

D-MILS M-BAC Tool User Guide

Version	Created By	Description	Date
1.0	Richard Hawkins (richard.hawkins@york.ac.uk)	Original version	01/10/15
1.1	Richard Hawkins	Added section 2.4	22/10/15

Contents

1.	Quick-Guide – How to run existing configurations	2
1.1.	Install M-BAC tool.....	2
1.2.	Create an instantiation of an argument pattern.....	2
2.	Advanced Guide – How to create new patterns and configurations	5
2.1.	Create Argument Patterns.....	5
2.2.	Create Weaving Model.....	7
2.3.	Create A Configuration.....	8
2.4.	Create Manual Instantiation Elements	8

1. Quick-Guide – How to run existing configurations

The M-BAC tool comes complete with pre-defined configurations that allow the instantiation of a number of D-MILS patterns.

1.1. Install M-BAC tool

Extract MBAC.zip and run eclipse.exe. When prompted to select a workspace select Workspace within the Eclipse Workspaces folder.

N.B. Windows places restrictions on path name lengths. To avoid this causing problems during installation it is highly recommended to extract the folder not too far from the file system root.

1.2. Create an instantiation of an argument pattern

Assurance case models are created using the MBAC tool.

Add target models

An XML file is generated from the MILS-AADL file of the system using the D-MILS compiler. The generated XML file is saved in the Target Models folder of the Eclipse Workspace of the MBAC tool.

Running an instantiation

To run an instantiation of an assurance case pattern using the target model, one of the available configurations should be chosen. To choose a configuration, click the down arrow next to the “Run Configuration” icon, as shown in Figure 1.

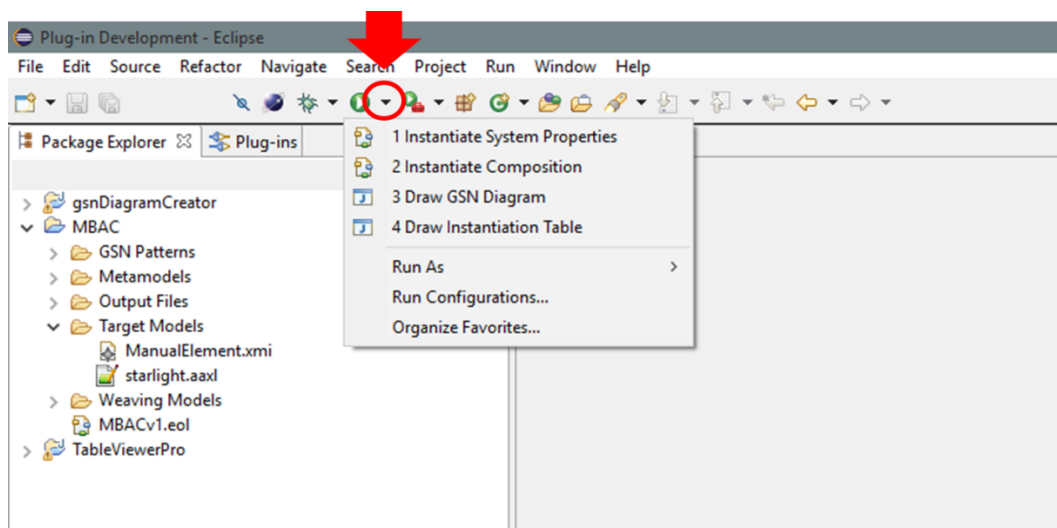


Figure 1 - Select configuration to run

The configurations named “Instantiate x” will instantiate the pattern x using the current target models. In Figure 1 options to instantiate the System Properties argument pattern and Composition argument pattern are provided (more options may appear in the current version of the tool). This will create models of the instantiated arguments.

Viewing an instantiated argument model

Once an instantiation has been run, there are a number of options provided for viewing the output model. These are:

1. Create an argument structure of the instantiated argument
2. Create an instantiation table
3. View the model with a model editor
4. View the model as an XML file

Each of these options is described below.

1. Choosing the configuration “Draw GSN Diagram” will create a GSN structure showing the element names and relationships, as shown in Figure 2.

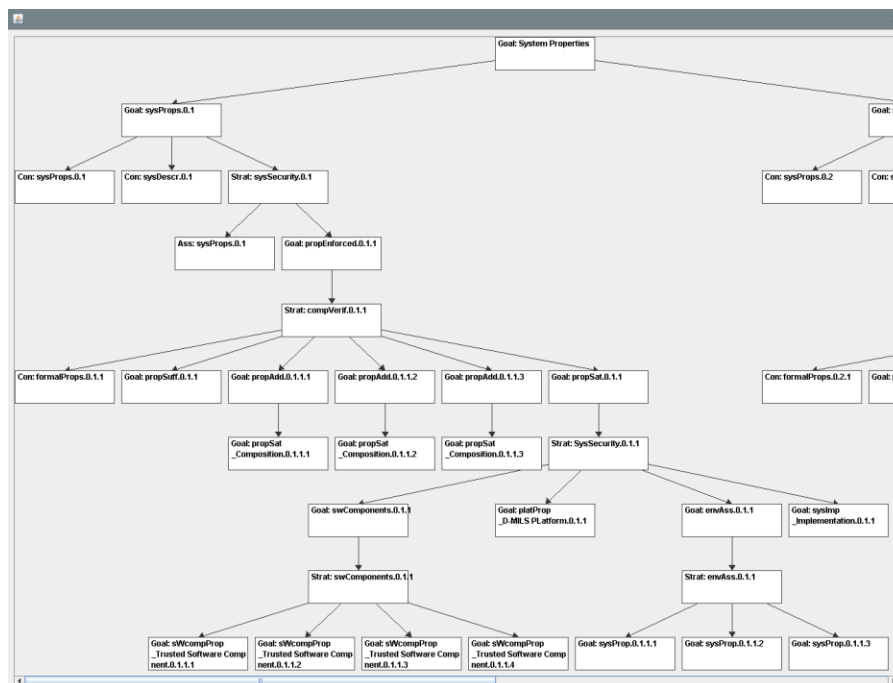




Figure 2 - Draw GSN Diagram

2. Choosing the configuration “Draw Instantiation Table” will create a table showing the instantiation information extracted from the target ID models to create the argument model. The table contains columns for the GSN Element ID, the Role Name of the role within the GSN element, and the Role Binding which is the element of the target model that is used to instantiate the role for that element. An example table output is shown in Figure 3.

GSN Element ID	Role Name	Role Binding
Goal: propSat.0.1	[formal property]	always ({cmd} implies in the future {return})
Goal: formalVerif.0.1	formal property	always ({cmd} implies in the future {return})
Goal: verifResults.0.1	formal property	always ({cmd} implies in the future {return})
Con: components.0.1.1	[trusted software component]	Usubject
Con: components.0.1.2	[trusted software component]	Dsubject
Con: components.0.1.3	[trusted software component]	Lsubject
Con: components.0.1.4	[trusted software component]	Hsubject
Con: platformProps.0.1.1	[assumed platform property]	No meta dep target because no role mapping in weaving model
Goal: verification.0.1	technique	ocra
Goal: propSat.0.2	[formal property]	always ({return} implies {last_data(return) = computation(last_data(cmd))})
Goal: formalVerif.0.2	formal property	always ({return} implies {last_data(return) = computation(last_data(cmd))})
Goal: verifResults.0.2	formal property	always ({return} implies {last_data(return) = computation(last_data(cmd))})
Con: components.0.2.1	[trusted software component]	Usubject
Con: components.0.2.2	[trusted software component]	Dsubject
Con: components.0.2.3	[trusted software component]	Lsubject
Con: components.0.2.4	[trusted software component]	Hsubject
Con: platformProps.0.2.1	[assumed platform property]	No meta dep target because no role mapping in weaving model
Goal: verification.0.2	technique	ocra
Goal: propSat.0.3	[formal property]	any
Goal: formalVerif.0.3	formal property	any

Figure 3 - Draw Instantiation Table

- Running an instantiation will create the file InstantiatedGSNPattern.gsnmetamodel in the Output Files folder. This is an ecore model of the argument. By default this will open in the MBAC tool in a GSNmetamodel Model Editor as shown in Figure 4. This view allows full details of the instantiated argument elements to be seen in the Properties tab. *N.B. to ensure the properties tab is visible, ensure that the Epsilon perspective is selected (as shown in figure ?).*

The  symbol next to an element in the model editor indicates that there remains some undeveloped entity below that element. By expanding branches containing this symbol, the undeveloped entity, indicated with a  symbol, can be identified. In figure ? the element Goal: propSuff.0.1.1 has not been instantiated from the Target Models and will require further development. At early stages in the process, there will remain many undeveloped entities.

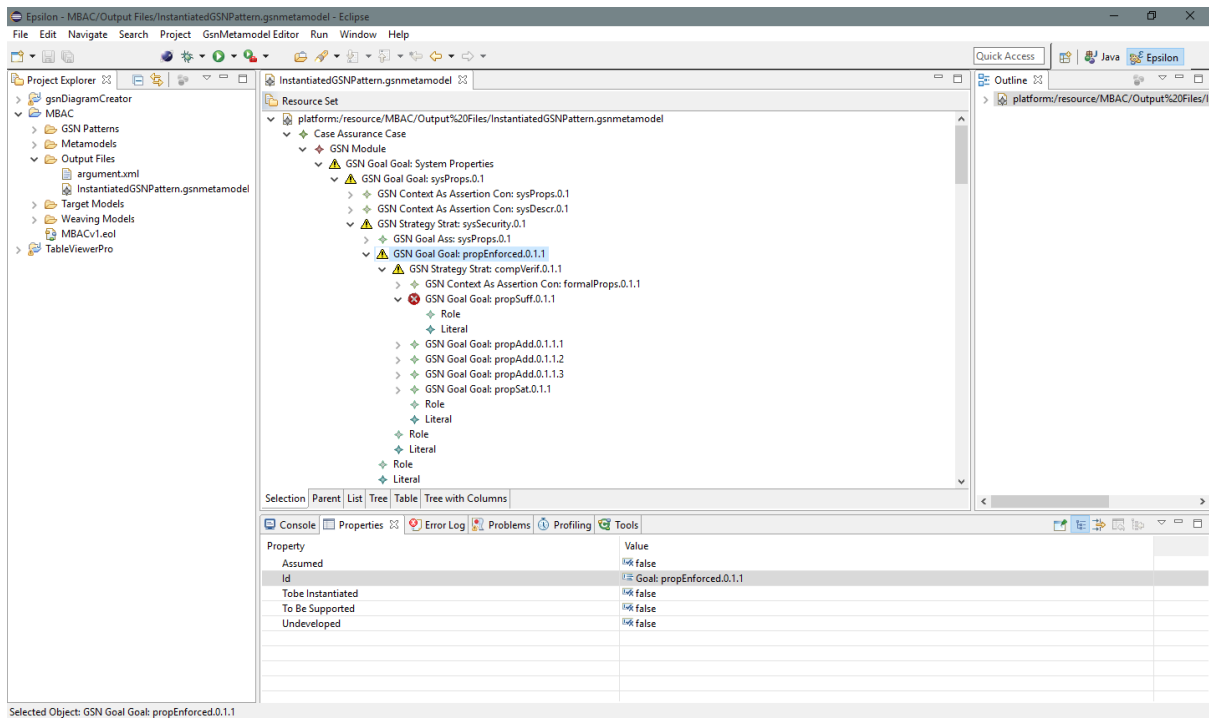


Figure 4 - View details of GSN model in model editor

4. Running an instantiation will create the file argument.xml in the Output Files folder. This is an xml representation of the argument model. This may be viewed in any text editor, or by selecting “Open With>Text Editor” within the MBAC tool.

N.B. For all the options above, the output displayed relates to the last instantiation performed. Models and diagrams created by the tool should therefore always be saved with unique file names before any further instantiations are performed if they are required for future use.

2. Advanced Guide – How to create new patterns and configurations

2.1. Create Argument Patterns

To create argument patterns using the M-BAC tool select File>New>Example and choose “GsnPatternEditor Diagram”. Select “MBAC>GSN Patterns” as the parent folder and give the pattern a File name. When you click Finish you will see a blank canvas and a palette of GSN elements as shown in figure Figure 5.

All elements of a pattern must be placed inside a Module. First click on GSN_Module in the palette and then click and drag in the canvas to create a Module box. Give the module a name by typing in the title bar of the Module box or clicking Properties tab below the canvas and adding a value to the Id.

Now create the top goal for the pattern by clicking on GSN_Goal in the palette and then click and drag inside the module to create a the Goal element. Give the Goal an ID by typing in the title bar of the Goal element or clicking Properties tab below the canvas and adding a value to the ID. The

properties tab also allows other properties of the element to be changed, such as “To Be Instantiated”.

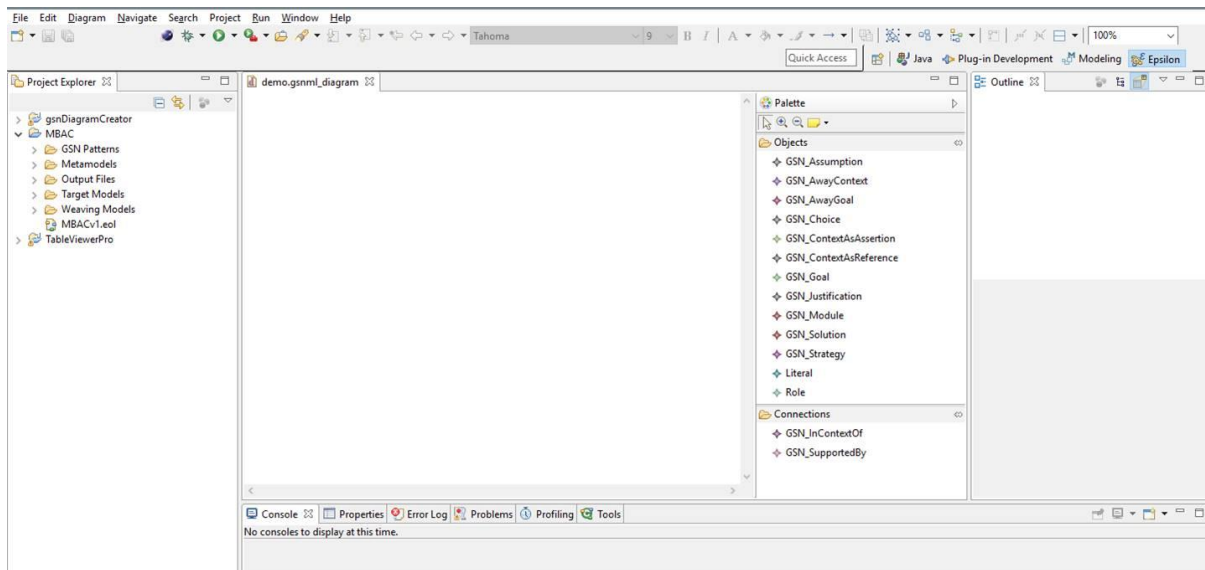


Figure 5 – Create a new argument pattern

To add content to the element click on either “Literal” or “Role” in the palette and create these inside the element.

Further elements can be added to the pattern by selecting the Goal and clicking and dragging on the arrow that appears below the element, as shown in Figure 6. Select the element from the context menu to which a SupportedBy or InContextOf relationship is required. The new element will be created; add properties and content as required.

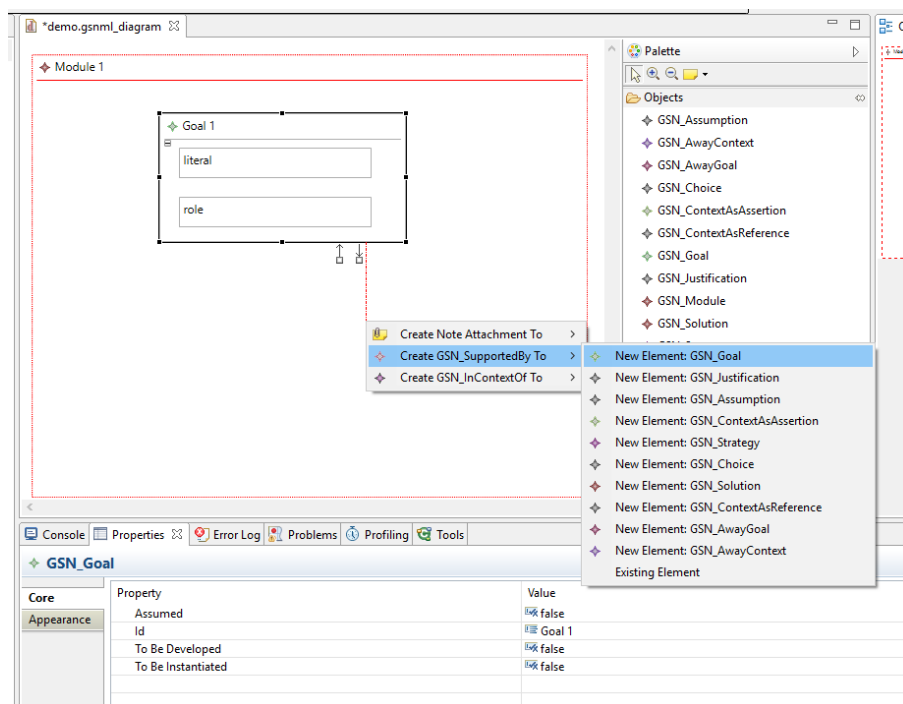


Figure 6 – Create new argument elements

The created pattern is saved as both a gsnml_diagram file and a raw gsnml file in the GSN Patterns folder.

2.2. Create Weaving Model

Weaving models are created in the form of graphml files using the graph editor yEd, available for free download from <http://www.yworks.com/en/products/yfiles/yed/>. The simplest way to create a new weaving model is to edit an existing model. Once yEd is installed, open one of the existing weaving model files contained with the Weaving Models folder of the MBAC tool.

Weaving models contain two types of node and two types of edge. To see the properties of a node, right-click on the node and select Properties. Click on the Data tab to see the Type and Properties of the Node.

Nodes of Type RoleBinding (Rounded Rectangles)

A node of this type should be created for each role in the argument pattern that is to be instantiated using the MBAC tool. The text of the node should correspond to the role name. No other properties are required.

Nodes of Type AADLmodelElement (Rectangles)

A node of this type should be created for each element type of the system metamodel to which a role will be linked. The text of the node should correspond to the name of the element in the metamodel. The following properties should also be defined for nodes of this type:

- String ElementType=<element_name> - where <element_name> is the name of the element in the metamodel. The value of <element_name> should normally be the same as the text of the node.
- String ModelName=<model_name> - where <model_name> is a unique identifier for the system model containing the required element. Note that <model_name> does not need to be the actual name of the system model itself, this is identified in the configuration.

Edges of Type Mapping

These edges represent links between roles, and elements of system models. The following properties should be defined for edges of this type:

- targetProperty=<attribute> - where <attribute> is the attribute of the target element that contains the information required to instantiate the role. <attribute> will often be “name”.

Edges of Type metaDep

These edges represent associations that exist between elements of a system model. The following properties should be defined for edges of this type:

- path=<path> - where <path> is the path defined by the relationships that exist between the elements in the metamodel.

The created weaving model is saved as a graphml file in the Weaving Models folder.

2.3. Create A Configuration

To instantiate a pattern using a weaving model and set of target models requires a configuration to be created. To do this click the down arrow next to the “Run Configuration” icon, as shown in Figure 1. Select “Run Configurations...”. The simplest way to create a new configuration is to edit an existing configuration, to do this select one of the existing configurations listed under EOL Progam, such as “Instantiate System Properties”. Right-click this configuration and select Duplicate. In this newly created configuration click on the Models tab. The models listed below *should not* be changed as they are required for all configurations:

- XMLDoc
- XMLDoc2
- InstantiatedGSNPattern
- FormattedGSNML

The other models should be altered to reflect the current instantiation. Select each of the models and click Edit:

- **weaving** - Change the file path to point to the weaving model to be used for this instantiation. The model name should not be changed.
- **patternInstantiation** - Change the file path to point to the pattern containing the top goal of the argument to be instantiation. No other details should be changed.
- **Other pattern models** – For all other pattern models used in the instantiation, the model name should be changed to match the ID of the top goal of that pattern model. Change the file path to point to the pattern model. Additional pattern models can be added to the configuration if required by selecting a pattern model from the model list and clicking Duplicate.
- **System models** – For each system model used in the instantiation, the model name should be changed to match the <model_name> defined in the weaving model. Change the Model file path to point to the system model. Add the corresponding metamodel files and remove any unneeded metamodels files. ***Note that each system model must have a corresponding metamodel defined in the configuration.***

Click “Apply” and “Run” to execute the instantiation.

2.4. Create Manual Instantiation Elements

It will sometimes be the case that for some roles within an argument pattern, the required instantiation information is not available in a model (either because the required model has not yet been created, or no such model exists). In these cases a method is provided for manually providing instantiation information for those roles, such that the M-BAC tool will include that information as part of an instantiated argument. Below we describe how to create new manual instantiation elements.

Open Manual.ecore file in the Metamodels folder. Right-click on starlightManual and click “create new child > EClass” (see Figure 7)

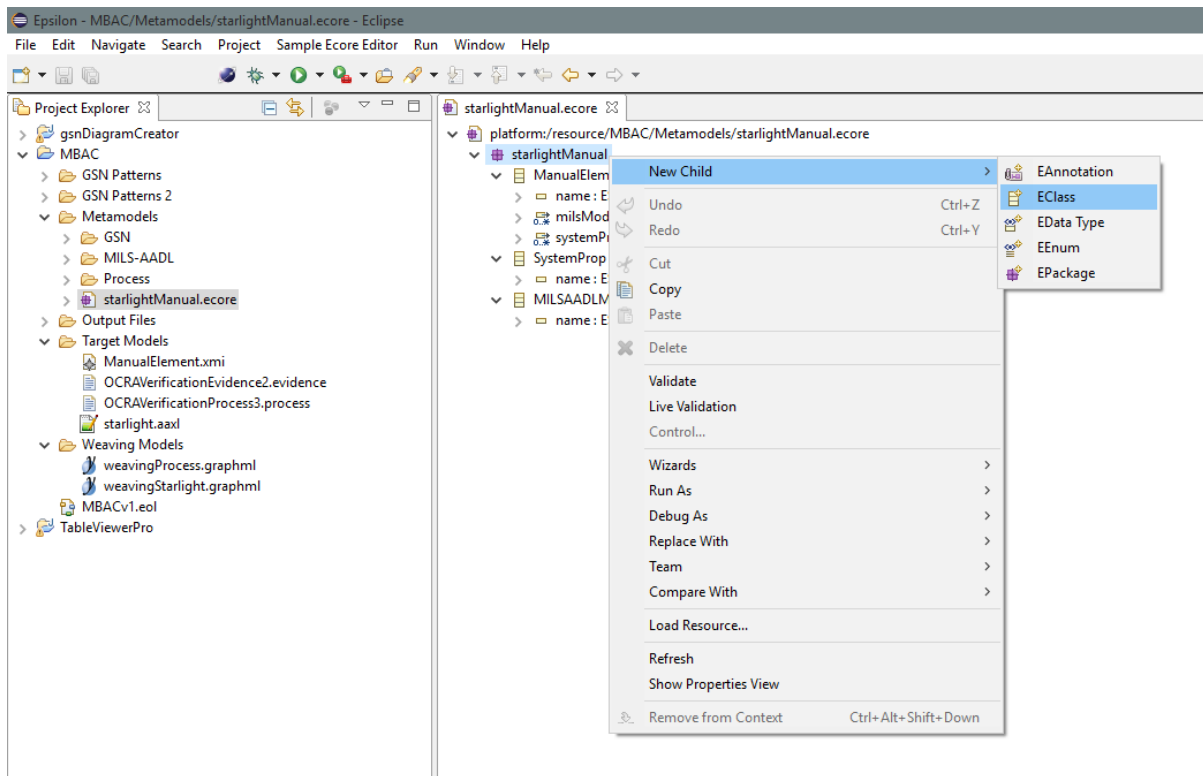


Figure 7 - Create new manual element type

Name your new element (in our example we create an element named “Constraint”. Right-click on the new element and click “new child > Eattribute”. Name the attribute and specify a type for the attribute (see the properties pane). You can create any number of attributes you require of any type (we create the attribute “name” of typew EString). Right click on ManualElement and click “new child > EReference”. Provide a name for the reference (in Figure 8 we call the reference “constraints”). In the properties pane click EType and from the dropdown box select the new element you create previously (see Figure 8). Ensure that Containment is set to true. Set the upper bound to -1 (meaning "many").

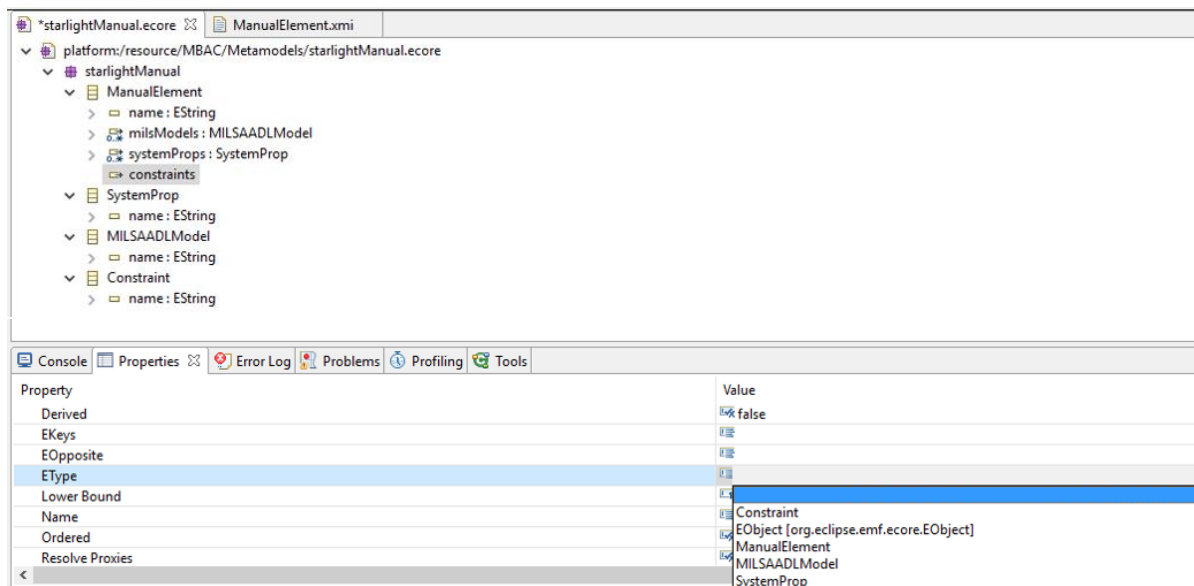


Figure 8 - Create EReference for new element

The metamodel is now created. You must now create a model containing the new element.

This can be done by either:

- Manually updating the file ManualElement.xmi to include required instances of the new metamodel element
 - for example to create an instance of the element Constraint with the name "new constraint" using the metamodel as shown in the Figure 9 would require the following to be included in the xmi file:
 - `<constraints name="new constraint"/>`

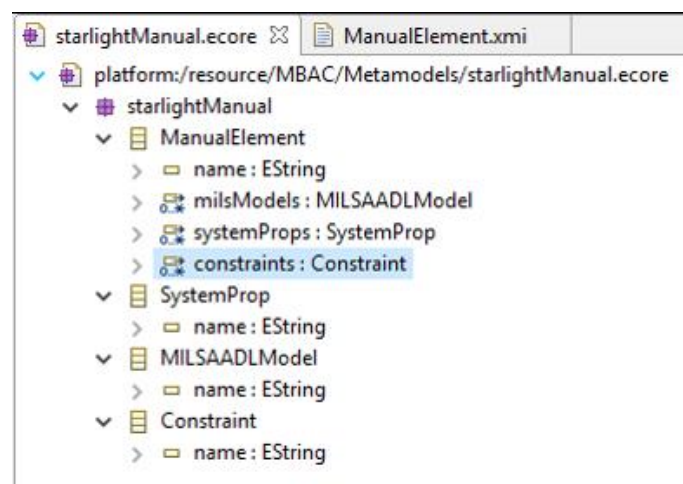


Figure 9 - ManualElement Metamodel

- An alternative method is to create a new model containing the new elements. This is achieved by right-clicking on the root element of the metamodel (in this case

"ManualElement") and selecting "Create Dynamic Instance" (see Figure 10). Save your model as a new file in the Target Models folder. When this file is opened you will be able to create instances of your new element by creating a new children of Manual Element. The new model must be added to the configuration.

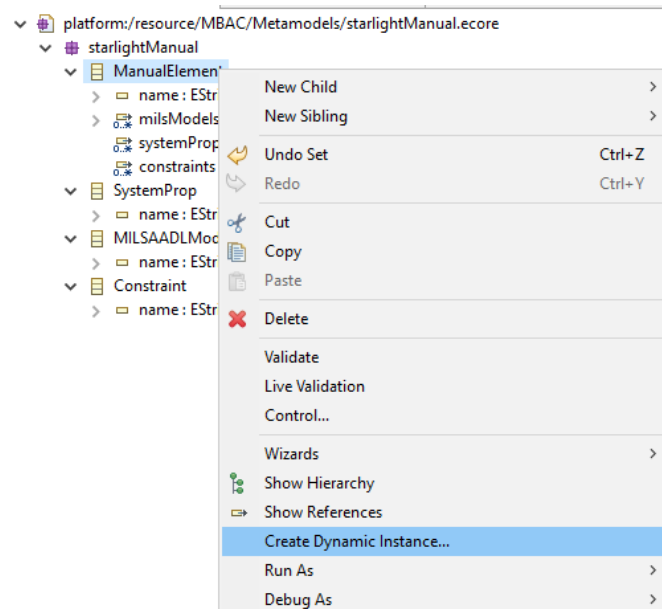


Figure 10 - Create new manual element model instance