# 1.BPMN

流对象：表示业务流程图的核心要素。

连接对象：用于连接BPMN核心对象

泳道：是在流程图上组织活动和责任的机制。

工件：允许流程设计者扩展基本BPMN表示法，以在流程图中包含有关流程的附加信息

# 2.activiti

## 2.1 activiti简介

## 核心API

ProcessEngine

RepsitoryService

RuntimeService

TaskService

IdentityService

FormService

HistoryService

ManagementService

DynamicBpmService

## 数据库

ACT\_GE\_\*

ACT\_RE\_\*

ACT\_

# 3.JBPM

开源工作流引擎很多，主要以Activiti为主，后来有Camunda等等，但是这些工作流引擎有其基因问题，因为是基因问题，属于原罪，也称为四重罪：

1. 对于使用者来说，如果需要精通工作流引擎，必须同时掌握Java语言 BPMN XML语法和图形符号，需要在这三者之间做到一一对应。 对参与人员的逻辑能力要求非常高，因为这些都是语言符号，只是表达逻辑的形式而已，不如直接用Java开发更简单方便。

2. 工作流引擎耦合了业务和技术架构，将可靠性（事务）、高性能（Job执行）和业务流程捆绑在一起，比如Job执行的方式就很难拓展到分布式弹性Job方式就比较难。流程事务和技术事务混淆在一起。 随着serverless和FaaS的兴起，K8s + Istio + Knative等将云架构技术将技术架构变成云平台的一部分，在此基础上使用Knative实现事件驱动的flow流程就更具有扩展性和可伸缩性。

3. 工作流引擎违背六角形架构，按照六角形架构，业务核心不应该依赖REST或JMS或数据库或界面等外围，一些工作流引擎无法通过Spring那样对权限等多切面进行解耦，比如如今流行的OAuth和JWT其实已经剥离了权限，但是很难结合到现有工作流引擎中，同理，有些工作流引擎依赖于分层架构，比如表单对象其实表单对象应该是一个领域模型对象，但是因为名称Form原因在实际中要么会依赖UI层，要么依赖JPA层，有的人会将其耦合到SpringMVC的Controller或Struts的Action，这导致整个系统的扩展性完全丧失，几乎很难过度到Spring Boot的微服务架构。

4. 将领域模型变成Map里面的数据，是先有领域边界？有界上下文？还是先有在这些上下文之间流动的流程呢？流程引擎默认是坚持后者，否定了有界上下文边界，将领域边界内的领域模型任意肢解成数据库的数据，通过Java的Map携带，这种不尊重领域模型的极端做法未来必然行不通，只要动态流程，不要静态结构了，没有结构，就没有不变，整个系统好像都可以变，其实是意大利面条混乱不堪。思路应该是：以聚合根为主，在不同聚合根之间采取异步的事件通讯，将这种事件通讯和流程管理器整合在一起，如Saga模式。

总结：

工作流引擎其实都是某个阶段技术和业务的组装者，但是由于技术不断发展，工作流引擎如果没有实现模块化或分层化，那么很难避免随着技术发展不断淘汰，用户也是先尝甜头后再尝苦头，掉进坑里，这也是工作流引擎不断更新迭代，但是鲜有恒强者的原因，但是由于美妙市场的存在，用户渴求不需要开发人员，以为自己动动鼠标就能搞定政务系统或办公系统。

在一个分布式服务环境下，如何整合调度服务走向，Spring cloud代表的微服务路由网关给出了一点答案，但是目前网关的路由还无法类似BPMN那样通过XML进行各种流程逻辑组合，未来应该会有，K8s + Istio + Knative代表无服务器架构就是一个方向。不过，BPMN本身思路方向可能有问题，希望抛弃代码和配置，用一套全新的语言表达整个业务流程概念，这种方式必然导致业务和技术的深度耦合。