JEngine Documentation

BP2014W1 Team

May 28, 2015

Date Performed: January 1, 2015

Todo list

Abstract

This documentation is created in the context of the bachelor project BP2014W1 at the research group of Prof. Dr. Mathias Weske. The goal of the project was to implement an Engine for supporting the concept of *Production Case Management* (PCM) in order to provide a proof of concept and a prototype for several uses cases of Bosch Sofware Innovations. This document serves as a detailed documentation of the result.

1 Introduction

Production Case Management describes a flexible alternative to traditional processmanagement which focuses the modelling and execution of static, predefined processes. PCM, on the contrary, supports the usage of processvariants granting simplification of complex processmodels and flexibility in the execution by splitting complex processes into less complex fragments.

2 JEngine

Overall JEngine

2.1 JCore

The JCore contains several main components of our engine. Those contain the evaluation of the database and the enablement of activities in the ExecutionService etc.
In the JCore reside some of the most fundamental funtionalities that our engine offers. They include the possibility to run activities sequentially. Supported activity-types are user-task and email-task.
With PCM activities can be enabled from controllflow or dataflow point of view. That is the reason why we support dataobjects and their state transitions. Further activities can be referenced in PCM, which has the effect during execution, that referenced activities, which are enabled and share the same set of pre- and postconditions, can both switch to the state 'running' if one of them is being activated.
Also PCM-fragments consist of a subset of BPMN. Currently we support start- and end-events, activities and XOR- and AND-Gateways.

2.1.1 Class diagram

The ExecutionService is the class that manages the complete JCore. The ExecutionService manages all ScenarioInstances and provides methods to interact with a scenario instance, for example to do an activity.

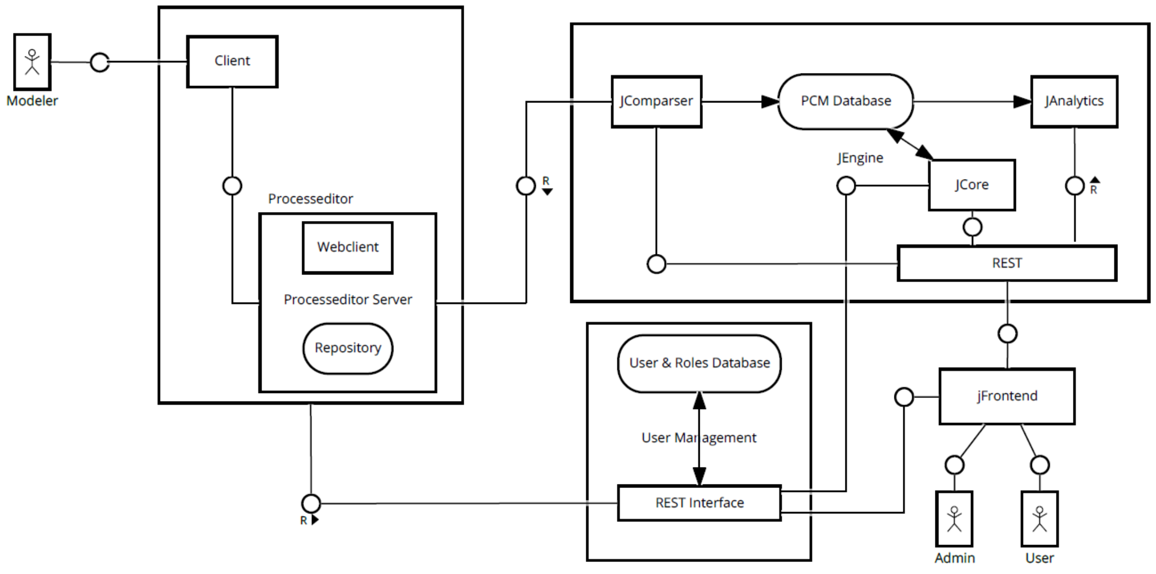

Figure 1: Meta Modell von PCM.

The ScenarioInstance represents an instance of a PCM scenario. It saves all FragmentInstances, ControlNodeInstances and DataObjectInstances. If a ScenarioInstance get initialized it initializes all FragmentInstances and DataObjectInstances.
The DataObjectInstances represent PCM data objects. They have a state and list of all there DataAttributeInstances. The DataAttributeInstance represent the data attributes and save the value and the type of the attribute.
The FragmentInstances have one function they initialize all ControlNodeInstances. ControlNodeInstances are ActivityInstances and GatewayInstances which are shown in the PCM model. All ControlNodeInstances have a StateMachine, a IncomingBehavior and a OutgoingBehavior. The StateMachine controls the state of the ControlNodeInstance and does the state transitions. The IncomingBehavior controls the incoming control flow. This is important for PCM gateways, because this behavior decides if a AND gateway for example can be activated. The OutgoingBehavior controls the outgoing control flow. For Example the data object output states for an activity are set here.
An ActivityInstance also have an TaskExecutionBehavior this behavior manages all things that happens if you execute the activity. This can be the HumanTaskExecutionBehavior that sets the data attribute states or a service task execution behavior like the EmailTaskExecutionBehavior that send automatic an eMail.

2.1.2 E-Mail-Tasks

The JEngine supports the execution of E-MailTasks. All of the E-Mail configurations are saved in the database in table *emailconfiguration*. Every E-Mail-Task which is modeled as described in section 4.2.1 gets its own database entry whith the corresponding controlnodeID (that's the databaseID, not the modelID from the XML) of the Mail Task.
The default template holds the databaseID -1. The values of the attributes from the default template are copied for each individual E-Mail-Task. In so far, the receivermailaddress, sendmailaddress, subject and messagecontent from the default template are used, if not specified differently. The databaseID and the controlnodeID are set automatically. If you wish to make an own configuration for the Mail Task, you can use the frontEnd ($Admin \rightarrow MailConfig$) by selecting the scenario, choosing the Activity and clicking on *Show Details*. After a window with the default content opens, you can enter the receiver, subject, and

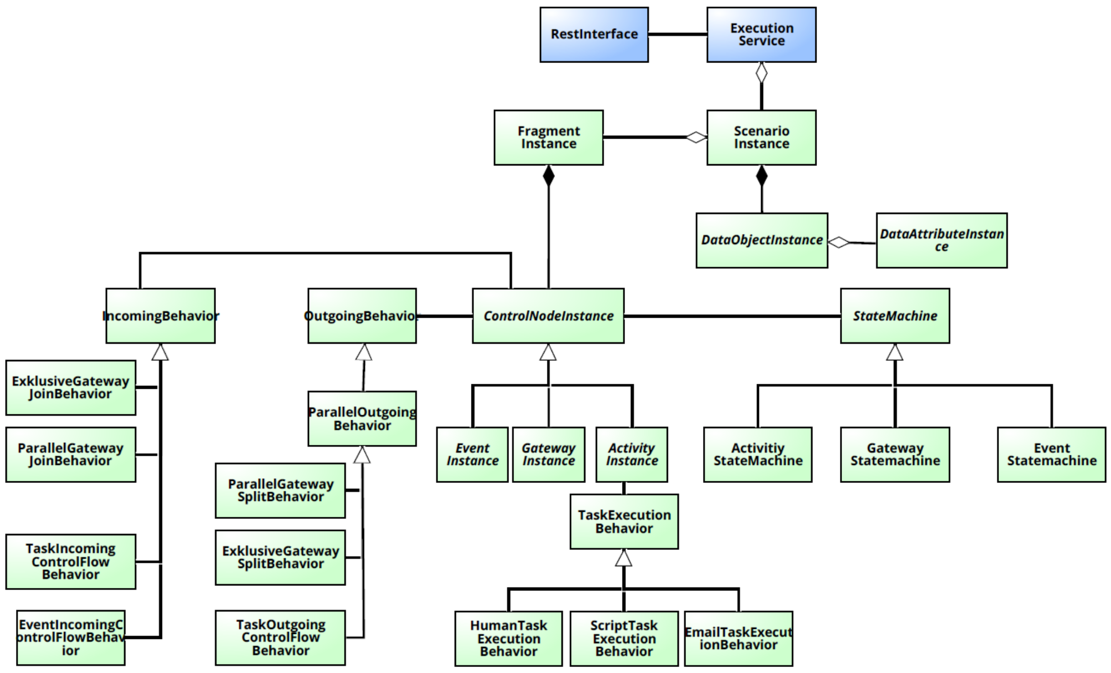
RestInterface
Execution Service
Fragment Instance
Scenario Instance
DataObjectInstance
DataAttributeInstance
IncomingBehavior
OutgoingBehavior
ControlNodeInstance
StateMachine
ExklusiveGateway JoinBehavior
ParallelGateway JoinBehavior
TaskIncoming ControlFlow Behavior
EventIncomingControlFlowBehavior
ParallelOutgoing Behavior
ParallelGateway SplitBehavior
ExklusiveGateway SplitBehavior
TaskOutgoing ControlFlow Behavior
Event Instance
Gateway Instance
Activity Instance
TaskExecution Behavior
HumanTask Execution Behavior
ScriptTask Execution Behavior
EmailTaskExecutionBehavior
Activitiy StateMachine
Gateway Statemachine
Event Statemachine

Figure 2: userview

message. Otherwise you can also change the values in the database.
Attributes. It is possible to access the values of dataAttributes in the message content, subject or e-Mail address by typing the following expression:

#dataobjectName.dataobjectAttributeName

This expression gets replaced by the value of *dataobjectAttributeName* which is an attribute of *dataobject-Name*.

2.1.3 WebServiceTasks

The JEngine supports the execution of WebServiceTasks. All of the WebService configurations are saved in the database in tables *webservicetasklink, webservicetaskpost, webservicetaskattribute*. Initial Webservice Tasks have no configuration entries in the database. The user have to configure the Tasks.
In the database table *webservicelink* is the method of the request defined. This can be *GET, PUT or POST*.
In the database table *webservicepost* gets the JSON for the PUT/POST request saved. The information from the dataobject and dataattributes can be used in the request. To access the values and states you have to type following expression: **#dataobjectName.dataobjectAttributeName** for the attribute value or **#dataobjectName** for the dataobject state.
In the database table *webserviceattribute* is defined which values from the JSON are saved in which dataat-tribute. The Engine uses the *key* to know how to access the JSON. The *order* ist the order in which the *keys* are used.

2.1.4 XOR Gateways

The JCore can execute XOR Gateways. The syntax of the conditions is shown by the following grammar. Expressions like *Dataobject.Dataattribute* or *Dataobject* are replaced with Dataattribute values and Dataobject states.

```
expression    ::=    expr2 (OPERATOR expr2)*;
expr2         ::=    NAME COMPARISON NAME;
NAME          ::=    CHARAC | CHARAC DOT CHARAC;
CHARAC        ::=    [#]('A'..'Z' | 'a'..'z' | '0'..'9')+;
COMPARISON    ::=    '=' | '<' | '>' | '<=' | '>=';
DOT           ::=    '.';
OPERATOR      ::=    ' & ' | ' | ' | '&' | '|';
```

2.1.5 REST-API

veraltet
for more informations about the REST please have a look at the REST Specification: https://github.com/BP2014W1/JEngine

2.1.6 ExecutionService

The ExecutionService is the component that is being used by the REST-API. So this component of the JEngine is essential to process queries that have been received via the REST interface. It establishes a database-connection to run queries on it. Most queries require the ID of an object to gather more information. Additionaly there are methods to instantiate activities and whole scenarios respectively.

3 Processeditor

4 PCM modelling using the Processeditor

This document explains how to use the Processeditor to create PCM models. A PCM-Process can be described by many PCM fragments and one PCM scenario.

4.1 Preparations

Currently you need both, the Processeditor Workbench and the Processeditor Server to model and Save PCM. You will use the Workbench for modelling and the Server as a global repository.

4.2 PCM Fragments

PCM Fragments are small Business Process models. They can be modelled using a subset of the BPMN-Notation:

- Tasks
- Events ** Blanko Start-Event ** Blanko End-Event
- Gateways ** Parallel Gateway ** Exclusive Gateway
- Data Objects
- Sequence Flow
- Data Flow

All these elements are offered by the model type PCM Fragment.

4.2.1 E-Mail Tasks

The engine supports the execution of E-Mail Tasks. For the modelling you need to open the relevant fragment and add a normal (unspecified) task. In the properties-editor (you can open it with a right-click on the task and then choosing *Properties...*) you simply set the stereotype to *SEND*.
For configurating the E-Mail-Task and the execution in the engine, see section 2.1.2.

4.2.2 Marking a Task as Global

PCM allows to use the same task in more than one fragment. To do so

1. model the Task (in one scenario)
2. Save the model to the repository
3. Right click on the Task and choose *Properties*
4. Set the *global flag*

4.2.3 Copy and Refer an existing Task

1. In another Fragment right click on any node
2. Choose "Copy and Refer Task"
3. Connect to the server if necessary
4. Choose the Model and the Task you want to refer
5. Click on Ok

4.3 PCM Scenario

A Scenario defines which PCM Fragments are part of one Process. All PCM Fragments have to be saved on the Server. You can alter the Scenario only by moving the nodes and adding/removing PCM Fragments.

4.3.1 Defining a PCM Scenario

1. Create a new PCM Scenario Model.
2. Right Click on one of the two nodes
3. Choose Add Fragments
4. Mark all Models you want to add in the left List (CTRL for multi select)
5. click on add than on ok

Now there should be entries for all the fragments (inside green node) and for all their data objects (inside white node).

4.3.2 Removing a Fragment From an Scenario

1. Right Click on one of the two nodes
2. Choose *Add Fragments*
3. Select all the models you want to remove from the right list
4. Click on *Remove* than click *Ok*

4.3.3 Set a Termination Condition

If a termination condition is full filled the process is terminated. Currently only one termination condition consisting of one Data Object in one specific state is possible.

1. Open your Scenario
2. Right Click on the canvas (not the Nodes)
3. Choose *Properties*
4. Fill out the *Termination Data Object* and *Termination State* fields

4.3.4 Copy and Alter a Complete Fragment

You can create a variation of an existing PCM Fragment using the Plug-in *Create Variant*.

1. First click on *Plug-Ins*
2. Choose *Create Variant*
3. Choose your Fragment and click on *Ok*

5 Addition

For further insides, we kindly refer to the created tutorials with their how-to section.