Business-Driven Software Engineering

Lecture 6 – Process Implementation

Jochen Küster jku@zurich.ibm.com
Agenda

- Motivation
- Different Abstraction Levels of Process Models
- The Concept of a Process Engine
- Overview of the Activiti Process Framework
- Process Implementation with the Activiti Process Engine
- Summary and References
Motivation
**Motivation**

- **So far:**
  - How to express process models with BPMN?
  - How to use a process model for documentation?

- **But:**
  - How to implement a process and make it executable?
  - How to execute the process model?
Different Abstraction Levels of Process Models
Different Levels of Abstraction of Process Models

- Different persons in a company want different view on processes
- Leads to different levels of process models

Example:
  - Business View for Business Analysts
  - IT View for IT architects

Levels of abstraction depend on the company and the domain of process modeling as well as the goals
Process Models at Different Abstraction Levels

- Process models are used both by business and IT people in companies
- Business-level Process Model
  - detailed understanding of the main behavior from business view
  - requirements specification for process implementation
- Technical Process Model
  - used for automated process execution
- Existence of process models at different abstraction levels is very common in companies today
Business-level Process Model
- Start
- Get Personal Details
- Get Claim Details
- Validate Claim
- Success?
- yes: Create Claim Document
- no: Reject Claim

Technical Process Model
- Start
- Get Request Details
- Log Session Data
- Validate Claim against Business Rules
- Success?
- yes: Create Documents
- no: Reject Claim
- Manual Handling
- Send Claim Letter
- End
Technical Process Model - Example

- Implementation details
  - human activities, script activities, services, communication
- Error handling
- Renaming and refactoring
- Refinement
  - additional IT activities, refinement of activities into subprocesses
Example: Camunda Levels of Abstraction

- Ebene 1: Strategisches Prozessmodell
  - Inhalt: Prozess im Überblick
  - Ziel: Schnelles Verständnis

- Ebene 2: Operatives Prozessmodell
  - Inhalt: Operative Abläufe
  - Ziel: Arbeits- und Umsetzungsgrundlage

- Ebene 3a: Technisches Prozessmodell
  - Mit Process Engine
- Ebene 3b: IT-Spezifikation
  - Ohne Process Engine

- Ebene 4a: Implementierung
  - Mit Process Engine
- Ebene 4b: Umsetzung
  - Ohne Process Engine

Source: Camunda
Camunda Levels (Overview)

- **Level 1: Strategic Process Model**
  - View for Business at high level
  - Responsibilities for resources
  - High level view of the process

- **Level 2: Operational Process Model**
  - Operational details of the process
  - Much more detailed than level 1

- **Level 3: Technical Process Model**
  - Automation with process engine
  - Many technical details depending on process engine and other systems
Example Level 1
From Level 1 to Level 2

- Introduction of pools for participants
- Modeling of special cases and error handling
- Completion of the process
- Introduction of additional flow and activities needed
Example for Level 2
From Level 2 to Level 3

- Decision about what parts of the process should be supported by process engine
- Refactoring of the process for this purpose
- Implementation details
  - human activities, script activities, services, communication
- Error handling
- Renaming and refactoring
- Refinement
  - additional IT activities, refinement of activities into subprocesses
Example: SAP Levels of Abstraction

Modelle zur Beschreibung allgemeiner unternehmensübergreifender Geschäftsszenarios (z. B. Auftragsabwicklung, Beschaffungsprozess)

Prozessebene, bei der jeder Schritt im Modell für eine komplizierte Aktivität steht

Aktivitätsebene, auf der ein Schritt aus Ebene 2 detailliert beschrieben ist – i. d. R. umfassen die Schritte dieser Ebene Aktivitäten, die von mehreren Personen ausgeführt werden

Aufgabenebene zur Beschreibung der spezifischen Maßnahmen, die zur Ausführung einer Aufgabe erforderlich sind

Source: SAP
The Concept of a Process Engine
The Concept of a Process Engine

- executes process models
- creates and manages process instances created from process models
- persists process instances in case of delay
- uses service calls for calling other software components
- provides mechanisms for terminating process instances
- manages human tasks included in process models
The Concept of a Process Engine

- Process engine distinguishes between two types of activities
  - Human interaction activities (human activities)
  - Automated activities (all others)
- Process engine executes the process model according to its semantics
- For human activities, the process engine includes a queue of human tasks
Architecture overview of process engines (jBPM)

- jBPM is an open source process engine

Source: jBPM
Architecture overview of process engines

IBM WebSphere Process Server

Source: IBM
Many different process engines available

- Appian [HUM]
- AristaFlow [INT]
- Fuege [HUM]
- Fujitsu Interstage [HUM]
- IBM WebSphere Process Manager [INT]
- Intalio [HUM]
- inubit [INT]
- Lombardi [HUM]
- Microsoft [INT]
- Oracle Business Process Manager [INT]
- Pegasystems [HUM]
- SAP NetWeaver Process Engine [INT]
- Savvion [HUM]
- SoftProject [INT]
- Software AG webMethods BPMS [INT]
- Stolz IT Consulting ProcessEngine [INT]
- Tibco iProcess [HUM]
- Tibco Rendezvous [INT]
- Vitria [INT]
- Enhydra Shark
- JBoss jBPM
- Activiti
Overview of the Activiti Process Framework
The Activiti Process Framework

- Activiti is a light-weight workflow and Business Process Management (BPM) Platform
- Target are business people, developers and system admins.
- Includes a BPMN 2 process engine for Java.
- Activiti is open-source and distributed under the Apache license.
- Activiti runs in any Java application, on a server, on a cluster or in the cloud.
Overview of Activiti Tool Stack

- Design tools
  - Web-based Activiti Modeler
  - Eclipse-based Activiti Designer
Overview of Activiti Tool Stack

- Supporting tools
  - Activiti Explorer for exploring and managing process instances
Overview of Activiti Tool Stack

Source: Activiti
The Activiti Explorer

- Activiti Explorer is a web application that is included when you download Activiti from the Activiti website.
- The purpose of Explorer is not a finished, end-user ready application, but rather to exercise and show the functionality of Activiti.
- The Explorer is meant as a demo, or maybe inspiration for people using Activiti in their own applications.
- Out of the box, Explorer uses an in-memory database, but it is easy to switch to your own database.
Starting Process Instances

Helpdesk process

Process Diagram

- First line support
  - normal end
- Handle escalated issue
  - escalated end
Administrating Process Instances

Expense process (144)

Process Diagram

Tasks

Variables

NAME | VALUE
--- | ---
Initiator | kermit
Process Implementation with Activiti Process Engine
Steps for Implementing a process with Activiti

- Model the process using Activiti Designer
- Decide for each activity how it shall be implemented
- Implement activities such as script activities and service activities
- Design user interface forms for user tasks
- Set conditions for exclusive gateways
- Deploy the process to the Activiti Engine
Modeling the process with Activiti Designer
Activities in BPMN for Implementation (Extract)

- A Script task can be used for executing a predefined script
- A Java Service task is used for invoking an external Java class
- A user task is used to model work to be performed by a human actor
- A Receive task for waiting for the arrival of a message
- A Business Rule task can be used for executing a business rule
**Script Tasks in Activiti**

- Process engine executes the script that is defined in the script task
- Activiti supports different scripting languages
  - Groovy, Clojure, JRuby, Jython, Javascript, Juel
- Groovy is default scripting language
- Jar file for the scripting language has to be added to the classpath
- In a script, we can read and set process variables

```xml
<scriptTask id="theScriptTask" name="Execute script" scriptFormat="juel" activiti:resultVariable="myVar">
  <script>
    #{echo}
  </script>
</scriptTask>
```
Script Tasks Examples

```xml
<scriptTask id="theScriptTask" name="Executescript" scriptFormat="groovy">
  <script>
    sum = 0
    for ( i in inputArray ) {
      sum += i
    }
  </script>
</scriptTask>

<scriptTask id="theScriptTask" name="Execute script"
  scriptFormat="juel" activiti:resultVariable="myVar">
  <script>${echo}</script>
</scriptTask>
```

- The return value of a script task can be assigned to an already existing or to a new process variable by specifying the process variable name as a literal value for the 'activiti:resultVariable' attribute of a script task definition.
Service Tasks in Activiti

- A Service task is a task that uses a form of a service
  - Web Service, Java service or other
- Is used for process-oriented integration of applications
- Example:
  - Querying customer information from a CRM application
  - Store order information into an ERP application
  - Querying credit rating details about a customer from an external service
- Activiti supports
  - Java Service tasks
  - Web Service tasks
Java Service Tasks in Activiti

- A Java Service task can be implemented as a Java class and then be called from the process.
- For deployment the *.jar file must be uploaded to the server.

```java
import org.activiti.engine.delegate.DelegateExecution;
import org.activiti.engine.delegate.JavaDelegate;

public class CreateApplicationTask implements JavaDelegate {

    @Override
    public void execute(DelegateExecution execution) throws Exception {
        LoanApplication la = new LoanApplication();
        la.setCreditCheckOk((Boolean) execution.getVariable("creditCheckOk"));
        la.setCustomerName((String) execution.getVariable("name"));
        execution.setVariable("loanApplication", la);
        System.out.println("Engine executed CreateApplicationTask");
    }
}
```
Human tasks in a process represent to dos for people

- Human tasks can be initiated either by a person or by an automated service
- Human tasks can be used to implement activities in business processes that require human interactions
  - Manual exception handling
  - Manual approvals
  - Manual input
- Human tasks can be used to invoke a service, or to coordinate the collaboration between people.
- A person from a group of people, to which the task is assigned, performs the work associated with the human task
User tasks in Activiti

- Human tasks are called user tasks in Activiti
- “A user task can be directly assigned to a user. This is done by defining a `humanPerformer` sub element. Such a `humanPerformer` definition needs a `resourceAssignmentExpression` that actually defines the user. Currently, only `formalExpressions` are supported.”
- “Only one user can be assigned as human performer to the task. In Activiti terminology, this user is called the `assignee`. Tasks that have an assignee are not visible in the task lists of other people and can be found in the so-called `personal task list` of the assignee instead.”
User tasks in Activiti

```xml
<userTask id='theTask' name='important task' >
    <potentialOwner>
        <resourceAssignmentExpression>
            <formalExpression>user(kermit), group(management)</formalExpression>
        </resourceAssignmentExpression>
    </potentialOwner>
</userTask>
```

- Tasks can also be put in the so-called **candidate task list** of people. In that case, the `potentialOwner` construct must be used. The usage is similar to the `humanPerformer` construct.

- Activiti can integrate with existing identity management solutions such as LDAP.
In order to execute user tasks, the process has to display data about the process instance and retrieve additional information from the user.

A **form** is a way of displaying and retrieving data from the user.
Forms in Activiti

- Activiti supports external forms support as well as built-in forms rendering
- For built-in forms rendering, forms are described in an xml extension for BPMN
- Fields of a form are stored as process variables in the process context
Forms in Activiti (XML Definition)

```xml
<startEvent ... />
<extensionElements>
  <activiti:formProperty id="numberOfDays" name="Number of days" value="${numberOfDays}" type="long" required="true"/>
  <activiti:formProperty id="startDate" name="First day of holiday (dd-MM-yyyy)" value="${startDate}" datePattern="dd-MM-yyyy hh:mm" type="date" required="true"/>
  <activiti:formProperty id="vacationMotivation" name="Motivation" value="${vacationMotivation}" type="string"/>
</extensionElements>
</userTask>

<startEvent id="startevent1" name="Start" activiti:formKey="your.form">
  <extensionElements>
    <activiti:formProperty id="name" name="Name" type="string" required="true"/>
    <activiti:formProperty id="emailAddress" name="Email Address" type="string" required="true"/>
    <activiti:formProperty id="income" name="Income" type="long" required="true"/>
    <activiti:formProperty id="loanAmount" name="Loan Amount" type="long" required="true"/>
  </extensionElements>
</startEvent>
```
Forms in Activiti

- Forms in Activiti can be defined for user tasks (human tasks)
- Forms in Activiti can be defined for start events of processes
Defining Forms in Activiti Designer

- Forms can be defined in Activiti Designer or directly in the process.xml
Defining an Approval Task Form

<userTask id="usertask1" name="Evaluate loan request">
  <extensionElements>
    <activiti:formProperty id="customerName" name="Customer Name" type="string"
      expression="${loanApplication.customerName}" writable="false" required="true"></activiti:formProperty>
    <activiti:formProperty id="income" name="Income of Customer" type="long"
      expression="${loanApplication.income}" writable="false"></activiti:formProperty>
    <activiti:formProperty id="requestedAmount" name="Requested loan amount"
      expression="${loanApplication.requestedAmount}" writable="false"></activiti:formProperty>
    <activiti:formProperty id="creditCheckOk" name="Outcome of credit check"
      expression="${loanApplication.creditCheckOk}" writable="false"></activiti:formProperty>
    <activiti:formProperty id="requestApproved" name="Do you approve the request?" type="enum"
      required="true">
      <activiti:value id="true" name="Yes"></activiti:value>
      <activiti:value id="false" name="No"></activiti:value>
    </activiti:formProperty>
  </extensionElements>
</userTask>

- In forms we can make use of process variables
  - we can define read/write fields
  - we can define new process variables
  - we can display values of process variables
Conditions of Exclusive Choice in Activiti

- Expressions are expressed in Unified Expression Language (UEL)
- Expressions can reference data of a process variable or call a method
Testing Processes in Activiti
Background

- In Activiti it is possible to use unit tests for processes
  - Processes can be tested without deployment to the server
- Activiti provides a built-in unit test generation facility
- Using the test, a process instance can be started and executed
- In addition, further functionality of Activiti can be used to fill in form values and manage user task during unit tests
Generating a unit test
public class ProcessTestCreditHandlingShort {
    ... 
    @Rule
    public ActivitiRule activitiRule = new ActivitiRule();
    @Test
    public void startProcess() throws Exception {
        RepositoryService repositoryService = activitiRule.getRepositoryService();
        repositoryService.createDeployment().addInputStream("CreditHandlingShort.bpmn20.xml",
                new FileInputStream(filename)).deploy();
        RuntimeService runtimeService = activitiRule.getRuntimeService();
        Map<String, Object> variableMap = new HashMap<String, Object>();
        ProcessInstance processInstance = runtimeService.startProcessInstanceByKey("CreditHandlingShort", variableMap);
        assertNotNull(processInstance.getId());
        System.out.println("id " + processInstance.getId() + " " + processInstance.getProcessDefinitionId());
    }
}
Deploying Processes to the Activiti Process Engine
Deploying Processes

- Model process model in Activity Designer
- Generate deployment artifacts
  - *.bar files
  - *.jar files
Deploying Processes

- Upload new deployable process into Activity Explorer
- Put additional classes into tomcat webapp directory
Summary of Lecture and References

- Different abstraction levels of process models
- Overview of process engines
- Implementation of processes with Activiti

References:
- J. Freund, B. Ruecker. Praxishandbuch BPMN 2.0, Hanser, 2012
- Activiti User guide, available online http://www.activiti.org/userguide/
Transactions in Activiti – Normal Execution

- Activiti executes processes in a transactional way which can be configured to suite your needs.
- Activiti is going to advance in the process, until it reaches wait states on each active path of execution.
Transactions in Activiti – Normal Execution

Activiti performs a depth-first search through the process graph and returns if it has reached wait states on every branch of execution.

A wait state is a task which is performed "later" which means that Activiti persists the current execution and waits to be triggered again.

The trigger can either come from an external source for example if we have a user task or a receive message task, or from Activiti itself, if we have a timer event.
When we reach the "generate invoice" task, we are creating a job "message" for Activiti to continue the process later and persisting it into the database.

This job is then picked up by the job executor and executed.