#### 工作流系统选型报告

前提：为满足公司内部工作流平台需要，特地调研了几种主流工作流平台的对比

##### 1.众多工作流

###### 1.1. JBPM（Java Business Process Management）

由JBoss公司开发，目前最高版本JPBM7，不过从JBPM5开始已经跟之前不是同一个产品了，JBPM5的代码基础不是JBPM4，而是从Drools Flow重新开始。下面要涉及的很多产品都是以JBPM4的代码为起点进行开发的。

###### 1.2. Activiti

Alfresco软件开发，基于JBPM4，后并入OMG，目前最高版本activiti 7。Activiti5版本的时候，核心团队发生了比较大的变动（离职），activiti6的开发团队在新版本中去除了PVM，纳入了DMN，重构XML解析，BUG较多，目前主要团队致力于activiti7,5&6已经官宣不维护。

###### 1.3. Osworkflow

完全用java语言编写的开放源代码的工作流引擎，具有显著的灵活性及完全面向有技术背景的用户的特点。由opensymphony组织维护，其不遵守XPDL等业务规范，完全使用XML编排业务。面向开发人员。

###### 1.4. Shark

靠山是Enhydra。是一个可扩展的工作流引擎框架，它包括一个完全基于 WFMC 规范的标准实现，它使用XPDL（没有任何自己新的扩展）作为自身的工作流流程定义格式。其持久层和设计器都是自己公司研发的，持久层实现采用的标准是轻量级的Enhydra DODS O/R mapping 工具，设计器可以用Enhydra JaWE 图形XPDL编辑器。

###### 1.5. Apache ODE

轻型的、可嵌入的组件，利用组件组装成一个完整的BPM系统。关键模块包括ODE BPEL编译器、ODE BPEL运行时、ODE数据访问对象（DAOs）、ODE集成层（ILs）和用户工具。虽然挂在Apache下面，但已经年久失修。

###### 1.6. Flowable

基于activiti6，最新的开源版本是flowable6，开发团队是从activiti中分裂出来的，修复了一众activiti6的bug，并在其基础上研发了DMN支持，BPEL支持等等。相对开源版，其商业版的功能会更强大。

###### 1.7. Camunda

基于activiti5，所以其保留了PVM，最新版本Camunda7，开发团队也是从activiti中分裂出来的，发展轨迹与flowable相似，同时也提供了商业版。

###### 1.8. JFlow

前身ccFlow，国产的工作流引擎，由济南驰骋公司开发维护，主打中国式的业务流程，由于是国产的软件，中文化程度比较深，业务开发也对用户比较友好。国产的开源工作流引擎还是挺多的，JFlow是其中功能比较完善的一个，同时对比activiti，流程上更加中国化，支持自定义流程跳转，加签等。其他国产工作流就不列举了。

还有很多工作流，比如ProcessMaker，SWF，oracle，Bonita，openwebflow，snaker等，不过做BPM的话，相对于上面列举的产品还是有些缺陷，比如流程过于简单，资料过少等。

##### 2.工作流架构以及社区情况对比

|  |  |  |
| --- | --- | --- |
| 工作流框架 | 遵循规范 | 备注 |
| Bonita BPM | XPDL | 流程过于简单 |
| Shark | XPDL | 不维护-2017 |
| Osworkflow | 自定义XML规范 | 不维护 |
| JBPM | BPMN2.0 | JBPM4.3后添加了对BPMN的支持，持续开源 |
| Apache ODE | WS-BPEL、BPEL4WS | 不维护 |
| Activiti | BPMN2.0，XPDL，JPDL | Activiti7维护，5和6bug多 |
| Flowable | BPMN2.0，XPDL，JPDL | 持续开源 |
| JFlow | BPMN2.0，Ccbpm | 2015年后为了与国际接轨，开发支持BPMN |
| Camunda | BPMN2.0，XPDL，JPDL | 持续开源 |

#### 3. 详细对比

###### 3.1 功能对比表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 功能 | Activiti 7 | Flowable 6 | Camunda bpm | JBPM 7 | JFLOW(国产的) |
| 会签 | √ | √ | √ | √ | √ |
| 回退 | × | √ | √ | 无 | √ |
| 驳回 | × | √ | √ | √ | √ |
| 自定义流转 | × | × | √ | 无 | √ |
| 加签、减签 | × | √ | √ | 无 | √ |
| 多实例 | √ | √ | √ | √ | √ |
| 事务子流程 | √ | √ | √ | √ | √ |
| 版本迁移 | × | × | √ | × | × |

###### 3.2 兼容性和二次开发

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 功能 | Activiti 7 | Flowable 6 | Camunda bpm | JBPM 7 | JFLOW(国产的) |
| 支持的流程格式 | BPMN2.0、XPDL、PDL | BPMN2.0、XPDL、PD | BPMN2.0、XPDL、PD | BPMN2.0 | BPMN2.0 |
| 开源情况 | 开源 | 商业版和开源版 | 商业版和开源版 | 开源 | 开源 |
| 开发基础 | JBPM4 | Activiti 5&6 | Activiti 5 | Drool Flow | 自开发 |
| 引擎核心 | 去除pvm | 去除pvm | 支持pvm | Drools | 自研 |
| Spring结合 | √ | √ | √ | √ | √ |
| 二次开发难度 | 一般 | 一般 | 一般 | 较难 | 一般 |

###### 3.3 其他对比

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 功能 | Activiti 7 | Flowable 6 | Camunda bpm | JBPM 7 | JFLOW(国产的) |
| 支持数据库 | Oracle,SQLServer,  mysql | Oracle,SQLServer,  Mysql,postgre | Oracle,SQLServer,  Mysql,postgre | Mysql,postgre | Oracle,SQLServer,  Mysql |
| 集群部署 | √ | 6.5版本支持 | √ | √ | √ |
| 云部署 | √ | 无 | √ | 无 | √ |
| 持久化框架 | Mybatis | Jpa 二次封装 | Hibernate | Jpa | 无 |
| 分布式事务 | √ | 无 | √ | √ | 无 |
| 运行模式 | Springboot 2 | Springboot 1.5 | Springboot 2 | Kie | Springboot 2(特别版本) |
| 集成接口 | SOAP、Mule、RESTful | SOAP、Mule、RESTful | SOAP、Mule、RESTful | 消息通讯 | SOAP、Mule、RESTful |
| 设计器 | 集成idea eclipse，web | 自提供、eclipse | 自提供、eclipse | eclipse | 自提供、.net开发 |
| 内部服务通讯 | Service间通过API调用 | Service间通过API调用 | Service间通过API调用 | 基于Apache Mina异步通讯 |  |

首先明确一点是，我们的aciviti 5 的版本应该被弃用，由于bug过多，以及和后续版本无法迭代，通过上面一些性能对比，更加倾向于在 Flowable 6 和Camunda中选择一个，在这二者之中更倾向于Camunda

###### 3.4 Flowable 6 和 Camunda 对比

1）camunda支持流程实例的迁移，比如同一个流程有多个实例，多个流程版本，不同流程实例运行在不同的版本中，camunda支持任意版本的实例迁移到指定的流程版本中，并可以在迁移的过程中支持从哪个节点开始。

（2）camunda基于PVM技术，所以用户从Activii5迁移到camunda基本上毫无差异。flowable没有pvm了，所以迁移工作量更大（实例的迁移，流程定义的迁移、定时器的迁移都非常麻烦）。

（3） camunda对于每一个CMD命令类都提供了权限校验机制，flowable没有。

（4）camunda继续每一个API都有批处理的影子，flowable几乎没有。比如批量挂起流程、激活流程等，使用camunda可以直接使用API操作，使用Flowable则只能自己去查询集合，然后循环遍历集合并操作。

（5）camunda很多API均支持批处理，在批量处理的时候可以指定是异步方式操作或者是同步方式操作。异步的话定时器会去执行。Flowable没有异步批处理的机制。比如批量异步删除所有的历史数据。

（6）camunda启动实例的时候支持从哪个节点开始，而不是仅仅只能从开始节点运转实例。Flowable仅仅只能从开始节点运转实例。

（7）camunda支持任意节点的跳转，可以跳转到连线也可以跳转到节点，并且在跳转的过程中支持是否触发目标节点的监听器。flowable没有改原生API需用户去扩展。

（8）camunda支持双异步机制，第一个异步即节点可以异步执行，第二个异步方式是：完成异步任务后，还可以继续异步去执行任务后面的连线。所以称之为双异步机制，flowable只有第一种异步方式。

（9）camunda支持多种脚本语言，这些脚本语言可以在连线上进行条件表达式的配置，开箱即用。比如python、ruby、groovy、JUEL。flowable仅仅支持JUEL、groovy。开箱即用的意思就是如果想用python直接引入jython包就可以用了，不需要额外配置。

（10）camunda支持外部任务，比如我们有时候想在一个节点中执行调用第三方的API或者完成一些特定的逻辑操作，就可以使用外部任务，外部任务有两种表，并支持第三方系统定期来抓取并锁定外部任务，然后执行业务完毕之后，完成外部任务，流程实例继续往下执行。外部任务的好处就是解决了分布式事物的问题。在flowable中我们可以使用httpTask任务，我个人更倾向于camunda外部任务，因为这个外部任务有外部系统决定什么时候完成，httpTask是不等待任务，实例走到这个节点之后，调用一个api就直接往下跑了，外部任务不会继续往下跑，有外部系统去决定啥时候往下跑。

（11）camunda支持为用户定制一些个性化的偏好查找API，比如张三每次查询任务的时候，一般固定点击某某三个查询条件过滤数据，使用camunda就可以将这三个查询条件进行持久化，下次张三来了，就可以直接根据他的偏好进行数据的过滤，类似机器学习。

（12）camunda支持历史数据的批量删除或者批量迁移到其他介质，比如批量迁移到es，flowable没有该机制。

（13）camunda支持在高并发部署流程的时候，是否使用锁机制，flowable没有该机制。

（14） camunda支持单引擎多组合、多引擎多库。flowable仅仅支持单引擎多组合。

（15） camunda支持流程实例跨流程定义跳转，flowable没有该机制。

（16） camunda支持分布式定时器，flowable没有该机制。

（17） flowable支持nosql,camunda只有nosql的解决方案。

（18）camunda支持优化流程，以及了解流程引擎的瓶颈所在和每个环节的耗时，flowable没有该机制。

（19） camunda修改了流程模板xml解析方式，相比flowable性能更好。

（20）camunda在解析流程模板xml的时候，去除了activiti5的双解析机制，相对而言耗时时间更短。flowable没有了pvm所以规避了双解析机制。

（21）camunda可以在任意节点添加任意的属性，flowable原生API没有，需要自己扩展。

（22）camunda框架没有为流程生成图片的API(所有流程图展示以及高亮均在前端动态计算)，activiti5/6/flowable5/flowable6有图片生成以及高亮的API.

（23）camunda可以在节点中定义定时作业的优先级，也可以在流程中进行全局优先级的定义。当节点没有定义优先级的时候可以使用全局的优先级字段。activiti5/6/flowable5/flowable6没有改功能。

（24）camunda可以再流程中定义流程的tag标记，activiti5/6/flowable5/flowable6没有改功能。

（25）camunda/activiti5/6/flowable5/flowable6 均不支持国产数据库，比如人大金仓 和 达梦。

（26）flowable6支持LDAP，openLDAP,camunda不支持。activiti5不支持。

###### 3.5 Flowable 和 Camunda性能对比

1.测试环境

PC笔记本，8G内存，CPU两核i5-8250  
压测工具：JMeter5.1  
数据库：mysql5.7  
Flowable版本：6.4.1  
Camunda版本：7.10

2.对比表

通过压力测试工具JMeter分别对flowable和camunda进行高并发压力测试，并发用户数从100到1000，JMeter设置Ramp-up为1，循环次数为10，测试数据如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 并发数 | Flowable | camunda | 性能对比（Camunda比flowable提升**）** |
| 100tps | 2269 ms | 1631 ms | +39% |
| 200tps | 3907 ms | 3489 ms | +12% |
| 400tps | 7250 ms | 6452 ms | +12% |
| 600tps | 9943 ms | 8700 ms | +14% |
| 800tps | 13700 ms | 12400 ms | +10% |
| 1000tps | 18800 ms | 15700 ms | +20% |

3.底层支持

Flowable数据库表和Activiti 5的数据库表基本一致

Camunda 数据库表在Activiti 5的基础上有变动，并且支持多租户

总结：从我个人角度而言，我比较倾向于 Camunda

**4.数据库差异性对比**

Flowable和camunda 都是基于activiti传承而来，各自都自定义了一些表，但是在

核心定义表：act\_ge\_bytearray camunda新增了五个字段，flowable完全保留了原来的格式

act\_ge\_property camunda，flowable完全保留了原来的格式

act\_procdef\_info flowable完全保留,camunda移除

act\_re\_model flowable中新增字段，camunda中移除此表

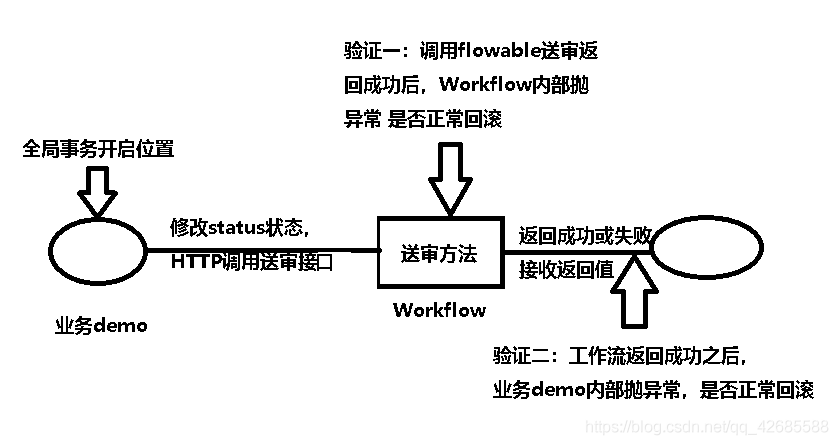
act\_re\_deployment flowable中新增字段,camunda中完全保留

act\_re\_procdef flowable新增字段，camunda中移除和新增字段

**5.分布式事务支持**

Seata的原理是全局事务，是通过数据库undo\_log保证事务的回滚，而我们可以将流程中

心，注册为一个微服务，并采用seata的@GlobalTrancational去向seata服务端传递事务上下文，全局事务的开启位置应该在发起流程的位置



由于流程中心的数据库都是基于mysql数据库，所以在支持分布式事务方面，flowable和camunda并没有啥区别

6.复杂业务场景的支持