



Enterprise Architect

User Guide Series

# DMN Modeling and Simulation

Author: Sparx Systems

Date: 26/07/2018

Version: 1.0

# Table of Contents

DMN Modeling and Simulation .....	3
Introduction .....	5
What is Decision Model and Notation .....	6
Why Use Decision Model and Notation .....	7
A First Example .....	8
DecisionTable .....	11
Literal Expression .....	16
Boxed Context .....	20
Invocation .....	24
Item Definition .....	28
Input Data .....	32
Data Sets .....	34
BusinessKnowledgeModel & Test Harness .....	36
DMN Expression Auto Completion .....	39
DMN Expression Validation .....	43
DMN Decision Table Validation .....	45
DMN Simulation .....	49
Configure DMN Simulation .....	51
Simulate DMN Model .....	54
DMN Module Code Generation & Test .....	58
Integrate DMN Module Into UML Class Element .....	63

# DMN Modeling and Simulation

Decision Model and Notation (DMN) is a standard published and managed by the Object Management Group (OMG). It provides a standard approach for describing, modeling and implementing repeatable decisions within an organization or an initiative. It is also intended to facilitate the sharing and interchange of decision models between organizations.

The modeling notation is comprised of a visual grammar that allows decisions and business rules to be documented in a way that makes them readable by both business and technical audiences thus ensuring that decisions and rules are not misinterpreted. The resulting Decision Model also provides a definition of how to evaluate the logic of decisions defined in Decision Tables using the Friendly Enough Expression Language (FEEL).

The purpose of DMN is to provide the constructs that are needed to model decisions, so that organizational decision-making can be readily depicted in diagrams, accurately defined by business analysts.

In this topic, we will introduce DMN Expression, DMN Simulation Artifact and how you can use Enterprise Architect to automate the decision-making process.

## DMN Expressions

- DMN Expression: Decision Table
- DMN Expression: Boxed Context
- DMN Expression: Literal Expression
- DMN Expression: Function
- DMN Expression: Invocation

## DMN Data

- DMN ItemDefinition
- DMN InputData
- DataSet for InputData

## DMN Simulation

- Configure a DMN Simulation Artifact
- Run/Step/Debug the DMN Simulation

## Code Generation & Connect to BPMN

- Generate DMN Server in language: Java/JavaScript/C++/C#
- Run/Debug of testing for Java version of DMN Server
- Connect DMN Server with EA BPSim Execution Engine

## Common Errors & Solutions

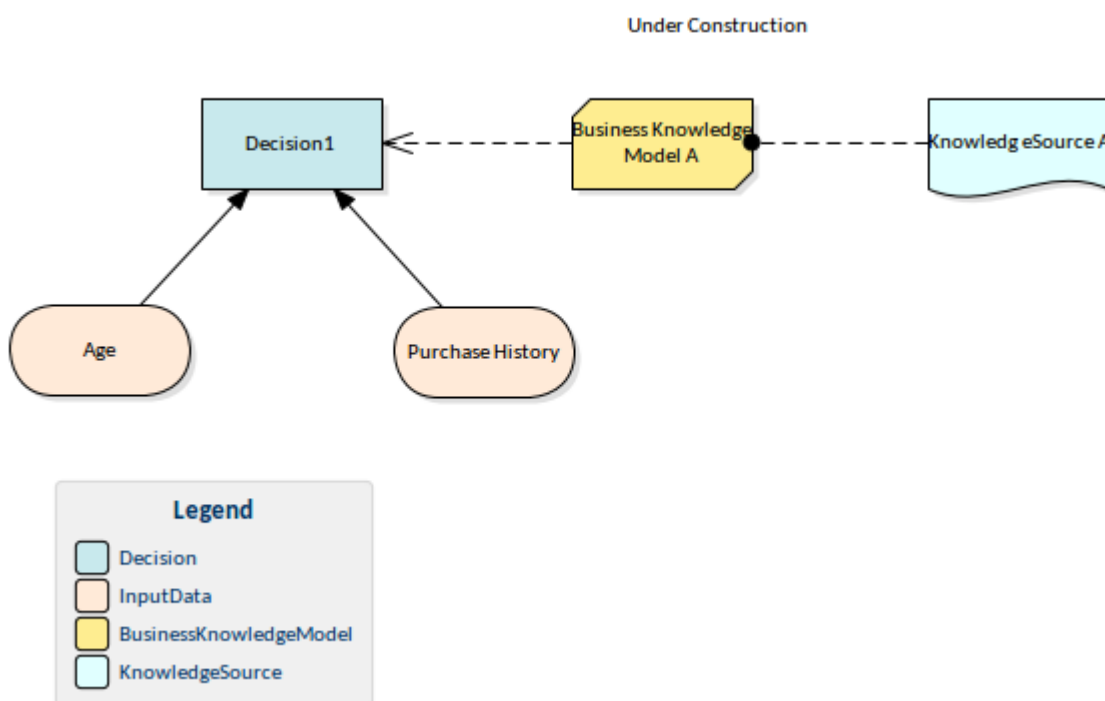
- Run validation will help you pick up most of the modeling issues. Run this before simulation and code generation.

- Variable Types: As DMN models use the FEEL language (Simulate with JavaScript), typing of variables is not compulsory. However, when generating code to languages that are compiled, the typing of a variable is required. There are context menu options and tag values for setting a type to the variables.
- Since a DMN expression allows for spaces, in order to clarify the composite Input Data there must be a space before and after the "." in the expression.  
For example, 'Applicant data . Age' is valid, whereas 'Applicant data.Age' is not valid.  
Note that when using the Auto Completion feature this issue will not occur.

# Introduction

Now more than ever, in a world turned on its head by new and innovative business and technical ideas and disruptive ways of working, does an organization need to have a clear understanding of its choices and the decisions it makes. Unmanaged complexity is the enemy and agility the friend that heralds business success and enables an organization to respond quickly to changes in its business circumstances. Without a clear and communicable model it is almost impossible for an organization to embrace the changes that confront them daily in the digital world.

The description and implementation of decisions which has been inexplicably and somewhat invisibly part of almost all disciplines has now been synthesized into a rigorous and formal discipline of its own with a new way of modeling and describing decisions, Inputs, Outcomes, Rules, Business Knowledge, Authorities and more. Indeed once you have seen the Decision Model and Notation in action and been introduced to the countless benefits it brings you will not be able to go back to old and arcane ways of working.



Enterprise Architect has become the tool of choice for many Business and Technical leaders because of its flexible, extensible, standards-based and pragmatic approach to modeling complex systems. As a collaboration platform it is a tool for all disciplines, and allows Decision Models to be created, integrated, managed, documented, simulated and generated to programming code. The models can be visualized and integrated with a range of other models including, Business Process Diagrams, Use Case Models, User Stories, Test Cases, Database Models, Implementation Artifacts and programming code to list just the main models.

# What is Decision Model and Notation

Decision Model and Notation (DMN) is a standard published and managed by the Object Management Group (OMG). It provides a standard approach for describing, modeling and implementing repeatable decisions within an organization or an initiative. It is also intended to facilitate the sharing and interchange of decision models between organizations.

The modeling notation is comprised of a visual grammar that allows decisions and business rules to be documented in a way that makes them readable by both business and technical audiences thus ensuring that decisions and rules are not misinterpreted. The resulting Decision Model also provides a definition of how to evaluate the logic of decisions defined in Decision Tables using the Friendly Enough Expression Language (FEEL).

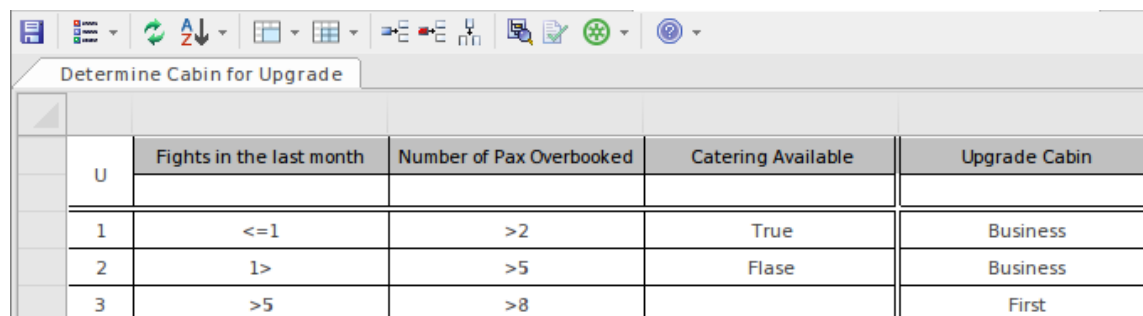
The DMN defines two level of the language which neatly align with organizational roles as follows:

## Requirements Level

Aligns with the business people who devise, analyze and define the decision rules. This provides a mechanism for the people who own the business to model them without the need for technical knowledge of how they are implemented.

## Logic Level

Aligns with implementation teams who augment the definitions to include decision logic in the form of decision tables that can be maintained by the business staff or expression logic managed by technical staff.



The screenshot shows a software interface for modeling a decision. At the top is a toolbar with various icons for file operations, undo/redo, and modeling. Below the toolbar is a tab labeled "Determine Cabin for Upgrade". The main area contains a decision table with the following structure:

	Fights in the last month	Number of Pax Overbooked	Catering Available	Upgrade Cabin
U				
1	$\leq 1$	$> 2$	True	Business
2	$1 >$	$> 5$	Flase	Business
3	$> 5$	$> 8$		First

# Why Use Decision Model and Notation

There are many reasons for using DMN at an Enterprise or Initiative level that will result in both business and technical value. There are many benefits described in the next topic but the most compelling of these are the following

## **Reduced Complexity**

As discussed earlier currently business rules and decisions are commonly located in a myriad of places and in a wide variety of formats. A common place to find them is in Business Process Diagrams that are overly complex with cascading sets of Gateways that describe the outcomes. The diagrams are commonly unwieldy and do little to reveal the inputs, business knowledge, authorities or the logic of how the decisions are made.

## **Automation and Execution**

As the Business Analysts define the decisions, inputs, business knowledge and the technologists elaborate the logic in the form of Expressions and Decision Tables potentially creating Decision Services the models can be used to generate programming language code that can be executed to automate the decisions and make them available to runtime systems.

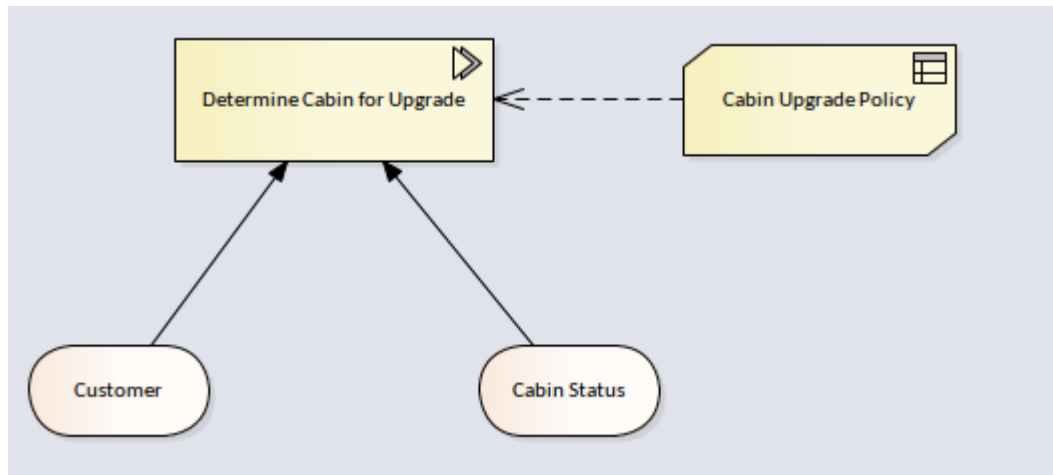
## **Agility and Responsiveness to Change**

As a result of separating the decision models from the Business Processes models and creating an implementation and automation pipeline the business and technology teams can respond at record speed to business change. These models then become a platform that facilitates agility and the seizing of opportunities.

## A First Example

Imagine you are an Airline reservation officer working at the check-in counter for a busy domestic airline. Getting the aircraft off on-time is critical as delays can result in fees applied by the airport controllers, needing to fly at a lower altitude increasing the cost of fuel, and other penalties.

A message from the supervisor appears on your screen saying that the economy cabin is overbooked you will need to upgrade some passengers to Business or First Class - but which passengers should be chosen and which cabin should they be upgraded to? A decision needs to be made but what factors should be considered? This can be recorded in a Decision Model using a Decision Requirements Diagram.



This is helpful but the busy check-in officer would still need to weigh up all the factors and make an unbiased decision. Should a disgruntled passenger be given priority over a Gold level frequent flyer, or should the fact that a particular passenger is connecting to an international flight take precedence. These 'rules' can all be recorded in a decision table making it clear which passengers should get an upgrade and to which cabin: Business or First class. This will make it much easier to make the decision and the rules can be formulated, agreed upon and checked for consistency back at head office. In this example we have kept it simple and used two factors: firstly the number of flights the passenger has made in the last month and secondly how overbooked the cabin is.

Cabin Upgrade Policy		Input Parameter Values for Simulation	
	( Flights in the last month, Number of Pax Overbooked )		
U	Flights in the last month	Number of Pax Overbooked	Upgrade Cabin
			Business Class, First Class
1	<=1	<=2	Business Class
2	<=1	(2..8]	Business Class
3	<=1	>8	First Class
4	(1..5]	<=2	Business Class
5	(1..5]	(2..8]	Business Class
6	(1..5]	>8	First Class
7	>5	<=2	Business Class
8	>5	(2..8]	Business Class
9	>5	>8	First Class

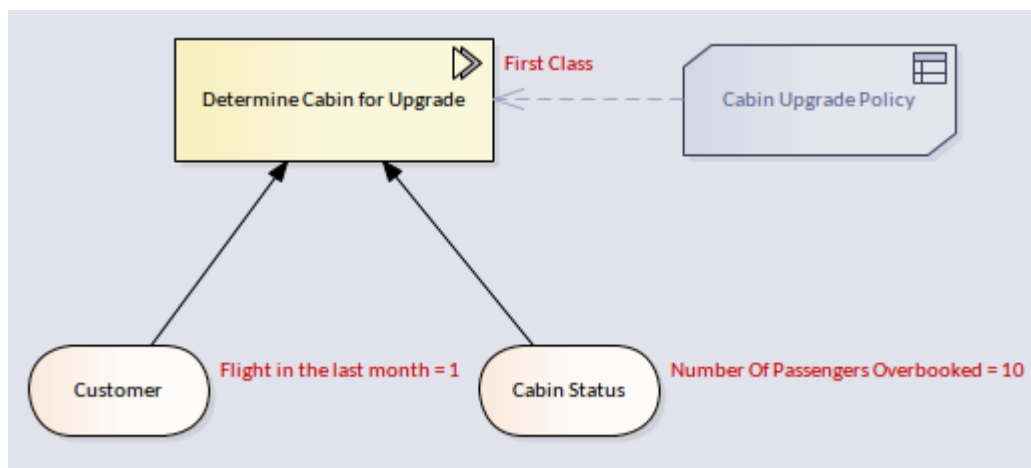
The table is divided into columns and rows. There are two types of columns: inputs that are required to make the decision



and outputs that are the result of applying the rules.

This is very helpful but still requires the busy check-in officer to be able to source all the required information required to find the right row in the decision table. Even if all this information were available a wrong decision could still result from human error in selecting the wrong row in the table

Fortunately the Decision Models can be automated and generated to programming code that can be executed by an application. So our busy check-in officer would not need to do anything or make any decisions, as she was checking in the passengers if a particular passenger was entitled to an upgrade it would be visible on the computer screen. In the diagram below the model has been simulated so that the business and technical staff can agree that the model has been defined correctly. Any number of user defined data sets can be used to test the model before generating out the programming code that will run in the check-in system and display the result to the end user.



When developing the models a business or technical user can step through the simulation and the system will show the user which row in the decision table was fired to determine the output. This is very helpful in models that are made up of multiple decisions.

Cabin Upgrade Policy			
Input Parameter Values for Simulation			
( Flights in the last month = 1, Number of Pax Overbooked = 10 )			
U	Flights in the last month	Number of Pax Overbooked	Upgrade Cabin
	1	10	First Class
1	<=1	<=2	Business Class
2	<=1	(2..8]	Business Class
3	<=1	>8	First Class
4	(1..5]	<=2	Business Class
5	(1..5]	(2..8]	Business Class
6	(1..5]	>8	First Class
7	>5	<=2	Business Class
8	>5	(2..8]	Business Class
9	>5	>8	First Class

It is common for the rules that govern the upgrade decision to change. For example the marketing department may decide they want to reward passengers that travel on long-haul flights. The Decision Requirements Diagram can be altered to include the new input, the Decision Table modified, and the programming code regenerated. Once the changes have been

pushed through to the airport systems magically the right passengers will be upgraded. The check-in officer could still view the decision tables during training and briefing session so as to understand the rules.












# DecisionTable


A decision table is a tabular representation of a set of related input and output expressions, organized into rules indicating which output entry applies to a specific set of input entries.

## Access

Ribbon	Simulate > DMN > Manage > DMN Expression, then select / create a Decision or BusinessKnowledgeModel
Other	Double-click on an DMN Decision or BusinessKnowledgeModel

## Toolbar Options

Option	Description
	Click on this button to save the configuration to the current Decision or BusinessKnowledgeModel.
	Click on this button to switch between rule-as-row and rule-as-column for the decision table.
	Click on "Sort By Input" will sort the rules by input columns; click on "Sort By Output" will sort the rules by output columns. The Columns can be dragged and dropped to organize the sorting order.
	Click on this button to merge adjacent input entry cells from different rules with the same content.
	Click on this button to unmerge entry cells that have been merged.
	Click on this button to edit parameters for the Business Knowledge Model.
	Click on this button to append an input column for the decision table.
	Click on this button to append an output column for the decision table.
	Click on this button to append a rule for the decision table
	Click on this button to show/hide the allowed values fields for the input and output columns. The allowed value defined for a input / output will be used for validation and auto completion editing.
	Click on this button to validate the Decision Table. EA will perform a series of validations to help the modeler to pickup errors in the decision table.

	<p>This button is Enabled when the decision table is defined for a BusinessKnowledgeModel.</p> <p>Activate tab "Input Parameter Values for Simulation", fill the values and click this button. The test result will be presented on the decision table, with the runtime values of inputs and outputs displayed and valid rule(s) highlighted.</p> <p>The user can use this functionality to unit test a BusinessKnowledgeModel without knowing the context and later on invoked by a Decision or other BusinessKnowledgeModel.</p> <p>Menu options are available for the this toolbar button. For more information, click the See also link.</p>
---	---

## Decision Table Hit Policy

The hit policy specifies what the result of the decision table is in cases of overlapping rules. The single character in a particular decision table cell indicates the table type and unambiguously understand the decision logic.

Single Hit Policies:

- **Unique:** no overlap is possible and all rules are disjoint. Only a single rule can be matched. This is the default.
- **Any:** there may be overlap, but all of the matching rules show equal output entries for each output, so any match can be used.
- **Priority:** multiple rules can match, with different output entries. This policy returns the matching rule with the highest output priority.
- **First:** multiple (overlapping) rules can match, with different output entries. The first hit by rule order is returned.

Multiple Hit Policies:

- **Output order:** returns all hits in decreasing output priority order.
- **Rule order:** returns all hits in rule order.
- **Collect:** returns all hits in arbitrary order. An operator ('+', '<', '>', '#') can be added to apply a simple function to the outputs.

Collect operators are:

- **+** (sum): the result of the decision table is the sum of all the distinct outputs.
- **<** (min): the result of the decision table is the smallest value of all the outputs.
- **>** (max): the result of the decision table is the largest value of all the outputs.
- **#** (count): the result of the decision table is the number of distinct outputs.

## Example of Unique hit policy

Unique hit policy is the most popular type of decision table and all rules are disjoint.

Post-bureau risk category table		Input Parameter Values for Simulation		
	( Existing Customer = true, Application Risk Score = 90, Credit Score = 590 )			
U	Existing Customer	Application Risk Score	Credit Score	Post Bureau Risk Category
	true	90	590	MEDIUM
1	false	<120	<590	HIGH
2	false	<120	[590..610]	MEDIUM
3	false	<120	>610	LOW
4	false	[120..130]	<600	HIGH
5	false	[120..130]	[600..625]	MEDIUM
6	false	[120..130]	>625	LOW
7	false	>130	-	VERY LOW
8	true	<=100	<580	HIGH
9	true	<=100	[580..600]	MEDIUM
10	true	<=100	>600	LOW
11	true	>100	<590	HIGH
12	true	>100	[590..615]	MEDIUM
13	true	>100	>615	LOW

## Example of Priority hit policy

In table with Priority hit policy, multiple rules can match, with different output entries. This policy returns the matching rule with the highest output priority.

Eligibility rules		Input Parameter Values for Simulation			
	( Pre-Bureau Affordability, Pre-Bureau Risk Category, Age )				
	P	Pre-Bureau Risk Categ...	Pre-Bureau Affordability	Age	Eligibility
					INELIGIBLE, ELIGIBLE
	1	DECLINE	-	-	INELIGIBLE
	2	-	false	-	INELIGIBLE
	3	-	-	<18	INELIGIBLE
	4	-	-	-	ELIGIBLE

Note: the output order is defined as INELIGIBLE, ELIGIBLE; which means INELIGIBLE has higher priority than ELIGIBLE.

One possible simulation result may look like this:

Eligibility rules		Input Parameter Values for Simulation		
	( Pre-Bureau Affordability = false, Pre-Bureau Risk Category = HIGH, Age = 25 )			
P	Pre-Bureau Risk Category	Pre-Bureau Affordability	Age	Eligibility
	HIGH	false	25	INELIGIBLE
1	DECLINE	-	-	INELIGIBLE
2	-	false	-	INELIGIBLE
3	-	-	<18	INELIGIBLE
4	-	-	-	ELIGIBLE

The matching rules are highlighted, but output with INELIGIBLE is the pick based on output order.

## Example of Collection-Sum hit policy

For decision table with Collect-Sum (C+) hit policy, the result of the decision table is the sum of all the distinct outputs.

Application risk score model		Input Parameter Values for Simulation		
( Age = 40, Marital Status = M, Employment Status = EMPLOYED )				
C+	Age	Marital Status	Employment Status	Partial score
	40	M	EMPLOYED	133
1	[18..21]	-	-	32
2	[22..25]	-	-	35
3	[26..35]	-	-	40
4	[36..49]	-	-	43
5	>=50	-	-	48
6	-	S	-	25
7	-	M	-	45
8	-	-	UNEMPLOYED	15
9	-	-	STUDENT	18
10	-	-	EMPLOYED	45
11	-	-	SELF-EMPLOYED	36

In this example, the output Partial Score is calculated as  $43 + 45 + 45 = 133$

## Set type for Input/Output Clause

The user need to set the type for Input Clause / Output Clause for the following reasons:

- Display: The string will be shown in Italic font (otherwise, each string literal need to be inside quotes)
- Validation: The range, gap, overlap validations are supported only on numbers.
- Code Generations for typed languages like C++/C#/Java

Some user may ask, since the BKM parameter already has types defined, why do we need to define type for the Input Clauses?

The answer is: the Input Clause could be an expression.

For example: a parameter "Credit Score" is a number type, but the Input Clause could be "Credit Score > 500" is a boolean type.

Context menu on the Input/Output Clause, choose the menu item to set type for it.

Credit contingency factor table		Input Parameter Values for Simulation	
		( Risk Category )	
▶	U	Risk Category	lit Contingency Factor
		DECLINE, HIGH, MEDIUM, LOW	
	1	HIGH, DECLINE	0.6
	2	MEDIUM	0.7
	3	LOW, VERY LOW	0.8

✓ type: string

type: boolean

type: number

type: date

type: time

type: duration

Delete Input Column









## Literal Expression

A Literal Expression is the simplest form of DMN expression. It is commonly used as one-liner or if-else block. When the expression is getting complicated, a Boxed Context is a better choice or encapsulate some logic as a function in DMN Library in order to improve readability. You will see an example at the end of this page.

### Access

Ribbon	Simulate > Decision Analysis > DMN > DMN Expression, then select / create a Decision or BusinessKnowledgeModel
Other	Double-click on an DMN Decision or BusinessKnowledgeModel

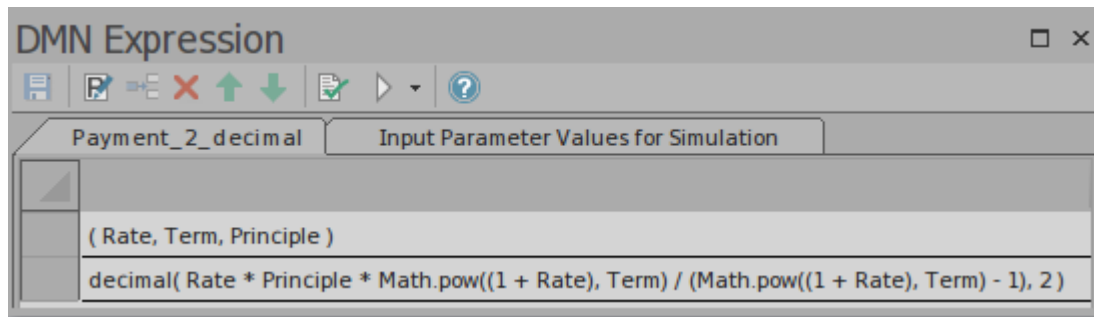
### Toolbar Options

Options	Description
	Click on this button to save the configuration to the current Decision or BusinessKnowledgeModel.
	Click on this button to edit parameters for the Business Knowledge Model.
	This option is disabled for Literal Expression.
	This option is disabled for Literal Expression.
	This option is disabled for Literal Expression.
	This option is disabled for Literal Expression.
	Click on this button to validate the Literal Expression. EA will perform a series of validations to help the modeler to pickup errors in the Expression.
	This button is Enabled when the literal expression is defined for a BusinessKnowledgeModel.

### Example - Payment of 2 Decimals

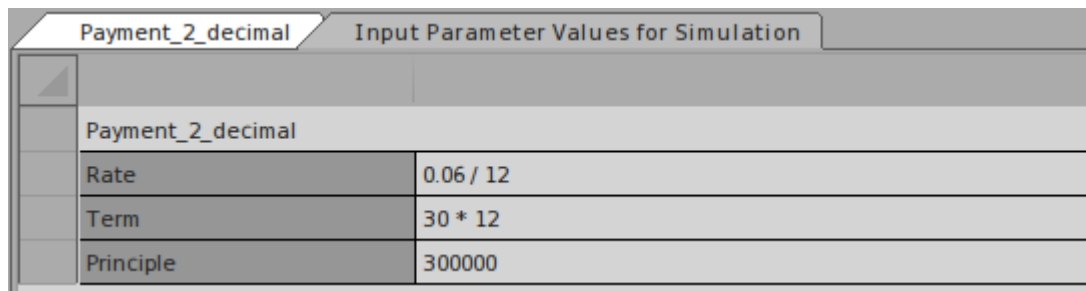
This Business Knowledge Model (BKM) *Payment\_2\_decimal* is implemented as Literal Expression.



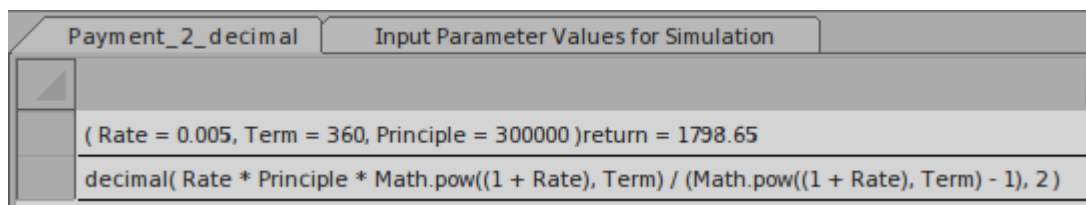


- The BKM defines 3 parameters: Rate, Term, Principle

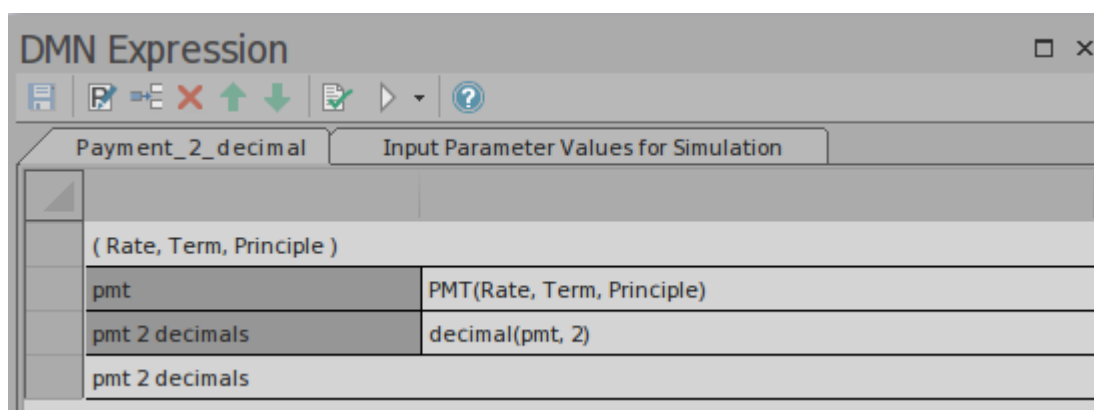
Give values for the Test Harness and Evaluate the model:



- The runtime parameter value will be displayed. E.g. Rate = 00.005
- The BKM's result will be evaluated by the literal expression and the value is displayed on the declaration line. E.g. return = 1798.65



Although the implementation is one liner, it is quite complicated. We can re-factor this model with Built-In function and Boxed Context to improve readability:



- The Boxed Context defines 2 variable-expression pairs entries, these variables serves as "local variables", which can be used in later expressions
- Return value: The expression can use the value of "local variables"
- Any expressions in a Boxed Context can use built-in functions which is defined in the customizable Template - *DMN Library*. For example, functions PMT(...) and decimal(...) are used in this example.

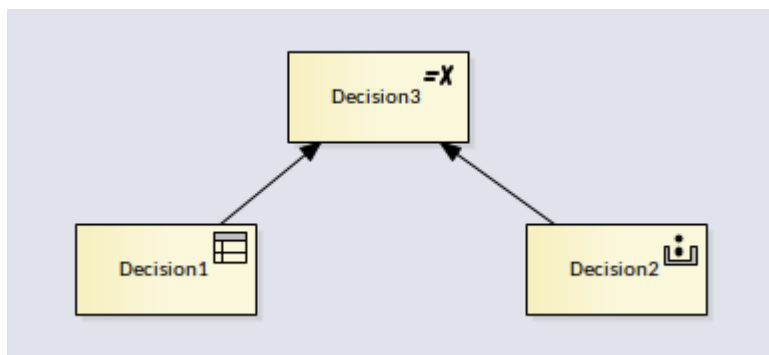
The simulation result is exactly the same as Literal Expression:

Payment_2_decimal		Input Parameter Values for Simulation	
	( Rate = 0.005, Term = 360, Principle = 300000 )return = 1798.65		
	pmt = 1798.6515754582708	PMT(Rate, Term, Principle)	
	pmt 2 decimals = 1798.65	decimal(pmt, 2)	
	pmt 2 decimals		

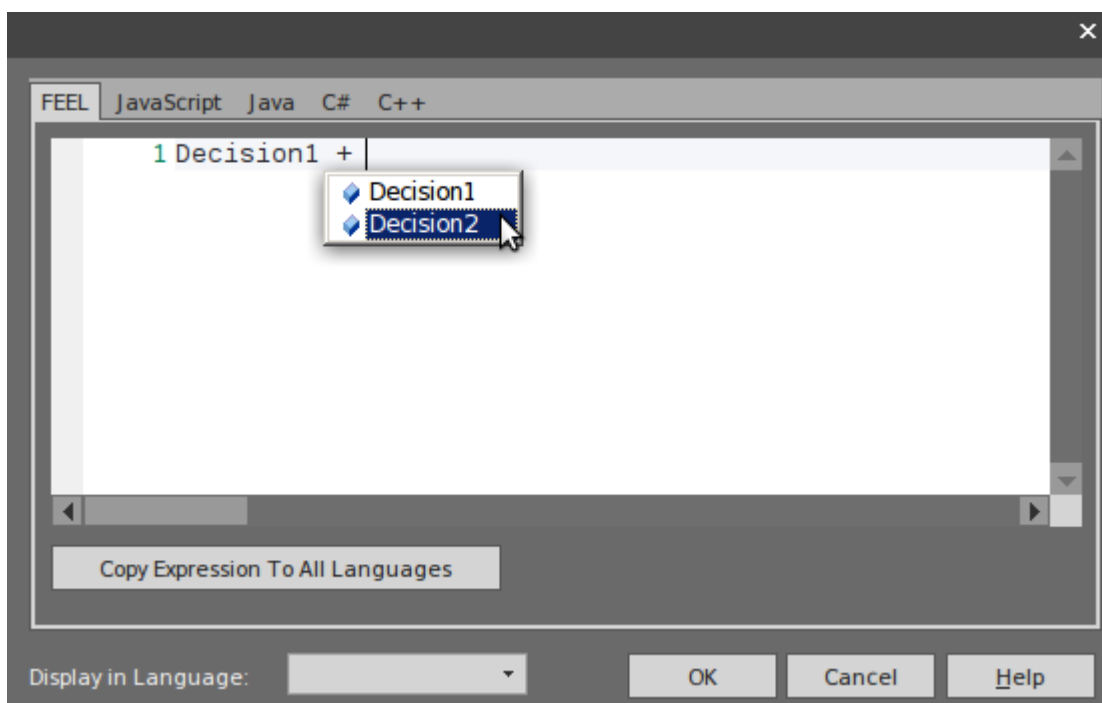
## Expression Editor and IntelliSense Support

The parameter names may contain space according to FEEL language specification. This feature makes the expression easy to read. In order to help the user to edit the expressions with fewer typing and make less mistake, EA provided IntelliSense support for editing expressions:

Given a decision hierarchy like this, the expression in Decision3 should be able to use the required decisions(variables ).



Context menu on the Expression | Click the menu item "Edit Expressions..." to bring up the Expression Dialog.



Press "Ctrl + Space" to show the IntelliSense menu:

- For BKM, all the parameters will be included.
- For Decision, all the required Decisions will be included.

The DMN Model may be generated as source code for language JavaScript/Java/C#/C++, since some languages may have different syntax for some expressions, EA provided language override pages for each language. If no override code is specified for a language, the expression defined for FEEL language will be used.

In the generated code, the space inside a variable name will be replaced by an underscore.









## Boxed Context

A boxed context is a collection of context entry. Each context entry is consisted of a variable and an expression. The Context also has an result value.

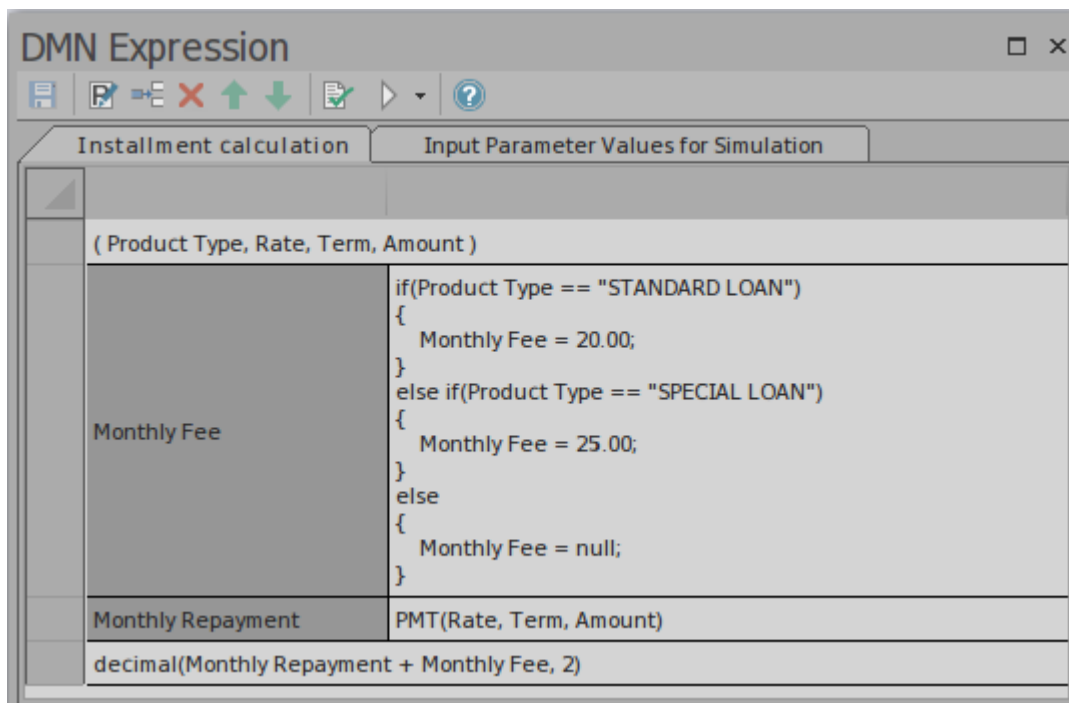
### Access

Ribbon	Simulate > Decision Analysis > DMN > DMN Expression, then select / create a Decision or BusinessKnowledgeModel
Other	Double-click on an DMN Decision or BusinessKnowledgeModel

### Toolbar Options

Options	Description
	Click on this button to save the configuration to the current Decision or BusinessKnowledgeModel.
	Click on this button to edit parameters for the Business Knowledge Model.
	Click on this button to add Context Entry to the boxed context. Each Context Entry is consisted of a variable and an expression.
	This button is Enabled when a ContextEntry's variable is selected. Click on this button to delete the selected Context Entry.
	This button is Enabled when a ContextEntry's variable is selected. Click on this button to move the selected Context Entry up.
	This button is Enabled when a ContextEntry's variable is selected. Click on this button to move the selected Context Entry down.
	Click on this button to validate the Boxed Context. EA will perform a series of validations to help the modeler to pickup errors in the Expression.
	This button is Enabled when the boxed context is defined for a BusinessKnowledgeModel.

### Example - Installment calculation



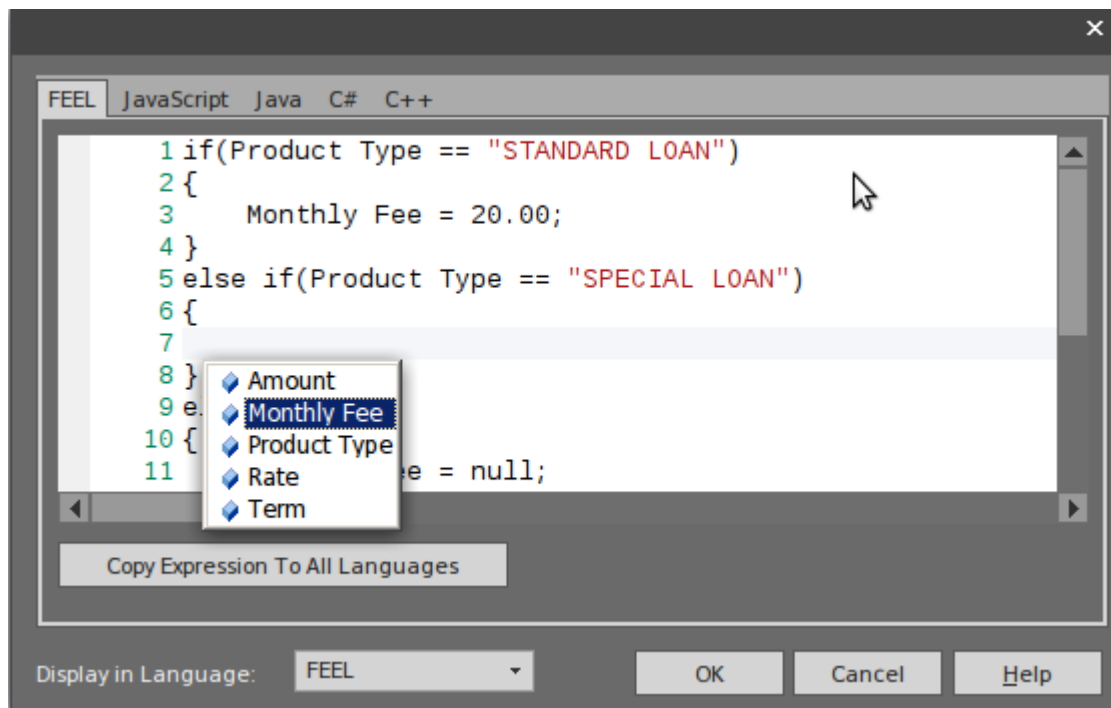
This Business Knowledge Model (BKM) *Installment calculation* is implemented as Boxed Context.

- The BKM defines 4 parameters: Product Type, Rate, Term, Amount
- The Boxed Context defines 2 variable-expression pairs entries, these variables serves as "local variables", which can be used in later expressions
- Return value: The expression can use the value of "local variables"
- Any expressions in a Boxed Context can use built-in functions which is defined in the customizable Template - *DMN Library*. For example, functions PMT(...) and decimal(...) are used in this example.

## Expression Editor and IntelliSense Support

The parameter and Context Entry's Variable name may contain space according to FEEL language specification. This feature makes the expression easy to read. In order to help the user to edit the expressions with fewer typing and make less mistake, EA provided IntelliSense support for editing expressions:

Context menu on the Expression | Click the menu item "Edit Expressions..." to bring up the Expression Dialog.



Press "Ctrl + Space" to show the IntelliSense menu:

- All the Context Entry Variables (the context entries later than the current one are excluded) will be included.
- For BKM, all the parameters will be included.
- For Decision, all the required Decisions will be included.

The DMN Model may be generated as source code for language JavaScript/Java/C#/C++, since some languages may have different syntax for some expressions, EA provided language override pages for each language. If no override code is specified for a language, the expression defined for FEEL language will be used.

In the generated code, the space inside a variable name will be replaced by an underscore.

## Test Harness for Business Knowledge Model

Activate tab "Input Parameter Values for Simulation", fill the values.

Installment calculation	
Input Parameter Values for Simulation	
Installment calculation	
Product Type	"STANDARD LOAN"
Rate	0.045/12
Term	12 * 30
Amount	300000

Click the Test Harness button on the toolbar, the test result will be presented on the boxed context.

Installment calculation		Input Parameter Values for Simulation	
		( Product Type = STANDARD LOAN, Rate = 0.00375, Term = 360, Amount = 300000 )return = 1540.06	
Monthly Fee = 20		<pre> if(Product Type == "STANDARD LOAN") {     Monthly Fee = 20.00; } else if(Product Type == "SPECIAL LOAN") {     Monthly Fee = 25.00; } else {     Monthly Fee = null; } </pre>	
Monthly Repayment = 1520.05...		PMT(Rate, Term, Amount)	
		decimal(Monthly Repayment + Monthly Fee, 2)	

- The runtime parameter value will be displayed. E.g. Rate = 0.00375
- The Context Entry variable's runtime value will be displayed. E.g. Monthly Repayment = 1520.05
- The BKM's result will be evaluated by the last entry and the values is displayed on the declaration line. E.g. return = 1540.06

The user can use this functionality to unit test a BusinessKnowledgeModel without knowing the context and later on invoked by a Decision or other BusinessKnowledgeModel.

Menu options are available for the this toolbar button. For more information, click the See also link.

## Invocation

An invocation is a container for the parameter bindings that provide the context for the evaluation of the body of a business knowledge model. There are two common use cases for Invocation:









- Bind Input Data to Business Knowledge Model
- Bind parameters or context entry variables to Business Knowledge Model

You will see two examples at the end of this page.

### Access

Ribbon	Simulate > Decision Analysis > DMN > DMN Expression, then select / create a Decision or BusinessKnowledgeModel with Invocation type
Other	Double-click on an DMN Decision or BusinessKnowledgeModel

### Toolbar Options

Options	Description
	Click on this button to save the configuration to the current Decision or BusinessKnowledgeModel.
	Click on this button to edit parameters for the Business Knowledge Model.
	Click on this button to synchronize with the invoked Business Knowledge Model. For example, if the Business Knowledge Model changed name, parameters, outputs and types, click on this button to synchronize these changes.
	Click on this button to set / change a Business Knowledge Model as invocation.
	Click on this button to open the invoked Business Knowledge Model in the DMN Expression window.
	When a Business Knowledge Model is implemented as a decision table, it may define multiple output clause, the invocation on this Business Knowledge Model may need to specify which output is requested. Click on this button to list all the available outputs in a context menu, the current configured output is checked.
	Click on this button to validate the Invocation. EA will perform a series of validations to help the modeler to pickup errors in the Expression.
	This button is Enabled when the Invocation is defined for a BusinessKnowledgeModel. Activate tab "Input Parameter Values for Simulation", fill the values and click this button. The test result will be presented on the decision table, with the runtime



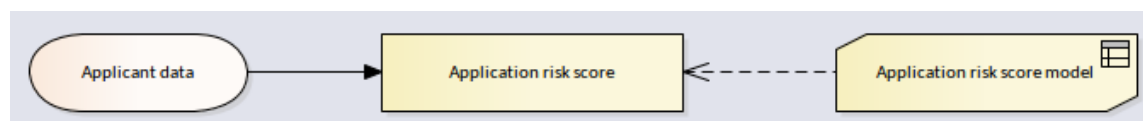
values of inputs and outputs displayed and valid rule(s) highlighted.

The user can use this functionality to unit test a BusinessKnowledgeModel without knowing the context and later on invoked by a Decision or other BusinessKnowledgeModel.

Menu options are available for the this toolbar button. For more information, click the See also link.

## Example 1 - Bind Input Data to Business Knowledge Model

The full example can be created with Model pattern (Access: Ribbon | Simulate | DMN | Apply Perspective | DMN Decision | Decision With BKM | Create Pattern)



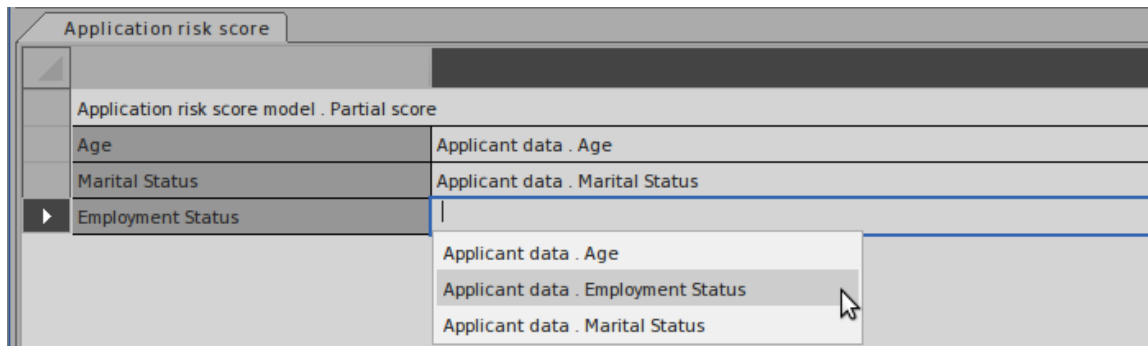
In this example, Input Data *Applicant Data* is typed to *Applicant Data Definition* which is a composition of 3 components.

Applicant data : Applicant data Definition		
Applicant data Definition	Marital Status : string	"M"
	Employment Status : string	"EMPLOYED"
	Age : number	40

The Business Knowledge Model *Application risk score model* is implemented as decision table with 3 inputs and 1 output.

Application risk score model				
Input Parameter Values for Simulation				
( Age, Marital Status, Employment Status )				
C+	Age	Marital Status	Employment Status	Partial score
	[18..120]	S,M	UNEMPLOYED,EMPLOYE...	
1	[18..21]	-	-	32
2	[22..25]	-	-	35
3	[26..35]	-	-	40
4	[36..49]	-	-	43
5	>=50	-	-	48
6	-	S	-	25
7		M	-	45
8		-	UNEMPLOYED	15
9			STUDENT	18
10			EMPLOYED	45
11			SELF-EMPLOYED	36

The decision *Application risk score* is implemented as Invocation to bind the Input Data's "leaf" components to the BKM's parameters.

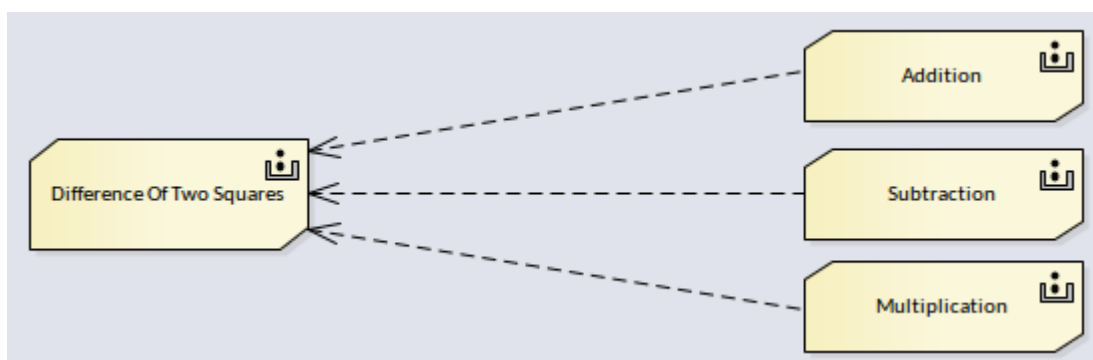


In order to make the binding easier, Auto-Completion is supported for the binding expression.

The full modeling and simulation instructions are available in the pattern's document.

## Example 2 - Bind Context Entry variables to Business Knowledge Model

The full example can be created with Model pattern (Access: Ribbon | Simulate | DMN | Apply Perspective | DMN Business Knowledge Model| Business Knowledge Model With Invocation | Create Pattern)



In this example, BKM *Difference Of Two Squares* is implemented as Boxed Context:

- The variable *sum of ab* is implemented as invocation by binding parameter *a* and *b* to BKM *Addition*.
- The variable *difference of ab* is implemented as invocation by binding parameter *a* and *b* to BKM *Subtraction*.
- The variable *difference of squares* is implemented as invocation by binding local variable *sum of ab* and *difference of ab* to BKM *Multiplication*.

Difference Of Two Squares		Input Parameter Values for Simulation	
( a, b )			
sum of ab	Addition		
	addend 1	a	
	addend 2	b	
difference of ab	Subtraction		
	minuend	a	
	subtrahend	b	
difference of squares	Multiplication		
	factor 1	sum of ab	
	factor 2	difference of ab	
difference of squares			

In order to make the binding easier, Auto-Completion is supported for the binding expression.

The full modeling and simulation instructions are available in the pattern's document.

## Item Definition






An important characteristic of data items in decision models is their structure.

The ItemDefinition element is used to model the structure and the range of values of the input and the outcome of decisions.

### Access

Ribbon	Simulate > DMN > Manage > DMN Expression, then select / create an ItemDefinition
Other	Double-click on an DMN ItemDefinition

### Toolbar Options

Option	Description
	Click on this button to save the configuration to the current ItemDefinition.
	Click on this button to add a child component to the selected ItemDefinition.
	Click on this button to add a sibling component to the selected ItemDefinition.
	Click on this button to delete the selected component.
	Click on this button to validate the ItemDefinition. EA will perform a series of validations to help the modeler to pickup errors in the ItemDefinition.

### Allowed Value Enumerations

Each "Leaf" component of the ItemDefinition may define a string of Allowed Value Enumerations.

For example:

**DMN Expression**

Applicant data (ItemDefinition)

Applicant data	Age : number	Type in Allowed Value Enumerations...	
	Existing Customer : boole...	true,false	
	Marital Status : string	"S","M"	
	Employment Status : string	"UNEMPLOYED","EMPLOYED","SELF-EMPLOYED","STUDENT"	
	Monthly	Expenses : number	Type in Allowed Value Enumer...
		Income : number	Type in Allowed Value Enumer...
		Repayments : number	Type in Allowed Value Enumer...

**DMN Expression**

Strategy (ItemDefinition)

Strategy : string	"BUREAU","DECLINE","THROUGH"
-------------------	------------------------------

The Allowed Value Enumerations plays an important role for Auto Completion. The idea is that the user type these values once, and choose from a list later on.

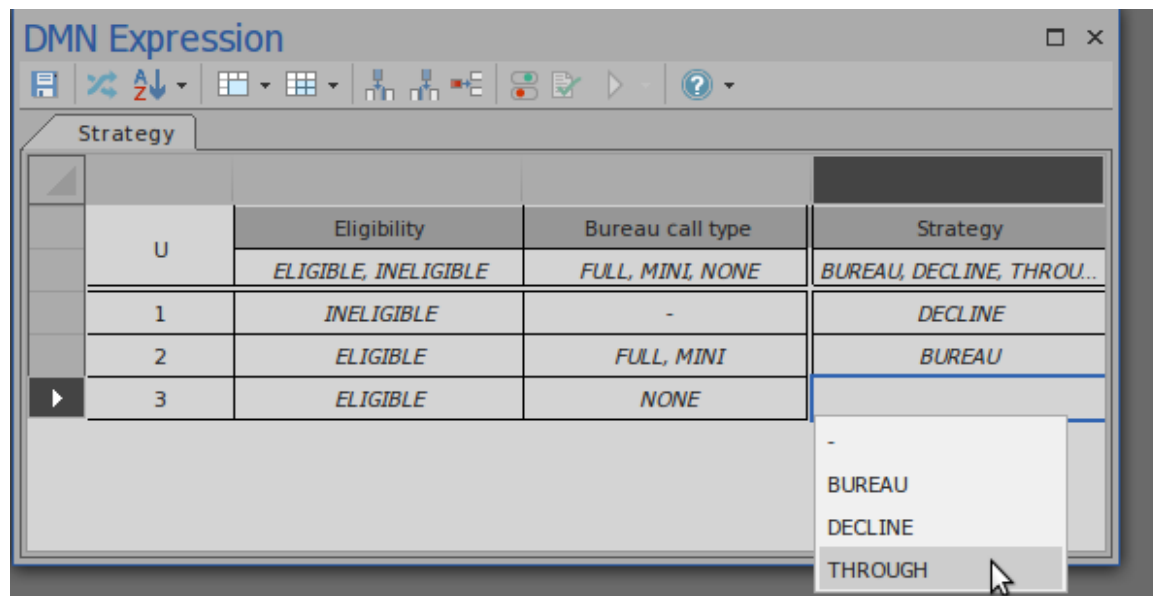
Take the above "Strategy" ItemDefinition as an example, we can fast fill the allowed values field for a decision table by selection:

**DMN Expression**

Strategy

	Eligibility	Bureau call type	Strategy
U	ELIGIBLE, INELIGIBLE	FULL, MINI, NONE	
1	INELIGIBLE	-	"BUREAU","DECLINE","THROUGH"
2	ELIGIBLE	FULL, MINI	BUREAU
3	ELIGIBLE	NONE	THROUGH

Then fast fill the decision table rules by selection:



## Type of Components

An ItemDefinition element SHALL be defined using only one of the alternative ways:

- Set to a built-in type
- composition of ItemDefinition elements.

In other words, if an ItemDefinition is "leaf"(no child components), it must set to a built-in type; if an ItemDefinition has child components, it can not set a built-in type.

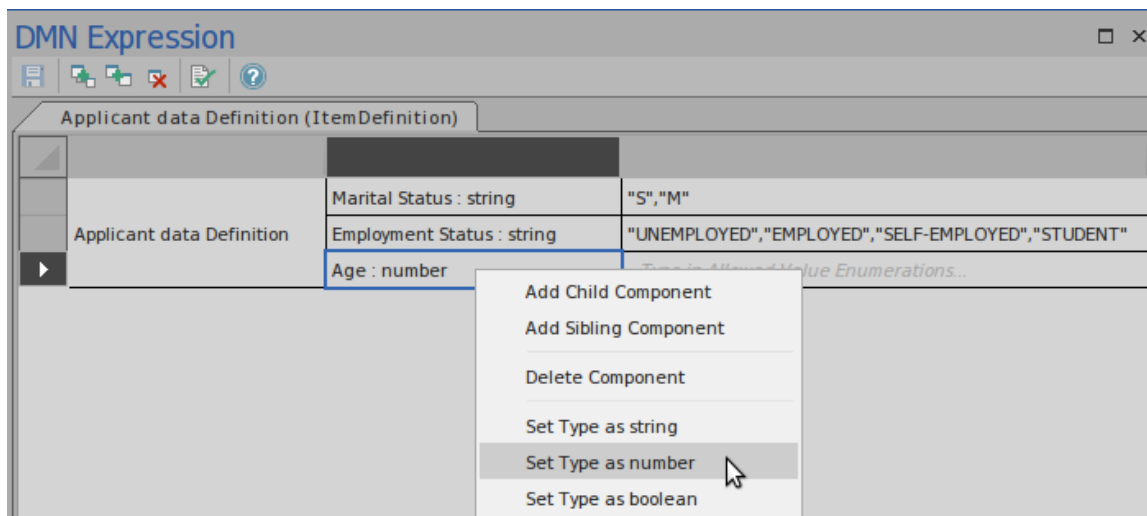
The FEEL language has following built-in types:

- **number,**
- **string,**
- **boolean,**
- days and time duration,
- years and months duration,
- time,
- date and time

Note: number, string and boolean are supported by EA for simulation.

In order to set type for "leaf" ItemDefinition, the user can use one of the following 3 methods:

- Use the context menu (Recommended)
- type in the cell after the name by appending " : string", " : boolean" or " : number"
- Input "string", "boolean" or "number" in the tag "type" for the ItemDefinition








# Input Data

An InputData types to an ItemDefinition and carries value used for Decisions.

## Access

Ribbon	Simulate > DMN > Manage > DMN Expression, then select / create an InputData
Other	Double-click on an DMN InputData

## Toolbar Options

Option	Description
	Click on this button to save the configuration to the current InputData.
	Click on this button to select a ItemDefinition as the type of this InputData.
	Click on this button to open the ItemDefinition element that types this InputData.
	Click on this button to open the dialog for editing data sets for this input data. Each InputData can define multiple data sets. With this feature, the DMN Simulation can quickly test some decision's result by choosing from different data sets.
	Click on this button to validate the InputData. EA will perform a series of validations to help the modeler to pickup errors in the InputData.

## Auto Completion

If the typing ItemDefinition has defined Allowed Value, then the value for the InputData can be simply chosen from the enumerations.



DMN Expression

Applicant data : Applicant data

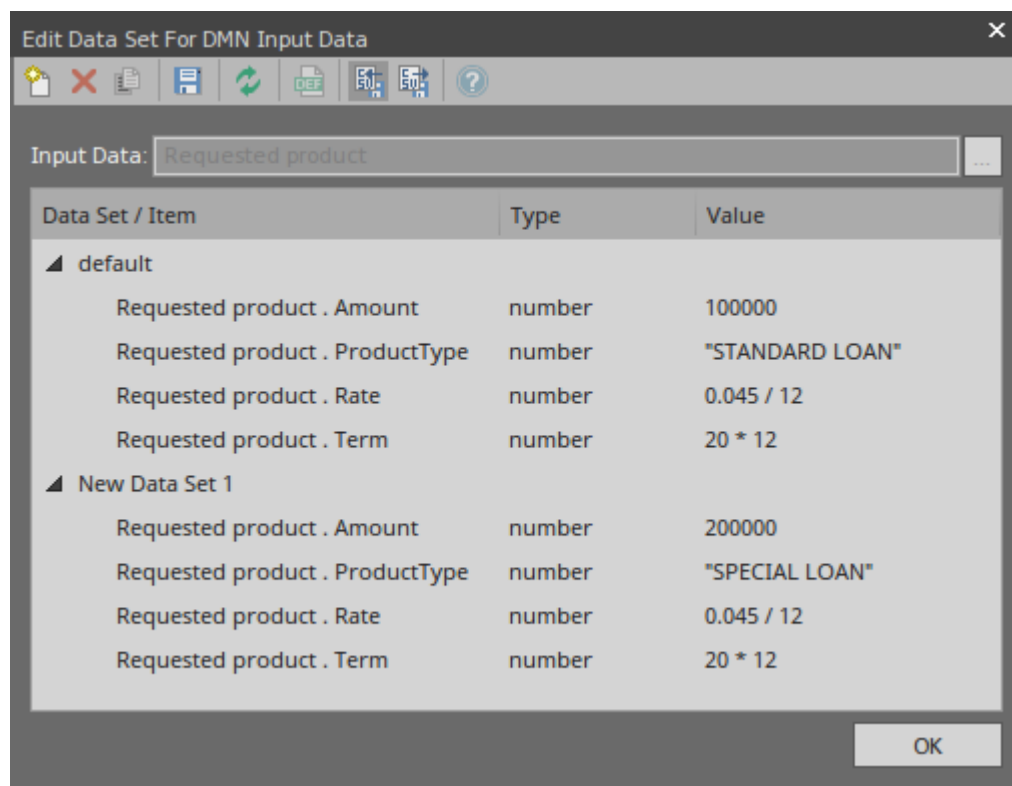
	Age : number	40
	Existing Customer : boolean	false
	Marital Status : string	"M"
▶ Applicant data	Employment Status : string	
	Monthly	

- "EMPLOYED"
- "SELF-EMPLOYED"
- "STUDENT"
- "UNEMPLOYED"

## Data Sets

Each InputData can define multiple data sets. With this feature, the DMN Simulation can quickly test some decision's result by choosing from different data sets.



- The values in the "default" data set will show in the DMN Expression window when the InputData is selected.
- The user can add/duplicate/delete dataset and set an existing dataset as default.
- The user can export the datasets to a CSV file and import from a CSV file.









## Access

Ribbon	Simulate > DMN > Manage > DMN Expression > InputData, Edit Data Set button on the toolbar
Other	Double-click on an DMN InputData, Edit Data Set button on the toolbar

## Toolbar Options

Option	Description
	Click on this button to create a new dataset.
	Click on this button to delete the selected dataset.

	Click this button to duplicate the selected dataset.
	Click on this button to save the datasets to the InputData.
	Click on this button to reload the datasets for the InputData.
	Click on this button to set the selected dataset as default.
	Click on this button to import datasets from a CSV file.
	Click on this button to export the datasets to a CSV file.

## BusinessKnowledgeModel & Test Harness

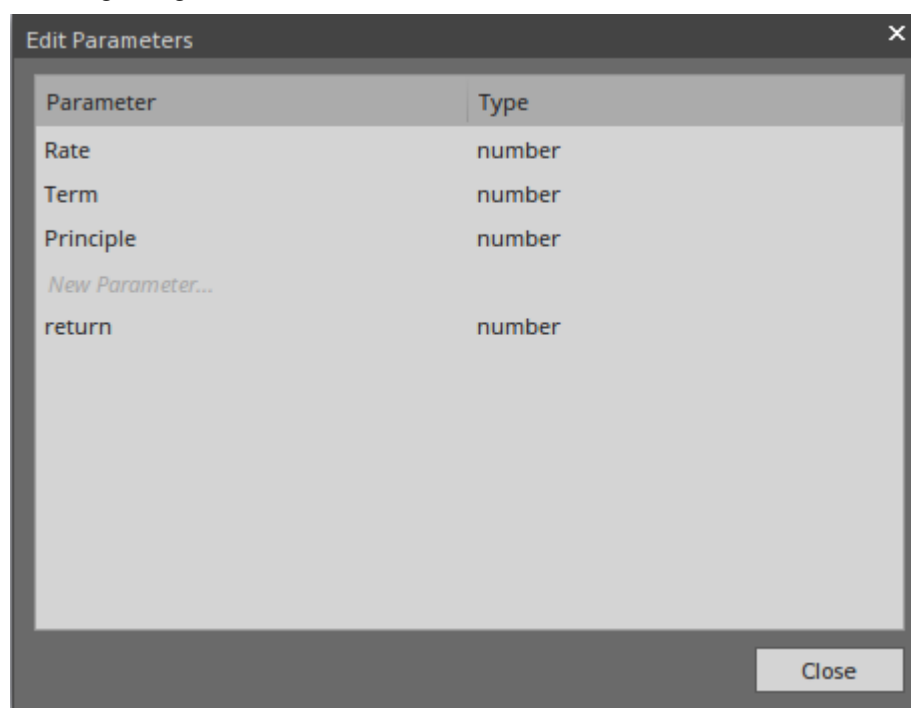
A Business Knowledge Model encapsulate logic without knowing the usage context, it is implemented as a function definition with parameters and a DMN expression as body(Decision table, Boxed Context, Literal Expressions etc).

Because the definition of a Business Knowledge Model is self contained, it is possible to a "Unit Testing" by providing arguments for the parameters. For this reason, a Business Knowledge Model will have a "Input Parameter Values for Simulation" tab in the Expression Window.

Here, we will show some examples available from Model Wizard.

### Parameters of a Business Knowledge Model

Context menu on the first row of the Business Knowledge Model and select "Edit Parameters..." to bring up the following dialog.



We can add/delete/modify/change type on parameters.

### Test Harness

#### Decision Table

When a decision table is created for a Business Knowledge Model, we can test this BKM by binding some values:

Eligibility rules				
Input Parameter Values for Simulation				
( Pre-Bureau Affordability, Pre-Bureau Risk Category, Age )				
P	Pre-Bureau Risk Category	Pre-Bureau Affordability	Age	Eligibility
	VERY LOW, LOW, MEDIU...			INELIGIBLE, ELIGIBLE
1	DECLINE	-	-	INELIGIBLE
2	-	false	-	INELIGIBLE
3	-	-	<18	INELIGIBLE
4	-	-	-	ELIGIBLE

In this example we provided test values like this:

Eligibility rules	
Input Parameter Values for Simulation	
Eligibility rules . Eligibility	
Pre-Bureau Affordability	true
Pre-Bureau Risk Category	"VERY LOW"
Age	16

Click the Run button on the toolbar, we will get the following result:

Eligibility rules				
Input Parameter Values for Simulation				
( Pre-Bureau Affordability = true, Pre-Bureau Risk Category = VERY LOW, Age = 16 )				
P	Pre-Bureau Risk C...	Pre-Bureau Afford...	Age	Eligibility
	VERY LOW	true	16	INELIGIBLE
1	DECLINE	-	-	INELIGIBLE
2	-	false	-	INELIGIBLE
3	-	-	<18	INELIGIBLE
4	-	-	-	ELIGIBLE

- The runtime parameter value will take the place of "Allowed Values" in simulation mode.
- Valid rule(s) are highlighted
- Since this decision table's hit policy is P (Priority) the final result is determined by the order of "output values". Since "INELIGIBLE, ELIGIBLE" are the output values and "INELIGIBLE" comes ahead of "ELIGIBLE", rule #3 will be final result and this applicant is "INELIGIBLE".

### Boxed Context / Literal Expressions

Similar to Decision table, the Business Knowledge Model implemented as Boxed Expression can be tested as well.

Take the following "Payment" BKM as example,

Payment	
Input Parameter Values for Simulation	
( Rate, Term, Principle )	
decimal( Rate * Principle * Math.pow((1 + Rate), Term) / (Math.pow((1 + Rate), Term) - 1), 2 )	

This BKM will calculate the monthly repayment based interest rate, number of terms, principle amount.

In this example we provided test values like this:

Payment		Input Parameter Values for Simulation	
	Payment		
	Rate		0.04 / 12
	Term		30 * 12
	Principle		300000

Click the Run button on the toolbar, we will get the following result:

Payment		Input Parameter Values for Simulation	
	( Rate = 0.003333333333333335, Term = 360, Principle = 300000 )return = 1432.25		
	decimal( Rate * Principle * Math.pow((1 + Rate), Term) / (Math.pow((1 + Rate), Term) - 1), 2 )		

- The runtime parameter and return value will showing with an equal sign "=", following by the runtime value.

In this example, given annual Rate 4% for 30 years, a principle of 300000, the monthly repayment is 1432.25

Note: The DMN Library already has a PMT function defined. This example mainly demonstrate how Literal Expression works and how to test it with a set of arguments.

## DMN Expression Auto Completion

DMN defines many expressions such as FunctionDefinition, DecisionTable, Boxed Context, Invocation and Literal Expression. The parameters, arguments and logic of expressions is implemented largely by 'text'.

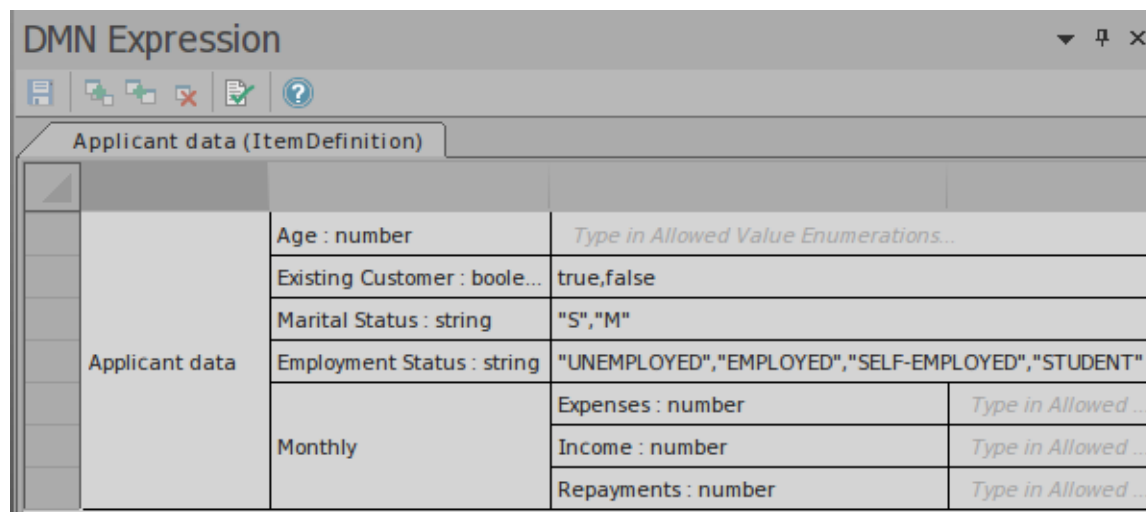
To make modeling easy and reliable, Enterprise Architect provides an Auto Completion facility

- Allowed Values of ItemDefinition
- Input/Output Entries of a Decision Table
- InformationRequirement

### Allowed Values of ItemDefinition

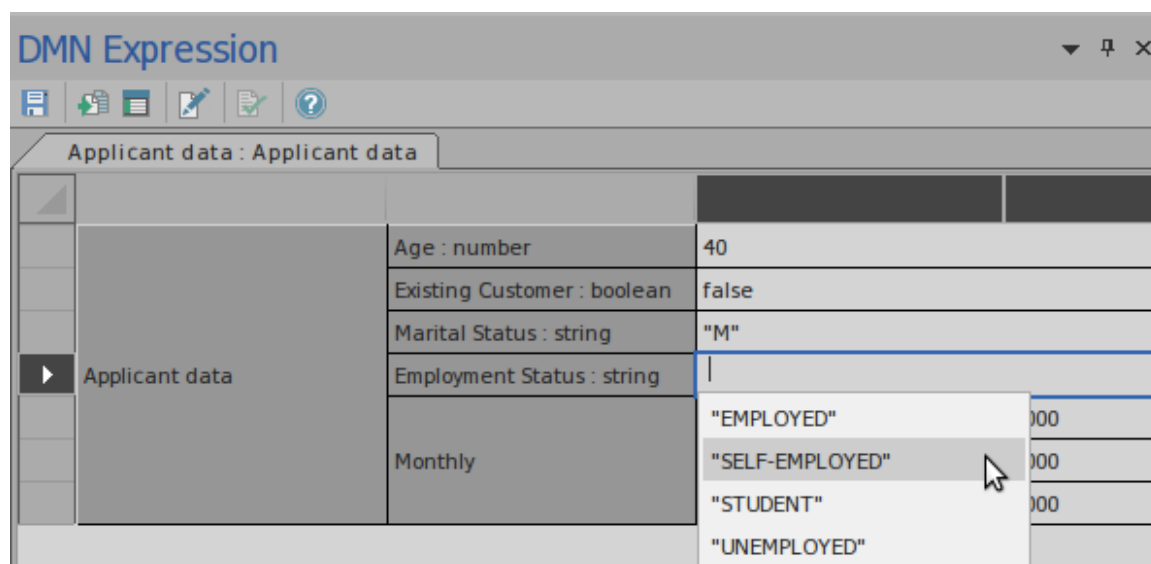
The idea is to define allowed value enumerations in ItemDefinition, then compose a list for selection whenever these values are requested.

In this example, ItemDefinition "Applicant data . Employment Status" defines an enumeration of allowed values.



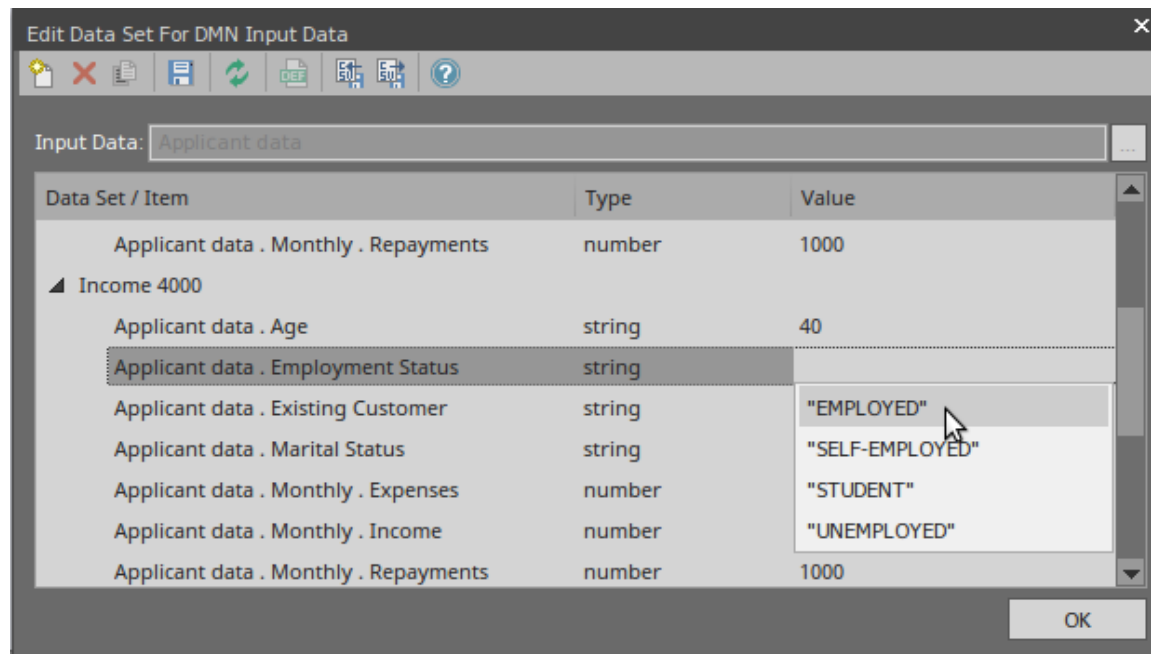
Applicant data (ItemDefinition)		
Applicant data	Age : number	Type in Allowed Value Enumerations...
	Existing Customer : boolean	true,false
	Marital Status : string	"S","M"
	Employment Status : string	"UNEMPLOYED","EMPLOYED","SELF-EMPLOYED","STUDENT"
	Monthly	Expenses : number Income : number Repayments : number

When editing values for the InputData typed to this ItemDefinition, press the Spacebar on the keyboard to display a list of values to select from.

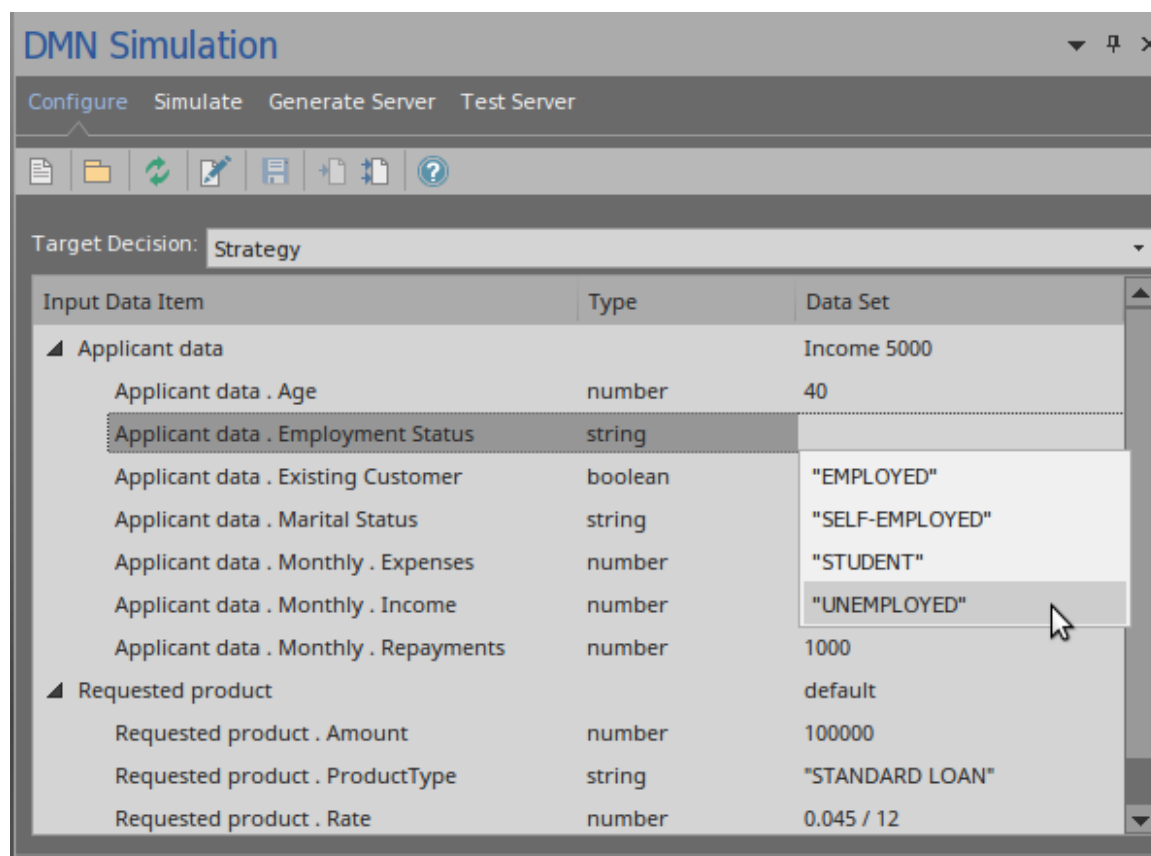


Applicant data : Applicant data		
Applicant data	Age : number	40
	Existing Customer : boolean	false
	Marital Status : string	"M"
	Employment Status : string	<div> "EMPLOYED"  "SELF-EMPLOYED"  "STUDENT"  "UNEMPLOYED" </div>
	Monthly	000 000 000

We may also define multiple data sets for the InputData, the Auto Completion Feature is available on this dialog.



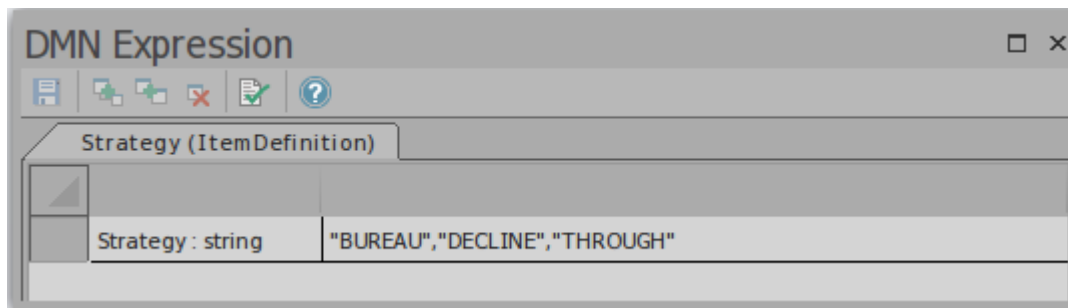
On the simulation window, the user may change the test value to simulate the model, the Auto Completion is available on this list as well.



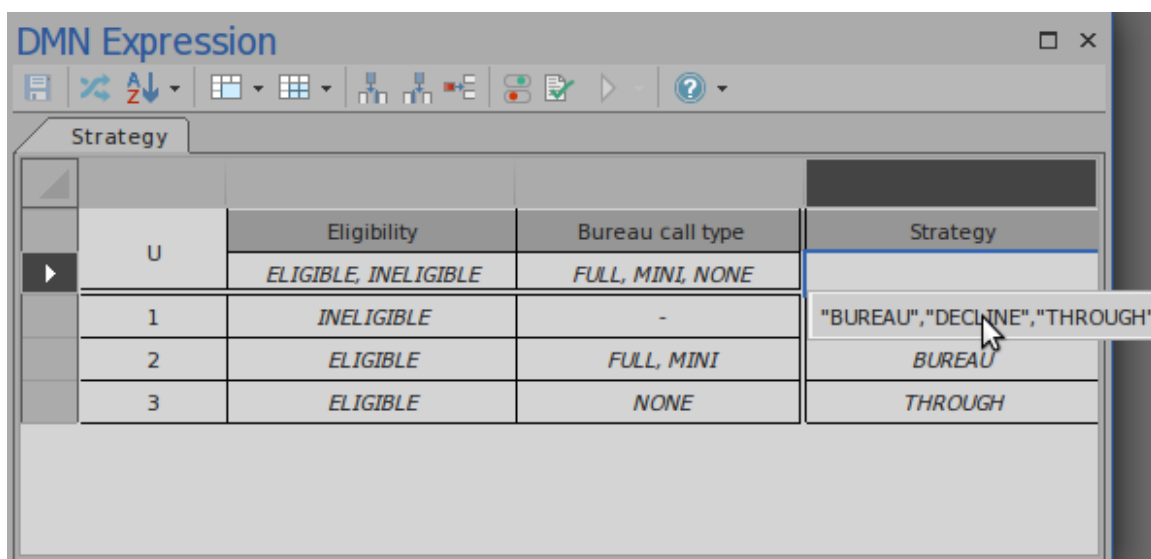
## Input/Output Entries of a Decision Table



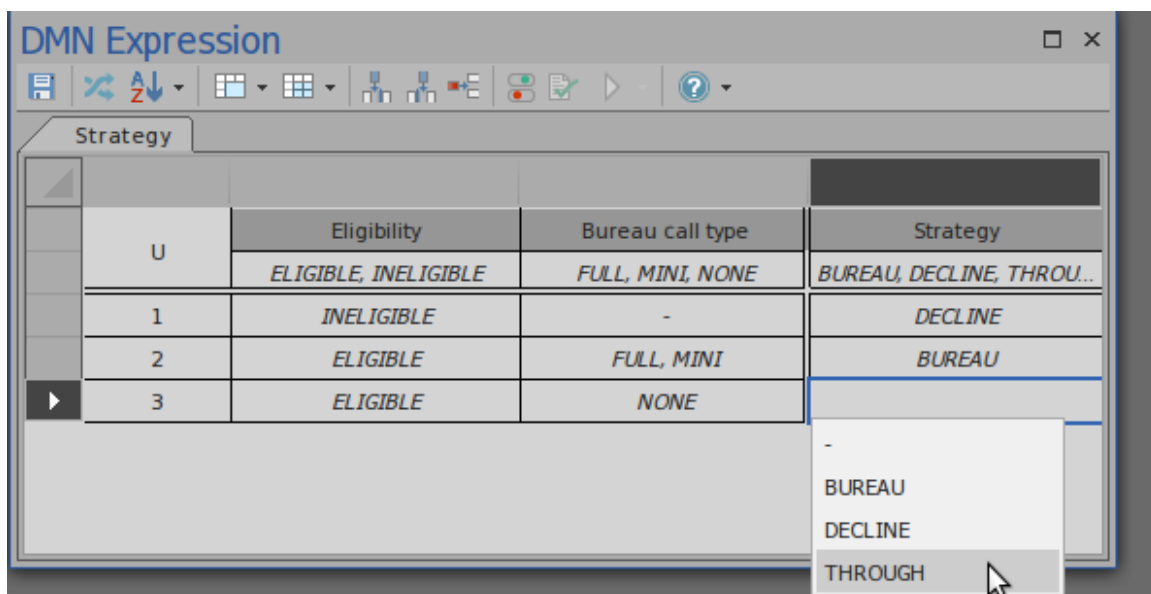
Take the above "Strategy" ItemDefinition as an example:



We can fast fill the allowed values field for a decision table by selection:



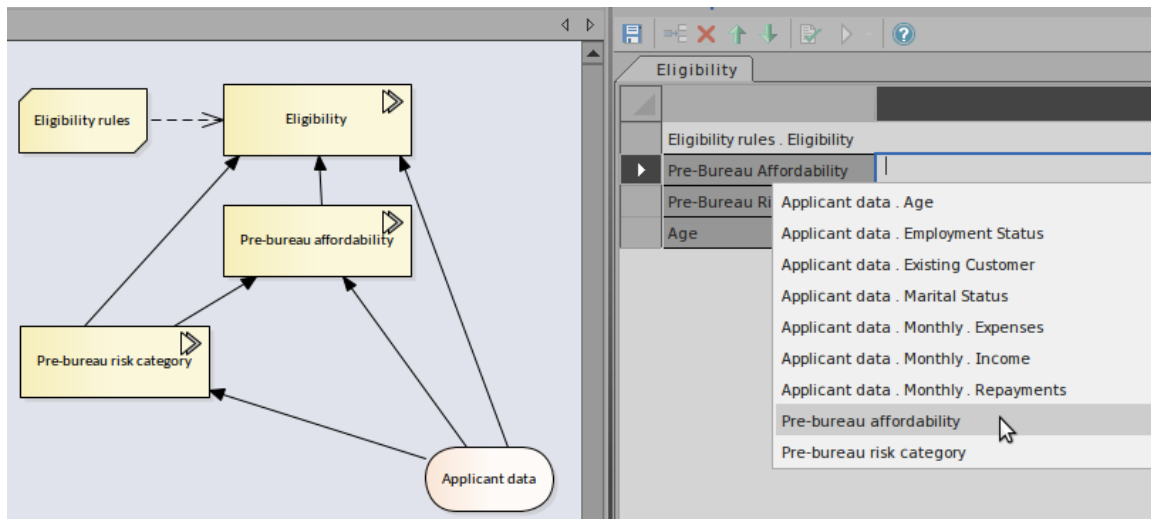
Then fast fill the decision table rules by selection:



## Information Requirement

On a decision hierarchy, a decision might access required decisions and input data; these required elements form a list of

variables that can be used by the decision.



In this example, Decision "Eligibility" requires two decisions - "Pre-bureau risk category" and "Pre-bureau affordability" - and one Input Data, "Applicant data".

When setting the binding values for the invoked BusinessKnowledgeModel "Eligibility rules", an Auto Completion list will prompt for selection.

In this list, there are sub-decision names, leaf components of the input data.

With this feature, the user will be able to easily set up an invocation.

## DMN Expression Validation

DMN defines many expressions such as FunctionDefinition, DecisionTable, Boxed Context, Invocation and Literal Expression. The parameters, arguments, logic of expressions is implemented largely by "text".

To make modeling easy and reliable, Enterprise Architect provides two features: Auto Completion and Validation.

- Auto Completion: The idea is to select a text string from a list of enumerations rather than type the text in
- Validation: this can pick up modeling errors caused by typo, logic completeness and consistency and so on

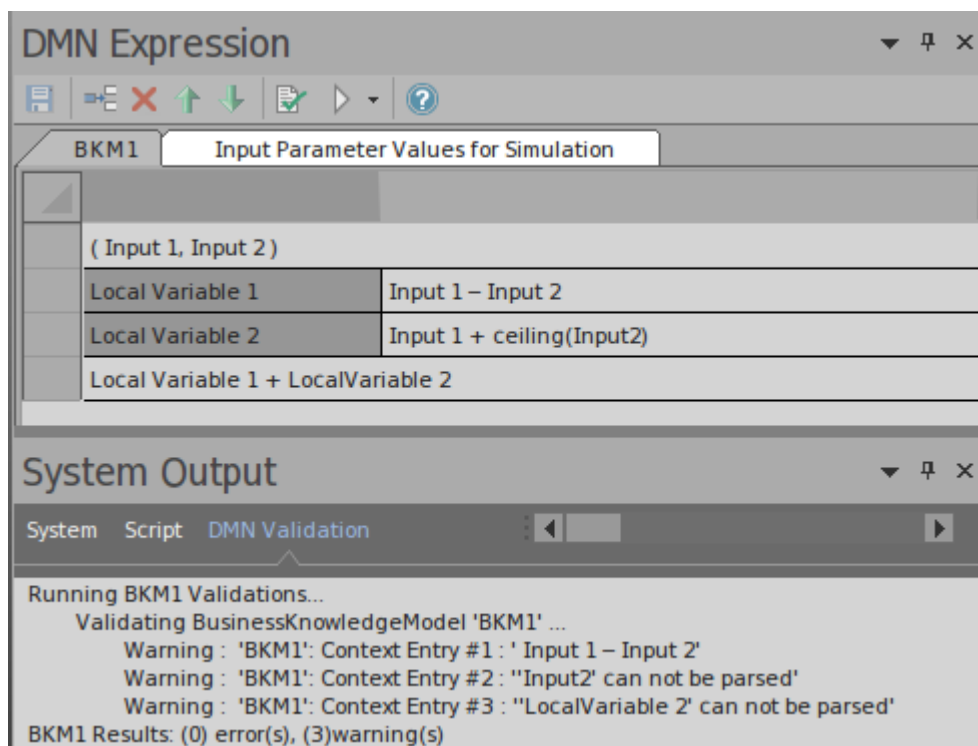
In this topic, we will show you how to validate a DMN Expression.

### Access

DMN Expression Widnow	Simulate > Decision Analysis > DMN > DMN Expression, then click the Validate button
DMN Simulation Window	Simulate > Decision Analysis > DMN > Open DMN Simulation > Simulate page, then click the Validate icon

### Common validations

#### Variable Name Validation



In this example, BusinessKnowledgeModel BKM1 defined two parameters: "Input 1" and "Input 2", two local variables: "Local Variable 1" and "Local Variable 2"

- Context Entry #1 failed because there is a typo: It should be operator "-", but typed in "–" instead.
- Context Entry #2 failed because there is no space between "Input" and number 2. Note: function ceiling is defined in

DMN Library so it can be successfully parsed.

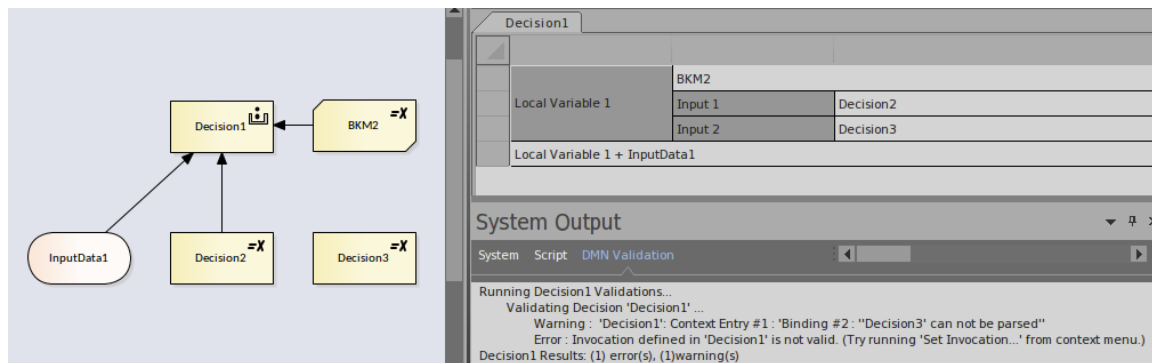
- Context Entry #3 failed because there is no space between "Local" and "Variable".

It is hard to identify these kind of errors by eye sight, running validation can help identify errors and the user may perform an easy fix.

## Dependency Validation

A decision may require other decisions, input datas and business knowledge models, these relationships are identified by informationRequirement and KnowledgeRequirement connectors.

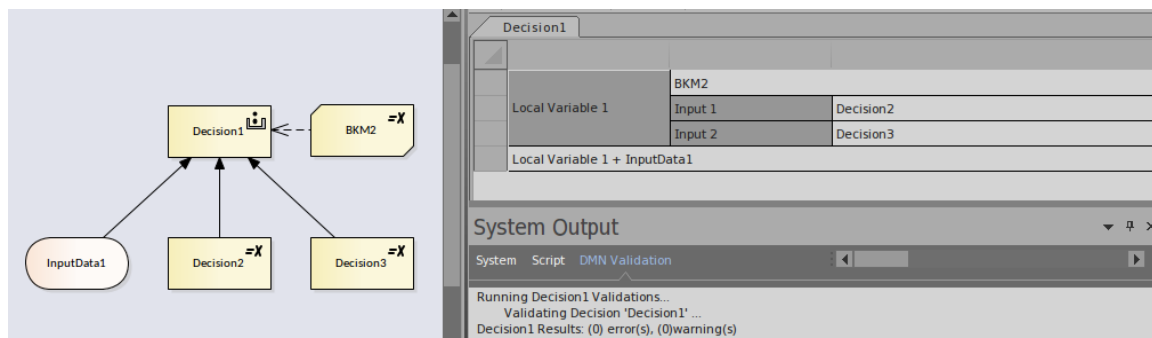
When the graph is getting complex, it is quite possible that some connectors were missing or the wrong connector type is being used.



In this example, click on the Validate button, EA will pickup:

- "Decision3" is used by "Decision1" by binding to a parameter of the called BKM2. However, it is not defined. The user may realize that an InformationRequirement connector is missing.
- Invocation defined in "Decision1" is not valid. The user may realize that the connector type from BKM2 to Decision1 should be an KnowledgeRequirement.

After fix this problems, run the validation again:



## DMN Decision Table Validation

Decision table is one of the most common and powerful DMN Expressions to express the decision logic. However, modeling a Decision table could also be complicated, especially multiple input clauses have combination for many decision table rules. Luckily, EA provided the feature to validate a decision table.

In this topic, we will show the user how to validate a DMN Decision Table.

### Access

DMN Expression Window	Simulate > DMN > Manage > DMN Expression, then click the Validate button
DMN Simulation Window	Simulate > DMN > Manage > Open DMN Simulation > Configure page, then click the Validate button

### Entries out of range detection

It is a good practice to define "allowed values" for input clause and output clause of a decision table.

Firstly, the enumerations (comma separated string) will enable the Auto Completion feature, where the user can choose the value from a list rather than typing them in the entry field.

Secondly, the "allowed values" enables EA to perform a range check for the entry values.

**DMN Expression**

Application risk score model    Input Parameter Values for Simulation

( Age, Marital Status, Employment Status )

C+	Age	Marital Status	Employment Status	Partial score
	[20..120]	S, M	UNEMPLOYED, EMPLOYED...	
1	[18..21]	-	-	32
2	[22..25]	-	-	35
3	[26..35]	-	-	40
4	[36..49]	-	-	43
5	>=50	-	-	48
6	-	S	-	25
7	-	D	-	30
8	-	M	-	45
9	-	-	UNEMPLOYED	15
10	-	-	STUDENT	18
11	-	-	EMPLOYED	45
12	-	-	SELF-EMPLOYED	36

**System Output**

System    Script    DMN Validation

Running Application risk score model Validations...

Validating BusinessKnowledgeModel 'Application risk score model' ...

Warning : DecisionTable "Application risk score model" Input Violation:Input Value is not allowed for "Rule[1].Age": [18..21]

Warning : DecisionTable "Application risk score model" Input Violation:Input Value is not allowed for "Rule[7].Marital Status": "D"

Application risk score model Results: (0) error(s), (2)warning(s)

In this example,

the "Age" input Clause defined a range of [20..120], caused the first rule [18..21] fail.

The Marital Status defined a enumeration "S, M", caused the rule #7 with value "D" fail.

The user may fix this issue by changing the "allowed value" range or the rule entry, based on the actual business rules.

## Completeness detection – report gaps in the rules

The gaps in rules for a decision table means that given a combination of input values, no rule is matched. This indicates that some logic/rule may missing (unless a default output is defined).

When the decision table defined rules with lots of number ranges, it is hard to detect gaps by either eye sight and too hard to compose test cases.

For example,

DMN Expression

Post-bureau risk category table    Input Parameter Values for Simulation

( Existing Customer, Application Risk Score, Credit Score )

U	Existing Customer	Application Risk Score	Credit Score	Post Bureau Risk Category
1	false	<120	<590	HIGH
2	false	<120	[590..610]	MEDIUM
3	false	<120	>610	LOW
4	false	[120..130]	<600	HIGH
5	false	[120..130]	[600..625]	MEDIUM
6	false	[120..130]	>625	LOW
7	false	>130	-	VERY LOW
8	true	<=100	<580	HIGH
9	true	<=100	(580..600]	MEDIUM
10	true	<=100	>600	LOW
11	true	>100	<590	HIGH
12	true	>100	[590..615]	MEDIUM
13	true	>100	>615	LOW

System Output

System    Script    DMN Validation

Running Post-bureau risk category table Validations...

Validating BusinessKnowledgeModel 'Post-bureau risk category table' ...

Warning : DecisionTable "Post-bureau risk category table" Completeness Violation: No rule exists for Existing Customer:"true", Application Risk Score:"<=100", Credit Score:"580"

Post-bureau risk category table Results: (0) error(s), (1)warning(s)

The validation reports a gap in the rules, the user may first perform a merge on the decision table to focus on the 3rd input "Credit Score" and easily detect the error in input entry (580..600], which should be [580..600].

## Rule Overlaps detection for Unique Hit Policy

When rules overlap, given a combination of input values, multiple rules are matched. This is a violation if the decision table specifies Unique as hit policy.

When the decision table defined rules with lots of number ranges, it is hard to detect gaps by either eye sight and too hard to compose test cases.

For example,

DMN Expression

Post-bureau risk category table    Input Parameter Values for Simulation

( Existing Customer, Application Risk Score, Credit Score )

U	Existing Customer	Application Risk Score	Credit Score	Post Bureau Risk Category
1	false	<120	<590	HIGH
2	false	<120	[590..610]	MEDIUM
3	false	<120	>610	LOW
4	false	[120..130]	<610	HIGH
5	false	[120..130]	[600..625]	MEDIUM
6	false	[120..130]	>625	LOW
7	false	>130	-	VERY LOW
8	true	<=100	<580	HIGH
9	true	<=100	(580..600]	MEDIUM
10	true	<=100	>600	LOW
11	true	>100	<590	HIGH
12	true	>100	[590..615]	MEDIUM
13	true	>100	>615	LOW

System Output

System    Script    DMN Validation

Running Post-bureau risk category table Validations...

Validating BusinessKnowledgeModel 'Post-bureau risk category table' ...

Warning : DecisionTable "Post-bureau risk category table" Consistency Violation: Rules "4, 5" overlap with input "false, [120..130], [600..610]"

Post-bureau risk category table Results: (0) error(s), (1)warning(s)

The validation reports an overlap in the rules, the user may first perform a merge on the decision table to focus on the 3rd input "Credit Score" and easily detect the overlap between "<610" and "[600..625]". The user may fix this issue either by changing rule #4 to "<600" or by changing rule #5 to "[610..625]", based on the actual business rules.



## DMN Simulation

After a Decision Model is created, the user may configure a DMN Simulation artifact and Validate/Run/Step/Debug the model.

By switching data sets, the user may do what-if analysis to ensure the model meet the requirement of business.

The user may also generate code for the DMN Server. Currently, 4 languages are supported: Java/JavaScript/C++/C#.

In order to simulate BPMN and DMN together, EA's BPSim Execution Engine accept a DMN Server in Java language. Before the BPSim Execution engine execute, we need to ensure the DMN Server is tested. Luckily, we can Run/Debug the testing for Java DMN Server easily in EA.

### Configure DMN Simulation

- Create a DMNSimConfiguration element in a package, double click to open it. Then all DMN elements in this package (Decision, BusinessKnowledgeModel, InputData ItemDefinition) will be loaded to the simulation window.
- The Target Decision combo box will be filled with all the decisions; choose a target decision, the dependent InputDatas will be filled in the list.
- Choose a defined dataset by clicking on the dataset cell in the list. (E.g. We chose Dataset "Income 5000" for InputData "Applicant data"; chose "default" for InputData "Requested product")

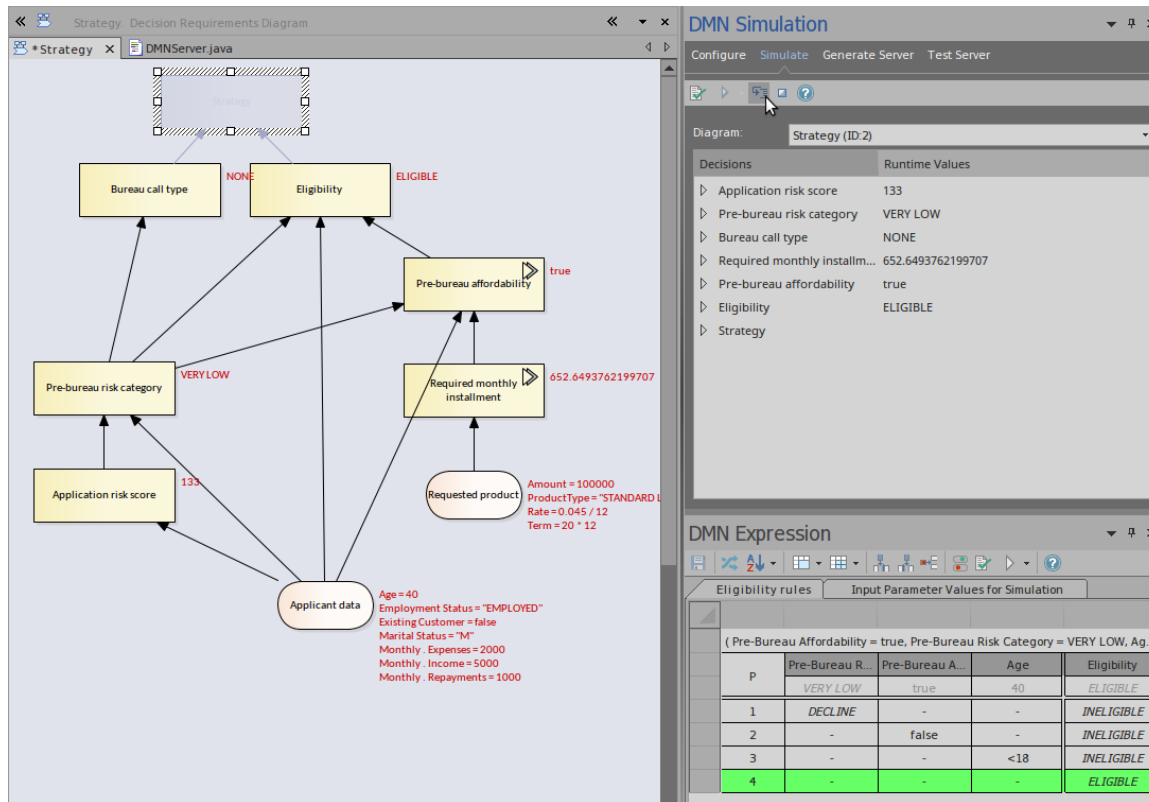
The screenshot displays the DMN Simulation configuration interface. On the left, a Strategy Decision Requirements Diagram is shown, featuring a central 'Strategy' decision node connected to several input nodes: 'Bureau call type', 'Eligibility', 'Pre-bureau risk category', 'Application risk score', 'Pre-bureau affordability', 'Required monthly installment', and 'Applicant data'. On the right, the 'DMN Simulation' window is open, showing the 'Configure' tab. The 'Target Decision' is set to 'Strategy'. Below it, a list of 'Input Data Item' is displayed, including 'Applicant data', 'Pre-bureau affordability', 'Required monthly installment', and 'Requested product'. The 'DMN Expression' window at the bottom shows a table with columns for 'Eligibility', 'Bureau call type', and 'Strategy'. The table contains four rows of data, including a header row 'U' and three data rows numbered 1, 2, and 3.

	Eligibility	Bureau call type	Strategy
U	ELIGIBLE, INELIGIBLE	FULL, MINI, NONE	DECLINE, BUREAU...
1	INELIGIBLE	-	DECLINE
2	ELIGIBLE	FULL, MINI	BUREAU
3	ELIGIBLE	NONE	THROUGH

### Simulate DMN Model

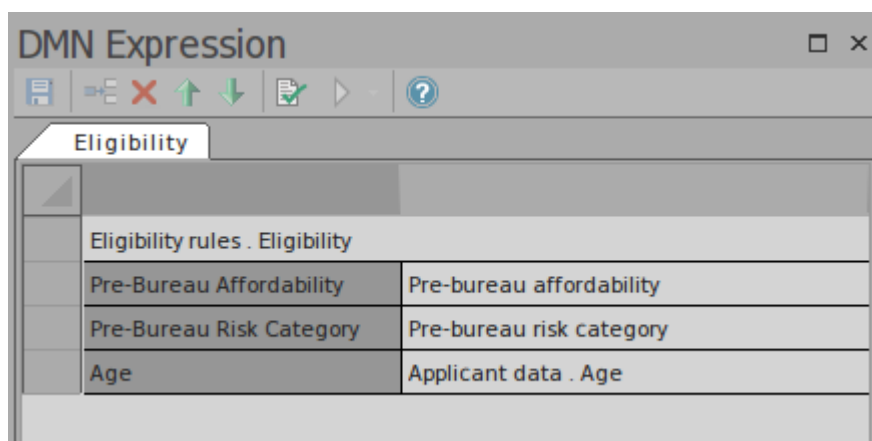
- When a Target Decision is specified, the simulation page will be filled with the decisions, in dependency order.
- Each decision row can be expanded to show the invoked BusinessKnowledgeModel.
- Click the "Run" button will evaluate all the decision values based on the values for input datas.
- Click the "Step" button will evaluate a single decision and the user can watch the DMN Expression window, which clearly showed the input value for the decision and output based on the input. The diagram containing the decision

hierarchy will highlight the executed decisions and show the runtime results on a label.



In this example, decision "Eligibility" returns a string "ELIGIBLE". Decision "Eligibility" invokes BusinessKnowledgeModel "Eligibility rules" by binding the parameters:

- Bind "Pre-Bureau Affordability" to dependent decision "Pre-bureau affordability" (runtime value: true)
- Bind "Pre-Bureau Risk Category" to dependent decision "Pre-bureau risk category" (runtime value: VERY LOW)
- Bind "Age" to a field "Age" in dependent input data "Applicant data" (runtime value: 40)



The BusinessKnowledgeModel "Eligibility rules" has a hit policy P (Priority), meaning that multiple rules can match, but only one hit should be returned, the ordering of the list of output values is used to specify the (decreasing) priority.

In this run time case (Pre-Bureau Affordability = true, Pre-Bureau Risk Category = VERY LOW, Age = 40), only one rule with output "ELIGIBLE" matches.








## Configure DMN Simulation

A DMNSimConfiguration artifact contains information to simulate a DMN model depicted by Decision Requirements Diagrams.

### Access

Ribbon	Simulate > DMN > Manage > Open DMN Simulation, Configure Page
Other	Double-click on an DMNSimConfiguration element

### Toolbar Options

Option	Description
	Click on this button to select/create a DMNSimConfiguration element.
	Click on this button to set a package for the DMNSimConfiguration artifact. All DMN elements under this package or sub packages will be loaded.
	Click on this button to reload DMN elements from the configured packages. For example, when some DMN elements were modified, run this command to reload the package so the changes will be taken into account for DMN Simulation.
	Click on this button to open the dialog for editing data sets for the selected input data.
	Click this button to save the DMN Simulation window information to DMNSimConfiguration element. Including: <ul style="list-style-type: none"><li>• Target Decision</li><li>• selected Dataset for each dependent InputData</li></ul>
	Click on this button to export the "name = value" records for all the inputDatas and their specified DataSets to a BPMN 2.0 DataObject(write to the Notes).
	Click on this button to export the "name = value" records for the selected InputData and its specified Dataset to a BPMN 2.0 DataObject (write to the Notes).  These value defined in BPMN 2.0 DataObject will be read by EA BPSim Execution Engine.

### DMNSimConfiguration Artifact

The user can create a DMNSimConfiguration element in the following ways:

- In the DMN Simulation Window, click the Artifact button on the toolbar.
- On a Decision Requirements Diagram, create the DMNSimConfiguration element from the toolbox.

By default, all DMN elements in this package (Decision, BusinessKnowledgeModel, InputData ItemDefinition) will be loaded to the simulation window.

## Decision Hierarchy

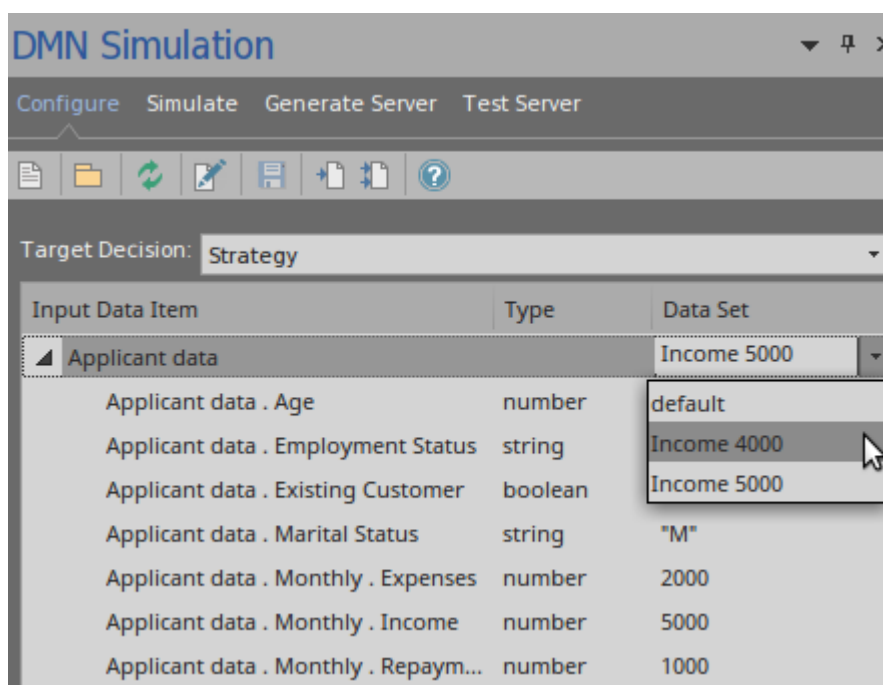
When a package is loaded, a Decision Requirements Graph (DRG) and decision dependency hierarchy is created. The DMN InformationRequirement connectors determines the hierarchy.

- All the decisions will be filled into the "Target Decision" combo box.
- When a "Target Decision" is specified, all the dependent Decisions, invoked BusinessKnowledges, InputDatas and the typed ItemDefinitions will be determined.

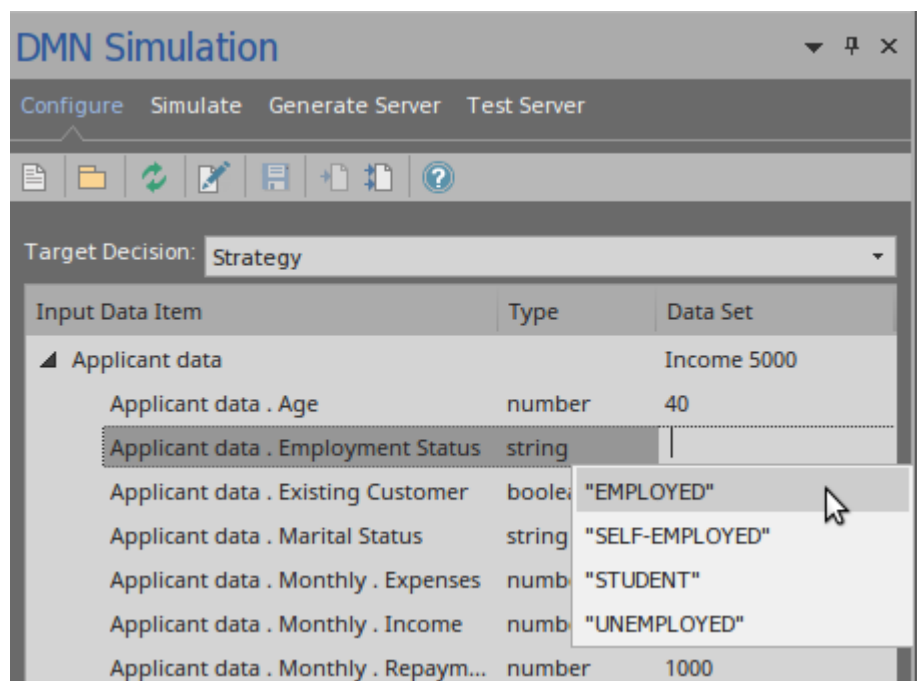
## DataSet & Input Data

When the Target Decision is selected. All the dependent InputDatas will be filled into the list.

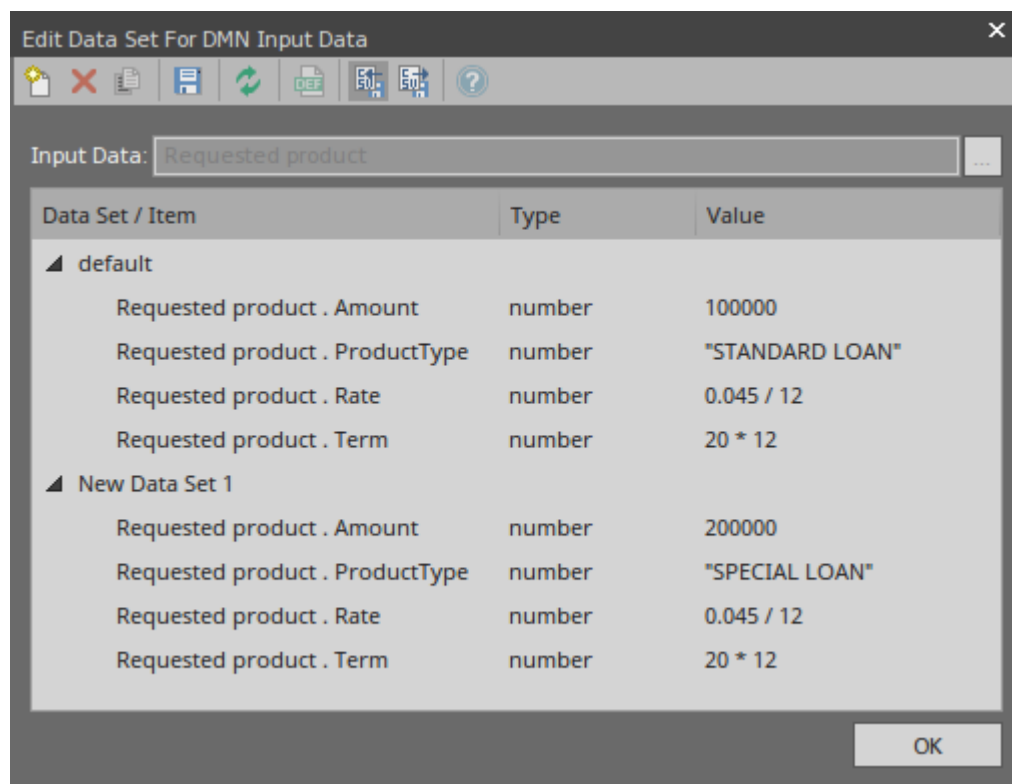
The user can choose a dataset for simulation from a list of defined datasets of an InputData.



The user can also change the Dataset value in the list, if ItemDefinition specified an Allowed Value Enumerations, auto-completion feature is supported and the user can simply choose from a list of enumerations.



The user can also edit the datasets for an InputData by clicking the "Dataset" button on the toolbar to open the Dataset Edit dialog.







## Simulate DMN Model

A DMNSimConfiguration artifact contains information to simulate a DMN model depicted by Decision Requirements Diagrams.

### Access

Ribbon	Simulate > DMN > Manage > Open DMN Simulation, Simulate Page
Other	Double-click on an DMNSimConfiguration element, Simulate Page

### Toolbar Options

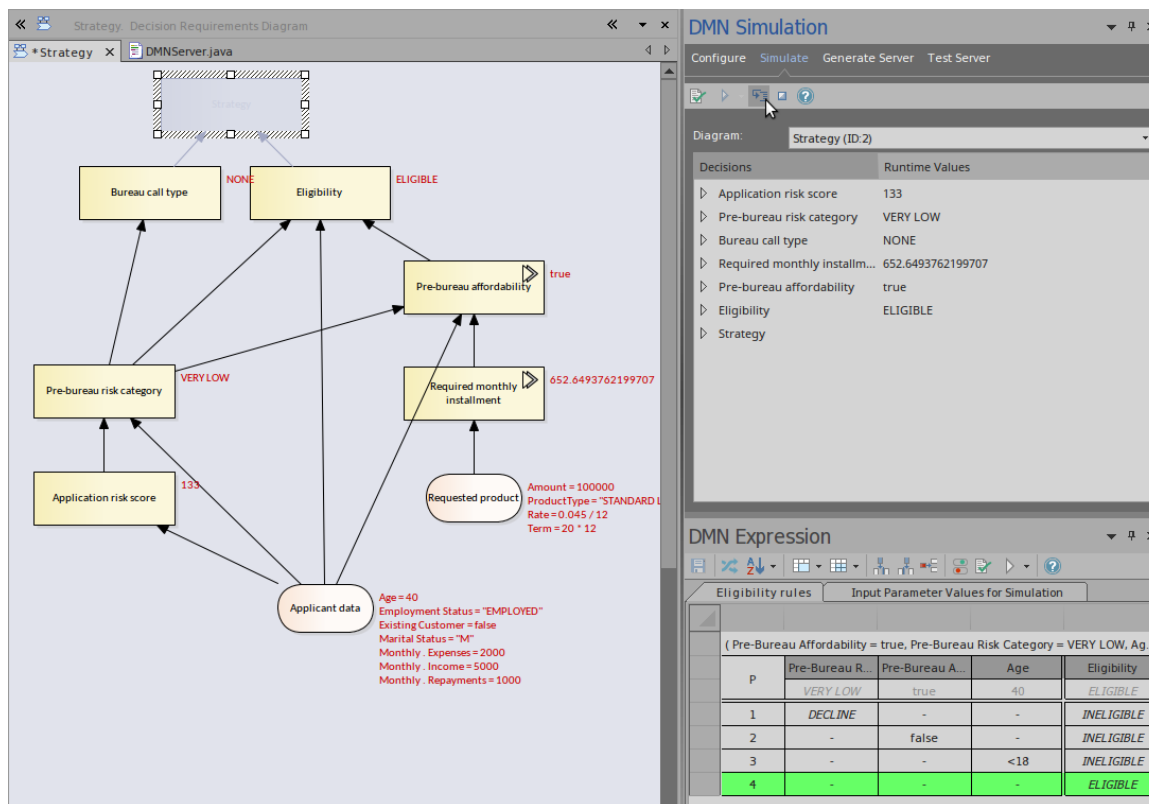
Option	Description
	<p>Click on this button to validate all the dependent DMN elements based on the Target Decision.</p> <p>Note: A Decision/BusinessKnowledgeModel/InputData/ItemDefinition that is on the target decision hierarchy will not be considered.</p> <p>For example, the user has some unfinished decision elements in the package, since they have no relationship the target decision, they will not impact the simulation.</p>
	Click on this button to execute the decision hierarchy in order. The result will be represented on the diagram and show in the "Runtime values" column.
	Click on this button to step run the decision hierarchy in order, one click will evaluate one decision element. With this feature, the user will be able to see the decision-making process, besides, the decision logic and runtime values will be displayed cleared in the DMN Expression window.
	Click on this button to exit the simulation mode.

### Simulation Overview

When a Target Decision is specified, the simulation page will be filled with the decisions, in dependency order.

When Executing or Step the Decision Hierarchy, the decisions will be evaluated in order:

- The runtime result will be showing in the decision row
- The runtime result will be displayed on the diagram
- The decision logic and input/output data will be presented in the DMN Expression window.



## Invocation Hierarchy

The user can expand the Decision in the list to see the invocation hierarchy.



For example,

- Decision "Pre-bureau affordability" invokes BusinessKnowledgeModel "Affordability calculation"
- BusinessKnowledgeModel "Affordability calculation" further invokes another BusinessKnowledgeModel "Credit contingency factor table"

## Advanced Debugging

Although EA provided a validation feature to help the user pickup many modeling issues and DMN expression issues, rarely but possible, the simulation may still fail due to some uncaught issues.

Luckily, EA provided the user an ability to debug the code that is running behind the simulation. The user may also modify the code, run it until the issue is found and fixed.

The Execute button on the toolbar has a menu with following items:

- Generate New Script (Scripting Window)
- Update Selected Script (Scripting Window)
- Run Selected Script (Scripting Window)
- Edit DMN Template

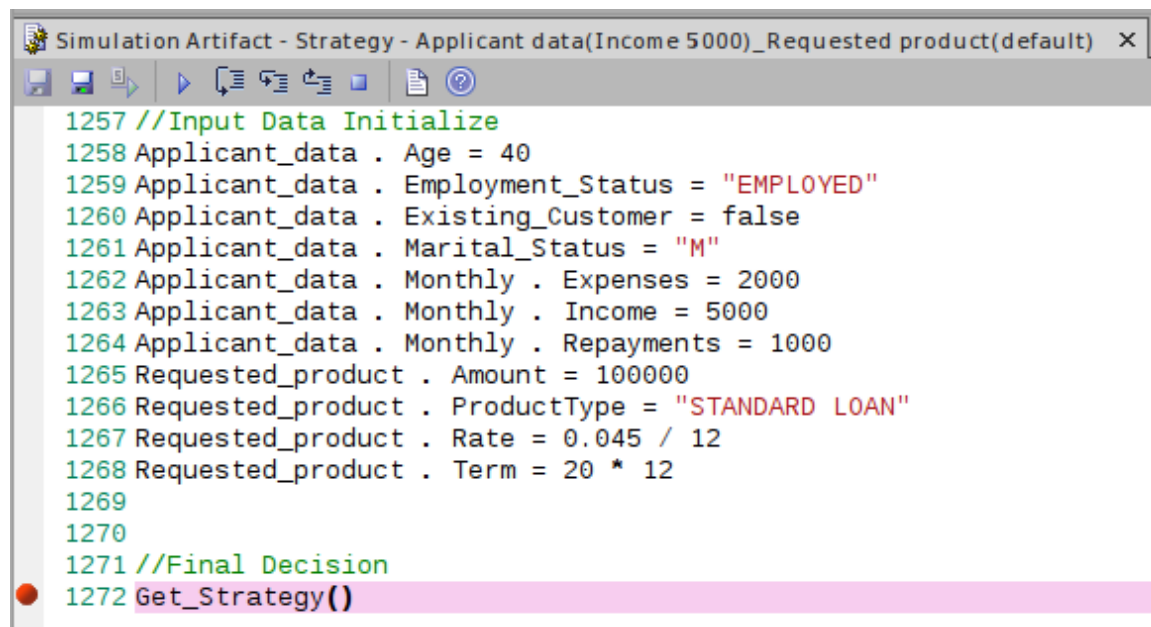
Click "Generate New Script (Scripting Window)" will open the Scripting window with a script created in a package named "DMN".



- The default script name is composed as "ArtifactName - TargetDecision - InputData1(DataSet)\_InputData2(DataSet)\_..."

Double click on this file to open it in EA script editor, set a breakpoint, and debug the file.





```
1257 //Input Data Initialize
1258 Applicant_data . Age = 40
1259 Applicant_data . Employment_Status = "EMPLOYED"
1260 Applicant_data . Existing_Customer = false
1261 Applicant_data . Marital_Status = "M"
1262 Applicant_data . Monthly . Expenses = 2000
1263 Applicant_data . Monthly . Income = 5000
1264 Applicant_data . Monthly . Repayments = 1000
1265 Requested_product . Amount = 100000
1266 Requested_product . ProductType = "STANDARD LOAN"
1267 Requested_product . Rate = 0.045 / 12
1268 Requested_product . Term = 20 * 12
1269
1270
1271 //Final Decision
1272 Get_Strategy()
```

By selecting the script in the Scripting Window, the menu "Run Selected Script" will be enabled if the script matches the model (by the "Simulation Script Identifier" in the script).

The user may customize the DMN Template to generate the correct script for simulation.

## DMN Module Code Generation & Test

After a Decision Model is created, simulated, the user may generate a DMN Module in language Java/JavaScript/C++/C#. The DMN Module may work together with EA BPSim Execution Engine, Executable Statemachine, or user's own project.

EA also provided a "Test Module" page, which is a preprocess for integrate DMN with BPMN. The idea is by providing one or more BPMN2.0::DataObject elements, then test if a specified target decision can be evaluated correctly or not.

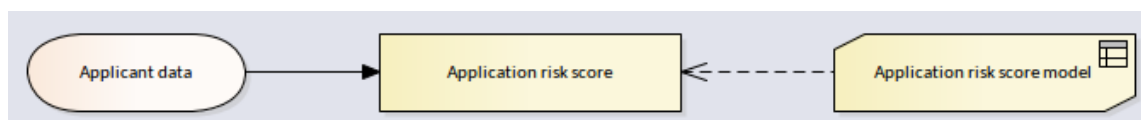
If any error or exception occurs, the user may create a Analyzer Script to debug the code of DMN Module and Test Client.

After this "Test Module" process, EA guaranteed the BPMN2.0::DataObject elements works well with the DMN Module.

The user then configure BPSim by loading DataObjects and assigning DMN Module decisions to BPSim Properties, which will be further used as conditions on the sequence flows outgoing from a gateway.

## Modeling & Simulation

We took the example available from Model Patterns: Ribbon | Simulate | DMN | Apply Perspective | DMN Decision | Decision with BKM



Here is a brief overview of the model:

- Item Definition:

DMN Expression			
Applicant data Definition (ItemDefinition)			
Applicant data Definiti...	Marital Status : string	"S", "M"	
	Employment Status : ...	"UNEMPLOYED", "EMPLOYED", "SELF-EMPLOYED", "STUDENT"	
	Age : number	Type in Allowed Value Enumerations...	

- Input Data

DMN Expression			
Applicant data : Applicant data Definition			
Applicant data Definition	Marital Status : string	"M"	
	Employment Status : string	"EMPLOYED"	
	Age : number	40	

- Business Knowledge Model (Decision Table)

DMN Expression				
Application risk score model				
Input Parameter Values for Simulation				
( Age, Marital Status, Employment Status )				
C+	Age	Marital Status	Employment Status	Partial score
	[18..120]	S,M	UNEMPLOYED,EMPLO...	
1	[18..21]	-	-	32
2	[22..25]	-	-	35
3	[26..35]	-	-	40
4	[36..49]	-	-	43
5	>=50	-	-	48
6	-	S	-	25
7	-	M	-	45
8	-	-	UNEMPLOYED	15
9	-	-	STUDENT	18
10	-	-	EMPLOYED	45
11	-	-	SELF-EMPLOYED	36

- Decision (Invocation)

DMN Expression	
Application risk score	
Application risk score model . Partial score	
Age	Applicant data . Age
Marital Status	Applicant data . Marital Status
Employment Status	Applicant data . Employment Status

- Simulation

Application risk score model				
Input Parameter Values for Simulation				
( Age = 40, Marital Status = M, Employment Status = EMPLOYED )				
C+	Age	Marital Status	Employment Status	Partial score
	40	M	EMPLOYED	133
1	[18..21]	-	-	32
2	[22..25]	-	-	35
3	[26..35]	-	-	40
4	[36..49]	-	-	43
5	>=50	-	-	48
6	-	S	-	25
7	-	M	-	45
8	-	-	UNEMPLOYED	15
9	-	-	STUDENT	18
10	-	-	EMPLOYED	45
11	-	-	SELF-EMPLOYED	36

- Export InputData to a BPMN2.0 DataObject

Activate "Configure" page, click Export DataObject buttons on the toolbar

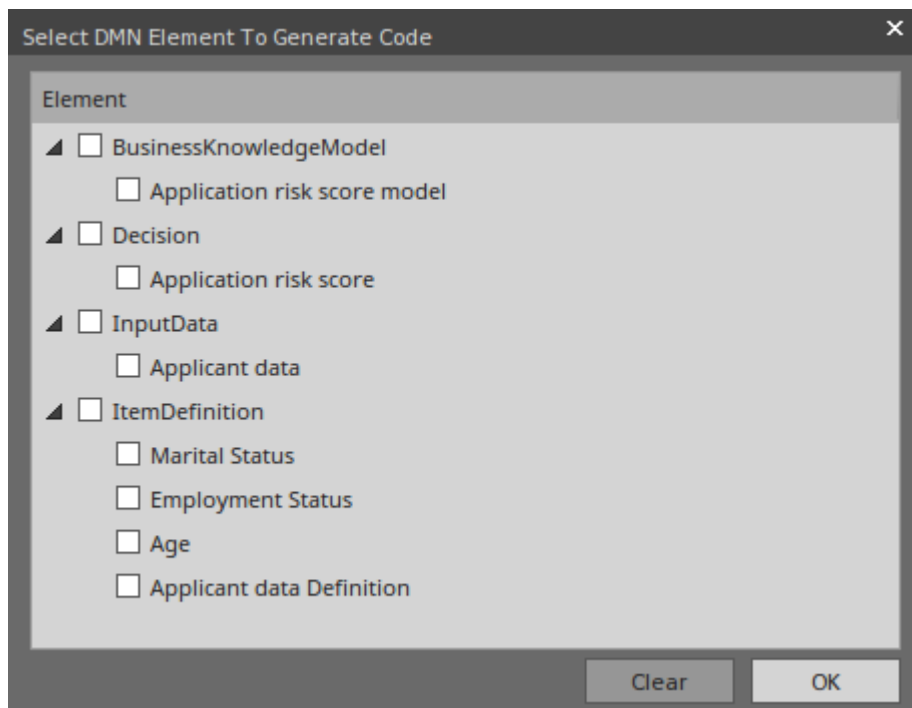
## DMN Module: Code Generation

Activate the 3rd page "Generate Module" of the DMN Simulation window, select the DMN elements you want to generate to the server, specify the file path and language, then click "Generate".

(Note: For Java language, the path need to match the package structure.)

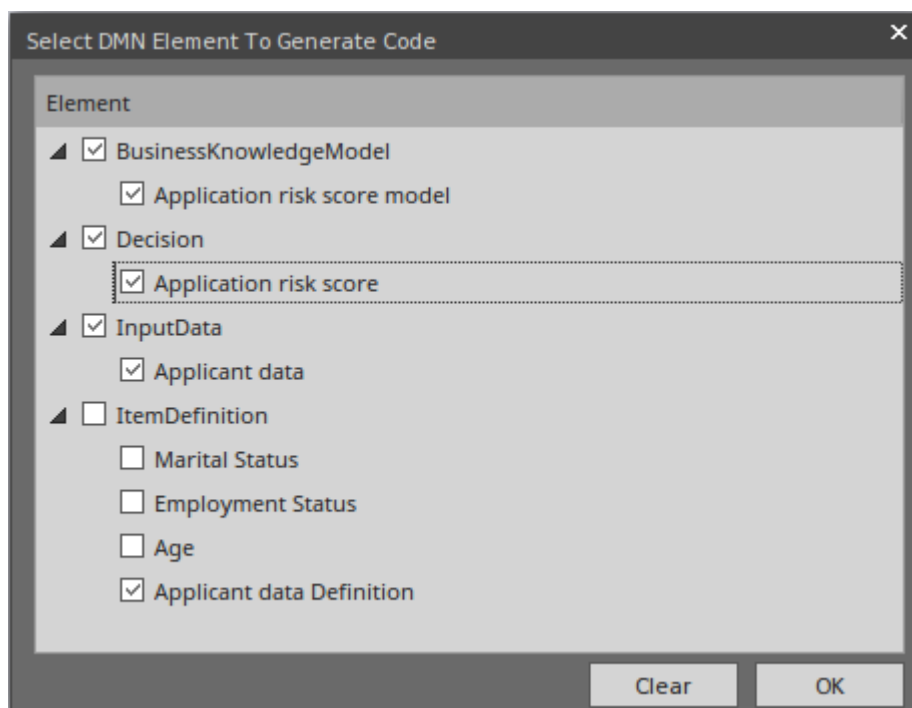
- Add elements to module

Click the Add button on the toolbar to open the DMN Element Selection dialog:



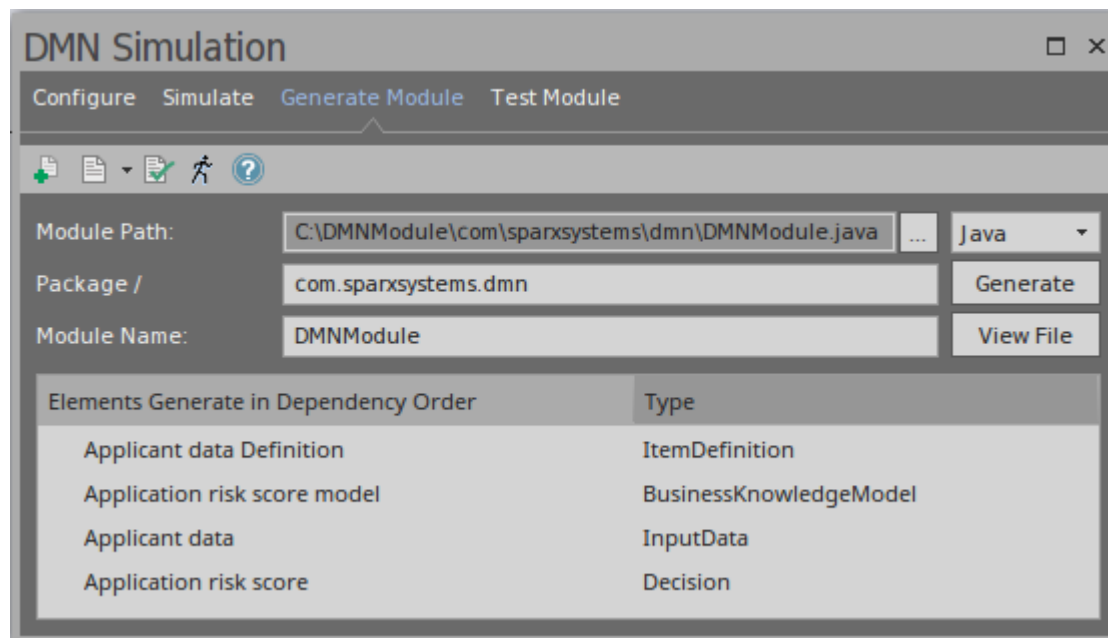
Click the decision you want to generate to the server, in this example, we selected decision "Application risk score".

Note: all the dependencies will be selected automatically.



- Generate DMN Module

Give the Java file a package name, click generate.



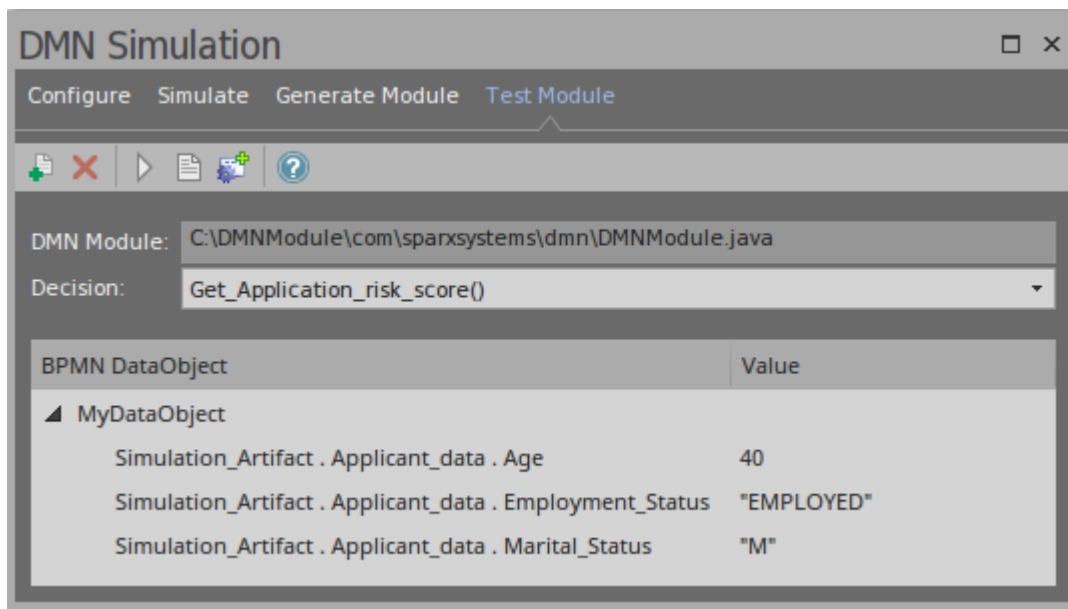
Note: in this example, the Module Path ended with "\com\sparxsystems\dmn", which matches the Package "com.sparxsystems.dmn".

- Click the Test button to enter the Test Server page

## DMN Server: Test client

If this page was activated from "Generate Server" page, the "DMN Module" field will be filled automatically with the generated DMN Server's path. Otherwise, click "..." button to browse for a DMN Server file.

Click the "Add DataObject" button to add one or more BPMN2.0 DataObject(s) to the list, choose a decision from the combo box, then click the run button on the toolbar.



In the System Output window, the following message indicates the DMN Server and BPMN2.0 DataObject may work well with each other to evaluate the selected decision.

Running Test Client for DMN Server...

dmnServer.Application\_risk\_score: 133.0

Result : 133.0

The Running completed successfully.

If there are errors, create a Analyzer script by clicking the toolbar button and fix the issue.

Important: This "Test Module" step is recommended before integrate DMNServer.java to EA BPSim Execution Engine.

# Integrate DMN Module Into UML Class Element

After a Decision Model is created, simulated, the user may generate a DMN Module in language Java/JavaScript/C++/C# and get it tested.

This DMN Module can be integrated into UML class element so the code generated from the class element can reuse the DMN model and be well structured. Since a class element may define statemachine(s), after integration with DMN module, the executable statemachine simulation will generically be able to use the power of DMN module.

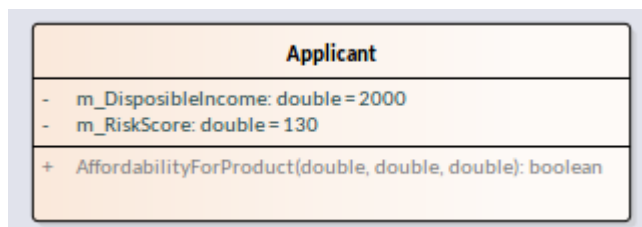
In this page, we will go through a whole process to integrate a DMN Model to a UML Class element:

- Class Element
- DMN Model(s)
- DMN Binding to Class & IntelliSense
- Code Generation on Class Element

## Class Element's Requirement

Suppose we have a class *Applicant* with an operation *AffordabilityForProduct*, which evaluate whether the applicant can afford a loan product.

A simplified model looks like this:



The class *Applicant* contains 2 attributes, which is actually calculated from more basic data like applicant's monthly income, expense, existing repayment, age, employment status etc.

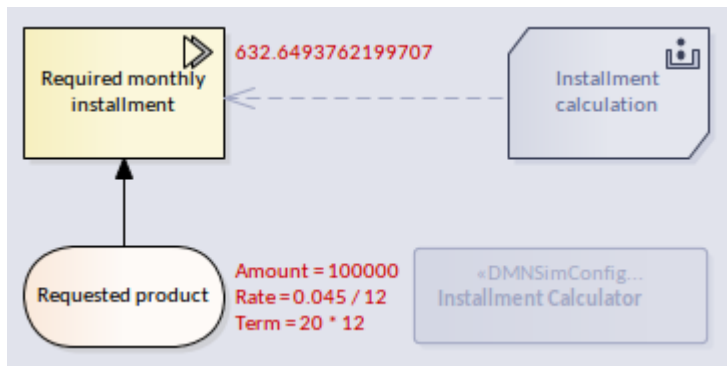
In this example, however, we simplify the model by skipping these steps and providing disposable income and risk score directly. In the DMN Complete Example (Model Patterns), you can see all the thorough steps.

## DMN Model(s)

In this example, we have 2 disjoint DMN Models to show that a UML class can integrate multiple DMN Models.

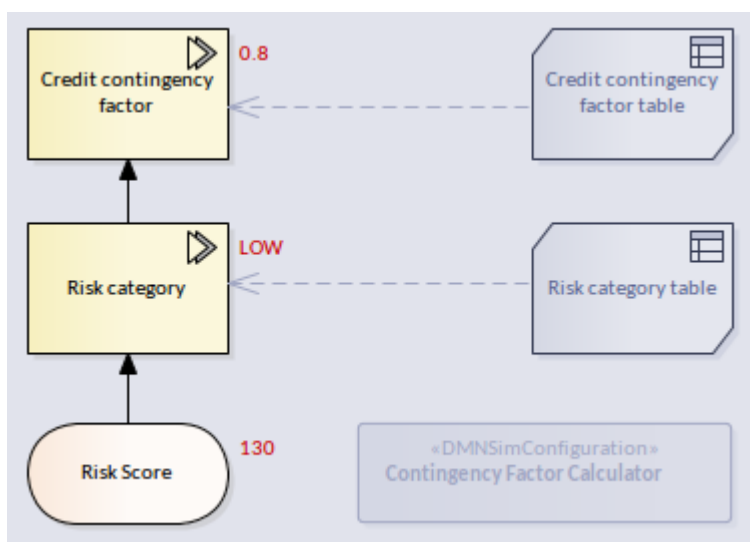
- Installment Calculator

This DMN Model computes the monthly repayment based on amount, rate and terms. It is composed of an InputData, a Decision and a Business Knowledge Model.



- Credit Contingency Factor Calculator

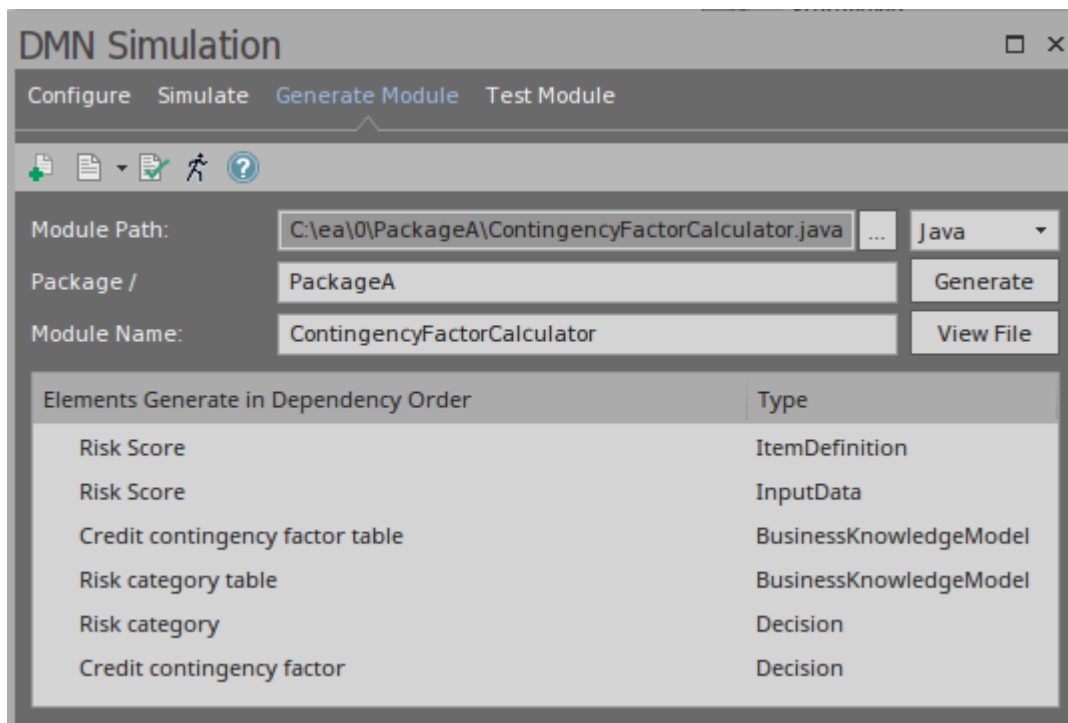
This DMN Model computes the credit contingency factor based on applicant's risk score. It is composed of an InputData, two Decisions and two Business Knowledge Models.



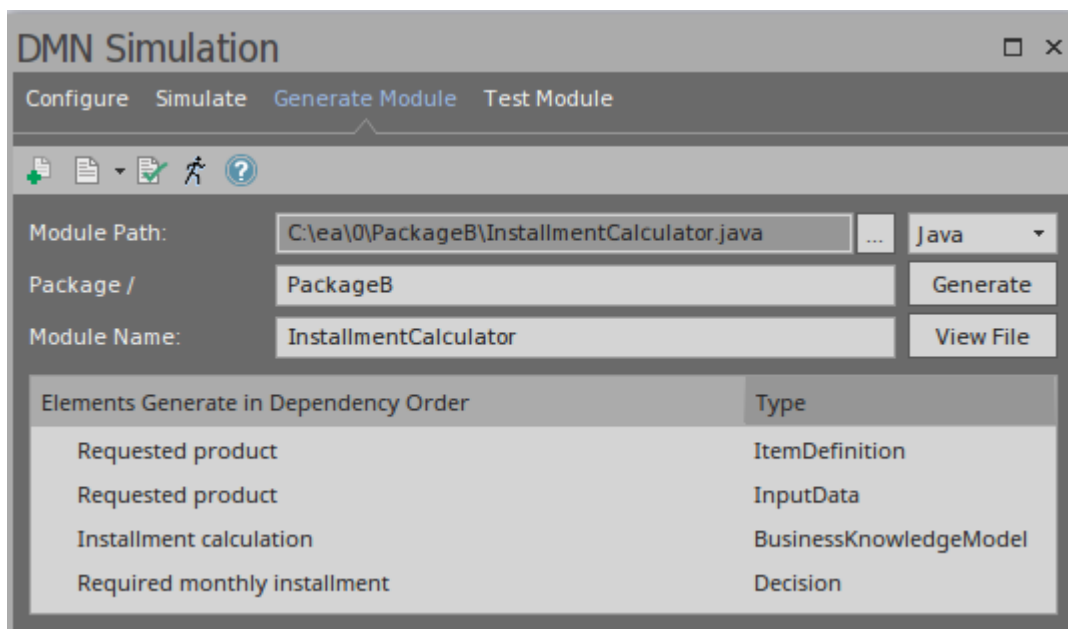
Note: In this example, we focus on how to integrate DMN Module into Class Element, the DMN Element's detail is not described in this page. The full example is available in Model Patterns and EAExample model.

- Generate code for both DMN Models

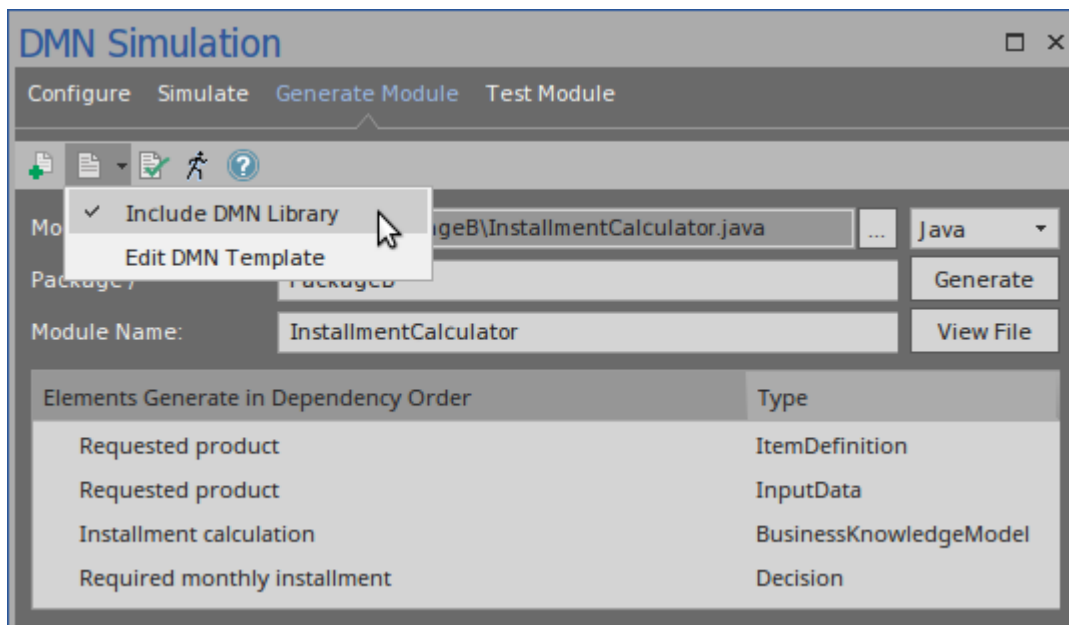




Click "Generate" button, ensure you can see the following string in the System Output window, DMN Page:  
*DMN Module is successfully compiled.*



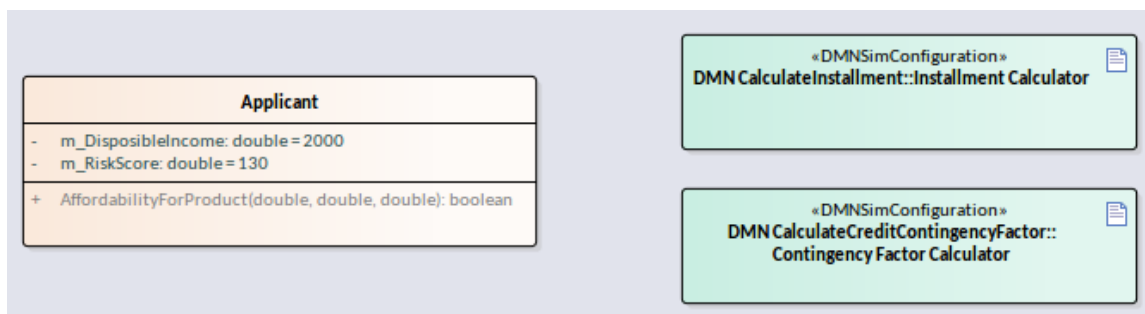
Note: Since this model uses a built-in function PMT, the DMN Library need to be included:



Click "Generate" button, ensure you can see the following string in the System Output window, DMN Page:  
*DMN Module is successfully compiled.*

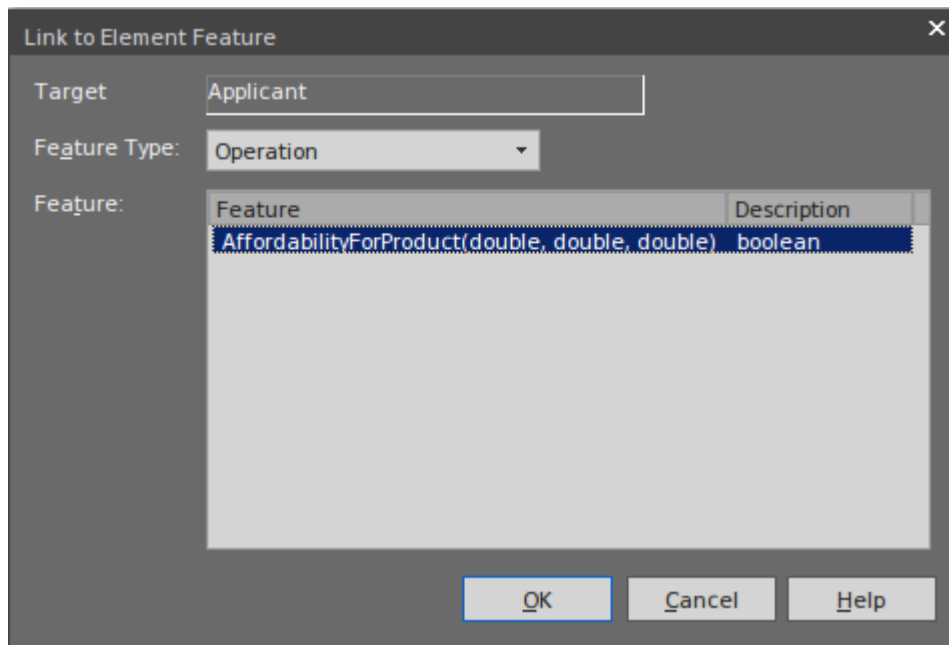
## DMN Binding to Class & IntelliSense

Put the two DMNSimConfiguration artifacts on the class diagram



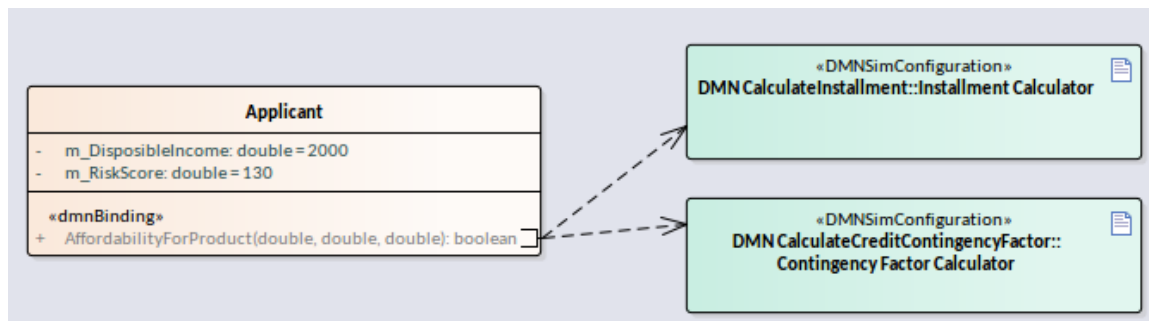
Use the quick linker to create a Dependency connector from Class *Applicant* to the DMN Artifact.

On the connector's creation, the dialog will prompt for the user to choose the operation which will be bound to DMN module.



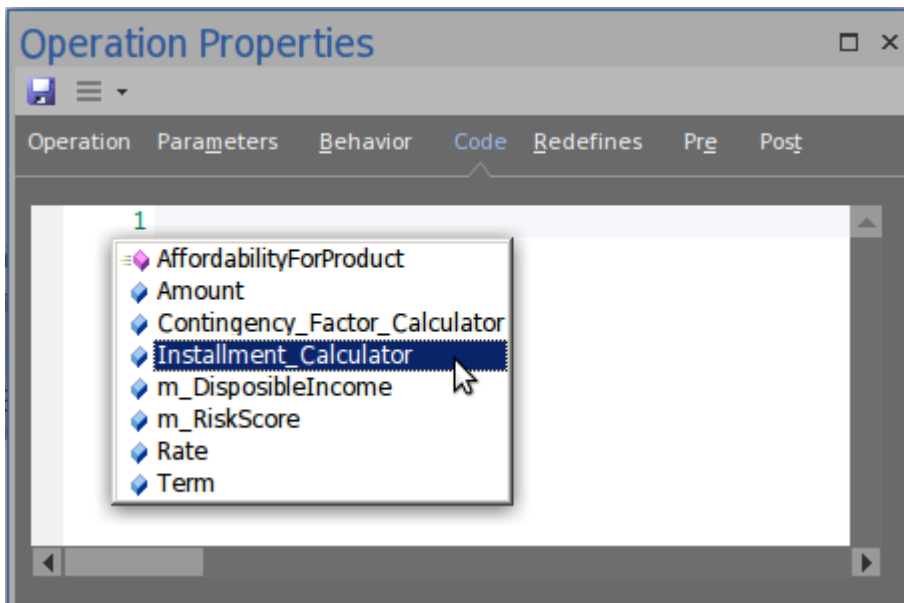
Here is what's happened after the DMN Binding:

- The operation will be be applied a stereotype "dmnBinding"
- The dependency connector is connecting to the operation
- Multiple DMN Artifacts can be bound to the same operation



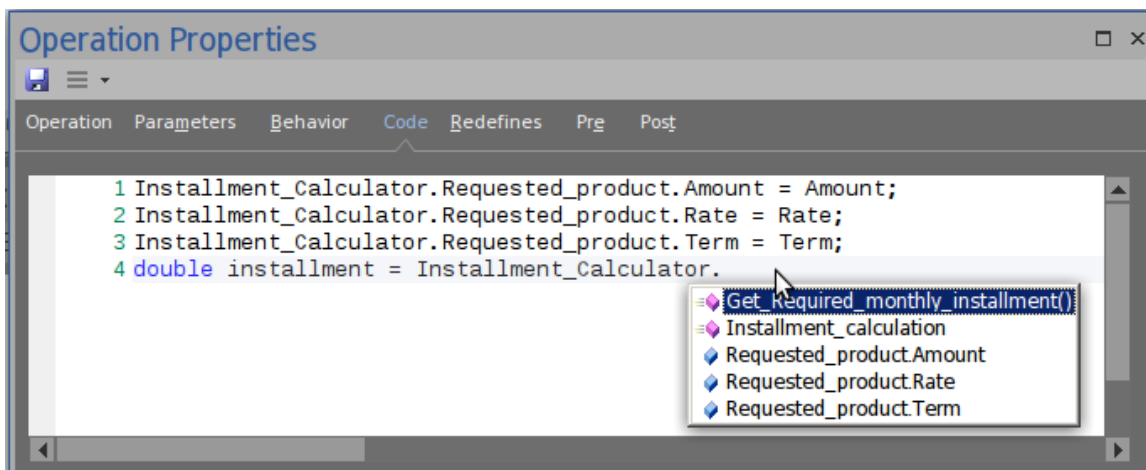
After DMN Bindings, IntelliSense for Operation's code editor will support for DMN Modules, the trigger the IntelliSense, use the following key combination:

- Ctrl + Space: For most of the cases
- Ctrl + Shift + Space: When "Ctrl + Space" does not work after a parenthesis "(". For example, function's arguments, inside if condition's parenthesis, etc.



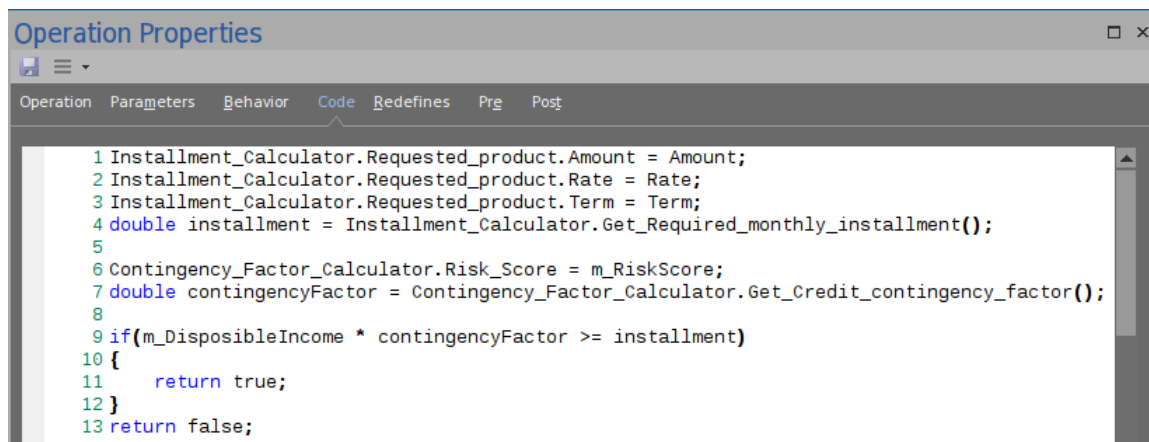
- Class attributes will be listed: m\_RiskScore, m\_DisposableIncome
- Operation Parameters will be listed: Amount, Rate, Term
- Operation will be listed: AffordabilityForProduct
- All bound DMN Modules will be listed: Contingency\_Factor\_Calculator, Installment\_Calculator

It is quite easy to compose the code with IntelliSense support. On accessing the DMN Module, all the Input Datas, Decisions, Business Knowledge Models will be listed for the user to choose.



This figure shows that we are selecting Get\_Required\_monthly\_installment() from Installment\_Calculator.

Here is the final implementation for operation.



## Code Generation for Class (With DMN Integration)

Generate Code on class Applicant, here is the code generated:

```

8 public class Applicant {
9
10     private double m_DisposableIncome = 2000;
11     private double m_RiskScore = 130;
12
13     PackageA.ContingencyFactorCalculator Contingency_Factor_Calculator = new PackageA.ContingencyFactorCalculator();
14     PackageB.InstallmentCalculator Installment_Calculator = new PackageB.InstallmentCalculator();
15
16     public boolean AffordabilityForProduct(double Amount, double Rate, double Term){
17         //WARNING: Code in this function will be overwritten when generate from EA because this operation has a flush type of stereotype
18         Installment_Calculator.Requested_product.Amount = Amount;
19         Installment_Calculator.Requested_product.Rate = Rate;
20         Installment_Calculator.Requested_product.Term = Term;
21         double installment = Installment_Calculator.Get_Required_monthly_installment();
22
23         Contingency_Factor_Calculator.Risk_Score = m_RiskScore;
24         double contingencyFactor = Contingency_Factor_Calculator.Get_Credit_contingency_factor();
25
26         if(m_DisposableIncome * contingencyFactor >= installment) {
27             return true;
28         }
29         return false;
30     }
31 } //end Applicant

```

- The DMN Module(s) are generated as attribute of class
- The dmnbinding operation's code is updated

Note: No matter the generation option is Overwrite or Synchronize, the operation's code will be updated if it has a stereotype dmnbinding.

