Rete 快速匹配算法

事实与规则之间的匹配算法是规则引擎的核心算法。如果采用不好的规则匹配算法，规则引擎的性能将大打折扣。在众多的匹配算法中比较有代表性的主要有线性匹配算法 Linear[32]，Rete 算法，Leaps 算法[33]，Treat 算法[34]。其中，Rete 算法及其衍生的算法是目前规则引擎所采用的普遍算法，因为它们的匹配效率是最高的。Rete 算法除了应用与规则引擎外，还在许多其他的领域中广泛应用，特别是在人工智能领域，因为 Rete 算法最初是针对基于规则的专家系统而提出的。Drools 使用的是 Rete，据说也有 Leap 的实现，后来因为无人使用也无人维护给放弃了。

2.3.1 Rete 算法简介

Rete 在拉丁语中是“net”，有网络的意思。1979 年美国卡耐基 梅隆大学的 CharlesL.Forgy 在他的博士论文中阐述了 Rete 算法。

Rete 算法的初衷是，利用规则之间各个域的公用部分减少规则存储，同时保存匹配过程的临时结果以加快匹配速度。Rete 模式匹配算法是在模式匹配中利用推理机的时间冗余性和规则结构的相似性，通过保存中间过程及模式共享来提高推理效率的一种模式匹配算法[35]。在模式匹配过程中，规则的前提(LHS)中可能会有很多相同的模块，因此在匹配规则的前提时，将进行大量的重复运算，这样就带来时间冗余性问题。例如:

```
RULE1:if (A>B) and D or C then E=100
RULE2:if (A>B) and (B<C) then E=200
RULE3:if (!(A>B) or (B<C)) then E=300
```

图 2-8　三条简单的规则

若要匹配这 3 条规则时，对于表达式 A>B 要进行三次计算，对 B<C 需要两次计算。Rete 采用的方法为:令 M1=A>B，M2=B<C；则规则可修改为如下所示。

```
RULE1:if (M1) and D or C then E=100
RULE2:if (M1) and (M2) then E=200
RULE3:if (!(M1) or (M2)) then E=300
```

图 2-9　采用 Rete 后的三条规则

这样只有当 A 或 B 发生变化时，才重新计算 M1；同样，当 B 或 C 发生变化时，重新计算 M2。这样就避免了在每次进行模式匹配都重复计算相同的表达式，而只要检测相关参数是否变化来决定是否更新表达式，因此在推理过程中可节省大量的时间开销，从而提高了推理效率。

2.3.2 Rete 算法优缺点

Rete 算法通过共享规则节点和缓存匹配结果，获得产生式推理系统在时间和空间上的性能提升。其优点可以总结如下[36]：

（1） 通过节点共享，减少或消除了特定类型的规则或事实冗余。

（2） 进行连接操作时存储部分的匹配结果，这反过来也使得产生式系统在每次有事实变化或增删的时候都可以避免对整个事实集的重评估，只需评估工作内存中变化的事实。

（3） 在事实从工作内存撤销时，Rete 允许记录单元有效地从网络中移除。

Rete 算法的节点共享和状态缓存虽然大大提高了匹配效率，然而也有很多缺点[37]：

（1） 存在状态重复保存的问题，比如匹配过模式 1 和模式 2 的事实要同时保存在模式 1 和模式 2 的节点缓存中，将占用较多空间并影响匹配效率。

（2） 处理逻辑有限，仅有一阶布尔逻辑。

（3） 处理大规模数据和快速变化的数据时效率较低。

2.3.3 Rete 算法研究现状

Rete 算法自从 1979 年提出以来，已经经历过各种改进与推广，国内外学者对对其研究和改进主要包括以下几个方面[38]。

（1） 结构优化亦即通过在原始 Rete 算法的规则网络上增加或修改一些结构，来提高网络的灵活性、高效性等。Rete 从提出至今，已经经历过不同的演化。对 Rete 本身网络结构的优化一直是研究的重要课题。

（2） Rete 算法最初用于优化产生式的匹配，是在确定性的一阶布尔逻辑

下进行规则匹配。然而当前各种其他逻辑的广泛应用、数据的缺失、模糊逻辑等问题对传统规则引擎的功能提出了新的要求。于是近年来出现许多对 Rete 匹配算法功能扩展的研究。

（3） 近年来，随着云计算的研究与应用，规则推理引擎在云平台上如何处理大规模数据也相应地成为研究重点。

尽管在过去的几十年间，对 Rete 算法及一般的产生式推理引擎有过大量研究和改进，但是我们实际面临的数据和逻辑依然以出乎意料的速度增长[39]，这既推动了对 Rete 推理方法的革新，也为 Rete 算法今后的研究提出了更大的问题和挑战。

3. 课题内容及进程

3.1课题内容

本课题源于某钢厂的全流程工艺质量在线判定与分析诊断系统，因此课题内容与钢厂紧密相关，所用数据也都源于钢厂实际生产采集的数据。

（1） Drools 开发环境的搭建与代码编写。一个完整的开发环境是课题研究基础之一。Drools 提供了一个基于 Eclipse 的 IDE，这个 IDE 提供了 Drools 的核心类库及运行 Drools 所需要的所有依赖包，它要求 Java1.4(J2SE)、Eclipse3.2 及 Eclipse 插件 EclipseGEF3.2(eclipse 图形化编辑框架)或更高版本。安装方法很简单，下载 Drools IDE 压缩包后，将解压后的 plugins 文件夹中的 org.drools.ide_3.0.5.jar 文件复制到 Eclipse 的 plugins 文件夹内即可。成功安装后，启动 Eclipse 就会在工具栏看到一个醒目的红色人头像轮廓的图案。Drools IDE 包括以下几个部分：向导、规则编辑器、规则验证器、DSL 编辑器、4 个视图：Working Memory View、Agenda View、Global Data View 和 Audit view，前 3 个用在调试时检查规则引擎本身的状态，Audit view 用来显示审计日志。

（2） 添加外部数据预处理模块，提高规则引擎的匹配效率。Drools 规则引擎的匹配过程并不一定适应钢厂的数据特点。因此，本课题将研究建立针对钢厂不同生产线和生产流程的数据提取配置模块，在外部解决数据的提取和优化过程，使得优化后的数据能更好的适应 Drools 规则引擎，以提高其匹配效率。

（3） 适当对规则引擎和匹配算法做优化。对规则引擎和 Rete 匹配算法做适当优化，使其更好的适应钢厂的数据特点和业务特点。将外部数据预处理的方法与规则引擎的优化相结合，尽可能的提高规则引擎的执行效率。只有规则引擎的高效运转才能使其满足钢厂产品质量的在线判定与评级。

（4） 规则文件的构建。从文献综述可以知道，规则文件对于规则引擎的运转十分重要，而且规则文件直接影响着规则引擎的执行策略。建立合理的规则文件对于产品在线判定评级十分重要。本课题将会结合钢厂业务特点建立灵活的通用的规则文件配置管理模块。

3.2课题难点

（1） 数据项的预处理模块的构建将是难点之一。钢厂数据存在数据量较大以及受不稳定因素影响的不确定性。而这几个数据特点都对 Drools 的匹配性能产生重要影响。因此，如何对数据项的提取过程和数据项的组合进行优化，使其适合 Drools 规则引擎的匹配过程，将是本课题的难点之一。

（2） 规则配置模块的设计。规则文件定义了匹配过程中的条件，以及满足相关条件后所需执行的具体动作，这将直接影响规则引擎在在线判定中的执行策略。本课题与钢厂的实际生产流程结合紧密，因此规则文件的编写应以钢厂的世纪流程状况为依据。如何设计适用于钢厂的规则配置模块，以及使得该配置模块方便灵活易于管理人员的使用和修改将是一个难点。

（3） 规则引擎和匹配算法的优化与改进。快速匹配算法是规则引擎中重要组成部分，匹配算法的速率和性能直接影响到了规则引擎在产品质量在线判定中效率和性能。生产线的产品质量存在诸多不稳定因素，过程参数也将会实时变化，这都对匹配算法的优化提出了挑战。除此之外，信息化的推进使得钢厂数据量增加迅速，如何使规则引擎适应钢厂的大数据量也是难点之一。

3.3课题进度

2013 年 7 月～2013 年 8 月	查阅相关文献，明确课题研究方向，初步分析课题研究方法，完成开题报告及文献总结；
2013 年 9 月～2013 年 10 月	搭建 Drools 开发环境，了解规则文件的结构与配置过程；
2013 年 11 月～2013 年 12 月	了解钢厂不同生产流程及其参数的含义，结合生产流程研究通用的规则文件配置方法；
2014 年 1 月～2014 年 2 月	结合行业特点优化推理引擎的数据提取和输入方法；
2014 年 3 月～2014 年 5 月	搭建完整的规则引擎，并完成数据测试；
2014 年 6 月～2014 年 8 月	分析数据结果，根据测试结果，不断调整规则引擎；

2014 年 9 月～2014 年 12 月	整理数据，评价优化方案的可靠度和适用性，评价规则引擎性能，撰写完成毕业论文。

4. 参考文献

[1] 芦永明，王丽娜.中国钢铁企业信息化发展现状与展望[J].中国冶金，2013，23（5）.

[2] 良战利.基于数据仓库技术的钢铁质量数据分析方法研究[D].昆明：昆明理工大学，2007:1.

[3] mingyekeji123.在线检测[EB/OL].http://baike.baidu.com/view/5965000.htm?fr=aladdin.2012-12-25.

[4] 刘伟.Java 规则引擎——Drools 的介绍及应用[J].微计算机应用，2005，11（6）.

[5] 尹朝庆.《人工智能与专家系统(第二版)》[M]，水利水电出版社，2009.4.

[6] 蔡自兴，徐光佑.《人工智能及其应》[M]，清华大学出版社，2004.08.

[7] shaliebao.规则引擎［EB/OL］.http://baike.baidu.com/view/1636209.htm?fr=aladdin.2013-03-12.

[8] 李国乐.详解 Java 规则引擎与其 API.http://dev.yesky.com/478/2034478.shtml，2005.

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

[10] 刘宇.Java 规则引擎技术研究[A]，Computer Era，2011，No.7.

[11] Geoffrey Wiseman.Real-World Rule Engines[EB/OL].http:www.infoq.eom/articles/Rule-Engines，2006.

[12] 缴明洋，谭庆平.Java 规则引擎技术研究[J].计算机与信息技术.2006(03):41-43.

[13] 郭芳，白建军.基于 Rete 算法的规则引擎 JBoss Rules[A]，Computer Era，2008，No.1.

[14] 黄翱，潘正运等.业务规则引擎 ILog JRules 工作引擎的工作机制分析[J].微计算机信息.2006.22(24).

[15] C.L.Forgy.On the Efficient Implementation of Production System[D].PhD thesis，Carnegie-Mellon University，Department of Computer Seienee，1979.

[16] 杨伟峰.基于业务规则引擎的保险业软件开发[D].上海:复旦大学，2007.

[17] Java Community Process，“Java Rule Engine APITM JSR- 94”，Jul．2002．

[18] Darren Jefford，Kevin B.Smith，Ewan Fairweather：Professional BizTalk Server 2006[M]，Wr0X，2007.05.

[19]Jon Mountjoy，Avinash Chugh，WebLogic：The Definitive Guide[M]，O'Reilly Media，Inc 2004.02.

[20]JBoss.org，JBoss Rule User Guide[EB/OL]，http://labs.jboss.com/drools/documentation.html，2007.02.

[21]Ernest Friedman-Hill，Jess in Action：Java Rule-Based Systems[M]，Manning Publications，2003.07.

[22]Barbara Von Halle，Business Rules Applied：Building Better Systems Using the Business Rules Approach[M]，Wiley，2001.09.

[23]张渊，夏清国.基于 Rete 算法的 Java 规则引擎[J].科学技术与工程.2006，6（11）.

[24]刘伟.Java 规则引擎——Drools 的介绍及应用.微计算机应用.Vol.26 No.6 Nov.2005.

[25]童毅. 规则引擎中模式匹配算法及规则引擎应用的研究[DB]. 北京:北京邮电大学，2010.

[26]张希尧.基于 J2EE 框架和规则引擎的信用卡风险监控系统的构建[D]，上海，复旦大学，2011，29-31.

[27]张娇.规则引擎在促销管理系统中的研究及应用[D]，上海，华东理工大学，2011，15-30.

[28]刘金龙.DROOLS 规则引擎模式匹配效率优化研究及实现[DB].四川：西南交通大学，2007.

[29]马秀丽，王红霞，张凌云.Drools 在网络故障管理系统中的应用[J]，计算机工程与设计，2009，30（8）.

[30]高杰.Drools5 规则引擎开发教程[EB/OL]，www．bstek．com，2009，13-14.

[31]Bob McWhirter.Why choose Drools[EB/OL].http://www.jboss.org/drools/，2007.

[32]Oracle.Linear Inferencing：High-Performance Processing[EB/OL].OraCle，2009.

[33]Don Batory.The Leaps Algorithm.Department of Computer Sciences[J]，University of Texas at Austin，1994.

[34]Daniel P.Miranker. Treat：A Better Match Algorithm for AI Production Systems[J].AAAI，42-47，1987.

[35]RETE 算法的描述[EB/OL].http：//www.cnblogs.eom/ipointer/archive/2005/09/28/246251.html.

[36]Wikipedia[EB/OL].http://en.wikipedia.org/wiki/Rete_algori-thm.2009-09-08.
[37]顾小东，高阳.Rete 算法：研究现状与挑战[J].计算机科学，2012，39（11）.
[38]张彪.基于 Rete 算法的数据库通知引擎技术研究.上海海事大学硕士学位论文.2004-8.
[39]杨智.基于 Rete 算法规则引擎的研究及其实现与应用[D].东北大学，2007.