Oracle Corporation

Oracle Decision Service

Decision Model and Notation

Gary Hallmark

May 10, 2016
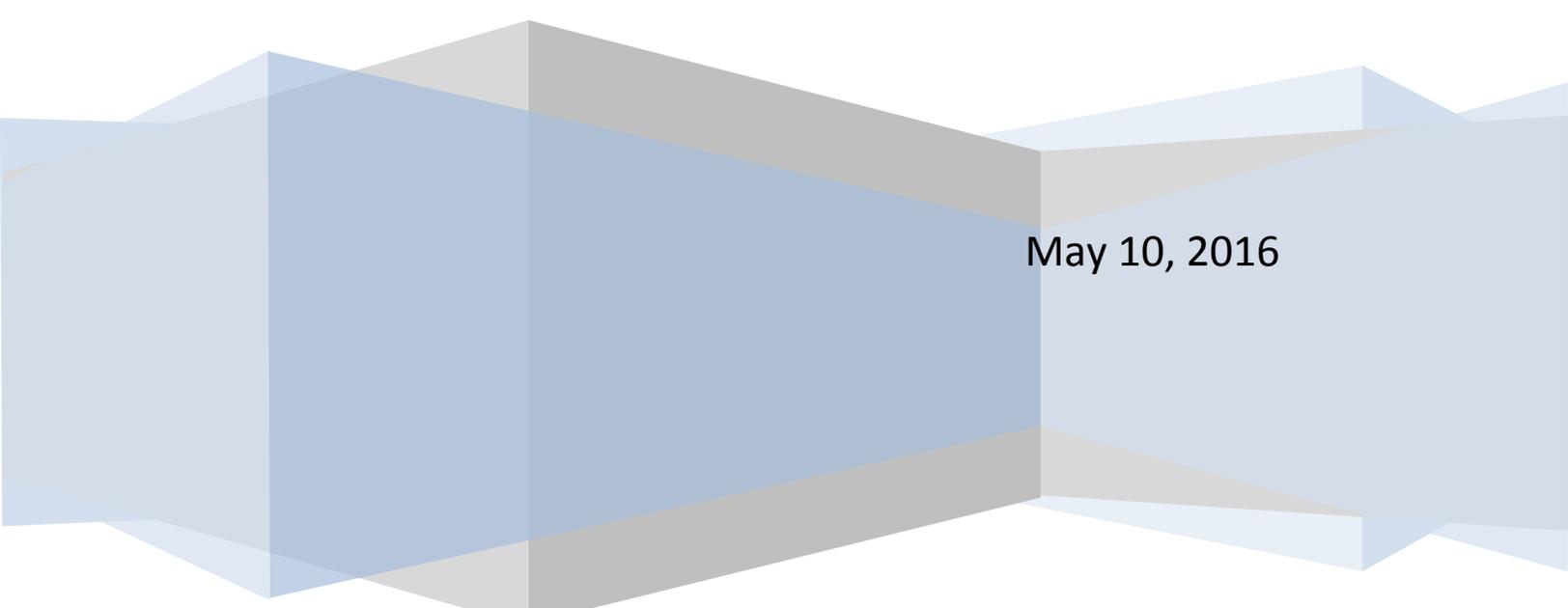
Oracle's Decision Service is a cloud-based service for modeling, automating, and executing business decisions. It supports all standard OMG DMN FEEL expressions, including decision tables, boxed expressions, and all FEEL expressions allowed at DMN Conformance Level 3.

We will demonstrate the service by modeling, testing, and running a decision model abstracted from Oracle Application's Approvals Engine. This is a workflow engine that has both human-powered and automated decision making. We will focus on a small automated component - given a purchase order with multiple lines, each line having potentially multiple cost centers, return a list of approvers who must sign off on this PO.

Our DMN standard solution uses functions to encapsulate decision tables that must be executed for each line and for each cost center, and FEEL for-loops to invoke the functions for each line and for each cost center.

Decision models can be tested in two ways. The conventional method is to encapsulate all or part of a model in one or more decision services, and use a service testing framework. For more immediate feedback, the modeling tool allows you to attach sample data directly to the model inputs and see the resulting outputs as you build the logic.

Decision services created in a DMN Model can be executed as RESTful web services by posting values for the inputs and then receiving values for the outputs.

Slide 1

Today we will talk about a new decision service tool and engine we are developing for Oracle Process Cloud.

Slide 2

Oracle Process Cloud Service is a BPMN-based process design, execution, and management service running in the Oracle Public Cloud.

DMN is a new standard notation for modeling and automating parts of decision-heavy processes (e.g. expense approval, loan approval, spending approval, document approval).

Once you have mastered DMN's expression language, then you've mastered DMN. There's nothing hidden or stateful about DMN (unlike production rules).

We will show how DMN easily handles some complex use cases that until now required a rules expert – or decision logic in your process diagram – including the processing of a master/detail/sub-detail hierarchy.

Slide 3
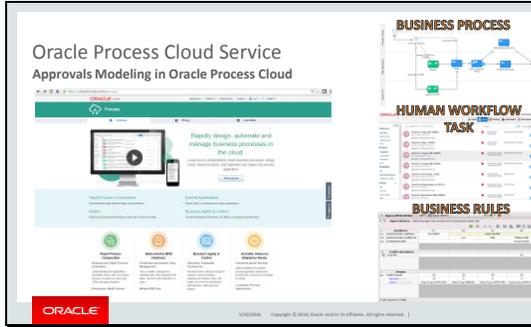
Oracle Process Cloud is our Platform-as-a-Service for business process automation. It offers many of the same features as the Oracle BPM Suite, including diagramming, KPI definition and dashboards, human workflow modeling and mobile-apps, and business rules based on production rules with layers of business-friendly syntax options.

Many use-cases are decision processes, e.g. approvals.

It is not always easy to see the decisions, and no standard place to document the decision/knowledge requirements.

Slide 4
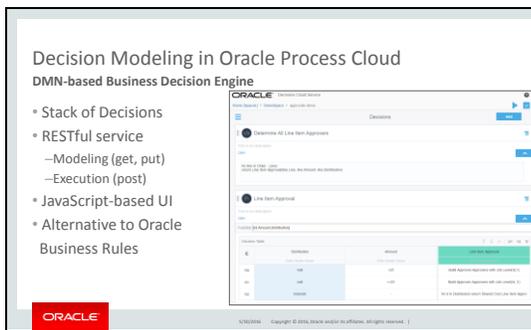
Our new Decision Model service is DMN-based decision modeling and execution engine.

We do not yet have a graphical editor, so the decisions are modeled in a stack, with the logic of one decision able to see the lower decisions in the stack, but not the higher decisions.

It will be offered in Oracle Process Cloud as an alternative to the existing Oracle Business Rules.

Slide 5
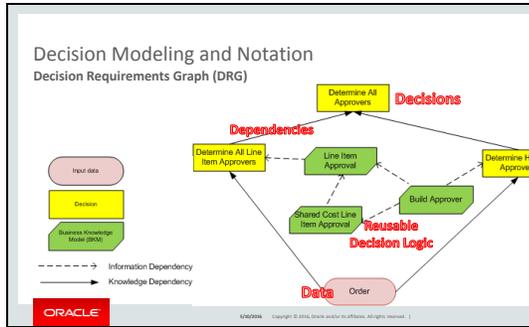
In DMN, a Decision Requirements Graph is for modeling decisions.

Here we have decisions, sub-decisions, dependencies, reusable decision logic called business knowledge models (BKMs), and input data.

In our DMN-based product, we do not yet have the DRG. We can use an equivalent decision stack.

BKMs are represented as a decision whose logic is a function.

In words, the decision model determines all the approvers for an order by combining the approvers for the line items with the approvers indicated by the order header information.

An approver here is a little data structure that gets returned, and the data structure is built by Build Approver. To determine approvers for all lines, you must iterate over each line. And to determine approvers for a single line, you may need to iterate the Shared Cost detail.

Slide 6
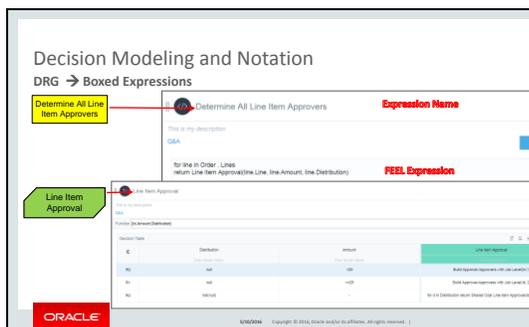
Within each box in the decision diagram is a boxed expression. Boxed expression is the notation for modeling decision logic.

The logic in the boxed expressions is implemented with an expression language called FEEL.

Slide 7

The logic in the boxed expressions is implemented with an expression language called FEEL, which stands for Friendly Enough Expression Language.
The boxed expressions can reference each other through their names.

Slide 8

FEEL expressions are interpreted with no side-effects. Every expression has a value. The value of a 'for' expression is a list of the return expression mapped to each element of the in expression. The value of an 'if then else' expression is the value of the 'then' or 'else' expression, depending on whether the 'if' expression is true or not.

Slide 9

The decision table takes input expressions, matches them with the rules, and returns an output.
The hit policy defines what happens when rules overlap.

Slide 10

FEEL – if then else expressions

If then else expression are commonly used in rules expressions. We've added a new boxed expression for it, in an obvious way.

Inside the boxes, we can use literal expressions or another boxed expression.

Slide 11

FEEL – for in return expressions

The for in return expression loops through a list and evaluates an expression for each element,
then returns the list result.

Slide 12

FEEL – for in return expressions

In the demo, we will show how to use the For expression to iterate through line items of an order.
Calculate the approvers for each line, and return a combined result.

Slide 13

Item Definitions is a type definition in DMN used for defining the type of
input data, function arguments, and decision results.

Slide 14
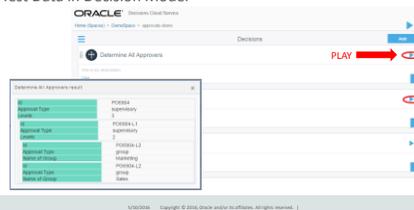
After a decision model is completed, users can define test data in the decision model for testing.
In our tools, we have "play" buttons to test the decisions against the sample data.

Slide 15
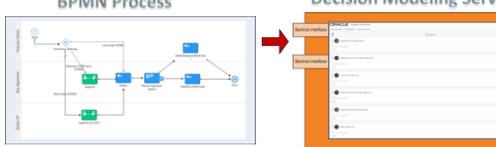
A Decision Service encapsulates one or more decisions in a decision model for external consumption.
For instance, a BPMN Process can call a decision service to evaluate a decision.

Slide 16

For our demo, we are going to show a decision model that takes input data from a Purchase Order and derive a list of approvers needed for approving the purchase order.

Slide 17

The demo shows how DMN can handle multiple line items.
The output is a list of approver attributes. The attributes tell how the item should be approved, and who should be approving it.

Slide 18

For header approval, we will need a manager from a particular supervisory level based on the PO total amount.
For line items, it may require one or more approvers from different cost centers.

Slide 19
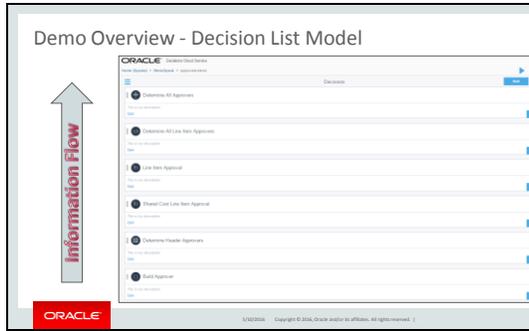
Slide 20

We will take a look at the live demo now.