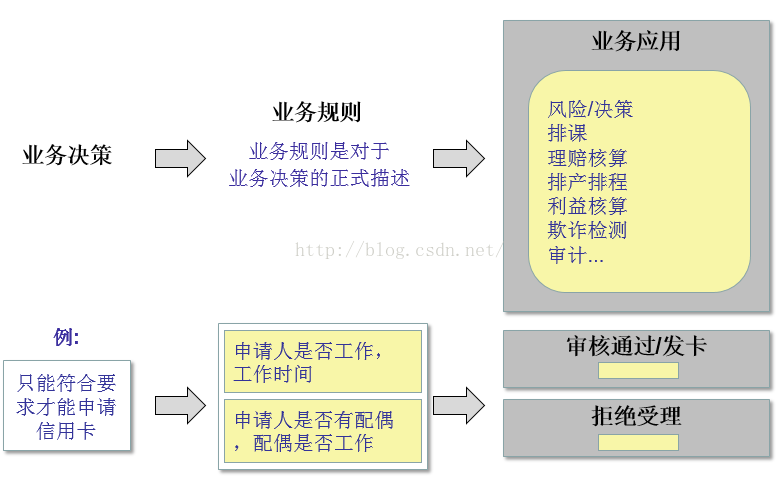
# **1. 什么是规则**

复杂企业级项目的开发以及其中随外部条件不断变化的业务规则(business logic),迫切需要分离商业决策者的商业决策逻辑和应用开发者的技术决策，并把这些商业决策放在中心数据库或其他统一的地方，让它们能在运行时（即商务时间）可以动态地管理和修改从而提供软件系统的柔性和适应性。规则正是应用于上述动态环境中的一种解决方法。



# **2. 规则产生的背景**

一个业务规则包含一组条件和在此条件下执行的操作．它们表示业务规则应用程序的一段业务逻辑。  
业务规则的理论基础是：设置一个或多个条件，当满足这些条件时会触发一个或多个操作。  
企业为适应市场的多变性，它的业务逻辑也必然是多变的。理想情况下，业务逻辑的改变应该能够实时的反映在企业信息系统中。  
但是，传统的方法往往会使这种反映出现滞后，进而可能使企业错过市场机遇。

# **3. 引入业务规则技术的目的**

## **对系统的使用人员**

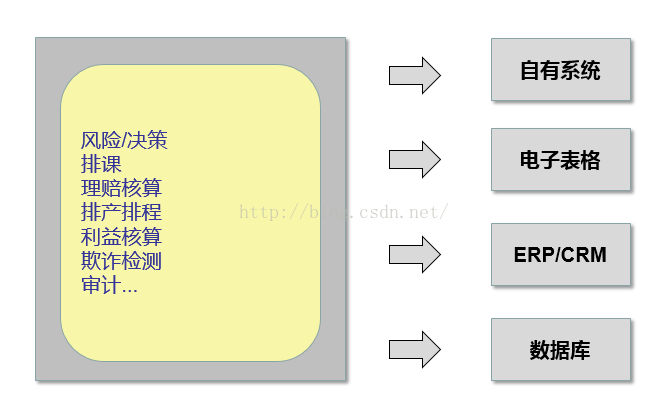
* 把业务策略（规则）的创建、修改和维护的权利交给业务经理
* 提高业务灵活性
* 加强业务处理的透明度，业务规则可以被管理
* 减少对IT人员的依赖程度
* 避免将来升级的风险

## **对IT开发人员**

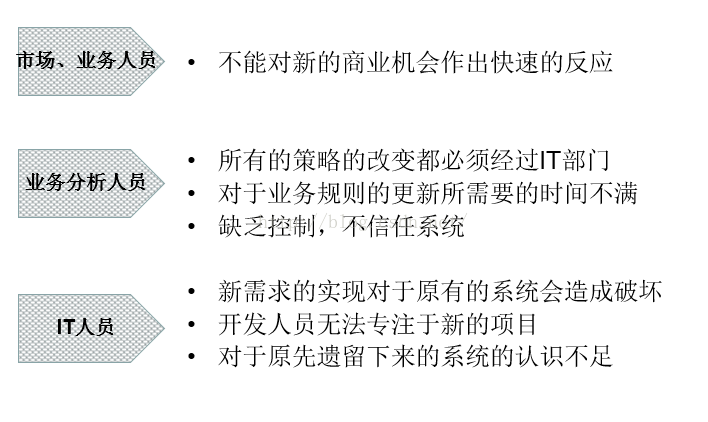
* 简化系统架构，优化应用
* 提高系统的可维护性和维护成本
* 方便系统的整合
* 减少编写“硬代码”业务规则的成本和风险

# **4. 传统IT项目实施与引入规则进行项目实施的比较**

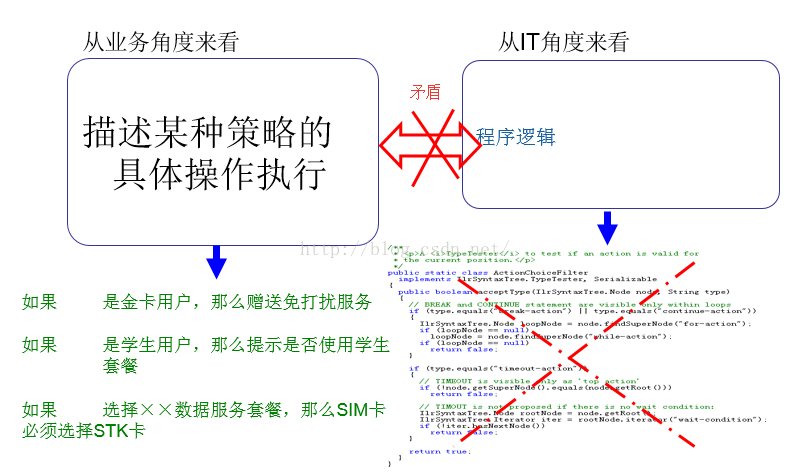
## **传统的IT项目实施**

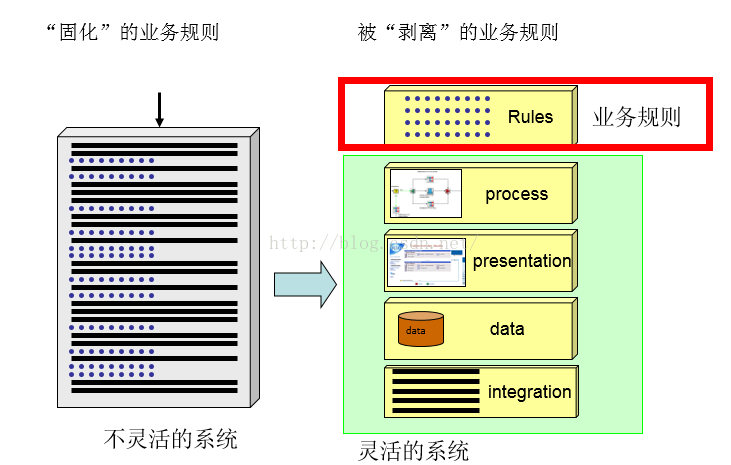


## **传统做法的缺点**

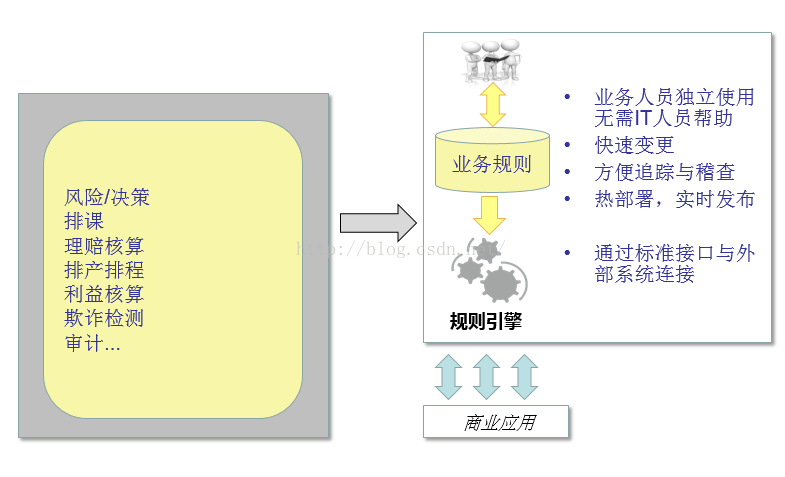


## **在传统的IT项目实施中业务与IT间存在的“矛盾”**

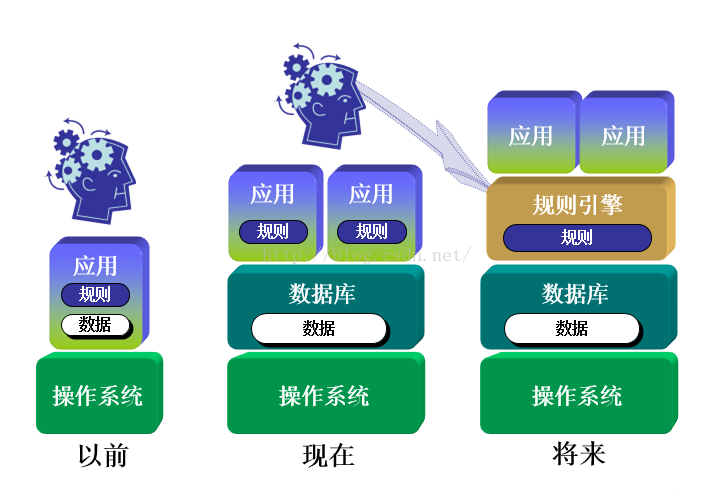




## **引入规则后的做法**



5. 规则是让业务人士驱动整个企业过程的最佳实践

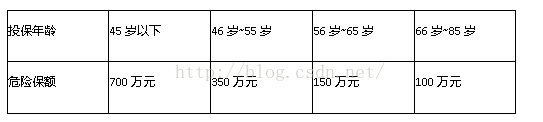


# **6. 从IT技术人员的角度看为什么使用规则引擎**

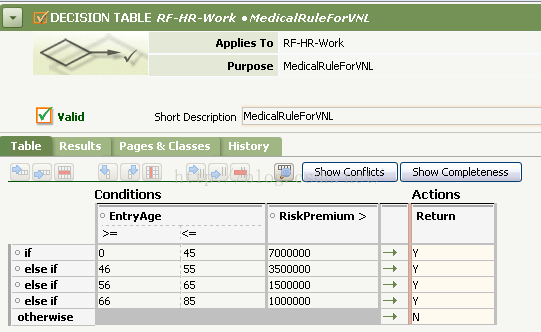
- 从应用逻辑和数据中将业务逻辑分离  
- 简单！ -规则有一个非常简单的结构  
- 让业务用户开发和维护规则以降低成本  
- 声明式编程  
- 性能和可伸缩性  
- 解决复杂的和复合的问题，其中有大量细粒度的规则和事实互动

## **来看一个简单的例子**

例：  
以全球人寿新卓越变额万能寿险投保规则的体检规则为例，输入投保年龄及危险保额来获取是否需要体检的结果。  
免体检累积最高限额表如下：

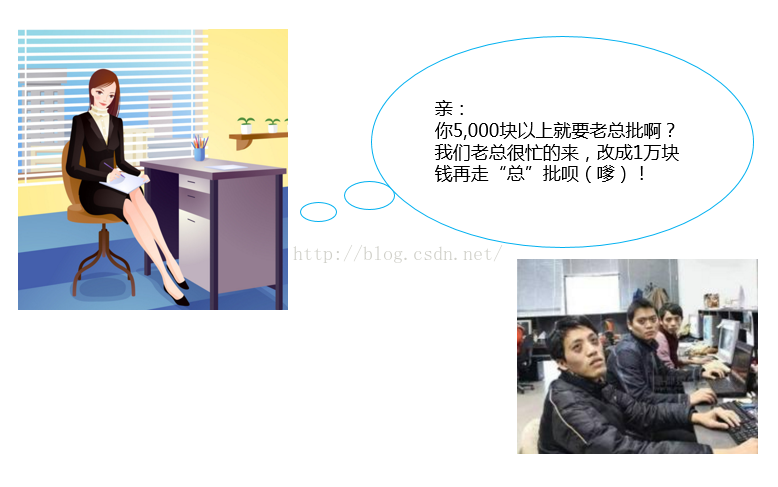


免体检累积最高限额表在规则中又是如何实现的呢？



## **结合现实生活中的一个例子来看变化的业务与IT的“矛盾”**

没有规则时：



有了规则后：



# **IT在面对上述例子时如何能够做到熊掌与鱼兼得呢？**



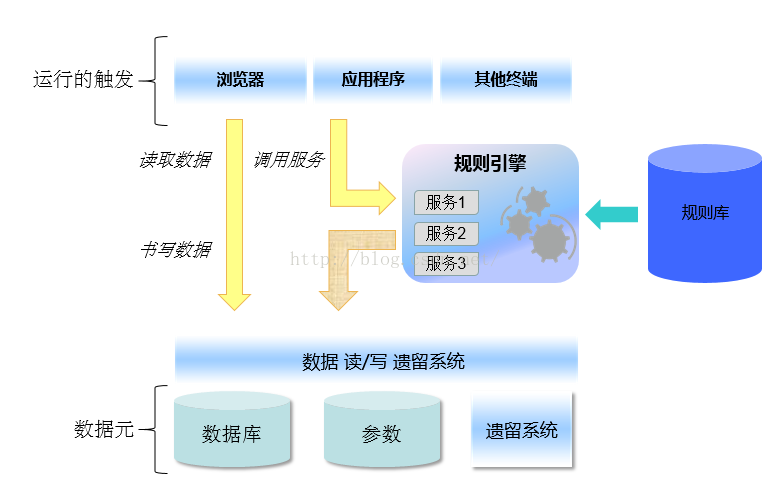
做到以下几点是否就可以“全得”？

* 规则更改不重启，即改即用
* 数据库访问可随意更改，即改即用
* 业务服务层可以随意更改，即改即用
* 开发人员不需要关心底层API，他只需要懂JSON（加快开发）

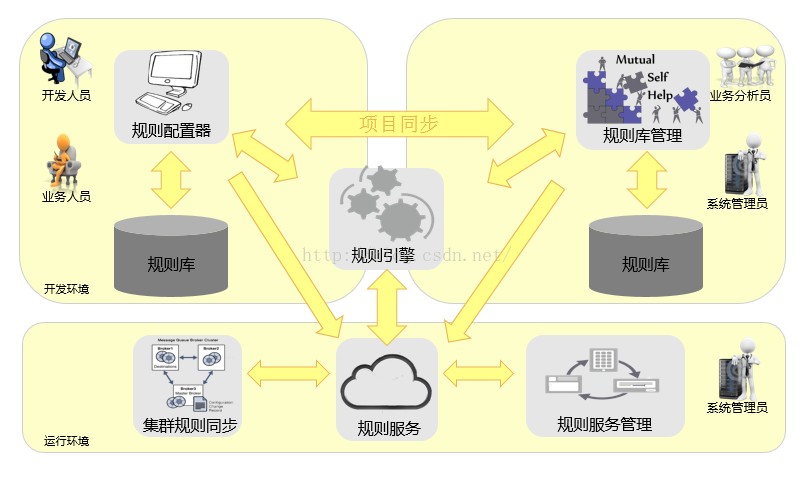
# **因此我们进一步引入了“规则引擎管理系统-BRMS”的概念**

规则引擎由推理引擎发展而来，是一种嵌入在应用程序中的组件，实现了将业务决策从应用程序代码中分离出来，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，并根据规则做出业务决策。

# **BRMS的技术架构**



## **一个完整的BRMS应该具有的特点**

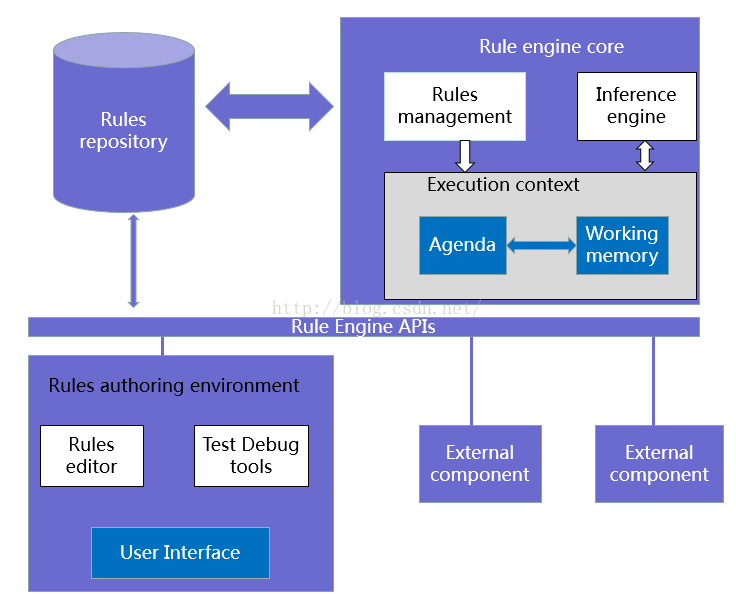


## **引入规则引擎后带来的好处**

* 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
* 使业务分析人员也可以参与编辑、维护系统的业务规则
* 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的简单

# **基于DROOLS打造自有BRMS解决方案**

我在我自己公司采用的是JBOSS Drools来实现的大规模生产环境下的业务规则的应用场景的。



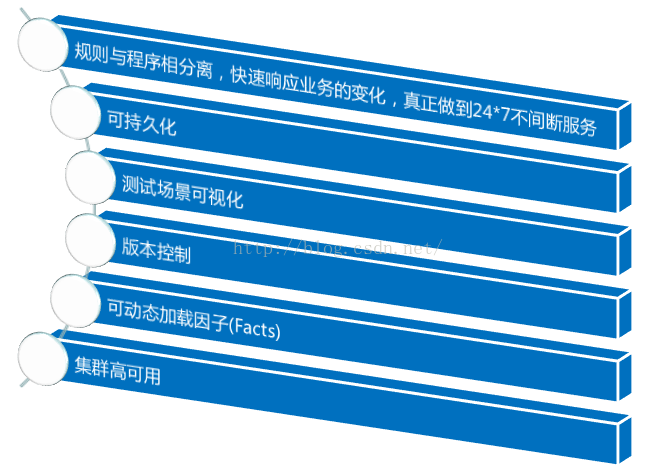
Drools是业界著名的开源规则引擎，它使用的是目前最快的Forward-Chaining推理算法。



RETE 算法可以分为两部分：

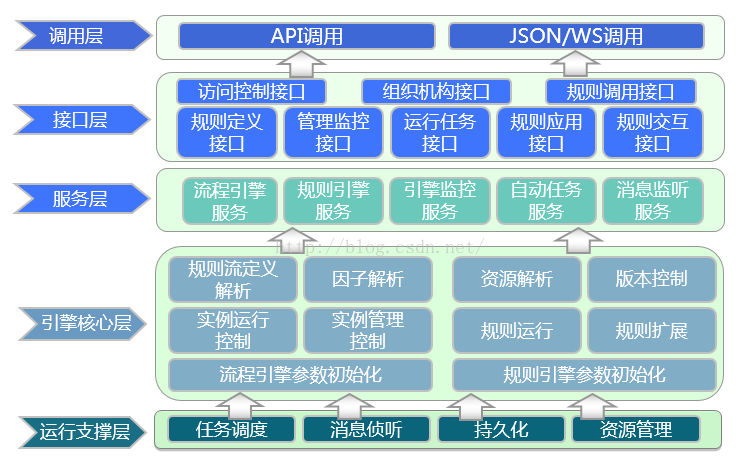
* 规则编译（ rule compilation ）
* 运行时执行（ runtime execution ）

它的核心思想是将分离的匹配项根据内容动态的构造匹配树，以达到显著降低计算量的效果。  
不过由于DROOLS是一个开源免费规则组件，因此在面向企业级商业应用时它还存在着以下的一些问题需要解决：

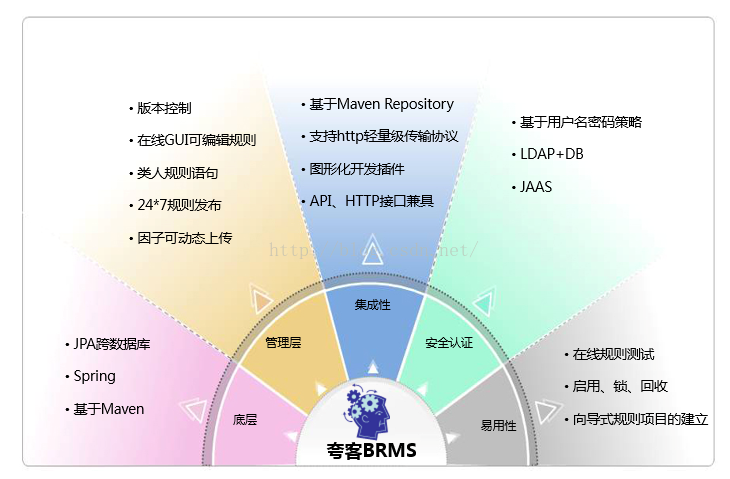


## **打造全方位的面向SOA2.0解决方案**

因此基于上述需求，我在我们公司内部（夸客金融）对于DROOLS进行了二次封装与开发，打造出了一个可用于在生产环境大规模开发和布署的“业务规则引擎”。



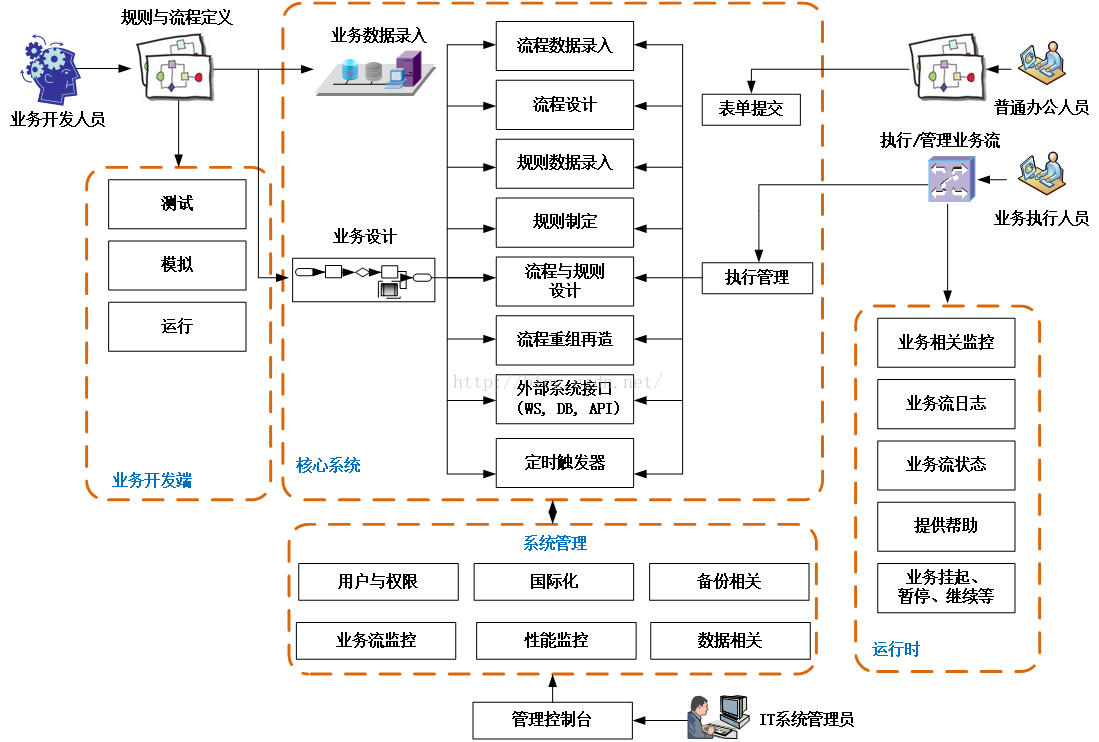
它即可以无缝的和我们原有的核心、ESB总线、支付系统进行连接，也可以和审核、合同系统进行连接，它降低了开发人员对于BRMS底层技术的依赖、増加了开发的“透明性”。同时这套BRMS做到了真正的24\*7可动态支持业务变化、快速响应迅速变化的市场与业务规则。



有了这套规则引擎后，本来在传统金融领域中的“线性开发”发生了变化。  
****传统金融领域中对于系统集成开发的线性方式****



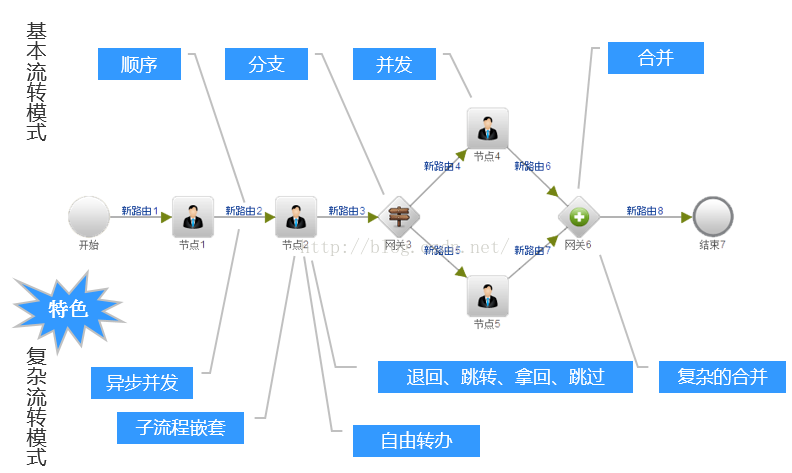
****引入BRMS后我们IT对于内部的系统集成开发方式****



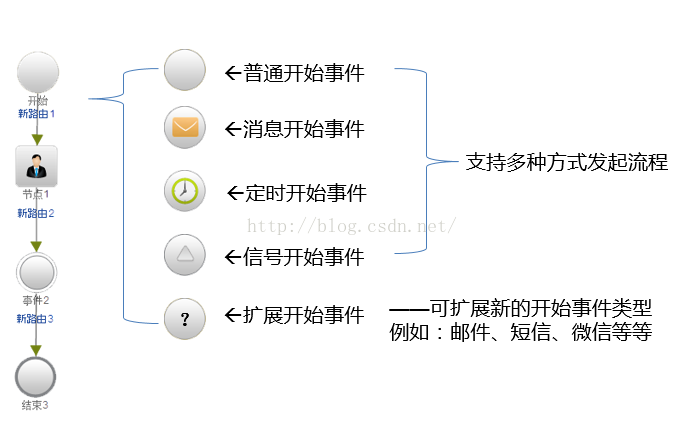
运用了这套BRMS后，我们的IT在面对随时、多变的业务场景时可以做到：



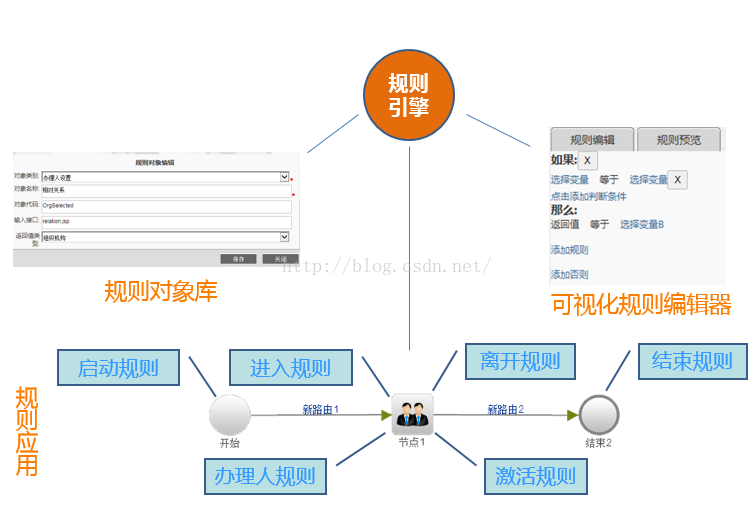
****特色一：支持基于规则的各种复杂流转模式****



****特色二：多种复杂事件发起模式****

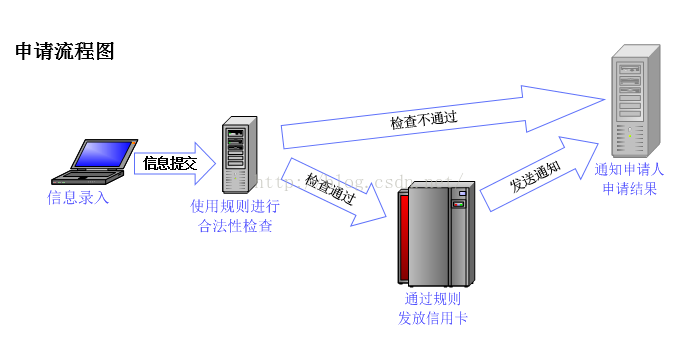


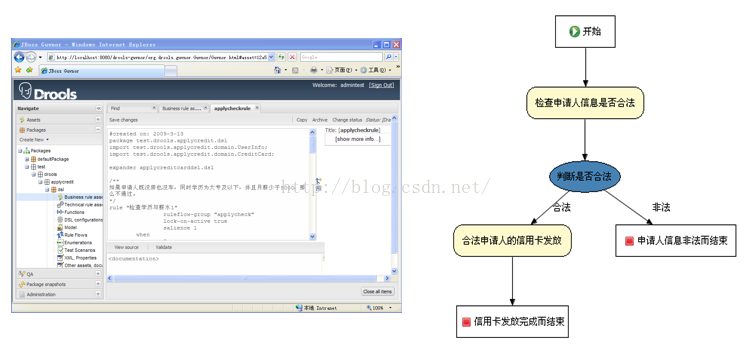
****特色三：灵活全面的规则设计****



# **BRMS在其它金融领域中的应用场景介绍**

## **规则引擎在信用卡申请场景中的应用**





## **规则引擎在反欺诈场景中的应用**





* 黑规则：甄别嫌疑订单
* 白规则：订单解除嫌疑
* 商家管理处罚系统，对违规商家处罚处理。
* 搜索排名，降低违规商家搜索排名权重。
* 自定义阈值规则指标体系
* 地址模糊聚类算法
* 手机嵌入欺诈模糊甄别算法
* 黑/白名单机制设置
* 灰名单机制设置
* 区域订单脉冲机制
* 全区管控机制
* 产品线与SKU监控管理机制