一、ilog介绍

业务规则系统,后被IBM 收购,改名IBM ODM 决策管理系统,商用

- 1、规则引擎的实际用途和价值
 - a、各地核保政策不同,业务变更频繁
 - b、hard code的代价非常大,参考echannel ,前端堆积大量逻辑判断 比如要求发送电子保单的分支机构有二级机构、有二级机构下边个别除外

码农的困惑:

- 1、if else 好多,嵌套好深啊。 前辈走了,我就得哭了。
- 2、规则制定者需求源源不断: 预算标准变化,交通情况变化 交通数据变化,组合条件变化

规则制定者的困惑:

- 1、我不知道以前的规则有哪些,不懂代码,一不小心踩坑里了
- 2、一个小需求,需要等待开发人员 安排一个开发周期,太慢了
- c、 通过下述过程缓解频繁变更的需求变化
 - 开发人员和业务专家梳理业务,定义统一的BOM (业务对象模型)
 - 业务专家可以快速的制定修改规则,然后交由规则引擎自动化地来处理分析
 - 规则引擎代替系统开发人员,解决由规则条件关联动作变化带来的开发工作

2、组件介绍

重要概念:

xom:业务执行对象,java对象给调用方和规则引用

bom: 业务对象模型,由xom转化而来,可显示为中文

组成:Res、Rts、RuleDesigner:

RuleDesigner:本地开发客户端,参考eclipse 中文版

Res: 规则集部署执行服务Rule Execution Server, res同时支持规则的完成性检查,可以对规则调用的次数及效率进行监控,支持生成excel文件类型的规则执行报告

Rts:决策中心Rule Team Server, Web版开发工具, 自带流程, it与业务协同操作。

*演示介绍各个组件

二、业务场景介绍

公司规则引擎调用部署:

Res + 调用Client服务+Weblogic

核心原有

- 1. PFLVRULE/ 规则引擎获取签单赔付率
- 2. PUWRULE/调用前置规则引擎,返回双自主系统和费用比例
- 3. <u>VUWRULE/</u> 自动核保
- 4. 自动核赔理算

o2o新增:智慧投保、自动核赔、理算

其他业务场景:

保险:信用卡中心发卡审核

电商:各种优惠活规则,满减、折扣等

三、思考问题:规则优先级问题? ->规则流

四、良好实践

1、参数化,业务线上修改规则参数

示例

(1) 核心自动理算规则

一、自核总金额及人员条件规则:			
01	01、纯车损定损金额≪		
5, 000. 00		1-是	•
2,000.00	元		
16	02、查勘定损人员是否配置核损权	1-是	V
限			
二、欺诈风险规则:			
12		1-是	\
核损			_
09	04、不存在同一驾驶员驾驶不同车	0-否	V
辆在我司多次出险			
10	05、出险时间不在承保起期	1-是	
5.00	天或到期 5.00 天内	1-Æ	•
11		1-是	
3.00	次	1 75	•

(2) 费用费率规则: 费率规则较规范、各分类费率的汇总(简单计算)



2、结合机器学习,调整具体业务规则参数:

业务场景: 自核

传统实现方式:规则执行结果统计、被拦截规则、人工分析调整

对已有数据,按照已有规则进行参数优化。

五、其他规则引擎

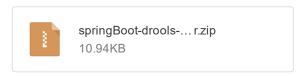
Drools一个基于Java的开源规则引擎

Easy Rules 轻量级规则引擎Easy Rules 适合一些独立的小型产品需求中,可以合理应用规则引擎实现业务与规则的分离,降低系统间耦合

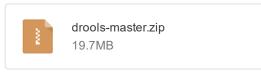
VisualRules 国产:学习成本(文档、培训中文),政治风险小

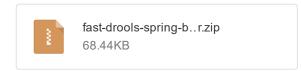
参考:http://blog.sina.com.cn/s/blog bd5db8370102xaax.html













rule-engine-master.zip 310.75KB

blog.csdn.net/wo541075754

https://blog.csdn.net/u013115157/article/details/53156989

https://blog.csdn.net/caicongyang/article/details/53056890