BPM Voices: Using business rules to mine business rules test scenarios

Pierre Berlandier (pberland@us.ibm.com)
Senior Technical Staff Member
I.B.M.

29 June 2011

In this column, Pierre Berlandier presents a flexible solution to the problem of creating and selecting of test scenarios, based on the different components of the WebSphere® ILOG® JRules Business Rules Management System (BRMS) platform. The solution leverages the Decision Warehouse to capture a detailed trace of the scenario executions, and the Business Action Language for mining the desired test scenarios.

Introduction

One critical challenge in the development and maintenance of a rule-based decision service is to gather a collection of relevant test scenarios. Relevance is, of course, subjective and can be defined from different points of view: for the business, relevance can mean testing the behavior of a very specific aspect of the business policy that has not been exercised extensively yet. For IT, a set of test scenarios may be considered relevant if it offers enough coverage among all the possible decision paths.

Building test scenarios from scratch for a real-life decision with tens (and often hundreds) of data points can be a daunting task, both for the business, which has to work inductively on the rules to select the proper scenario values, and for the IT groups, which have to debug the test scenarios to ensure that they are complete, well-formed and eventually executable by the rule engine. The activity thus needs the combined knowledge of the business, the rules implementation, and the BRMS platform.

Meanwhile, the production database that supports a business application often contains a treasure trove of potential test data. It is usually possible to extract request data that is (or has been) submitted to the production system and transform it, so that it conforms to the request format that is expected by the rule-based decision service. However, to exploit this large number of potential test scenarios, we have to provide an easy way for the business to mine and select them, based not only on characteristics of the test data, but also on the execution details and the outcome of the scenario.

The goal of this article is to present a flexible solution to the problem of creation and selection of test scenarios, based on the different components of the ILOG JRules BRMS platform. It proposes, in particular, to leverage:

- The Decision Warehouse to capture a detailed trace of the scenario executions.
- The expressiveness and the business-friendly nature of the Business Action Language (BAL) as an efficient tool for mining the desired test scenarios.

Test scenario selection process

The information needed to perform test scenario selection is composed of the input and output data, as well as the detailed record of the steps that led to the output data using the given input. For example, one may want to gather the test scenarios where the primary applicant of a loan has a credit score that is less than 650 and for which the rule "Retired Applicant Discount" has been applied.

The information needed to design such criteria is typically captured by the execution trace recorded by the JRules Decision Warehouse component. The first step to prepare the mining of test scenarios from a production system database is therefore to run all candidate scenarios extracted from this production database through the business rule service and capture the execution traces in the Decision Warehouse, as shown in Figure 1.

Once the execution trace objects are available, they can be used as the input of a rule-based decision which rules are implementing the desired selection criteria.

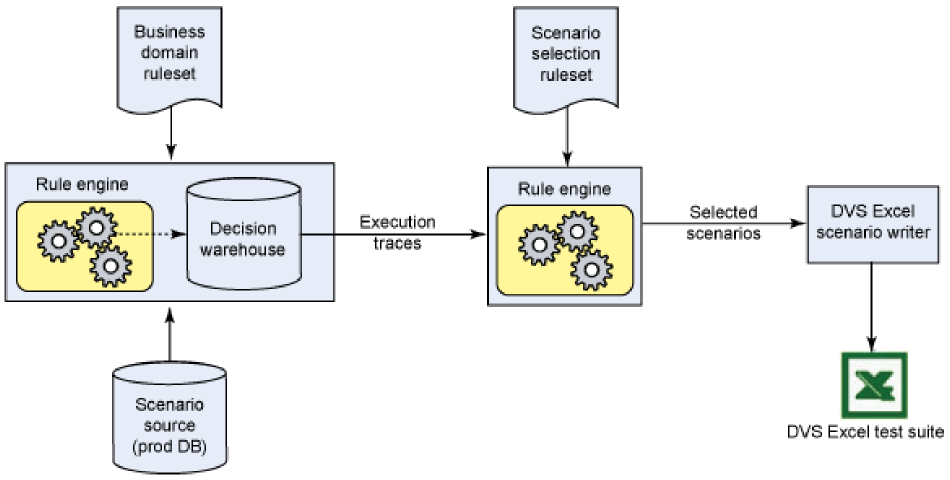
As a result of the selection process, the business will expect test suites that are captured in the business friendly format supported by the Excel® spreadsheets for the Decision Validation Services (DVS) component. The last step is thus to transform the raw collection of scenarios into a well-organized and manageable DVS Excel test suite, using the DVS serialization API.

The three steps are illustrated on Figure 1:

1. Capture of the execution trace by the Decision Warehouse.
2. Selection of relevant scenarios by applying rules to the execution traces.
3. Serialization of scenarios and output to a DVS Excel test suite.

The following sections describe in detail how to perform these three steps.

Figure 1. Test scenario selection process



Note that we are assuming in the rest of this article that the eXecution Object Model (XOM) for the business domain ruleset is based on XML Schemas, which facilitates the serialization and deserialization of information captured by the Decision Warehouse traces.

Capturing the execution traces

The Decision Warehouse provides a flexible way to store execution traces through a custom DAO. In our case, we will rely on the default DAO, which will enable us to store and retrieve the execution trace as an instance of the JRules `IlrDWTrace` API class. Enabling the capture of traces by the Decision Warehouse is controlled through ruleset properties, which can be either defined when the ruleset is deployed or dynamically managed through the Rule Execution Server console:

- The `monitoring.enabled` property shall be set to `true`, so that traces are captured.
- The `ruleset.bom.enabled` property shall be set to `false`, so that the input and output parameters of the candidate scenarios are recorded as XML code, and not the BOM representation.

Selecting the scenarios

Our goal here is to prepare the environment for a rule project in which rules can be written against potential test data but also the properties of the execution of business rules against this test data.

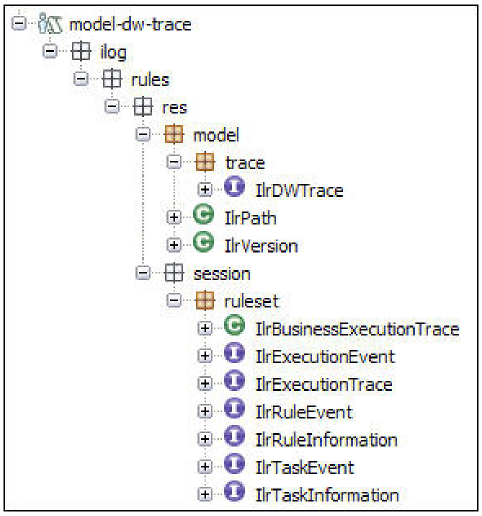
To support our examples, we will use a simple Lending business rules application, composed of a rule project defining a BOM entry (`loan-bom`) with three classes: `Borrower`, `LoanRequest` and `LoanResponse`, and a separate rule project (`loan-rules`) which implements the Eligibility and Pricing rules, using the previous BOM. The `loan-rules` project has a `LoanRequest` input parameter and a `LoanResponse` output parameter.

BOM for trace objects

In order to be able to write rules on the different information provided by the execution trace, we need a BOM entry to model the trace object structure and verbalize its properties.

We thus need to define a rule project (we will call it `scenario-selection-bom`) that defines the BOM entry illustrated in Figure 2.

Figure 2. DW trace BOM entry



```
model-dw-trace
  ilog
    rules
      res
        model
          trace
            IlrDWTrace
          IlrPath
          IlrVersion
        session
          ruleset
            IlrBusinessExecutionTrace
            IlrExecutionEvent
            IlrExecutionTrace
            IlrRuleEvent
            IlrRuleInformation
            IlrTaskEvent
            IlrTaskInformation
```

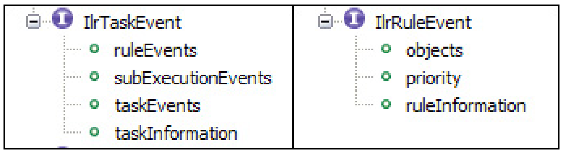
This entry can be created by pointing the Java Execution Object Model property of the project to the jrules-res-execution.jar file, which is part of the ILOG JRules platform distribution. The methods of the `IlrDWTrace` and its parent interface `IlrExecutionTrace` are particularly relevant to writing discriminating conditions as they allow access, among other properties, to:

- The input and output parameters
- The execution duration
- The list of executed rules and tasks (execution events)
- The rules not fired, and the tasks not executed

The list of execution events captured by the trace contains:

- `IlrTaskEvent` elements, a recursive structure composed of the tree of events (rules events and task events) that have been fired.
- `IlrRuleEvent` elements, a record of a rule instance that has been fired.

Figure 3. Execution events captured by the trace



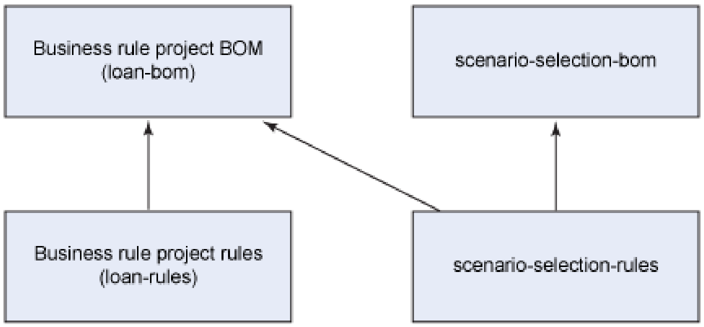
```
IlrTaskEvent
  ruleEvents
  subExecutionEvents
  taskEvents
  taskInformation
```

```
IlrRuleEvent
  objects
  priority
  ruleInformation
```

Selection rule project

Armed with the generic BOM for trace objects, as well as the specific BOM used by the business project, we can now define a rule project (we will call it `scenario-selection-rules`), which receives the execution trace to examine as an input parameter and returns as an output a Boolean string telling whether the scenario represented by the trace should be selected or not. The references between the different rule projects are shown in Figure 4..

Figure 4. Rule project relationships


Business rule project BOM (loan-bom)
scenario-selection-bom
Business rule project rules (loan-rules)
scenario-selection-rules

Ruleset variables

The execution trace object coming from the Decision Warehouse will contain a serialized version of the input and output parameters in either the BOM form or as XML strings for projects using an XML XOM.

We need to bind this data to objects inside the engine so that we can manipulate them. For an XML XOM, this binding can be performed by using an XML data driver that translates an XML Schema to an execution object model (XOM).

We can thus declare a set of variables in the scenario-selection-rules project and initialize them using an implementation of an `IlrXmlDataDriver` interface (for example, `IlrXmlDefaultDataDriver`). Once an XML data driver instance is available, we can create the internal JRules objects from their XML serialized form using an IRL expression of the form, as shown here:

```
ObjectClass input =
  (ObjectClass)xmlDataDriver.readObject(new StringReader(inputString));
```

In our Lending example, we would define two ruleset variables, `loanRequest` and `loanResponse`, which could then be initialized using a Technical Rule on the following model:

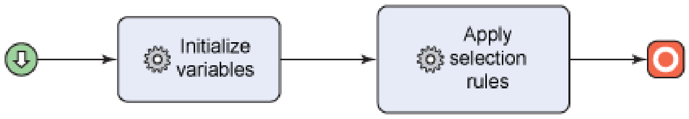
```
when {
  IlrContext();
}
then {
  String loanRequestString =
    (String)trace.inputParameters.get("loanRequest");
 loanRequest =
  (LoanRequest)driver.readObject(new StringReader(loanRequestString));

  String loanResponseString =
    (String)trace.outputParameters.get("loanResponse");
  loanResponse =
    (LoanResponse)driver.readObject(new StringReader(loanResponseString));
}
```

Rule Flow

The rule flow for the scenario selection project is straightforward: the first task is dedicated to the initialization of the ruleset variables (see previous section), and the next task is the collection of scenario selection rules (see next section), as shown in Figure 5.

Figure 5. Rule flow



Rules

With the ruleset parameters and variables available, we can now start writing scenario selection rules. For example, the following rule selects all the scenarios for which all the rule tasks in the rule flow have been executed.

```
if
  the total tasks not executed of trace equals 0
then
  mark scenario derived from trace as selected ;
```

The following rule is more complex and involves conditions on both the scenario data and the execution information: it will select the scenario if the primary borrower is retired but the `Retired Discount` rule has been fired for the request.

```
if
  the employment status of the primary borrower of 'loan request' is "retired"andthere is one rule info in
 the rules not fired of trace
      where the business name of this rule info ends with "Retired Discount" ,
then
  mark scenario derived from trace as selected ;
```

In the last example below, the selection rule is looking for scenarios where more than three adjustment rules were applied to the request, and the request was still accepted.

```
if
  therearemorethan 3 ruleeventsin the execution events of tracewhere the name of the rule information of each
 rule event contains "Adjustment" , and
  the status of 'loan response' is "accepted"
then
  mark scenario derived from trace as selected ;
```

Obviously, the rules can be more complex, involving conditions on the sequence of rules fired, the sequence of tasks executed, the objects used in the rules, and so on. It may also become necessary to instrument the raw BOM entry obtained from the JRules trace classes to extend the navigation capabilities through the execution event tree.

Still, since the selection rules are using a combination of the business vocabulary and an extra vocabulary derived from the execution trace properties, they are easily manageable by the business, which can write its specific test scenario mining queries without assistance from the IT group.

Creating the DVS Excel spreadsheet

Once the scenarios have been selected and collected, they need to be exported to a DVS spreadsheet. This last step is facilitated by the existence of the DVS API, which allows you to write, through the `IlrExcel2003ScenarioSuiteWriter` class, a map of input parameter names and input parameter objects as a test scenario to a given DVS Excel test suite template.

A detailed example of how to use this API can be found in the documentation of the Populate Excel scenario file template sample.

Conclusion

In this short column, I have shown an application of the business rules approach that operates not only on business application objects but also on objects coming from the rule platform API itself. The goal was to avoid as much as possible the design and implementation of a complex and ad hoc technical tool when the requirements were typically suited to the use of business rules.

The test scenario selection problem is indeed a typical validation problem, where the rules need to be written by business people and are bound to change often. These characteristics are a perfect match for the business rule approach.

The same type of approach can be used for other type of problems, where some analysis and processing is needed on objects that belong to the JRules BRMS platform itself. For example:

- For the authoring phase, the writing of meta-rules (that is, rules acting on rule objects) to explore and validate some characteristics of the business rules written by the business.
- For the execution phase, the writing of rules on reified rule engine artifacts, such as the execution trace we have presented in this article.

Resources

- WebSphere ILOG JRules BRMS: Get product information.
- IBM WebSphere ILOG JRules BRMS Version 7.1 Information Center
- Populate Excel scenario file template sample in the WebSphere ILOG BRMS V7.1 Information Center.
- developerWorks BPM zone: Get the latest technical resources on IBM BPM solutions, including downloads, demos, articles, tutorials, events, webcasts, and more.
- IBM BPM Journal: Get the latest articles and columns on BPM solutions in this quarterly journal, also available in both Kindle and PDF versions.

About the author

Pierre Berlandier

Pierre Berlandier has worked for ILOG services for the past 15 years, during which time he has helped ILOG's clients leverage the successive incarnations of the rules programming paradigm from expert systems to real-time intelligent agents, and now business rules applications.