OpenRules®

Open Source Business Decision Management System

Release 6.3.1

Getting Started

OpenRules, Inc.

www.openrules.com

May-2014

Table of Contents

INTRODUCTION

Business Decision Management

OpenRules® is an open source Business Rules and Decision Management System developed by OpenRules, Inc. in 2003. It is oriented to subject matter experts (business analysts) allowing them to represent, test, and maintain their business logic using MS Excel or Google Docs. Today OpenRules® is a winner of several software awards and is used worldwide by multi-billion corporations, major banks, insurers, health care providers, government agencies, online stores, universities, and many other institutions.

OpenRules® offers the following decision making components:

- Rule Repository for management of enterprise-level decision rules
- Rule Engine for rules execution
- Rule Learner for rule discovery and predictive analytics
- Rule Solver for solving constraint satisfaction and optimization problems
- Finite State Machines for event processing and "connecting the dots"
- Rule Dialog for building rules-based Web questionnaires.

Integration of these components with executable decisions has effectively made OpenRules® a reliable and easy to use product oriented to "decision-centric" application development.

This document helps a user to get started with OpenRules®. It describes how to install OpenRules® and use it starting with simple examples. This document is aimed at business analysts and software developers who may assist them in development of decision making applications. More information is available in the User Manual.

Document Conventions

The regular Century Schoolbook font is used for descriptive information.

The italic Century Schoolbook font is used for notes and fragments clarifying the text.

```
The Courier New font is used for code examples.
```

INSTALLATION

To start working with OpenRules® you need to download and unzip one folder "openrules.decisions" from http://openrules.com/download.htm. You also need to make sure that you have free Java Platform (JDK) and Apache Ant installed. See the detailed installation instructions at http://openrules.com/pdf/OpenRulesInstallationGuide.pdf.

INTRODUCTORY EXAMPLE

This section will demonstrate how to build simple executable decision projects with underlying decision tables using only OpenRules® and Excel. We will take a very simple example (similar to a traditional "Hello World!") and will explain how to implement it as an executable decision project.

Decision "DetermineCustomerGreeting"

The following example demonstrates how to develop a very simple application that should decide how to greet a customer during different times of the day. The proper decision might be a part of an interactive voice response (IVR) system. For example, if a customer Robinson is a married woman and local time is 14:25, we want our decision to produce a greeting like "*Good Afternoon, Mrs. Robinson!*"

We will use Excel to represent decisions, related decision tables, and several test cases. Then we will demonstrate how to connect them with data coming from two different sources:

- From Excel Data tables
- From Java objects.

Starting with Decision

We will call our decision "DetermineCustomerGreeting" because it should "Determine" (a decision word) the correct value for the unknown decision variable "Customer Greeting". To make this decision we should make three sub-decisions:

- Define Current Time (to greet the customer based on her own time of the day)
- Define Greeting Word (e.g. "Good Afternoon")
- Define Salutation Word (e.g. "Mrs.")

We will need to create rules for each of these sub-decisions. OpenRules® provides special tables for representing decisions where each sub-decision is implemented as a Decision Table. So, first we will create an Excel Decision table "DetermineCustomerGreeting":

Decision DetermineCustomerGreeting	
Decisions	**Execute Decision Tables**
Define Current Time	DefineCurrentHour
Define Greeting Word	DefineGreeting
Define Salutation Word	DefineSalutation

The first column of this table defines the decision name and the second column defines the decision table that implements the related (sub) decision.

Defining Decision Tables

Let's first define the current hour that preferably should be the current hour at the customer's location. The following decision table with only one conclusion

DecisionTable DefineCurrentHour	
Conclusion	
Current Hour	
Is	::= customer(decision).currentHour

defines the decision variable "Current Hour". We may assume that there is a method `customer(decision)` that returns a customer that will be passed as a parameter to our decision. We also may assume that each customer has an attribute "currentHour". That's why we may use the expression `::=customer(decision).currentHour` inside our decision tale.

Now, we will define a decision table that based on the Current Hour determines the proper value for the decision variable "Greeting":

DecisionTable DefineGreeting					
Condition		Condition		Conclusion	
Current Hour		Current Hour		Greeting	
>=	0	<=	11	Is	Good Morning
>	11	<=	17	Is	Good Afternoon
>	17	<=	22	Is	Good Evening
>	22	<=	24	Is	Good Night

Here the first row defines the name "DefineGreeting" of the decision table following the keyword "DecisionTable". The second row specifies types of columns in this decision table using keywords "Condition" for conditions and "Conclusion" for actions. The third row describes decision variables for the proper columns. The decision table "DefineGreeting" uses the variable "Current Hour" in condition-columns and the variable "Greeting" for the corresponding conclusion.

The rows 4-7 contain rules with operators and operands appropriate to the variables in the column headings. For instance, the second rule can be read as:

"IF Current Hour is more than 11 AND Current Hour is less or equal to 17 THEN Greeting is Good Afternoon".

OpenRules® allows a user to define decision tables in different ways. For example, replacing keywords "Condition" to "If" and "Conclusion" to "Then" we may represent the same decision table in a much more compact way:

DecisionTable DefineGreeting	
If	Then
Current Hour	Greeting
0-11	Good Morning
11-17	Good Afternoon
17-22	Good Evening
22-24	Good Night

However, here we will rely on the default assumption that the interval such as "11-17" includes its bounds.

Now, we may similarly define a decision table "DefineSalutation" that determines a salutation word:

DecisionTable DefineSalutation		
If	If	Then
Gender	**Marital Status**	**Salutation**
Male		Mr.
Female	Married	Mrs.
Female	Single	Ms.

Let's assume that for customers born in 2007 and later we want to generate a salutation "Little" independently of their Gender (and, of course, Marital Status). To do that, we will simply add another condition for a decision variable "Date of Birth":

DecisionTable DefineSalutation							
Condition		Condition		Condition		Conclusion	
Gender		**Marital Status**		**Date of Birth**		**Salutation**	
Is	Male					Is	Mr.
Is	Female	Is	Married	<	January 1, 2007	Is	Mrs.
Is	Female	Is	Single			Is	Ms.
				>=	January 1, 2007	Is	Little

All 3 decision table are now specified.

Defining Business Glossary

Our decision tables deal with the following variables:

- Current Hour
- Gender
- Marital Status
- Date of Birth
- Greeting
- Salutation

Now we need to associate these variables with business concepts and their attributes. We will assume that variables Current Hour, Gender, Marital Status, and Date of Birth are related to the business concept Customer. The output variables Greeting and Salutation are related to an object Response. So, we may define our business glossary that basically connects all (!) variables with business concepts and concrete data:

Glossary glossary		
Variable	**Business Concept**	**Attribute**
Gender	Customer	gender
Date of Birth		dob
Marital Status		maritalStatus
Current Hour		currentHour
Greeting	Response	greeting
Salutation		salutation
Result		result

We added one more output variable "Result" to the Response, as we plan later on to produce the complete string like "Good Afternoon, Mrs. Robinson!" and save it in this variable as well.

Defining Test Data in Excel

Now, we want to create a test case for the developed decision. Initially, we will define test data in Excel using OpenRules® tables of the types "Datatype" and "Data". Based on the Glossary, we need to define two Datatypes "Customer" and "Response":

Datatype Customer	
String	name
String	maritalStatus
String	gender
Date	dob
int	currentHour

Datatype Response	
String	greeting
String	salutation
String	result

Then we create several test-customers:

Data Customer customers				
name	maritalStatus	gender	dob	currentHour
Customer Name	**Marital Status**	**Gender**	**Date of Birth**	**Current Hour**
Robinson	Married	Female	1/15/1990	20
Smith	Single	Male	10/19/2008	11

and one variable for a response:

Variable Response response		
greeting	salutation	result
Greeting	**Salutation**	**Result**
?	?	?

Note that the second rows should contain exactly the same names (with no spaces) as in the proper data types and in the Glossary. All variables inside the response should be defined by our decision tables.

Finally, we will use a table of the predefined type "DecisionObject" to map business concepts Customer and Response (defined in the glossary above) with business objects defined in the test data:

DecisionObject decisionObjects	
Business Concept	**Business Object**
Customer	:= decision.get("customer")
Response	:= response

Executing Decision

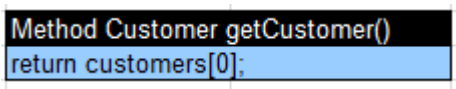
Now we are ready to execute the defined decision against the above test data. You may create and run a simple Java launcher for our decision "DetermineCustomerGreeting":

```
import com.openrules.ruleengine.Decision;

public class Main {

    public static void main(String[] args) {
        String fileName = "file:rules/main/Decision.xls";
        Decision decision = new Decision("DetermineCustomerGreeting",fileName);
        Object customer = decision.execute("getCustomer");
        decision.log("INPUT: " + customer);
        decision.put("customer", customer);
        decision.execute();
        decision.log("Decision: " + decision.getOutput());
    }
}
```

Here we create an instance of the standard OpenRules class Decision using the decision name "DetermineCustomerGreeting" and the name of the Excel file in which we keep our main table "Decision".

Then we create on object "customer" by getting it from the already defined array of

"customers". To do that, we execute the decision for the method "getCustomer" that in Excel may look like:



This code uses a predefined OpenRules® class "Decision" that extends HashMap and allows a user to put and get any object to the decision using a key "customer":

```
decision.put("customer", customer);
```

Then we call the method `decision.execute()` that executes our decision. If we Execute this Java launcher, it will produce the following results:

```
*** Decision DetermineCustomerGreeting ***
Decision has been initialized
INPUT: Customer(id=0) {
  name=Robinson
  currentHour=20
  dob=Mon Jan 15 14:51:58 EST 1990
  gender=Female
  maritalStatus=Married
}
Decision Run has been initialized
Decision DetermineCustomerGreeting: Define Current Time
Conclusion: Current Hour Is 20
Decision DetermineCustomerGreeting: Define Greeting Word
Conclusion: Greeting Is Good Evening
Decision DetermineCustomerGreeting: Define Salutation Word
Conclusion: Salutation Is Mrs.
```

You may see how our decision tables have been executed correctly producing vales for the decision variables Greeting and Salutation. However, we haven't added a decision Result yet.

Adding Decision Output

We will add two more sub-decisions our decision table "DetermineCustomerGreeting":

Decision DetermineCustomerGreeting	
Decisions	**Execute Decision Tables**
Define Current Time	DefineCurrentHour
Define Greeting Word	DefineGreeting
Define Salutation Word	DefineSalutation
Define Result	DefineResult
Define Decision Output	:= decision.setOutput(${Result})

The fourth sub-decision will define the decision variable "Result" using the following decision table:

DecisionTable DefineResult	
Conclusion	
Result	
Is	::= ${Greeting} + ", " + ${Salutation} + customer(decision).name + "!"

Gere we again used a Java expression that concatenates values of the decision variable "Greeting", "Salutation", and the customer's name. We also demonstrate here how to use OpenRules macros like `${Greeting}` to get the current value of the decision variable "Greeting".

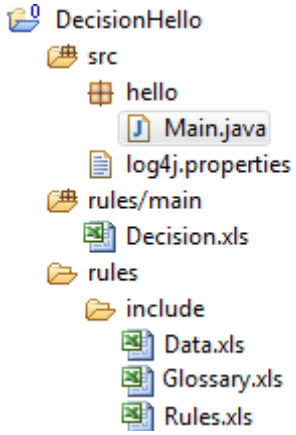
And finally, the fifth sub-decision above simply uses a Java snippet to set the output of the decision using the macro `${Result}` to get the generated value of the variable "Result".

If we run our decision again, along with the previous output it will also display the following result:

```
Decision DetermineCustomerGreeting: Define Result
Conclusion: Result Is Good Evening, Mrs.Robinson!
Decision DetermineCustomerGreeting: Define Decision Output
Decision has been finalized
Decision: Good Evening, Mrs.Robinson!
```

Decision Project

The described decision is available in the project "DecisionHello" included in the OpenRules® installation. This project has the following structure:



The folder "rules" is a rules repository. Its sub-folder "rules/main" contains the main methods for different decisions. For this example, it is one file "Decision.xls" with the table "DetermineCustomerGreeting". The sub-folder "rules/include" contains Excel files for data, glossary, and decision tables described above. The file "src/hello/Main.java" contains the standard Java launcher for executing Decision.xls.

A user may distribute Excel tables between xls-files placed in different directories and subdirectories. To define a desired structure for the rules repository, a user should include a special OpenRules® table "Environment" in the main file "Decision.xls". In this example the "Environment" table looks as follows:

Environment	
include	../include/Rules.xls
	../include/Data.xls
	../include/Glossary.xls
	../../../openrules.config/DecisionTemplates.xls

The "../include/" in front of files Rules.xls, Data.xls, and Glossary.xls point to the location of these files relative to the file rules/main/Decision.xls. All decision tables utilize the standard file "DecisionTemplates.xls" that is located in the OpenRules® configuration project "openrules.config" (3 levels above rules/main).

Accepting Data from Java

In the real-world, decisions are incorporated in business applications that supply the decisions with data and expect to receive back data-specific decisions. Above we have shown how to define data types and data instances in Excel tables. Now we will explain how the same decision deals with data defined in Java objects. The standard

OpenRules® installation includes the proper decision project "DecisionHelloJava".

This project has a Java package "hello" with two Java classes "Customer" and "Response" that are simple Java beans with the following organization:

```
public class Customer {
	String	name;
	String	maritalStatus;
	String	gender;
	double	salary;
	Date	dob;
	int	currentHour;
	// getters and setters
}

public class Response {

	String greeting;
	String salutation;
	String result;
      // getters and setters
}
```

To tell the decision that the objects "customer" and "response" now come from Java we will modify the table "decisionObjects":

DecisionObject decisionObjects	
Business Concept	**Business Object**
Customer	:= decision.get("customer")
Response	:= decision.get("response")

We also may remove the file Data.xls from the Environment table and add the following "import.java" statement:

Environment	
include	../include/Rules.xls
	../include/Glossary.xls
	../../../openrules.config/DecisionTemplates.xls
import.java	hello.*

The main Java class Main.java now additionally creates instances of the classes Customer and Response, puts them into the instance of Decision, and executes OpenRulesEngine for this decision using the following Java launcher:

```
public static void main(String[] args) {
    try {
        String fileName = "file:rules/main/Decision.xls";
        Decision decision = new Decision("DetermineCustomerGreeting",fileName);
        Customer customer = new Customer();
        customer.setName("Robinson");
        customer.setGender("Female");
        customer.setMaritalStatus("Married");
        DateFormat df = DateFormat.getDateInstance(DateFormat.SHORT);
        customer.setDob(df.parse("2/15/1990"));
        decision.put("customer", customer);
        Response response = new Response();
        decision.put("response", response);
        decision.saveRunLog(true);
        decision.execute();
        decision.log("Decision: " + decision.getOutput());
        decision.printSavedRunLog("results.txt");

    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

Now it will produce the following output:

```
*** Decision DetermineCustomerGreeting ***
Decision has been initialized
Decision Run has been initialized
Decision DetermineCustomerGreeting: Define Current Time
Conclusion: Current Hour Is 0
Decision DetermineCustomerGreeting: Define Greeting Word
Conclusion: Greeting Is Good Morning
Decision DetermineCustomerGreeting: Define Salutation Word
Conclusion: Salutation Is Mrs.
Decision DetermineCustomerGreeting: Define Result
Conclusion: Result Is Good Morning, Mrs.Robinson!
Decision DetermineCustomerGreeting: Define Decision Output
Decision has been finalized
Decision: Good Morning, Mrs.Robinson!
Print saved run log to results.txt
```

SUMMARY OF INTRODUCED CONCEPTS

In the above introductory example we demonstrated the use of several predefined OpenRules® tables:

- **Decision** – to specify a decision structure as a combination of intermediate decisions and related decision tables

- **DecisionTable** – to specify business logic behind different decisions
- **Glossary** – to specify all used decision variables and related business concepts and their attributes. The glossary remains independent of any particular implementation of business objects that could be defined in Excel data tables, in Java classes, or in XML.
- **DecisionObject** – to map business concepts defined in a glossary with concrete business objects defined in Java or Excel.
- **Datatype** – to specify data types for test data
- **Data** – to specify concrete test instances
- **Method** – to specify access to different objects (like Excel Data or Java-based instances) or using Java snippets to express more complicated calculations (like local time)
- **Environment** – to specify a structure of the decision projects.

These examples demonstrate how business analysts can create and test decisions without coding. At the same time it demonstrated how software developers can help to integrate a tested decision into an existing Java application.

The following simple decision projects will demonstrate a different use of already introduced concepts.

Other Decision Examples

The standard OpenRules® installation includes many other simple and more complex examples – see http://openrules.com/eval_standard_projects.htm. You may start analyzing, modifying, and executing these projects . Then you may look at a more complex example in the tutorial "Calculating Tax Return". This user manual covers the core OpenRules® concepts in greater depth. Additional OpenRules® components are described in separate user manuals: see Rule Learner, Rule Solver, and Rule Dialog.

Technical Support

Direct all your technical questions to support@openrules.com or to this Discussion Group. Read more at http://openrules.com/services.htm.