

Integrating Decision Tables with State Machines using Enterprise Architect

Phil Chudley – Principal Consultant

20 October 2022

Dunstan Thomas Consulting Limited is part of the Dunstan Thomas Group of companies.



dunstan thomas



Contents

С	Contents				
1 Introduction			tion	3	
	1.1	Pre	-Requisites	3	
	1.2	The	Example	3	
2	Wo	orkflov	Ν	4	
3	Wo	orked	Example	5	
	3.1	Мос	del the Business Rules as a DMN	5	
	3.2	Cre	ating Data Sets for the Decision Table	16	
	3.3	Run	nning a Simulation	20	
	3.4	Ger	nerating the Java Classes for the Decision Table	21	
	3.5	Cre	ating a Java Based Executable State Machine	24	
	3.6	Inte	grating the Decision Table and the State Machine	29	
	3.6	5.1	Add Public Attributes to the Java Class	30	
	3.6	6.2	Add the Public Operation for Invoking the DMN	31	
	3.6	6.3	Add a Public Operation to Invoke the Operation	31	
	3.6	6.4	Bind an Operation to the DMN Simulation	32	
	3.6	6.5	Add Java Code to the Bound Operation	33	
	3.6	6.6	Adding a Behaviour Operation in the State Machine	35	
	3.6	6.7	Set the Run State for the Property Instance	36	
	3.7	Run	n the Simulation	37	
	3.8	IMP	PORTANT Gotcha	40	
4	What to do Next				
5	Conclusion				



1 Introduction

Sparx Systems Enterprise Architect (EA) supports the modelling, simulation and code generation for decision tables (DMN) and state machines (ESM).

This document provides a step-by-step guide on:

- How to model DMN, simulate, and generate Java code.
- How to model ESM, simulate, and generate Java code.
- How to integrate a DMN and ESM together using a Java Class.

1.1 Pre-Requisites

- To model, simulate and integrate DMNs and ESMs the **complete** Java Development Kit (JDK) **must** be installed on the same workstation as EA. The JDK is available free of charge from Java Downloads | Oracle.
- A perspective containing the following MDGs **must** be created and active:

Technologies:
Included Technology
🛞 Basic UML 2 Technology
🛞 Class Modeling
🐼 State Machines
(S) DMN1.2
🕞 Executable State Machines

• Set the Code Engineering to Java:



1.2 The Example

For this document the example used will be a simplified car insurance quotation, where the business rules for determining the insurance premium will be modelled using a DMN, and the process of obtaining a quote will be modelled using an ESM which will invoke the DMN to obtain the premium,

The business rules (fictitious and not representative of actual insurance premiums) for this example are as detailed in the table below:

Dunstan Thomas Consulting Limited, Building 3000, Lakeside North Harbour, Portsmouth, PO6 3EN +44 (0)23 9282 2254 • info@dthomas.co.uk • www.dthomas.co.uk



Age	Car Type (Saloon, SUV, Sports)	NCD	Offences (Yes / No)	Premium
		0		500.00
	Saloon	1 to 4		450.00
		5 to 8		400.00
		0		600.00
	SUV 1 to 4 No 550.00 5 to 8 500.00 500.00 0 700.00			
5 to 8 50 0 70 Sports 1 to 4		500.00		
	700.00			
	Sports	1 to 4		650.00
17 to 25		5 to 8		600.00
17 10 25	Saloon	0		700.00
		1 to 4		650.00
		5 to 8		300.00 450.00 400.00 600.00 550.00 500.00 550.00 500.00 600.00 650.00 600.00 650.00 600.00 600.00 700.00 650.00 600.00 700.00 900.00 850.00 800.00
0	0		800.00	
	SUV	1 to 4	Yes	750.00
		5 to 8		700.00
		0		900.00
	Sports	1 to 4		850.00
		5 to 8		800.00

2 Workflow

The workflow for creating an integrated DMN and ESM consists of the following steps:

- 1) Model the business rules as a DMN model.
- 2) Test the DMN model using multiple datasets.
- 3) Generate the DMN using the Java language.
- 4) Create a Java class to integrate (via an operation) with the DMN.
- 5) Generate the Java source code for this class.
- 6) Add a State Machine to this Java Class.
- 7) Add an Executable State Machine (ESM) artefact element to the model.
- 8) Add an instance (property) of the Java class to the ESM element.
- 9) Set the run-state of this instance.
- 10) Test the State Machine.
- 11) Add an invocation to an operation defined in step 4) to at least one state in the State Machine model.
- 12) Test the State Machine.
- 13) Either:
 - a. Modify the run-state values to test a different set of business rules, and repeat for other business rules.

Dunstan Thomas Consulting Limited, Building 3000, Lakeside North Harbour, Portsmouth, PO6 3EN +44 (0)23 9282 2254 • <u>info@dthomas.co.uk</u> • <u>www.dthomas.co.uk</u>



or

b. Add multiple instances (properties), one for each set of business rules, to the ESM artefact and add multiple invocations of the operation defined in step 4) to at least one state in the State Machine Model.

3 Worked Example

In this section a worked example is presented using the business rules as presented in section 1.2

3.1 Model the Business Rules as a DMN

In EA the DMN 1.2 MDG provides multiple ways of modelling business rules, but every DMN model **must** have the following:

- The data (parameters) to be used as input to the decision table.
- The decision table.
- A DMNSimConfiguration element.

The elements above are provided in a toolbox within the DMN MDG DMNDiagram.

The following steps are required:

- 1) If necessary, create a new local EA project.
- 2) Create a repository structure like that shown below:



- 3) To the package named Business Rules (Decision Table) add a DMNDiagram
- Using the diagram toolbox add Item Definition elements such that there is one Item Definition element for each input to the Decision Table, plus one extra Item Definition element will "own" the other Item Definition elements:





NOTE

These can be named anything you like, and the names can included spaces, but I prefer to not use spaces in the names, prefix the "owning" element with **t** and name all other elements to correspond with the inputs as per the business rules defined (in this example) in section 1.2

5) **IMPORTANT** structure the elements above in the Browser to reflect the "ownership" as shown below:



- 6) These elements are **not** required on the diagram and can be deleted if you wish.
- 7) Using the diagram toolbox add an Input Data element to the diagram. I usually name this element to be the same as the "owning" Item Definition element but without the t prefix:

QuotationData			
«ItemDefinition» tQuotationData			
«ItemDefinition» Age	«ItemDefinition» CarType	«ItemDefinition» NCD	«ItemDefinition» Offences

- 8) **IMPORTANT** Bind the **Input Data** element to the "owning" Item Definition element, by:
 - a. Double-clicking the **Input Data** element to open the **DMN Expression** window:

DMN Expression				
	- 📝 🔞			
QuotationData				

b. Click the toolbar icon Set Item Definition:

DMN Expression				
	- 📝 🔞			
QuotationData				

c. Using the **Navigation Window** display, navigate to and select the "owning" Item Definition element named, in this example, **tQuotationData**:

DMN Expression					
QuotationData					
	Age	Type in Input Data Values			
tOustationData	CarType	Type in Input Data Values			
louotationData	NCD	Type in Input Data Values			
	Offences	Type in Input Data Values			



d. Click the icon Open Item Definition:

DMN Expression				
	- 2 0			
QuotationData				

This will change the **DMN Expression** window to:

DMN Expression					
tQuotationData (ItemDefinition)					
tQuotationData	Age	Type in Allowed Value Enumerations			
	CarType	Type in Allowed Value Enumerations			
	NCD	Type in Allowed Value Enumerations			
	Offences	Type in Allowed Value Enumerations			

- e. Set the **type** for each data item, by **right clicking** the data item and selecting the type from the drop down menu.
- f. For a data item that can be one of a set of values (enumeration) enter the permitted values in the last column.

DMN Expression				
tQuotationData (ItemDefinition)				
	Age : number	Type in Allowed Value Enumerations		
	CarType : string	"Saloon", "SUV", "Sports"		
QuotationData	NCD : number	Type in Allowed Value Enumerations		
	Offences : string	"No", "Yes"		

NOTE

String values **must** be surrounded by double-quotes.

- g. Save the changes, by clicking the **Diskette** toolbar icon.
- 9) In this example the decision table will be modelled using a Business Knowledge Model element. Add a Business Knowledge Model element to the diagram, select Decision Table from the menu and give an appropriate name such as Quotation Calculation. Once added to the diagram the DMN Expression window will display an empty decision table:

11	DMN	Components		-			
	08	usiness Knowledge Model					
		nowledge Source					
	O Ir	put Data					
	28 R	em Definition					
		ecision Service					
	2.1	formation Requirement					
- 1	2.0	nowledge Requirement	Oustation Calculation				
	2 A	uthority Requirement	countrol carcolation				
	E Te	ext Annotation					
	17 A	ssociation					
	🚯 SI	mulation Configuration					
	Come	100					
		Palationships					
	Comin	ion keadonships					
	Artifa	cts					
	• • •						
C	DMN Expression						
	3 2	- 44 - 田 - 田 - 岡太太区 - 三〇〇	- b + 😰				
	0.00	tation Calculation Input Parameter Values for Simula	tion				
- If	dan						
	(Input 1, Input 2)						
		Input 1	Input 2	Output 1			
	_	-					
		1	-				
		2 .					
		3	and the second se				
10							

Dunstan Thomas Consulting Limited, Building 3000, Lakeside North Harbour, Portsmouth, PO6 3EN +44 (0)23 9282 2254 • info@dthomas.co.uk • www.dthomas.co.uk



a. Using the Add Input toolbar icon add two more input columns:



b. Using the Edit Parameters toolbar icon, set the parameters these names must be the same as the "owned" Item Definition elements created earlier as shown below:

DMN Expression □ × · 2↓ · □ = = = • □ · □ · □ · □ · □ · □ · □ · □ · □ · □						
Parameter	Туре					
Age	number					
CarType	string					
NCD	number					
Offences	string					
New Parameter						
	Close					

- c. Change the names of the input / output columns to correspond to these input parameters (as you begin to type in an **input column**, you will see EA present a suggested name which you can then select), for an **output column** just type a name.
- d. Right-click in the columns named **CarType** and **Offences** and select the type **string** from the drop-down list
- e. For **string enumerations** (CarType and Offences) in our example, enter these values below (as you start to type you should see a suggestion from EA, which you should select) in section below the input column header:

Dunstan Thomas Consulting Limited, Building 3000, Lakeside North Harbour, Portsmouth, PO6 3EN +44 (0)23 9282 2254 • info@dthomas.co.uk • www.dthomas.co.uk



D

DM	DMN Expression 🔹 🔹 🗧							
E	╕_┝☆・2↓・ 田・田・ 図 牀 牀 2 峠 8 ☆ ♪・ @							
	Quotatio	on Calculation Input Parameter Values f	or Simulation					
				(Age, CarType, NCD, Offences)				
		Age	CarType	NCD	Offences	Premium		
			"Saloon", "SUV", "Sports"		"No", "Yes"			
•	1	-	-	-	-	-		
	2	-	-	-	-	-		
	3	-	-	-	-	-		

Now the laborious part, using the Business Rules defined in section 1.2 f. complete each row of the decision table. Extra rows can be added by selecting the Add Rule toolbar icon:

DMN Expression									
Image: Second state Image: Second state<									
DM	DMN Expression								
				(Age, CarType, NCD, Offences)					
	U	Age	CarType	NCD	Offences	Premium			
			Saloon, SUV, Sports		No, Yes				
	1	[1725]	Saloon	0	No	500.00			
	2	[1725]	Saloon	[14]	No	450.00			
	3	[1725]	Saloon	[58]	No	400.00			
	4	[1725]	SUV	0	No	600.00			
	5	[1725]	SUV	[14]	No	550.00			
	6	[1725]	SUV	[58]	No	500.00			
	7	[1725]	Sports	0	No	700.00			
	8	[1725]	Sports	[14]	No	650.00			
	9	[1725]	Sports	[58]	No	600.00			
	10	[1725]	Saloon	0	Yes	700.00			
	11	[1725]	Saloon	[14]	Yes	650.00			
	12	[17.25]	Saloon	[58]	Yes	600.00			
	13	[1725]	SUV	0	Yes	800.00			
	14	[1725]	SUV	[14]	Yes	750.00			
	15	[1725]	SUV	[58]	Yes	700.00			
	16	[1725]	Sports	0	Yes	900.00			
	17	[1725]	Sports	[14]	Yes	850.00			
•	18	[1725]	Sports	[58]	Yes	800.00			

- g. Save the changes by clicking the **Diskette** toolbar icon.
- h. Optionally, you can merge cells by selecting Merge All from the menu displayed when selecting the Merge Cells toolbar icon. This can be reversed (unmerged) by selecting **Unmerge All** from the menu displayed by selecting the next toolbar icon (to the right) of the one shown below, so you can restore to non-merged cells if you wish.



Dunstan Thomas Consulting Limited, Building 3000, Lakeside North Harbour, Portsmouth, PO6 3EN +44 (0)23 9282 2254 • info@dthomas.co.uk • www.dthomas.co.uk



	V Exp - 2 uotatio	oression ↓ - 田 - 田 - ℗ ポロ ポム 図 =€ n Calculation ⊂ Input Parameter Values f	😨 😪 ▷ 🚽 🔞			↓ ‡ :
				(Age CarType NCD Offences)		
-		Age	CarType	NCD	Offences	Premium
	U		Saloon, SUV, Sports		No, Yes	
	1			0	No	500.00
	2		Saloon	[14]	No	450.00
	3			[58]	No	400.00
	4			0	No	600.00
	5		SUV	[14]	No	550.00
	6	[1725]		[58]	No	500.00
	7			0	No	700.00
	8		Sports	[14]	No	650.00
	9			[58]	No	600.00
	10		Selece	0	Yes	700.00
	11		5010071	[14]	Yes	650.00
	12	[17.25]	Saloon	[58]	Yes	600.00
	13			0	Yes	800.00
	14	SUV	[14]	Yes	750.00	
	15	(17. 25)		[58]	Yes	700.00
	16	[1725]		0	Yes	900.00
	17		Sports	[14]	Yes	850.00
•	18			[58]	Yes	800.00

10) Add a **Decision Element** and choose **Invocation** from the menu from the toolbox to the diagram, in this example a suitable name is **Quotation**.



- 11) Using the Quicklink create an **Information Requirement** relationship from the **Input Data** element named **QuotationData** to the newly created **Decision Element**.
- 12) Similarly using the Quicklink create a **Knowledge Requirement** relationship from the **Business Knowledge Model** element named **Quotation Calculation** to the newly created **Decision Element**.



Dunstan Thomas Consulting Limited, Building 3000, Lakeside North Harbour, Portsmouth, PO6 3EN +44 (0)23 9282 2254 • <u>info@dthomas.co.uk</u> • <u>www.dthomas.co.uk</u>



- 13) **IMPORTANT** Bind together the **Decision** element and the **Business Knowledge Model** element by:
 - a. Double-click the **Decision** element named **Quotation** to open the **DMN Expression** window.
 - b. Click the Set Business Knowledge icon

	MN Expression Image: Constant state of the second state
Select Element	×
Browse	Go To Namespace: <any></any>
Search	MINESM Integration Example 1 Car Insurance Quotation Business Rules (Decision Table) SubsinessKnowledgeModel*Quotation Calculation <none></none>
Types : 'Activity' Stereotypes : BusinessKnowle	dgeModel,DecisionService
	OK Cancel Help

- c. If necessary, navigate to and select the **BusinessKnowlegdeModel** in this example it is named **Quotation Calculation**.
- d. Click OK

DMI	DMN Expression					
	R 💠 🖨 R 🗄 🛛 - 🛛					
	uotation					
	Quotation Calculation . Premium					
	Age : number					
	CarType : string					
	NCD : number					
	Offences : string					

- 14) **IMPORTANT** Bind together the **Decision** element and the **Input Data** element by:
 - a. Click in the text area adjacent to one of the input parameters listed and select **Edit Expression...**



Edit DM	IN Expressio	n							×
FEEL	JavaScript	Java	C#	C++					
	1								
	Copy Expres	ssion To) All Li	anguages					•
Display	in Language	: [-	ОК	Cancel	Help	

NOTE

The tab named **FEEL** is used for direct simulation and testing of the DMN model within EA. The remaining tabs are completed if the DMN is to be generated to code, for example **Java**.

- b. The syntax for the FEEL entry is InputData element name . Parameter Name
- c. The syntax for Java is InputData element name.Parameter name

DMN Expression							
Quotation							
Quotation Calculation . Premium							
Age : number	QuotationData . Age						
CarType : string	QuotationData . CarType						
NCD : number	QuotationData . NCD						
Offences : string	QuotationData . Offences						

15) Validate the Decision Table by:

- a. Selecting the **BusinessKnowledgeModel** element named **Quotation Calculation**.
- b. Clicking the Validation icon in the toolbar:





c. The decision table will be validated, and the results shown in the **System Output** tab:

System Output System Script DMN Validation Running Quotation Calculation Validations ... Validating BusinessKnowledgeModel 'Quotation Calculation' ... Warning : DecisionTable "Quotation Calculation" Completeness Violation: No rule exists for Age:"<17", *</td> Warning : DecisionTable "Quotation Calculation" Completeness Violation: No rule exists for Age:"<25", *</td>

Quotation Calculation Results: (0) error(s), (2)warning(s)

NOTE

In this example the validation provides two warnings regarding possible missing rules. In this example these missing rules are not important, but in other examples you may need to add the missing rules.

Other validation errors are syntax errors, for example entering [17.25] instead of [17.25] result in an **error**.

Warnings can be ignored. Errors **must** be corrected.

16) Using the diagram toolbox add a **Simulation Configuration** element to the diagram and give this element a suitable name, for example, **Quotation Calculation**.





- 17) Bind this **Simulation Configuration** element to the **Package** containing the elements created so far:
 - a. Double-click the Simulation Configuration element
 - b. Click the Configure Package toolbar icon:

DMN Simulation					
🛅 🖡 🤣 🖹 Required monthly instal 👻 😰					
Simulate Generate Module Test Module					
▶ - Ç≣ □ ‡] - @					

c. Select the Package (in this example the Package named **Business Rules** (Decision Table) in the Select Package dialog:



Select DMN elements Packag	je.	
Browse	Go To Namespace: <any></any>	Ŧ
Search	 ▲ DMN ESM Integration ▲ Car Insurance Quotation → Business Rules (Decision Table) ↓ → State Machine <none></none> 	
Types : 'Package'		
	OK Cancel He	lp

- d. Click OK.
- e. Select **Quotation** (the name of the **Decision** element) from the drop-down list in the DMN Simulation window:

DMN Simulation	
E F 2 Quotation V 💿	
Simulate Generate Module Test Module	
Element	Value
O QuotationData	Default
1 Quotation	

f. The entry for the Input data element named Quotation is shown with the value Default. This refers to the Data Set that will be used to run the simulation. Each Input Data element has a Default data set which cannot be deleted but can be copied to create multiple data sets for simulation. This process is described in the next section.

3.2 Creating Data Sets for the Decision Table

Before generating any code for the decision table, it should be tested to ensure all is working correctly.

A simulation requires an input data set that belongs to the **Input Data** element.

The recommended approach is:

- **Copy** the Default data set
- Assign values to **all** input parameters that will test a rule.
- Repeat this process to test as many rules as required.

In this example we will create the data sets (the data set name corresponds to the rule number in the decision table that is to be tested) as detailed in the table below:



Data Set Name Age		CarType	NCD	Offences	Expected Result
Rule 2	20	Saloon	2	No	450.00
Rule 6	Rule 6 20 SUV		6	No	500.00
Rule 7	20	Sports	0	No	700.00
Rule 11 20		Saloon	2	Yes	650.00
Rule 15	20	SUV	6	Yes	700.00
Rule 16 20 Sports		Sports	0	Yes	900.00

To create a Data Set for each row in the table above:

- 1) Double-click the Input Data element named QuotationData
- 2) Click the **Edit Data Set** toolbar icon:

DMN Expression						
R 🕫 🖬 😰 🔄 🔽 🕐						
QuotationData : tQuotat	ionData					
Edit Data Set For DMN ItemDefinition		×				
1 X 타 R V N N N N						
Data Set / Item	Туре	Value				
⊿ Default						
tQuotationData . Age	number					
tQuotationData . CarType	string					
tQuotationData . NCD	number					
tQuotationData . Offences	string					
		ОК				

- 3) Copy the **Default** data set by:
 - a. Selecting the Default data set.
 - b. Click the **Duplicate** icon in the toolbar:





4) Name the data set and enter values for each input parameter using the table above:

Edit Data Set For DMN ItemDefinition					
🏠 🗙 🗊 🗐 🗶 🖪 🏹					
ItemDefinition: tQuotationData					
Data Set / Item	Туре	Value			
⊿ Default					
tQuotationData . Age	number				
tQuotationData . CarType	string				
tQuotationData . NCD	number				
tQuotationData . Offences	string				
▲ Rule 2					
tQuotationData . Age	number	20			
tQuotationData . CarType	string	"Saloon"			
tQuotationData . NCD	number	2			
tQuotationData . Offences	string	"No"			
		ОК			

Dunstan Thomas Consulting Limited, Building 3000, Lakeside North Harbour, Portsmouth, PO6 3EN +44 (0)23 9282 2254 • <u>info@dthomas.co.uk</u> • <u>www.dthomas.co.uk</u>



NOTE

If there has been a selection defined, for example, the CarType, then a value can be selected from a drop-down list.

5) Repeat steps 4) and 5) for each row of the table above. You can either copy the Default data set **or** a data set that you have created.

dit Data Set For DMN ItemDefinition		\$
) 🗙 🗈 🖪 🗶 🖪 🖬 🖉		
emDefinition: tQuotationData		· · · · · · · · · · · · · · · · · · ·
Data Set / Item	Туре	Value
⊿ Default		
tQuotationData . Age	number	
tQuotationData . CarType	string	
tQuotationData . NCD	number	
tQuotationData . Offences	string	
▲ Rule 2		
tQuotationData . Age	number	20
tQuotationData . CarType	string	"Saloon"
tQuotationData . NCD	number	2
tQuotationData . Offences	string	"No"
▲ Rule 6		
tQuotationData . Age	number	20
tQuotationData . CarType	string	"SUV"
tQuotationData . NCD	number	6
tQuotationData . Offences	string	"No"
⊿ Rule 7		
tQuotationData . Age	number	20
tQuotationData . CarType	string	"Sports"
tQuotationData . NCD	number	0
tQuotationData . Offences	string	"No"
▲ Rule 11		
tQuotationData . Age	number	20
tQuotationData . CarType	string	"Saloon"
tQuotationData . NCD	number	2
tQuotationData . Offences	string	"Yes"
▲ Rule 15		
tQuotationData . Age	number	20
tQuotationData . CarType	string	"SUV"
tQuotationData . NCD	number	6
tQuotationData . Offences	string	"Yes"
▲ Rule 16		
tQuotationData . Age	number	20
tQuotationData . CarType	string	"Sports"
tQuotationData . NCD	number	0
tQuotationData . Offences	string	"Yes"
		01/
		OK

Dunstan Thomas Consulting Limited, Building 3000, Lakeside North Harbour, Portsmouth, PO6 3EN +44 (0)23 9282 2254 • <u>info@dthomas.co.uk</u> • <u>www.dthomas.co.uk</u>



6) Click **OK**

3.3 Running a Simulation

Now that we have created the Decision table, bound all the elements together and defined input data sets, we can run a simulation to test the decision table.

To run a simulation, use the following steps:

- 1) Decide which data set to use for the simulation, for example Rule 2.
- 2) Note the expected value, in this example 450.
- 3) Double-click the Simulation Configuration element.
- 4) Select Rule 2 from the drop-down in the DMN Simulation window:



5) Click the **Simulation** toolbar icon:

DMN Simulation	▼ ₽ ×	
🛅 🛱 🤣 🔐 Quotation 👻 🔞		
Simulate Generate Module Test Module		
↓ ↓1 + ()		
Liement	Value	
O QuotationData	Rule 2	
[*] Quotation		

6) The simulation will run, and the results shown:



We can verify the correct result by observing the **red** text adjacent to each element. The text adjacent to the **Input Data** element is the data set used in the simulation, the other red text

Dunstan Thomas Consulting Limited, Building 3000, Lakeside North Harbour, Portsmouth, PO6 3EN +44 (0)23 9282 2254 • info@dthomas.co.uk • www.dthomas.co.uk



is the result after evaluating the decision table rules. Comparing the result against the expected result, we can verify that the decision table is correct.

7) End the simulation by clicking the **Stop Simulation** icon in the toolbar:

DMN Simulation	↓ ‡ >
E R 2 V Quotation · O	
Simulate Generate Module Test Module	
↓ ↓ □ ↓ 1 · ↓ ②	
Element	Value
O QuotationData	Rule 2
t ¹ Quotation	

 Repeat the steps above for each data set to verify that each data set produces the expected result.

3.4 Generating the Java Classes for the Decision Table

Once the decision table has been tested using simulation and data sets, Java code can be generated using the following steps:

- 1) Double-click the Simulation Configuration element named Quotation Calculation
- 2) Click the section named Generate Module
- 3) Select Java as the language from the drop-down list:

1			
DMN Simulatio	n		
🖿 🗏 🍫 🕃 Quotat	ion 👻 🔞		
Simulate Generate Mo	dule Test Module		
B¥ ▼ Java	• 🗗 🖹 🗟 🛠 🖉		
Module Path:			
Package and Name:	com.sparxsystems.dmn.Quotation_Module		
Elements			
📑 tQuotationData			
Quotation Calc	ulation		
⊿ 😁 Global Decision Service			
🔵 QuotationD	ata		
Quotation			

4) Note the Package and Name. This follows typical Java reverse domain name convention, and the Module Path MUST refer to a file structure that matches this reverse domain name. Therefore, use Windows File Explorer to create folders to reflect this structure. Although these folders could be created anywhere, I usually create them as child folders within the folder where my EA repository is located:





5) Using the ... navigate button, set the **Module Path** to the **dmn** folder created above:

DMN Simulatio	n
🖿 🗏 🍫 🛃 Quota	tion 👻 💿
Simulate Generate Mo	odule Test Module
B≸ ▼ Java	
Module Path:	C:\EA-Projects\DMN & ESM Integration\com\sparxsystems\dmn\Quotation_Module.java
Package and Name:	com.sparxsystems.dmn.Quotation_Module
Elements	

6) Generate the code using the **Generate Code** icon in the toolbar:

DMN Simulatio	n
🖿 🗏 🍫 🛃 Quota	tion 👻 🔞
Simulate Generate Mo	odule Test Module
💱 🔻 Java	
Module Path:	C:\EA-Projects\DMN & ESM Integration\com\sparxsystems\dmn\Quotation_Module.java
Package and Name:	com.sparxsystems.dmn.Quotation_Module
Elements	

7) You may see the following error message in the **System Output** window:



8) If you do, then follow the instructions and define **JAVA_HOME** to refer to the package where you have installed the JDK. For example:



Local Paths				×
Path:	C:\Program Files\Ja\	va\jdk-18.0.1.1		
ID:	JAVA_HOME		Apply Path	Expand Path
Type:	Java	v		
Relative Pa	aths		New Save	Delete
Туре	ID	Path		
Java	JAVA_HO	C:\Program Files\Java\jdk-18.0.1.1		
•				
2			Close	Help

9) Repeat step 6) and check in the **System Output** window that all generated and compiled OK:

System Out	put				
System Script	DMN Validation DM	N Build			: •
Generating DMN	Module to file: 'C:\EA	-Projects\DMN & ES	M Integration\com	\sparxsystems\dmn\	Quotation_Module.java'
The DMN module	e is successfully gene	ated.			
Generating Analy	zer Script for DMN M	odule			
Analyzer Script fo	r DMN Module is suc	cessfully updated.			
Compiling DMN	Module				
DMN Module is s	uccessfully compiled				

EA Gotcha and TIP

If you get any build errors, these are likely due to typos when entering the Java expression when binding the **Decision** element to the Input Data. (Section 3.1 step 14).

You can correct this of course, **BUT gotcha** you still get build errors! The changes have **not** been applied when the decision table code is generated.

The work around, is **save all diagrams**, close the repository and then re-open the repository.

That should solve the problem, and all should build OK.

This has caught me out several times.

If you are curious you can examine he source code generated, not shown here, by clicking the **View Code** icon in the toolbar.

Dunstan Thomas Consulting Limited, Building 3000, Lakeside North Harbour, Portsmouth, PO6 3EN +44 (0)23 9282 2254 • info@dthomas.co.uk • www.dthomas.co.uk



DMN Simulatio	on
🖿 🗏 🍫 🕃 Quota	ition 👻 🔞
Simulate Generate Mo	odule Test Module
🐺 🔻 Java	
Module Path:	C:\EA-Projects\DMN & ESM Integration\com\sparxsystems\dmn\Quotation_Module.java
Package and Name:	com.sparxsystems.dmn.Quotation_Module
Elements	

3.5 Creating a Java Based Executable State Machine

The next step is to create a simple Java based executable state machine (ESM).

The steps involved should be familiar especially if you have model state machines in EA before.

For the example in this document, we do not require a very complex state machine and the following steps are used:

- 1) Create a Package named **Class and State Machine** using the Package named **Car Insurance Quotation** as the Parent.
- 2) Add a **UML Class** diagram to this Package and choose a suitable name for the diagram, for example, **Java Class**
- 3) Using the diagram toolbox and a Java class (a Class with its language property set to Java) to the diagram. Enter a suitable name for the class (do not put spaces in the class name). A suitable name in this example is Driver. At this point, no attributes or operations are required. These will be created in the section 3.6



- 4) Right-click the class in the Browser and choose Add -> State Machine from the menu. The default name of State Machine is fine for this example. The child state machine will be created and the diagram for the state machine will be opened.
- 5) Create the following simple state machine:





We now need to make this state machine a Java based ESM. The following steps are used:

- 1) In the package named Class and State Machine create a UML Class Diagram named State Machine Simulation.
- 2) Using the **Artefacts** page in the diagram toolbox add to this diagram an **Executable StateMachine Artefact**. Name this element **QuotationApplication**:



3) Drag the class named **Driver** onto this **Executable StateMachine Artefact** element and drop as a **Property**. **Do NOT drop as a link**. Name this property **Example1**:



We can Build and Run the state machine to simulate it and ensure all the transitions work correctly between the various states. The following steps are used:

- 1) Select the Executable StateMachine artefact.
- Select Generate, Build and Run from the Statemachine drop-down menu, in the Executable States section of the Simulate ribbon. The following dialog will be displayed:



Executable Statemachi	ne Code Genera	ation		×
Artifact:			Language:	
QuotationApplication			Java	
Project output directory	:			
1				
Class Name	File Path			
ConsoleManager	\ConsoleMana	ager.java		
ContextManager	\ContextMana	iger.java		
Driver	\Driver.java			
EventProxy	\EventProxy.ja	va		
SimulationMana	\SimulationM	anager.java		
Executable Statemach	ine Target Machi	ne		
	O Remote			
() Local	Onemote			
Local Path: Lo	cation of Java JD)K installation din	ectory.	
JAVA_HOME	: \Program Files \J	ava\jdk-17.0.2		
Compilation Option	s: O 32bit	• 64bit		
	Generate	Cancel	Help	

4) Using the navigate button ... navigate to and select the **same parent** folder as where the generated DMN code is located:

dunstan thomas

Executable Statemachi	ne Code Generation		×
Artifact:		Language:	
QuotationApplication		Java	
Project output directory	:		
C: \EA-Projects \DMN	& ESM Integration		
Class Name	File Path		
ConsoleManager	C:\EA-Projects\DMN &	ESM Integration\Co	
ContextManager	C:\EA-Projects\DMN &	ESM Integration\Co	
Driver	C:\EA-Projects\DMN &	ESM Integration\Dri	
EventProxy	C:\EA-Projects\DMN &	ESM Integration\Eve	
SimulationMana	C:\EA-Projects\DMN &	ESM Integration\Sim	
Everytable Statemach	ing Taract Maching		
Executable Statemach	ine Target Machine		
 Local 	⊖ Remote		
Local Path: Lo	cation of Java JDK installatio	on directory.	
JAVA_HOME	:\Program Files\Java\jdk-18	3.0.1.1	
Compilation Option	s: 032bit 💿 64bit		
	Generate Canc	el Help	

5) Click **Generate** the Java code will be generated, built and the state machine simulation started:





NOTE

The name of the Java class property instance is shown in red next to the **Initialising** state.

6) Select **Open Simulation Window** from the **Simulation** drop-down menu, in the **Dynamic Simulation** section of the **Simulate** ribbon:



NOTE

The Simulation window displays **Waiting for Trigger**, **BUT** unlike simulating a State Machine using Javascript, a list of Triggers is **NOT** displayed. This can be verified by selecting **Events** in the **Dynamic Simulation** section of the **Simulate** ribbon.

How to fire a trigger in a Java based ESM simulation puzzled me for a long time until I found the solution. Which is:

In the Simulation Window there is text field below the toolbar. To fire a trigger, enter

broadcast tigger name or

send trigger name to instance name

and press Enter

This will fire the trigger, repeat this process to test all transitions.

Dunstan Thomas Consulting Limited, Building 3000, Lakeside North Harbour, Portsmouth, PO6 3EN +44 (0)23 9282 2254 • <u>info@dthomas.co.uk</u> • <u>www.dthomas.co.uk</u>



- stm StateMachine
- 7) In the Simulation Window enter the command broadcast GO

The state machine has transitioned correctly to state Applying.

8) In the **Simulation Window** enter the command **broadcast BACK** to return to the state **Initialising**.



9) Finally, to end the simulation test, In the **Simulation Window** enter the command **broadcast QUIT** to end the simulation.

TIP

You can use the **up and down** arrow keys on the keyboard to scroll through the broadcast commands that have been entered during this simulation.

3.6 Integrating the Decision Table and the State Machine

Now that we have a working Decision Table in Java and an ESM also in Java, we can integrate them using the following workflow:

Dunstan Thomas Consulting Limited, Building 3000, Lakeside North Harbour, Portsmouth, PO6 3EN +44 (0)23 9282 2254 • info@dthomas.co.uk • www.dthomas.co.uk



- Add **public** attributes to the Java class that correspond with the input data within the data sets used to test the decision table, plus **public** attribute(s) that correspond to the result(s) returned from the decision table.
- 2) Create a **public** operation to the Java class which will call an operation that has been generated within the Decision table Java classes. This operation will return the result from the decision table.
- 3) Create a **public** operation that will call the operation defined in 2) above and set the result in an attribute defined in 1) above.
- 4) Add code to this operation to call the operation defined in 2) above and store the returned value in a suitable attribute,
- 5) Bind operation created in step 2) to **DMN Simulation** element(s).
- 6) Add Java code to this operation, that will:
 - a. Set data in the Java code for the Decision Table
 - b. Call the operation defined in the Java code for the decision table
 - c. Return the result from the decision table
- 7) Add a call to the operation defined in step 4) above, within a state **do** behaviour.
- 8) Set the **run state** of the property instance for the class to be the values in **one** of the input data sets used to simulate the decision table.
- 9) Ensure the Local Variables window is visible.
- 10) Generate, Build and Run the ESM, fire triggers and observe the values in the local variables.

3.6.1 Add Public Attributes to the Java Class

First an explanation as to why the attributes are defined as **public** rather than **private**. The answer is simple, we need **public** attributes so that the run state of the Java property instance can be set. If **private** attributes are used, the Java class for the state machine simulation will fail to compile.

Attributes are added in the usual manner using the **Features** window. I have found that the following type mapping works between the Decision table and the Java types:

Decision table type Number - maps to Java double type

Decision table type String - maps to Java String type

NOTE

I have not tried other decision table types, for example **Boolean**, but I would guess this would map to the Java type **bool**. Similarly for the other decision table types.

From our decision table we require Java attributes for:

- Age
- Car Type
- NCD
- Offences
- Premium



	Driver
+	age: double
+	carType: String
+	ncd: double
+	offences: String
+	premium: double

3.6.2 Add the Public Operation for Invoking the DMN

Any name can be used, but the operation **must** have parameters that correspond to the data inputs for the decision table and return a value which corresponds to the result of the decision table.

	Driver
+	age: double
+	carType: String
+	ncd: double
+	offences: String
+	premium: double
+	ApplyForInsurance(age: double, carType: String, ncd: double, offenœs: String): double

3.6.3 Add a Public Operation to Invoke the Operation

The operation can be added in the usual manner, and once created:

- 1) Open the **Properties** window for this operation
- 2) Select Code
- 3) Enter the code for the operation, for example:

```
premium = ApplyForInsurance(age, carType, ncd, offences);
```

Operation		
Operation	1 premium = ApplyForInsurance(age, carType, ncd, offences);	
Parameters		
Code		
Behavior		
Redefines		
Pre		
Post		
Tagged Values		
		•
	Code Editor Save	э
	Close	eln

4) Click Save

Dunstan Thomas Consulting Limited, Building 3000, Lakeside North Harbour, Portsmouth, PO6 3EN +44 (0)23 9282 2254 • <u>info@dthomas.co.uk</u> • <u>www.dthomas.co.uk</u>



	Driver
+ + + + +	age: double carType: String ncd: double offences: String premium: double
+ +	ApplyForInsuarnce(age: double, carType: String, ncd: double, offences: String): String GetAQuote(): void

3.6.4 Bind an Operation to the DMN Simulation

This is the heart of the integration and is performed using the following steps:

1) Re-use as a **Link** the **DMNSimConfiguration** element from the Decision Table model on the Java Class diagram:

	Driver
+ +	age: double carType: String
+ + +	ncd: double offences: String premium: double
+ +	ApplyForInsuarnce(age: double, carType: String, ncd: double, offences: String): String GetAQuote(): void
_	

	«DMNSimConfiguration» Quotation Calculation	
--	--	--

2) Using the Quicklink create a **Dependency** relationship from the Java Class to the **DMNSimConfiguration** element. The following dialog will be displayed:

Link to Element F	eature		×
Target Element:	Driver		
Feature Type:	Operation	T	
Feature:	Feature	Description	
	ApplyForInsuranc GetAQuote()	double void	
	OK Can	cel Help	

- 3) Select the operation that will invoke the DMN code, in this example, **ApplyForInsuranceQuote** in the list of Features.
- 4) Click **OK**. This will **bind** the selected operation to the DMN simulation:



Driver		
+ age: double + carType: String + ncd: double + offences: String + premium: double		«DMNSimConfiguration» Quotation Calculation
 GetAQuote(): void «dmnBinding» ApplyForInsurance(age: double, carType: String, ncd: double, offenœs: String): double 	an and a second s	

3.6.5 Add Java Code to the Bound Operation

The code entered for this operation, in theory, can be any valid Java code. But **at least** the following **must** be included:

- The values in each parameter **must** be stored in variables declared within the Java classes generated for the decision table.
- The correct method within the Java classes generated for the decision table **must** be called.

The challenge is what classes, what variables and what method must be used?

According to the Sparx documentation, all of these is available using **code complete**. Whilst this may be true for Javascript based ESMs, (I haven't verified this), it is only **partially** true for Java based ESMs.

I have found the following steps work:

- 1) Select the operation that is bound to the **DMNSimConfiguration** element and open its Properties window.
- 2) Select Code.
- Enter assignment statements, one per operation parameter using the following syntax:

Java Class for the DMNSimConfiguration.Java Class Name for the Data Input Element.Data Item Name.

An example should make more sense of this:

Referring to the models created so far:

In our example the name of the DMNSimConfiguration element is **Quotation Calculation**, so the Java class name is **Quotation_Calculation**. This **is** available via code complete.

In our example the name of the Data Input element is **QuotationData**, so the Java class name is **QuotationData**. This **is not** available via code complete.

The names of the Data Items are Age CarType, NCD and Offences. These are the names used in the Java class. These are **not** available via code complete.

Hence the statements to store the parameter data from the operation to these Data Items are:

```
Quotation_Calculation.QuotationData.Age = age;
Quotation_Calculation.QuotationData.CarType = carType;
Quotation_Calculation.QuotationData.NCD = ncd;
```



Quotation_Calculation.QuotationData.Offences = offences;

Operation		×
Operation Parameters Code	<pre>1 Quotation_Calculation.QuotationData.Age = age; 2 Quotation_Calculation.QuotationData.CarType = carType; 3 Quotation_Calculation_QuotationData.NCD = ncd; 4 Quotation_Calculation_QuotationData.PCD</pre>	
Behavior Redefines Pre Post Tagged Values	+ Quotation_Laiculation.QuotationData.orrences = orrences;	
	Code Editor	Save
	Close	Help

TIP

To use code complete press Ctrl + Space and you will see a list of valid entries. Once a class is selected from this list, entering . will display a valid list of operations. (Usually only one since a decision table usually returns one value. I assume for decision tables that return multiple values, there will be one operation for each result returned).

In our example we simply need to call the operation that returns the result from the Decision Table. This **is** available using code complete. The completed code for the operation is:

return Quotation_Calculation.Get_Quotation();

Operation		
Operation Parameters Code	<pre>1 Quotation_Calculation.QuotationData.Age = age; 2 Quotation_Calculation.QuotationData.CarType = carType; 3 Quotation_Calculation.QuotationData.NCD = ncd;</pre>	
Behavior Redefines	<pre>4 Quotation_Calculation.QuotationData.Offences = offences; 5 6 return Quotation_Calculation.Get_Quotation();</pre>	
Pre Post		
ragged Values		
	Code Editor	Save
	Close	Help

Do not forget to Save the code



3.6.6 Adding a Behaviour Operation in the State Machine

The next piece of the DMN / ESM integration is to add a behaviour operation, for example **do** to a state.

In our state machine:



We will add a **do** behaviour to the state **Applying** using the following steps

- 1) Select the state Applying
- 2) Right-click and select **Features -> Operations...** from the menu:

Features		
Behavior Internal	Triggers Connection Points	
Туре		Name / Comment
ſ .		
entry		Name / Comment
entry do		Name / Comment Name / Comment

- 3) Enter a name for the **do** operation, for example **Apply**
- 4) Select this operation and open the Properties window and select Code
- 5) Enter the code that will call the method named **GetAQuote()** created earlier in our Java class that owns the state machine as shown below:

this.GetAQuote();

Operation		
Operation	1 this.GetAQuote();	
Parameters		
Code		
Behavior		
Redefines		
Pre		
Post		
Tagged Values		
		•
	Code Editor Sa	ve
	Close	Help
		-

Dunstan Thomas Consulting Limited, Building 3000, Lakeside North Harbour, Portsmouth, PO6 3EN +44 (0)23 9282 2254 • <u>info@dthomas.co.uk</u> • <u>www.dthomas.co.uk</u>



NOTE

The keyword this **must** be used to invoke the method. This refers to the Property instance added to the **executable statemachine** element.

3.6.7 Set the Run State for the Property Instance

Our state machine simulation needs data, just as did our decision table simulation. This data is created by setting the **run state** of the Property instance of the Java class added to the **executable statemachine** element.

The following steps are used:

 Right-click the Property Instance within the executable statemachine element and select Features -> Set Property Values... from the menu:

Example1 : Features					×
Set Property Values	Variable	Operator	Value		
	age				
	carType				
	ncd				
	offences				
	premium				
	New Variable				
				Close He	p

 All the attributes, including the attribute for the result are listed. Using the = operator set values for age, carType, ncd and offences. A good set of values would be the same as those used in a data set used for simulating the decision table. The value of premium should be 0.



Set Property Values				
set Property values	Variable	Operator	Value	
	age	=	20	
	carType	=	"Saloon"	
	ncd	=	2	
	offences	=	"No"	
	premium	=	0	
	1			
	1			
	1			
	1			
	[
	[

3) Click Close



3.7 Run the Simulation

Ensure the following windows are visible (they can be accessed using the **Simulate** ribbon):

- Simulation Window
- Local Variables

Run the simulation as before by:

- 1) Select the **executable statemachine** element on the diagram named **State Machine Simulation**.
- 2) Select **Generate, Build and Run** from **Statemachine** drop-down menu. Assuming there are no build errors, (if errors are present, these are usually typos which can be corrected) then the state machine will begin to execute:





The local variables are:

Locals			→ ∓ ×
Variable		Value	Туре
🔄 🧅 thi	s		SimulationManager
	ContextMai		ContextManager
•	activeState(1	int
A 9	Example1		Driver
	🖕 carType	"Saloon"	String
	🖕 age	20	double
	🧄 ncd	2	double
	premiur	0	double
	offence:	"No"	String
►	🖕 Quotatii		com.sparxsystems.dr
•	🔷 m_State		StateMachinelmpl
	🔶 m_slnsti	"Example1"	String
	🐞 m_sΤγpε	"Driver"	String
	m_state	Driver_ENUM_STATEN	Driver\$StateEnum
	🔶 m_state	NOSTATE	Driver\$StateEnum
- b Q	m_pComm		java.lang.StringBuilde
	m_commar		java.util.Queue
• •	activeStates	Example1,{BF3C9EB3-	String[1024]

Expand the entry named **Example1** and note the values of the variables that correspond to attributes defined in the Java class have been set to the values defined in the Property instance run state.

3) Enter **broadcast GO** in the simulation window:





The local variables have changed to:

Locals		▼ ₽ ×	101
Variable	Value	Туре	0
 this ContextMana activeStateCo Example1 	1	SimulationManager ContextManager int Driver	
 carType age ncd premium offences Quotation m_StateM m_sInstan m_sType m_statema m_statema m_statema m_command 	"Saloon" 20 2 450 "No" "Example1" "Driver" Driver_ENUM_STATEM Driver_ENUM_STATEM	String double double String com.sparxsystems.c StateMachinelmpl String String Driver\$StateEnum Driver\$StateEnum java.lang.StringBuil java.util.Queue	
activeStates	Example1,{66FCDCB8	String[1024]	

NOTE

The value of premium has been set to **450** which is the value returned after executed the decision table and is the correct result.

- 4) Enter broadcast Back in the simulation window.
- 5) Enter **broadcast QUIT** in the simulation window.



3.8 IMPORTANT Gotcha

When the executable state machine builds for the first time, it **copies** the code generated for the DMN from its location into a new folder with the same parent as the ESM Java class:



Therefore, for the **first** Generate, Build and Run, the folder named **QuotationApplication** will be created and the DMN files copied. **BUT** on second and subsequent simulation runs, the files are **not** copied.

What is the impact of this? Quite serious, since if the Decision Table is **modified** and code re-generated, simulating the StateMachine **does NOT** use the changed decision table code, since the files have been copied already!

The workaround is simple:

- If the decision table has been modified, then using File Explorer navigation to where the folder (in our example named QuotationApplication) is located and delete this folder.
- Select Generate, Build and Run as usual, and since the folder named QuotationApplication in our example, does not exist, it will be created and the decision table files copied, and the simulation will use the updated decision table.

NOTE

The folder named **QuotationApplication** above is created when the simulation starts (if it does not exist already) and the name corresponds to the name of the **executable statemachine** element selected for the simulation.

If you have chosen a different name for the **executable statemachine** element then the folder created will be named according to the name you used for the **executable statemachine** element.



4 What to do Next

Having achieved a working integration between a Java based ESM and a Java based DMN, you could create multiple Property instances, one for each data set used to test the decision table, setting their run states accordingly. This would provide a thorough test of the integration:



When the simulation is run, one state machine is created for each instance, and when the decision table is invoked, the results are return to **all** instances. The results can be verified by examining the local variables window:

Locals		~	γ Ţ X
Variable	Value	Туре	Address
🔺 🧅 this		SimulationManager	
🕨 🧅 ContextMana		ContextManager	
activeStateCo	6	int	
🕨 🧄 Example1		Driver	
🕨 🧅 Example2		Driver	
🕨 🧄 Example3		Driver	
🕨 🧅 Example4		Driver	
🕨 🧄 Example5		Driver	
🕨 🧅 Example6		Driver	
🕨 🧄 🖕 🕨 🕨		java.lang.StringBuilde	
m_command:		java.util.Queue	
octiveStates	Example1,{BF3C9EB3-	String[1024]	

Initial state



LOCAIS	×
Variable Value Type Address	
▲ ♦ this SimulationManager	
ContextMana ContextManager	
ο activeStateCo 6 int	
Example1 Driver	
🖕 carType "Saloon" String	
🖕 age 🛛 20 double	
oncd 2 double	
premium 450 double	
offences "No" String	
Quotation com.sparxsystems.dr	
m_StateM StateMachinelmpl	
m_sinstan "Example1" String	
m_sType "Driver" String	
m_statem; Driver_ENUM_STATEN Driver\$StateEnum	
m_statem; Driver_ENUM_STATEN Driver\$StateEnum	
🖌 🧅 Example2 Driver	
carType "SUV" String	
🧄 age 20 double	
🧅 ncd 6 double	
premium 500 double	
offences "No" String	
Quotation com.sparxsystems.dr	
m_StateM StateMachinelmpl	
m_sinstan "Example2" String	
m_sType "Driver" String	
m_statem; Driver_ENUM_STATEN_Driver\$StateEnum	
m_statem; Driver_ENUM_STATEN Driver\$StateEnum	
✓	
cariype Sports String	
age 20 double	
nca U double	
offences "No" Gouble	
Outercles No String	
m StateM StateM StateM	
m dattan "Example?" StateWachinempi	
m_sinstan Examples String	
m statem: Driver ENUM STATEN Driver\$StateEnum	
m statem: Driver FNUM STATEN Driver\$StateEnum	
Example4 Driver	
Example5 Driver	
▶ ♦ Example6 Driver	
🕨 🧄 m_pComman java.lang.StringBuilde	
▶ 🆕 m_command: java.util.Queue	
activeStates Example1,(66FCDCB8 String[1024]	
Properties Locals	_

After transition to State Applying - Example 1, Example 2 & Example 3



Locals				×
Variable	Value	Туре	Address	
🔺 🧅 this		SimulationManager		
ContextMana		ContextManager		
activeStateCo	6	int		
Example1		Driver		
Example2		Driver		
Example3		Driver		
🖌 🦕 Example4		Driver		
🖕 carType	"Saloon"	String		
🖕 age	20	double		
ncd	2	double		
premium	650	double		
offences	"Yes"	String		
Quotation		com.sparxsystems.dr		
🕨 🧄 m_StateM		StateMachineImpl		
🔷 m_sinstan	"Example4"	String		
🖕 m_sType	"Driver"	String		
🔶 m_statem;	Driver_ENUM_STATEN	Driver\$StateEnum		
🖕 m_statemi	Driver_ENUM_STATEN	Driver\$StateEnum		
🔺 🧅 Example5		Driver		
🖕 carType	"SUV"	String		
🧄 age	20	double		
🧄 ncd	6	double		
premium	700	double		
offences	"Yes"	String		
Quotation		com.sparxsystems.dr		
m_StateM		StateMachineImpl		
🔶 m_sinstan	"Example5"	String		
m_sType	"Driver"	String		
m_statem;	Driver_ENUM_STATEN	Driver\$StateEnum		
m_statem;	Driver_ENUM_STATEN	Driver\$StateEnum		
Exampleb	"Constat"	Driver		
Gariype	Sports	string		
age age	20	double		
ncu nremium	900	double		
premium offences	900 "Vec"	String		
Outation	105	com snarvsvstems dr		
Market Ma		StateMachinelmnl		
m sinstan	"Example6"	String		
m sType	"Driver"	String		
m_statem;	Driver ENUM STATEN	Driver\$StateEnum		
m statem:	Driver ENUM STATEN	Driver\$StateEnum		
m pComman		java.lang.StringBuilde		
m_command:		java.util.Queue		
activeStates	Example1,{66FCDCB8	String[1024]		
Properties Locals	•			
 age ncd premium offences Quotation m_StateM m_slnstan m_sType m_statem: m_statem: m_pCommand: activeStates 	20 0 900 "Yes" "Example6" "Driver" Driver_ENUM_STATEM Driver_ENUM_STATEM Example1,{66FCDCB8	double double String com.sparxsystems.dr StateMachineImpl String Driver\$StateEnum Driver\$StateEnum java.lang.StringBuilde java.util.Queue String[1024]		

After transition to State Applying - Example 4, Example 5 & Example 6

Should you wish to trigger **individual** instances so that you can control each instance separately, then fire a trigger using the following format:

send trigger name to instance name

for example

send GO to example4



You can also use **send** to send a trigger to all instances (same as **broadcast**) by using the format:

send trigger name to all

for example

send BACK to all

5 Conclusion

This document provides a step-by-step tutorial of the integration of DMN and ESM using Java. This document cannot possibly address all the possibilities of DMN / DMN integration but I really hope that it gets you started on your DMN / ESM integration.