

jBPM – a experience report

Business people has known it for years: business courses in enterprises follow describable processes to a great percentage, the so called business processes. This knowledge lead to Business Process Management (BPM). jBPM is a small flexible Opensource-BPM-Engine, who offers effective BPM even to small enterprises.

Because of the understanding, that workflows can be described in Business Processes, a mass of quality-managment projects took place in the 1990s which had aimed detailed description of the processes. In the meantime the realization grows that documented Business Processes are only useful when they run off IT – supported. In the best case the documentation and implementation occure from the same model of process. But this important conversion from process description into software cannot be found in many enterprises today.

There you realize, that most of all small and medium enterprises lack of solutions to tackle these problems. This is not at least because of the high prices of the corresponding software-products from the established manufactures. But that is not necessary, because even today you can find well-developed Open-Source-Tools for BPM-Projects.

Short summary of BPM

In the area of BPM are two important standards at the moment:

- BPEL (Business Process Executive Language), a language for the description of processes, which is completly based on Web-Services and
- XPDL (XML Process Definition Language)

It is true that other "standards" exist beside, but they will only play a minor role in the future. There are only a few products of Open-Source Workflow-Engines on the market so far, based on these standards. Instead proprietary solutions are often the case. This motivated us to include also products in our evaluation, where you find BPEL just on the Roadmap. An overview about Open-Source BPM-Engines can be found at [Manageability] for example.

In this article we want to take a look at the project jBPM [jBPM], which has joined the JBoss Group [Jboss] by now and enoys a growing community in the internet. We can imagine that jBPM is on the best way to become the de-facto-standard for BPM in Java (like JBoss Application Server). And because jBPM offers a very good roadmap, we regard it as a good choice for a new Java-BPM-projects with Open Source, even when it is not supporting BPEL in its current version.

Our Opensource-Project

These ideas about jBPM have lead to a small Opensource-Project, which helps to intgrate jBPM into typically business applications, the "camunda toolkit for jBPM" (see [tk-jbpm]). Some of the concepts described here you can find illustrated in Java-Code there.

Technical Basics of jBPM

But what is jBPM? Actually you can call jBPM a classical Workflow-Engine: You describe your process as a state-machine which the runs off in the jBPM-Engine. The state-machine has to be described in a XML-file.

Because the basic principle is easy to understand with an example, you can see a simple procurement-process from the erp-sysytem [CCS], developed in a project at a customer,

in illustration 1. For the business administration aspects of processes see the wide range of books available. Listing 1 shows extracts of this process in XML (but in an own format, enriched with metaformations).

```
<process-definition name="Procurement">
  <swimlane name="procurement" delegation="Einkauf" />

  <start-state name="ProcurementStart" swingForm="CreateProcurementProcessComponent">
    <assignment swimlane="procurement" />
    <transition name="createDemand" to="GoodsNeeded" />
  </start-state>

  <task-node name="GoodsNeeded" swingForm="GoodsOrderedComponent">
    <assignment swimlane="procurement" />
    <transition name="goodsOrdered" to="decideOrderComplete">
      <action>
        <businessService facade="StockJBpmFacade" method="addAnnouncement">
          <param name="announcement">announcementValue</param>
          <result>article</result>
          <resultType>key</resultType>
        </businessService>
      </action>
    </transition>
    <transition name="demandDeclined" to="end" />
  </task-node>

  <decision name="decideOrderComplete">
    <handler class="OrderCompleteDecisionHandler" />
    <transition name="complete" to="decideIdentifiedGoods" />
    <transition name="incomplete" to="splitProcurementProcess">
      <action>
        <delegation class="StartSecondGoodsNeededProcessAction" />
      </action>
    </transition>
  </decision>
  [...]
</process-definition>
```

This XML can be transformed easily into the required jPDL [JPDL] with XSLT. jPDL is also a XML-Format, and it is very similar to the presented processdescription. You can look up jPDL, like the whole example, **in our wiki [camundaWikiBsp]**. At the moment we are trying to generate the XML directly from the graphic model or to use the jbpm graphical designer to get XML and graphic single source.

The most important structures of the process are: state, decision, and transition between states. The majority of all processes can be described with just these few methods. If demands are more complicated, jBPM also can handle fork, join, sub-processes or milestones.

But what does the engine with these processes? At first it can start process instances and after that it controls states and transitions of each of the instances. This sounds much more complicated as it is, if you look at the following simple code example:

```
JbpmSession session = JbpmSessionFactory.buildJbpmSessionFactory().openJbpmSession();

// Variablen für Prozess
Map variables = new HashMap();

// Prozess-Definition laden
ProcessDefinition pd = session.getGraphSession().findLatestProcessDefinition("Procurement");

// Prozess-Instanz "Einkauf" starten
ProcessInstance pi = pd.createProcessInstance();

// Variablen zum Prozess hinzufügen
pi.getContextInstance().addVariables(variables);

// Transition "Bedarf abgelehnt" ausführen
pi.getRootToken().signal("demandDeclined");

// Alle Aufgaben für "Bernd Rücker"
```

```
TaskInstance[] tasks = (TaskInstance[])session.getTaskMgmtSession().findTaskInstances("Bernd Rucker").toArray(new TaskInstance[0]);
```

Other constructs of the process-description are the so-called swimlanes. They regulate in which group (or employee) is responsible for tasks. With these swimlanes it is possible to create Todo-Lists for each member of the staff.

But jBPM neither implement the maintaining of groups nor categorization of the employees. On the contrary the engine works with pure strings, who are describing an 'actor' in general (this 'actor' can be a group, a single person or another system). In our project we implemented a component which resolves the Todo-List for the individual user by loading all task for him and all of his groups from the engine (this can also be implemented with JAAS by the way). Also the concrete implementation of the workflow-engine is hidden by this component (for example it transforms the tasks after the DataTransferObject-Pattern into a jBPM neutral "TaskDTO").

Another important feature of the process-engine is the process context, so every process instance can contain variables. In jBPM you can use every Java-Object. To get an abstraction of jBPM in Java and to achieve fast development, we generate Java-classes from the processdescription. These classes form a process-model and with JGoodiesDataBinding [JGoodies] they offer fields for the GUI, which are bound to the processvariables.

How do the processes now call business-logic? Developers of jBPM place on flexibility and the users are free to do everything, with one constraint only: actions have to be implemented in Java. Then these actions can be added to different processconstructs and they perform at corresponding transitions or events. This doesn't sound like flexibility, but with your own little framework you are able to do nearly everything. In our project for example we have designed the access to businesslogic (implemented as Service) like this, that we are able to "call" it up in the processdescription (see listing 1). The same procedure is possible for web-services, other processes, other systems or anything else.

Environment for jBPM

jBPM can run in every Application-Server which is J2EE-conform or in every stand-alone application, too. The running in an applicationserver offers a great advantage after all: both the BPM-Enging and the the J2EE Services can be performed in a context of transaction, and this helps to simplify ollbacks at errors.

Undo-Mechanism could be used as an alternative in such processes. This concept, like it is pursued especially in BPEL, is even in jBPM possible, but more complex.

jBPM uses Hibernate [Hibernate] as persistence mechanism, which also allows you custom Querys on the objectmodel of jBPM, when bordfeatures are not sufficient. In our case we also implemented Querys which pick out the process instances with special values of variables (all processes of procurement for a particular article for example).

Advantages of a BPM-Engine

If you work with a BPM-engine you will realize soon what the advantages of a process - oriented development are. The process controls the basic services, that's why the structure of the application and the control-flow are easy to understand. Business processes can be easy adapted as well and you can deploy changes even in the running system. A versioning prevents the mess with long running processes that are already started. Due to this fact maintaining the applications is much easier, because demands of the process are changing much faster then demands of the services.

The possibility to reuse the Business-Services is another big advantage. They can be implemented independent from another, because the interaction depends only on the process. Therefore you can "compose" an individual process with these different services which are available (this is called "orchestration").

BPM in context of SOA

This procedure is connected with service-orientated architecture (SOA) very tightly, although SOA is not necessarily connected with Web-Services! I think the following definition from Karlsruher Versicherung is very useful: " Eine Service-Orientierte Architektur ist eine Software-Architektur, die idealerweise für alle Anwendungssysteme eines Unternehmens gilt und auf den Prinzipien der Service-Orientierung aufbaut." (SOADef) These principles are:

- Encapsulation of business functionality
- Uncoupling of the interface
- Neutrality of technology
- Services are the elements of processes and application

In this context you can say that BPM-Engine is the decisive centre where single services were connected to a working businessprocess. And of course these processes can be provide as services again.

GUI of Business Processes

But how do I integrate BPM-Engine in the GUI of my application? We have implemented a worklist in our framework, that means an individual To-Do-List for every single employee, which is filled with data from jBPM.

This Worklist is then able to get itself the Java process models for current or new processes by a resolver. In addition to the fields for process variables the process model offers Actions, which call up the according transitions. Also only actions are active, who are possible in the current process state.

The worklist can get and GUI component for the recent process-state too. The corresponding component is configured in a properties-File and initialized with the ProcessModel. So the component can be implemented completely flexible and it easy to use fields, variables or actions which are supplied from the processmodel. The only constraint for the gui component is the implementation of an abstract component from the used GUI-Framework (we use an implementation which follows Genuine [Genuine] as GUI-Framework). However you can adapt the implementation to your own framework by a simple adaption of the resolver.

The future

This solution worked perfect in our project. Although there are still some things to improve, everything will be developed further more. Finally jBPM will gain from the advantages of an active Opensource-project.

We have not mentioned some more features of our tk-jbpm yet: a Java Swing Admin-Client to overview processinstances which is also able to intervene manually in their flow.

But jBPM isn't inactive, too. An integration of the standardized Process-Description-Language BPEL is planned and on the roadmap and in general we have great expectations from this project.

Java & BPM

Regarding "BPM and Java" you realize a quiet healthy boom of this topic. Perhaps the often quoted service-orientated architecture is prerequisite for recognizing the great

potential of BPM or there aren't any good and free products in this sector. We have no idea why this technique so slowly gains acceptance, but we are sure it will speed up software development together with SOA and MDA noticeable over the next time.

We also expect a correction of the market, with its uncountable Opensource-Workflow-Engines, and that's why jBPM has done a good job with its cooperation with JBoss. We also expect some changes in the commercial sector, where high developed products are available, because of the new competition out of the Open Source Community.

Summary

The use of a BPM-Engine in a typical business-process is definitely worthwhile. At best the process orchestrates only existing services into different business processes. Economic process documentations and working process description can originate from the same source, if the procedure is suitable. With jBPM you get a small, flexible Opensource-BPM-Engine, but it can handle even big projects. Compared with good commercial products you've got to do more manual work and write some small extensions or frameworks, see our toolkit for jbpm for example. Altogether you surely can expect more supports from the Open Source Community in the future.

References

[Manageability]	www.manageability.org/blog/stuff/workflow_in_java/view
[jBPM]	www.jboss.com/products/jbpm
[JBoss]	www.jboss.org
[jira]	jira.jboss.com/jira/browse/JBPM
[tk-jbpm]	tk-jbpm.camunda.org
[CCS]	www.camunda.com/ccs/
[SchmSe04]	Hermann J. Schmelzer, Wolfgang Sesselmann: Geschäftsprozeßmanagement in der Praxis, Hanser Wirtschaft 2004
[jPDL]	http://www.jbpm.org/jpdl.html
[Hibernate]	www.hibernate.org
[SOADef]	www.eec2005.de/speaker/EB_04_Muench_Karlsruher.pdf
[Jgoodies]	genuine.sourceforge.net/