Decision Model and Notation

https://documentation.dcr.design/documentation/decision-model-and-notation/

Table of Contents

The DCR Portal now supports the DMN notation as outlined by the OMG group. DMN empowers business people to write complex business rules as *simple* Excel-like models. Once written DMN models can be executed by a computer and embedded into IT systems. This avoids the risk of developers re-writing the business rules into some programming language, e.g. C# or Java, introducing bugs. Using the native DMN model business people are given control of the execution and can change the business rules whenever they like. In this sense, DMN and DCR are very alike. Both empowers business people to take control of the business.

We've created a simple DCR process model with a decision model inside, please look at this model.

Decision Model and Notation

Let's start with a simple example of a business rule. If the amount of any expense report exceeds 1 mio the CEO must also approve. This can be expressed in an *Excel style* model as outlined below:

	When	Then	Annotations
	Amount	Rules	
	integer	integer	
1	>=100000	1	
2	-	0	
+			

Simple DMN model

If the amount if greater than 1.000.000 then the result is 1 (=yes). In any other situation the result is 0 (=no).

DMN and business processes

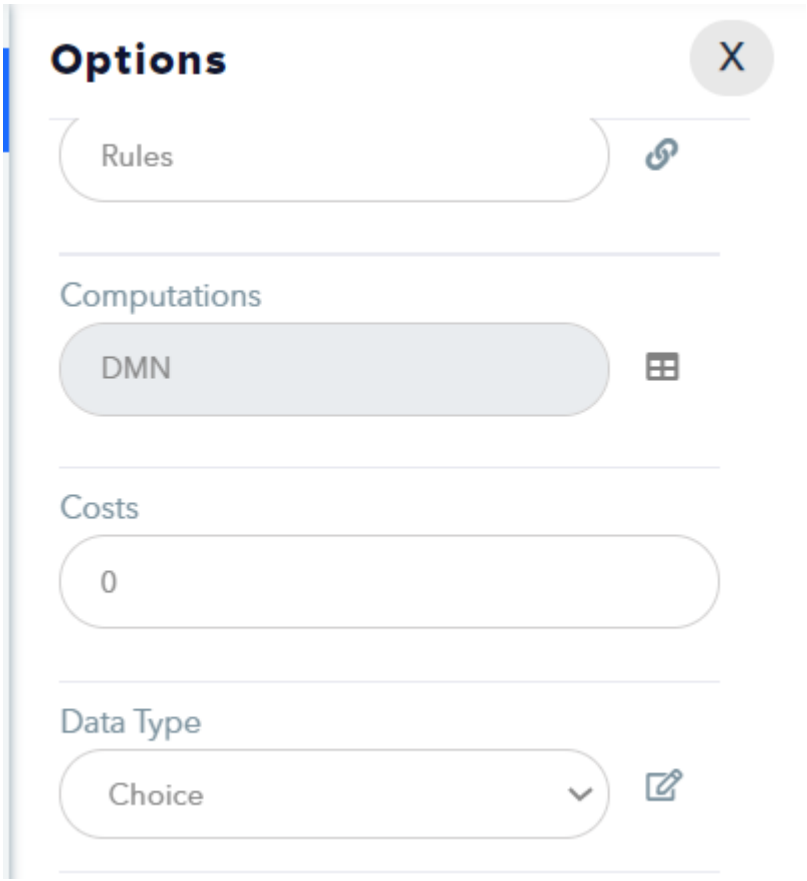
Traditionally business processes has been written down in the BPMN notation. However, in order to do this real life business processes must be simplified and large change management projects must be done in order to *transform* the organization to use the new processes. DCR, dynamic condition response, is a declarative business process management notation that can handle the complexity of

real life. Using DCR is simpler and faster than BPMN, and it is easier to change the process as laws and regulations or new ideas and requirements meets your organization.

DCR has an expression language where users can write business rules. However, business people have a hard time writing the *if-then-else* construct often seen in programming languages. Decision Model and Notation, DMN, is a more *intuitive* standard that can easily be used by business people. DCR now allows certain activities in the business process to be modelled as DMN tables.

DMN in DCR models

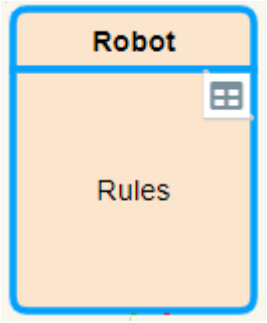
In order to use DMN in a DCR model you must define an activity with the data type *choice* field. Once you do that a table icon to the right of the computation gets visible:



When creating DMN expressions you must refer to DCR activities with proper data types. Currently we do not validate the existence of the corresponding DCR activity.

The result of the DMN expression is written as integers from the choice field. We plan to support a drop down list with the corresponding labels from the choice field later.

Any DMN activity will be shown in the DCR model with a table icon in the upper right corner:

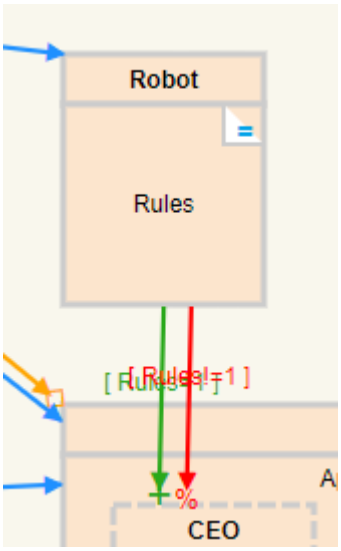


A DMN activity in a DCR process model

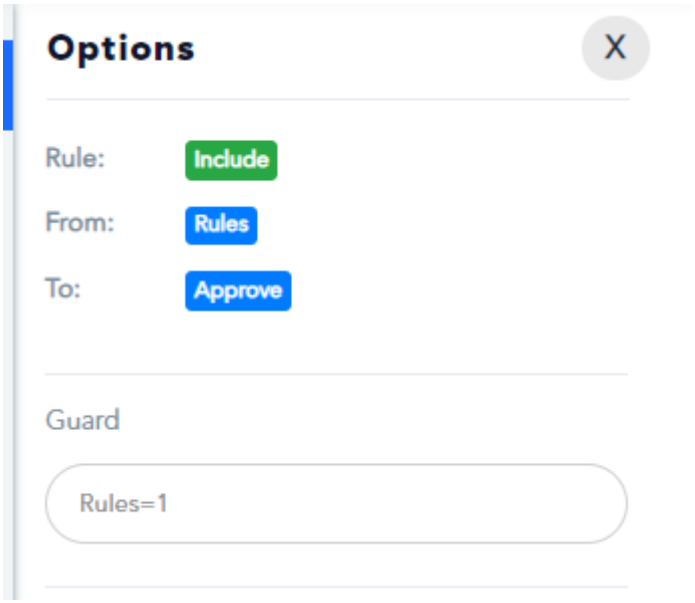
Notice that the role *Robot* is associated with the activity. The reason is that the DCR process engine automatically re-executes the DMN activity if the role is Robot every time the activity becomes *Pending*.

Using DMN expressions in DCR

DCR uses *guards* to control which business rules are applicable at a certain state of the DCR model as outlined here. The example process model use guards from the DMN Rules activity to the CEO Approve Activity in order to control whether this activity is part of the process or not:



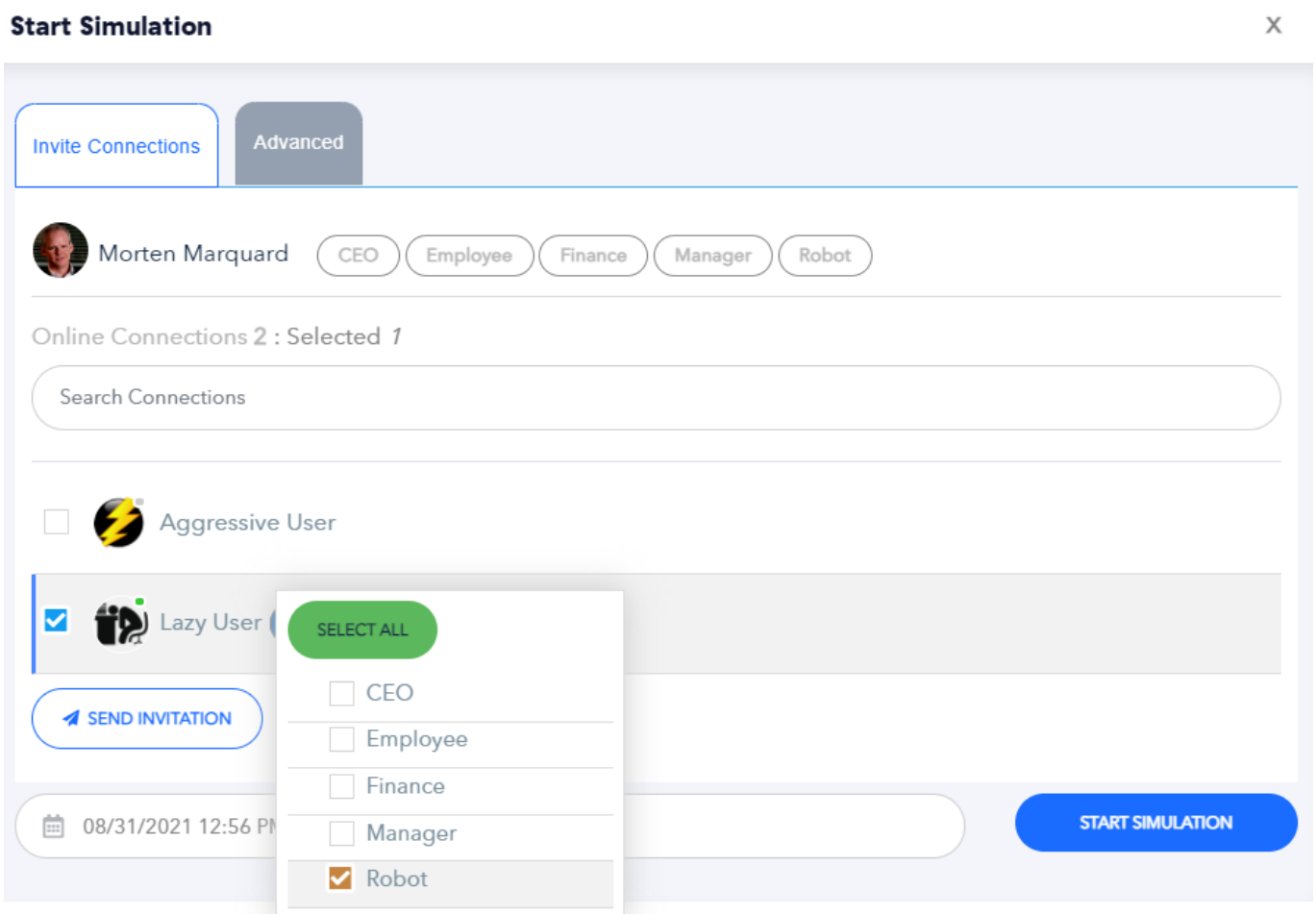
If you click the include arrow (green) you will notice the specific rule:



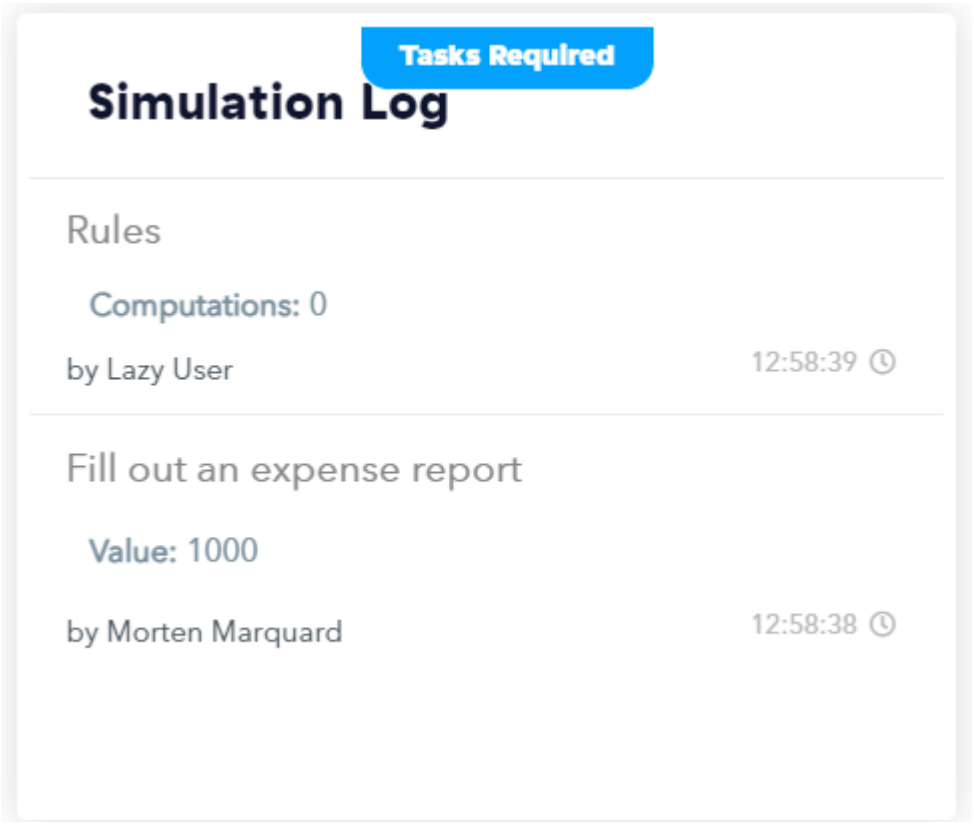
The expression "Rules=1" means that if the expression evaluates to true then the business include, here the include rule, is part of the model. When the DMN Rules activity is executed by the process engine the rule will have the effect of including the activity CEO Approval.

Simulation with DMN data

Given the example mentioned above we can start the DCR Simulator. When we start we assign the Robot to the Lazy user. The Lazy user executes any activity that is enabled and pending if it is pending initially or each time it becomes pending again.

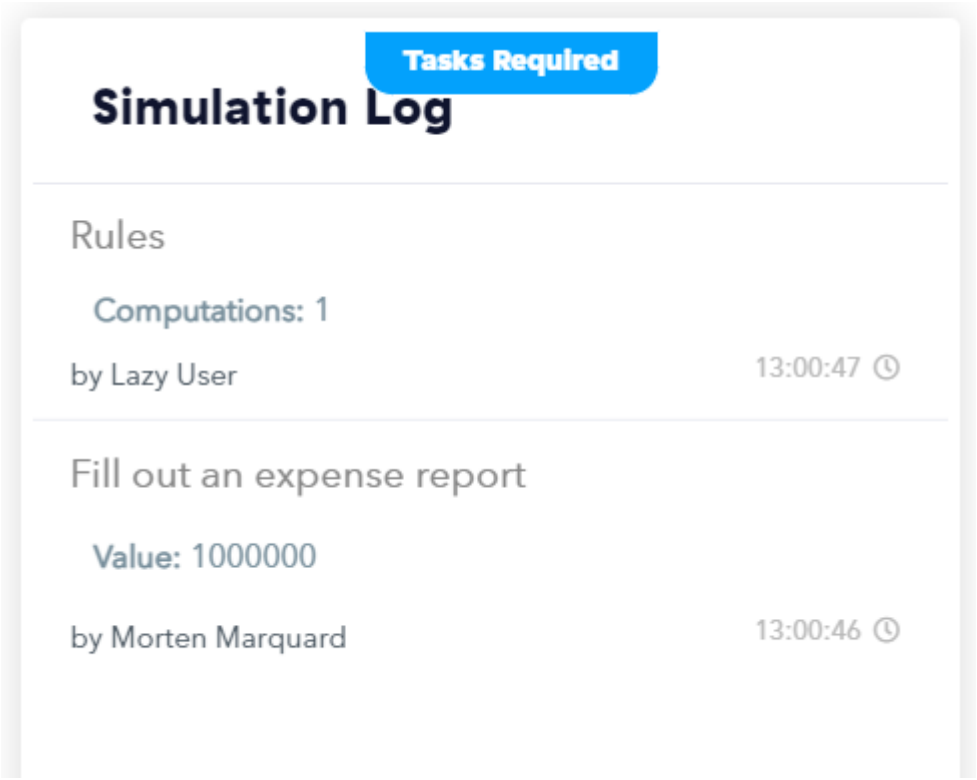


First you fill out the expense report with a 1.000 amount. Once this is done the Lazy user will automatically execute the Robot activity as illustrated from the Simulation Log below.



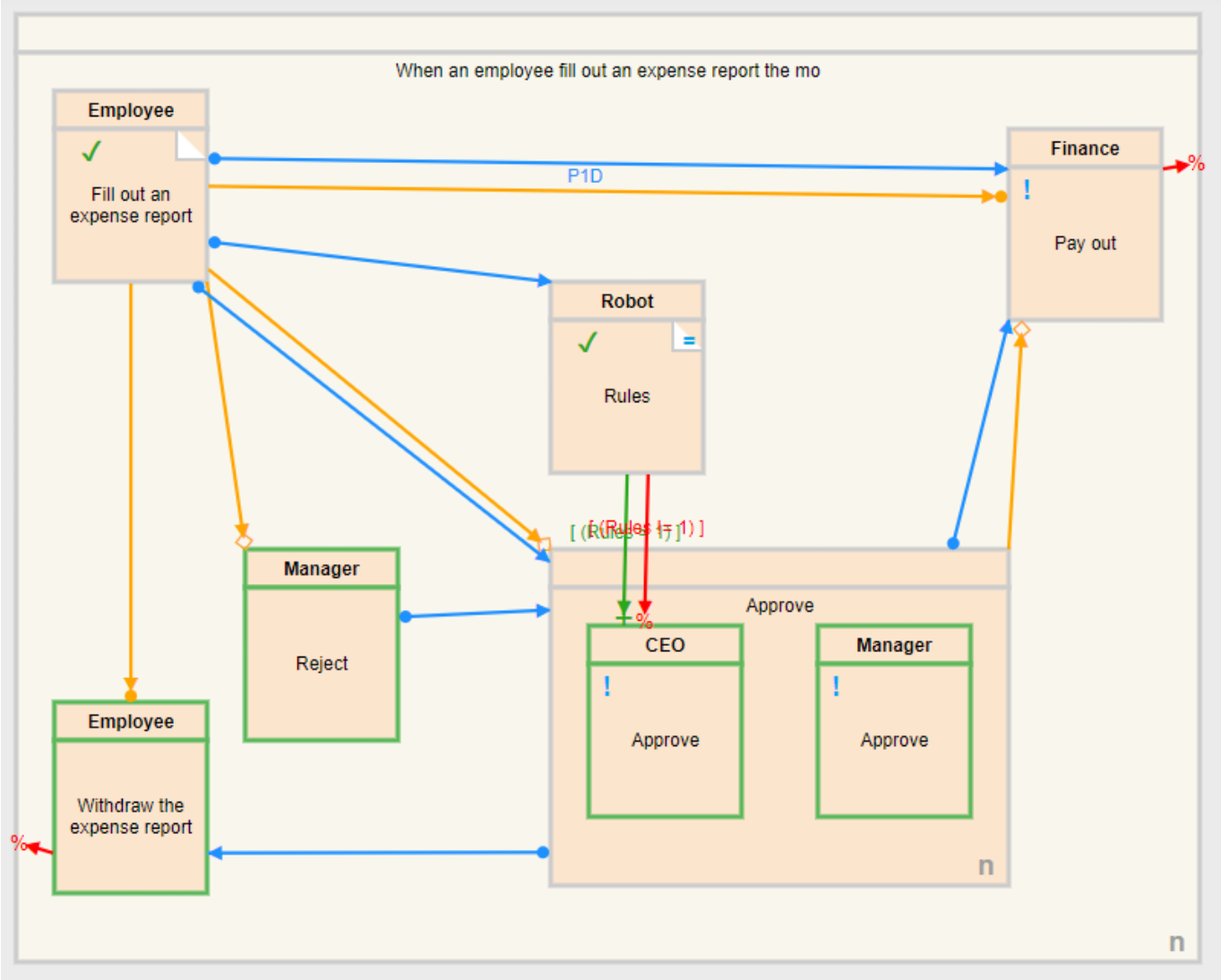
Notice that the result of the *Rules* is 0, which correspond to the rules found in the DMN table above.

If we try the same again but enters 1.000.000 as the amount we get the following Simulation Log:



Notice that the result of the computation is now 1.

After execution of the Rules with 1 the CEO Approve activity is included:



Accessing the DMN xml

The DMN table is stored inside the DCR process model using the DMN xml.

The DMN xml from the above example is shown below:

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions xmlns="https://www.omg.org/spec/DMN/20191111/MODEL/"
id="definitions_15idq6g" name="definitions"
namespace="http://camunda.org/schema/1.0/dmn" exporter="dmn-js
(https://demo.bpmn.io/dmn)" exporterVersion="11.0.1">
        <decision id="decision_0w5qbd2" name="">
                <decisionTable id="decisionTable_0lpmax3">
                        <input id="input1" label="Amount">
                                <inputExpression id="inputExpression1"
typeRef="integer">
                                        <text/>
                                </inputExpression>
                        </input>
                        <output id="output1" label="Rules" name=""
typeRef="integer"/>
                        <rule id="DecisionRule_1i05xtn">
```

```
                                <inputEntry id="UnaryTests_1lkh3gl">
                                        <text>&gt;=100000</text>
                                </inputEntry>
                                <outputEntry id="LiteralExpression_1e2xtyl">
                                        <text>1</text>
                                </outputEntry>
                        </rule>
                        <rule id="DecisionRule_0bcouon">
                                <inputEntry id="UnaryTests_0jkmgus">
                                        <text>-</text>
                                </inputEntry>
                                <outputEntry id="LiteralExpression_1a9s13j">
                                        <text>0</text>
                                </outputEntry>
                        </rule>
                </decisionTable>
        </decision>
</definitions>
```

Date: 07/09/2021
Type: User guide
Audience: Modelers