MDK User's Guide

Model Development Kit (MDK) is a MagicDraw plugin that syncs models to the MMS, enabling interoperability between MagicDraw and other tools integrated with MMS, like View Editor. This guide is designed to introduce users to the various features of MDK.

Table of Contents

List of Tables

List of Figures

List of Equations

1 Initialization

The following views are designed to guide a user through initializing and configuring a project to enable the full functionality of MDK.

1.1 Linking to MMS

Actions described below instruct the user to set up a brand new MagicDraw project that is MDK-enabled. An MDK-enabled project means that the entire project model will be able to sync with MMS.

1.1.1 Create New Project

Actions

1. Open MagicDraw
2. File > New Project
3. Select “SysML Project” under the "Systems Engineering" section in the left side of the window
4. Name the Project
5. Choose project save location
6. Select "OK"
7. See message about System Engineering perspective - choose either
    - Perspectives switch the application to the graphical user interface designed for a specific role (such as systems engineer or analyst). Perspective options as well as details about the perspectives may be found by going to the top ribbon Options > Perspectives

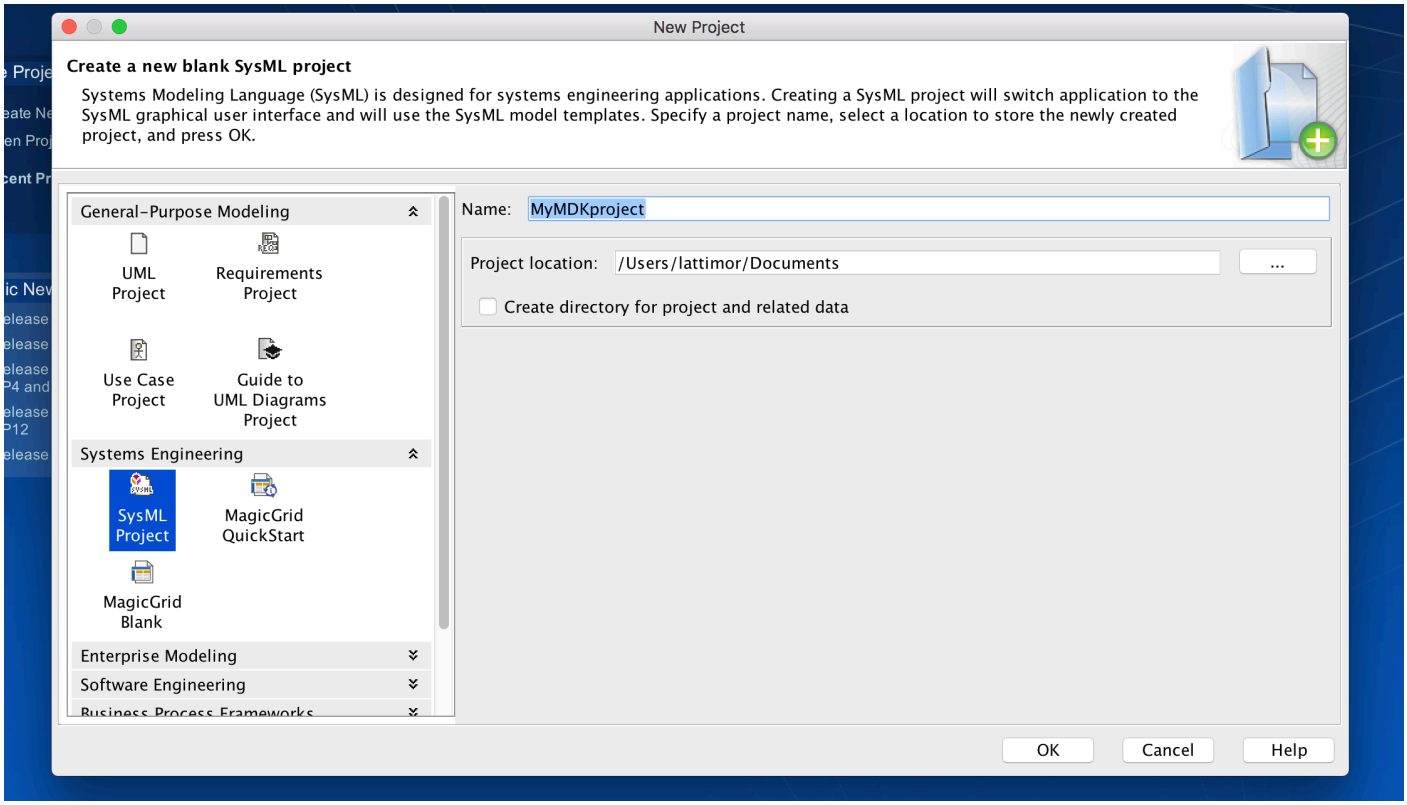


Figure 1. Create a new SysML project

1.1.2 Import MDK Module to Project

Actions

1. Go to Options>>Project Usages
2. Select "Use Project"
3. Select "From predefined location"
4. Select "<install.root> /profiles" under paths to used projects
5. Select “SysML Extensions.mdxml” from the list of projects
6. Select "Finish"
7. See message about showing Auxiliary Resources - choose either
    - Showing Auxiliary resources allows project usages to be seen in the containment tree
8. Click "OK"

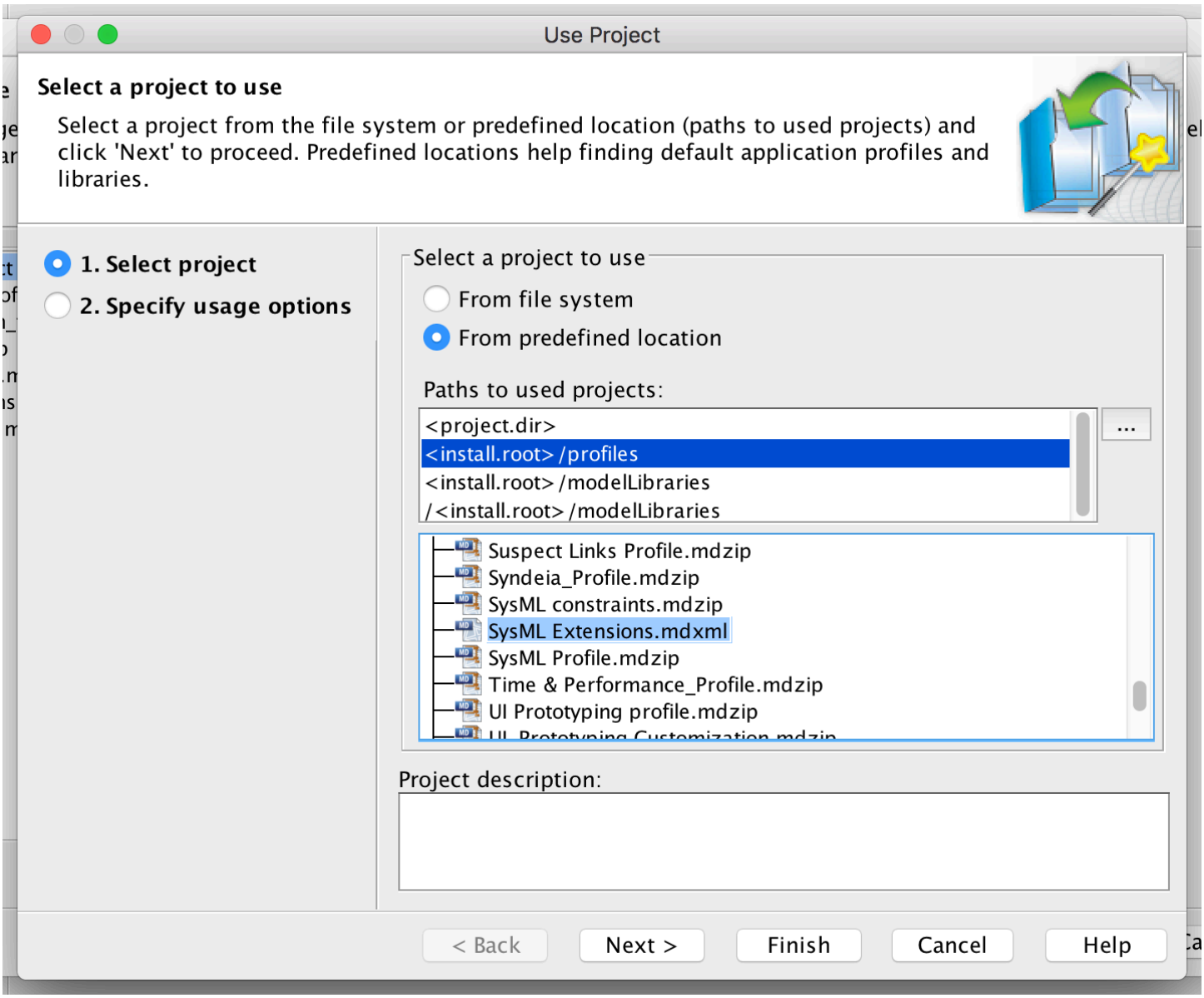


Figure 2. Importing SysML Extentions

1.1.3 Add Model Management Stereotype

Actions

1. Right click "Model" package > Specification
2. If greyed out, Model needs to be locked for teamwork before editing: Right click "Model" package > Lock element for edit
3. Select “Applied Stereotype” and "..." in the top right corner of the section to browse for stereotype
4. Search applied stereotypes for “MMS”

5. Select “Model Management System” stereotype
6. Select "Apply"
7. Model specification should now have Model Management System stereotype in its specification

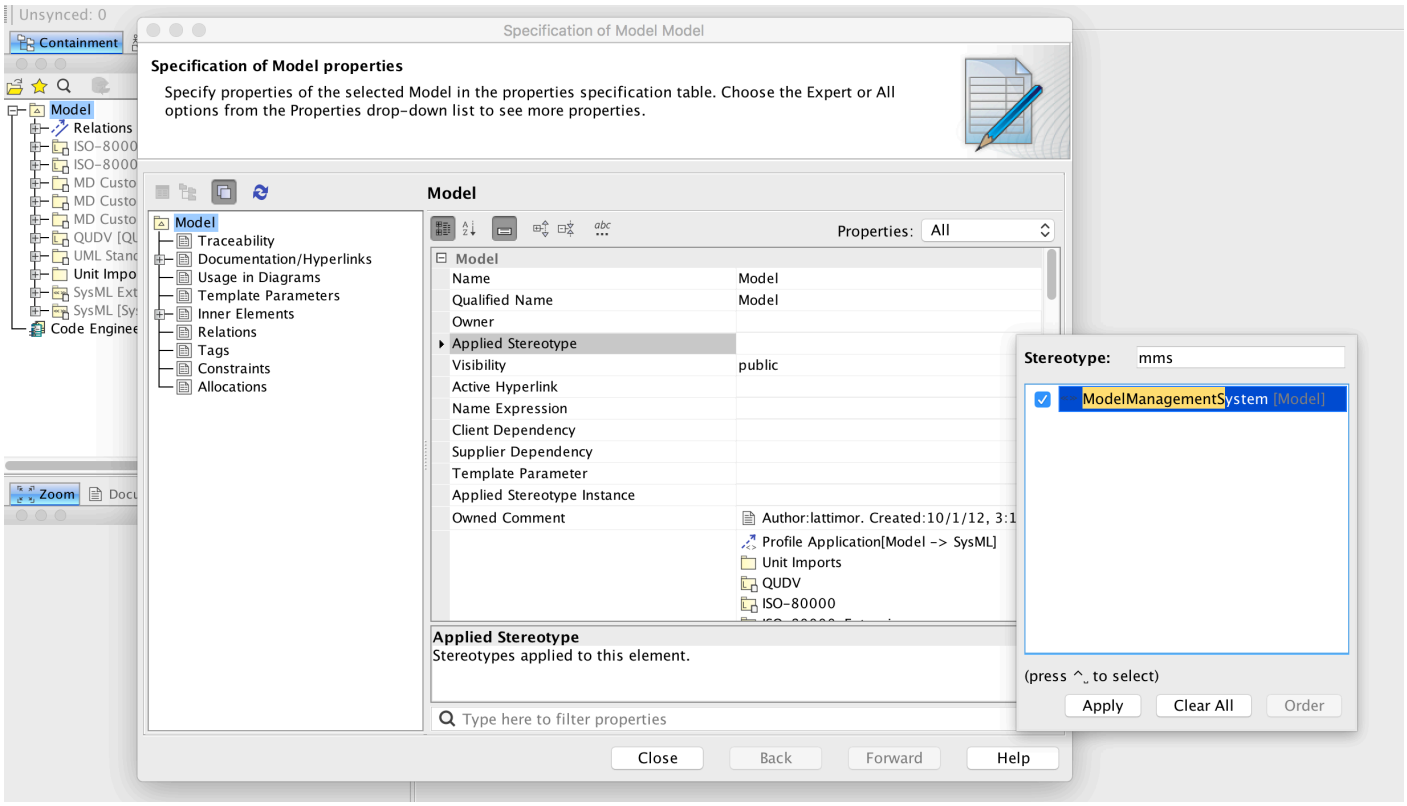


Figure 3. Adding MMS Stereotype

1.1.4 Assign Project to MMS Server

Actions

1. Open “Model” package specification (double click package or right click package > specification)
2. Scroll down to MMS section
3. Enter server name as MMS URL (e.g. https://mms.hostname.com)
4. Select "Close"

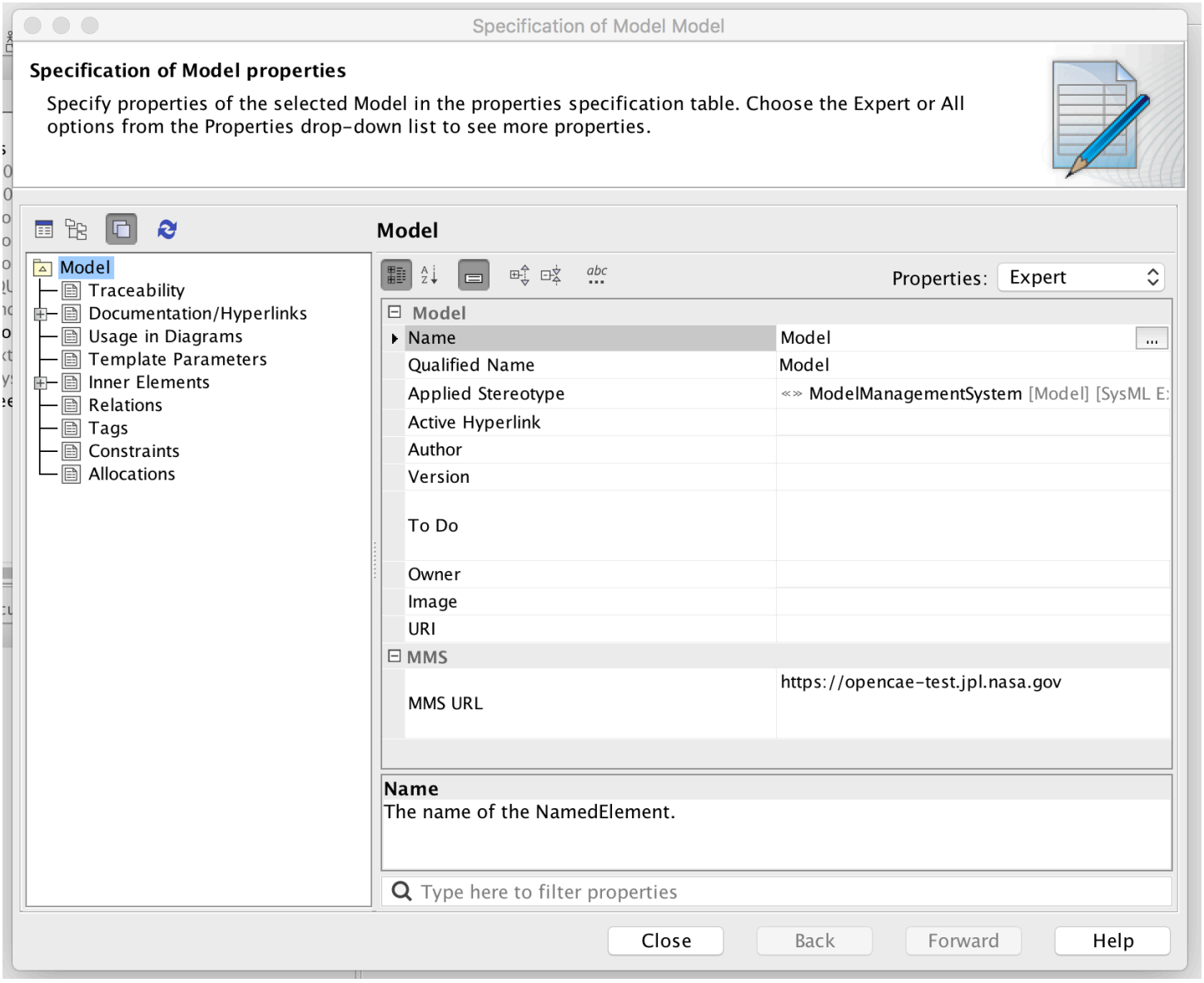


Figure 4. Specification of Model Properties

1.1.5 Initialize Project

Add Project to MMS

1. Login to MMS: Select "MMS" tab from top ribbon > Login > enter username and password > ok
2. Right click “Model” > MMS > Validate Models
3. Wait for validation window to see error
4. Right Click error in Validation window > Commit Project and Model. Note: Error will remain visible until validation window is closed
5. Wait for "Choose MMS Org" dialog to pop up > Select desired Org > ok
    - New Org may be created by selecting the "New..." option from the bottom of the list and following the prompts
6. Again, Right click "Model" > MMS > Validate Models
7. Wait for validation window to see "holding bin missing in client" error
8. Right click error > update element from MMS (holding bin should now be present in the containment tree)
9. Save project locally by File > Save Project

2 Functions

The following views are designed to guide the user through using all aspects of the MDK.

2.1 Logging in to MMS

Because many MDK features expect that you are logged in to the MMS, you may be prompted to login to the MMS when a MDK-enabled project is loaded. This login prompt occurs near the end of the load cycle, but will not block MagicDraw from completing the model load. If the login fails or is cancelled, MDK will revert to offline mode. Any changes will be persisted in the model, and you will be unable to receive changes from or send changes to MMS until you successfully log in. Other offline features of MDK are unaffected. You may login at any time by selecting "Login" from the "MMS" menu and submitting your credentials through the dialog box.

If you commit a MDK-enabled project to Teamwork Cloud (TWC) but are not logged in to MMS, you will be prompted to log in the MMS. This prompt occurs immediately after you submit the commit message, and will block MagicDraw from completing the commit to TWC until it is submitted or cancelled. If this login fails or is cancelled, you will continue to not have access to the online functionality as described above.

2.2 MMS Menu

The following provides an overview of the different operations that are offered through the top MMS menu. For information about the MMS menu that appears when selecting an element, such as a document, refer to Section .

- Login - Trigger a prompt for credentials to be used to log in to the MMS. When logged in, the options that require MMS interaction will be enabled.
- Logout - Clear your MMS authentication. It will also disable any options that require a connection to the MMS.
- Generate All Views - Generates all Views in the project. It is functionally equivalent to finding all Views, right clicking, and selecting "MMS" > "Generate View Contents".
- Validate - Validate elements or project attributes in bulk, such as Views and branches.

2.3 MMS Syncing

Syncing with MMS is an automated process that is typically transparent to the user. These views explain how this occurs, cases where manual intervention is required i.e. conflicts, and how to recover from a loss of parity.

2.3.1 Coordinated Syncing

Coordinated Sync ties the MMS synchronization functionality to MagicDraw's Teamwork Cloud (TWC) commit action. When a TWC commit is initiated in MagicDraw, coordinated sync triggers a similar commit to the MMS. This minimizes the user interactions required to keep the model up to date on the MMS and ensures that parity is maintained between the TWC model and the MMS model. This also vastly reduces the need for alternative sync methods such as Section 2.3.2.

Description:

While you are editing the model, element changes are collected in memory to be processed on your next save / commit. This includes direct changes to your model, such as new elements or updated documentation, as well as reference changes that can arise from changes to mounted projects. MMS changes by other users, in MagicDraw or View Editor, are also stored in memory to be processed during the sync operation (these changes are not stored in memory if you are not logged in to the MMS).

When a TWC commit is initiated and a coordinated sync occurs, these two change lists are processed. MMS element changes are updated into the MagicDraw model as possible. After this model update, MagicDraw model changes are committed to the MMS in the background of the TWC commit. Finally, the model save/commit occurs.

Any conflicts caused by changes to elements in both MagicDraw and MMS introduced by any user on either end will be stored in the model for any user to resolve as desired. They will also be presented to the user in the validation window upon save/commit.

These conflicts will be presented to the user on every save/commit until resolved. The presence of conflicts does not prevent completion of the TWC commit operation.

Usage:

- On Model Open
    - Coordinated Sync will prompt the user to log in to MMS. Logging in allows it to listen to messages track changes in real time, and be ready to gather and apply them at the appropriate time.
        - Cancelling this login, or submitting invalid credentials, will not prevent you from working on the model. It will only prevent you from receiving messages about elements changed in the MMS and prevent you from committing to the MMS until after you log in manually. Changes made to the model will be persisted locally for later commit.
- On Model Save / TWC Commit
    - Coordinated Sync will automatically perform model synchronization with MMS, committing MagicDraw elements to MMS or deleting elements no longer present in MagicDraw.
    - Coordinated Sync will update elements in MagicDraw based on MMS changes if the element can be edited and there is no version conflict between the MagicDraw and MMS versions of the elements.
        - Elements with pending updates will be presented to the user as a validation violation so that the user can plan to lock them before next commit, if desired. These will be stored in a list in the model for future checking.
    - Coordinated Sync will not update elements in MagicDraw of MMS that have pending changes on both the MagicDraw and MMS side, as it cannot determine which version of the element is correct.
        - Elements with these conflicts will be presented to the user as a validation violation for manual resolution. These will be stored in a list in the model for future checking.
    - Coordinated Sync will also validate all elements in its list of elements whose updates failed due to locks or conflicts, according to the above rules. If an update can be performed, the element will be updated in MagicDraw and removed from the list. Similarly, if the element is equivalent between MagicDraw and MMS, the element will be removed from the list. Any elements that remain out of sync with MMS will be again displayed as a validation violation.

2.3.2 Manual Syncing

Manual validation is intended to only be used to re-establish parity if it is lost, i.e. by application error, MagicDraw project reversion, etc., but models should generally be synced using Coordinated Syncing. If you have to manually validate, it is best to follow the CRUD rules about order of operations: 1) create 2) update 3) delete.

Manual validation options can be found by right clicking any element and selecting the "MMS" category. The three options are as follows:

- Validate Models: Run validation on the element(s) selected and all those contained by them.
- Validate Models (specified depth): Run validation on the element(s) selected and elements that are contained by them to the specified depth.
- Validate Element: Run validation on the element(s) selected.

2.3.3 Resolving Conflicts and Errors

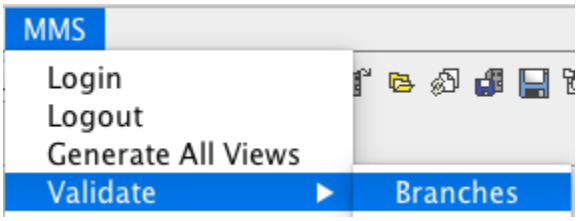
During any of the MMS synchronization operations, the MDK checks for conflicts between the MMS and the MagicDraw model. These conflicts may indicate a difference between what is in the current project model and an edit in View Editor, or they may be caused by data being updated on the MMS by a different application.

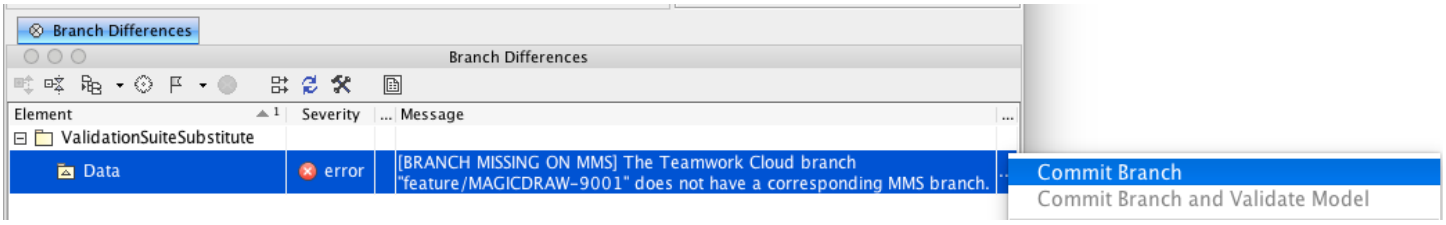
Conflicts will appear either in a Validation window, and each one may be interacted with via the context (right-click) menu in order to commit the MagicDraw version or updated from the MMS version. Additional information about the conflict may be available in the context menu as well. Error messages will appear on the Notification window, and may not be interacted with.

2.3.4 Branch Syncing

Note: This applies to Teamwork Cloud (TWC) projects, but not local projects as the latter do not have branching.

Both TWC and MMS support branching as a part of their version control functionality. MDK syncs branches created in TWC by committing them to MMS. To validate the branches on both sides, select "MMS" from the main menu > "Validate" > "Branches". Validation rule violations will be presented if they are not in sync, and resolutions can be selected by right clicking each violation.





2.4 MDK Specific Features

The following views are designed to provide users with information about key MDK features, including explanations of their use and how they affect a modeler on a daily basis.

2.4.1 Documents and Views

Documents and Views can be created and modeled in MagicDraw and generated to View Editor.

Documents are a key part of systems engineering; through the practice of Model Based System Engineering (MBSE), documents have also been adapted so that they are able to be generated and produced from a model. Documents consist of Views, which are sections of a document. Both Documents and Views are based on Classes in UML and therefore are treated as elements in the model, with their own associated attributes and formatting metadata.

Implications:

A SysML model is not required to produce a document; however, the main interface between MagicDraw and View Editor, the web application, is primarily done through the interactive capabilities of documents and views. They are used to provide access to modeling data outside of the model itself.

Documents and views are built and configured using Section 2.4.2. More information can be found in the next view.

2.4.1.1 Open in View Editor

You may wish to inspect a view after generating it with the MDK, to confirm that it matches your expectations. The "Open in View Editor" feature allows you to easily open a Document of View on View Editor from its element representation in MagicDraw.

To use this feature, right-click the element in a diagram, containment, or similar view and select "Open in View Editor" from the context menu. This feature will build a full web link to the selected View by navigating up the view hierarchy to find the Document that contains the View, and then simply open the web link in your default browser.
If a Document or View is a child of multiple Documents, the feature will instead build a link for each Document and open a new window with a button for each Document. Clicking on one of the Documents listed will open the appropriate web link.
If a View is not a child of any Documents, the feature will display a message in the Notification Window of MagicDraw and open the View in View Editor without a Document context.

Use of this feature does not require you to be logged in to MMS, but does require that your model have a MMS server specified in its ModelManagementSystem stereotype.

Note: In the event that Java fails in its attempt to open a link in your default browser, a message will appear in the Notification Window of MagicDraw that includes the web address. This can by copied into your web browser to open the page manually.

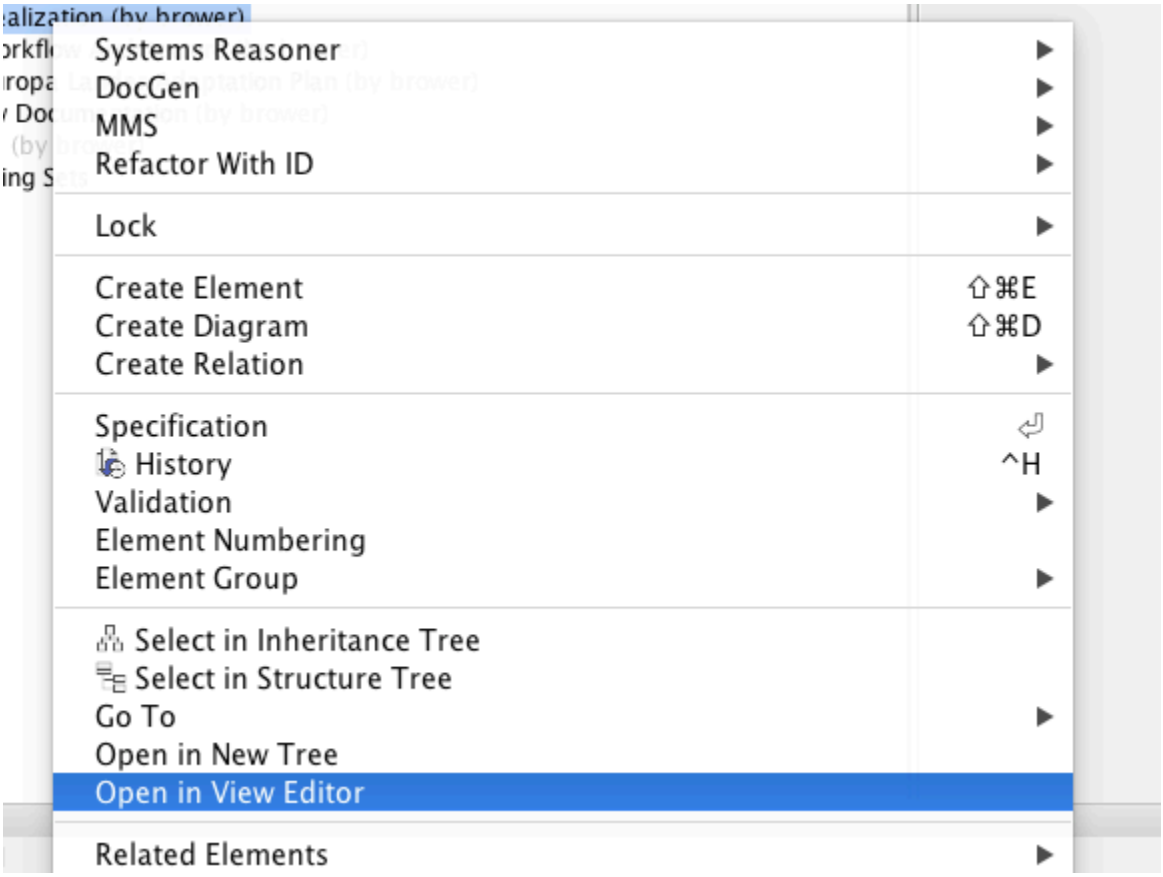


Figure 5. Open In View Editor

2.4.2 Viewpoints and Viewpoint Methods

One of the defining moments for widespread adoption of SysML at JPL was when the community created "DocGen" (see DocGen), a precursor of the MDK plug-in, which gives MBSE practitioners the ability to produce documents from their models. In order to create these documents, the view and viewpoint method was introduced.

A pattern is a set of rules governing model construction that provide standardization and consistency across models. The method for document creation is one such pattern adopted by the Object Management Group (OMG, the standards body behind SysML) and was incorporated
into SysML 1.4.

Viewpoints and Viewpoint Methods can be created in MagicDraw and will result in View Editor. Refer to Section 2.5.1 and Create a Reusable Cover Page for examples of using Documents, Views, Viewpoints, Viewpoint Methods, and Exposing elements.

Viewpoints:

- Can be thought of as the "compiler" for a view.
- Dictate what will actually be displayed in a view.
- Uses the viewpoint methods and the exposed elements to produce a view.

Viewpoint Methods:

- Are a set of rules that govern model construction to give standardization and consistency across models.
- Can be thought of as the "constructor".
- Consist of activity flows that are specialized to be in charge of building what the view will be.
- The most common viewpoint method are those that make Rapid Tables - they take the exposed elements and iteratively go through them to produce the desired table. More information on how to build such a table can be found here: Section 2.5.1.3.

Implications:

Using the model, a user will consistently utilize Viewpoints and Viewpoint Methods to construct documents and views. These documents and views may contain any number of important modeling information, based on what the user desires. The user will then generate these formatted and configured documents and views so that users on View Editor can have access to the model information and data.

NOTE: If the viewpoint method diagram is created by methods other than right clicking the viewpoint itself on the view diagram (such as right clicking the viewpoint in the containment tree), an error may occur during document generation regarding a viewpoint method. This error may be resolved by right clicking the error in the notification window > Set Nested Behavior as Viewpoint Method.

There are two ways to check for this error before view generation:

- Right click a View > DocGen > Validate View
    - Any not compliant views will appear in the notification window. Right click error > Set Nested Behavior as Viewpoint Method
- Select "MDK" from the top ribbon > Validate > Views
    - Any not compliant views will appear in the notification window. Right click error > Set Nested Behavior as Viewpoint Method

2.4.3 _MMSSync_ Package

The _MMSSync_ package is part of a number of sync solutions for the MMS and View Editor. Its purpose is to allow continual collaboration while a project is consistently being updated by multiple users. It does this by capturing changes between the model and the MMS server and persisting them in the model. The idea is that when an element is updated from the server, whoever is using the syncing options will capture the changes and the MDK will try to update the model itself. However, if there is some non-editable content or errors of any sort that would prevent the MDK from automatically updating, these changes will be saved in the _MMSSync_ package. Once stored, the changes will be tried to be resolved/updated the next time it is run.

Although the _MMSSync_ package looks like an ordinary Package, **users must NOT edit the content.** A user may try to unlock the elements in Teamwork Cloud projects, but should not edit the elements themselves. If the user manually modifies this Package or its contents, parity and/or data could be lost.

Usage:

- Every project that is connected to MMS will have an _MMSSync_ package.
- References to all changed (categorized as created, updated, and deleted) elements on both MagicDraw and MMS will be persisted in elements in this package.
- These changed elements will be analyzed during sync operations to generate the necessary operations to sync the model.
- Each element in this package will be categorized and timestamped.
- Each element in this package is automatically managed and deleted by the MDK. No user intervention is required or recommended. See version specific implications for more information.
- In the case that elements in this package cannot be deleted due to insufficient locks, a second element is created that signifies to sync operations that the first one can be safely ignored as its contents have already been processed.

2.4.4 Holding Bin Package

The "Holding Bin" Package is created for every project. Its purpose is to provide a default place to put elements that were created without an owner specified. This is often the case for Documents and Views that originated in View Editor. Elements in the Holding Bin can be moved to other locations in the project just like other model elements.

2.5 Selected Modeling Tutorials

The following views are dedicated to guiding users through some of the most commonly used workflows that involve MDK.

2.5.1 Create and Generate Documents

The following views focus on foundational training to get any user to be able to interact with the MMS and subsequently View Editor. The goal at the end of these views is for the MDK user to be able to Section 2.5.1.1 , Section 2.5.1.1.1.1 , and Section .

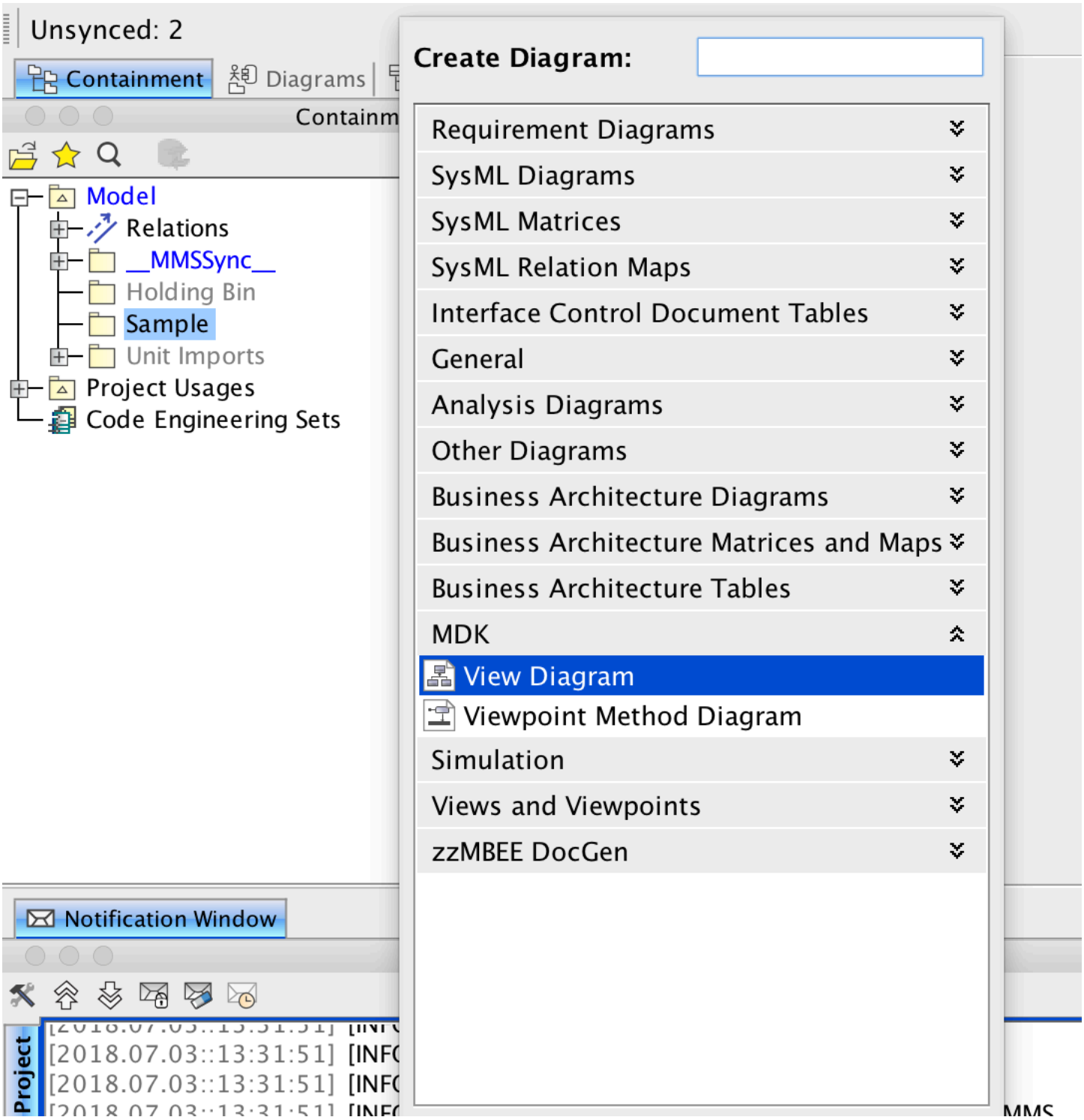
2.5.1.1 Create A Document With A View

In these instructions, the user creates a new (blank) document in MagicDraw using MDK's tools for Documents and Views. Committing these new elements is required for proceeding to the next step, Section 2.5.1.1.1, where the user will generate the document so it appears on View Editor.

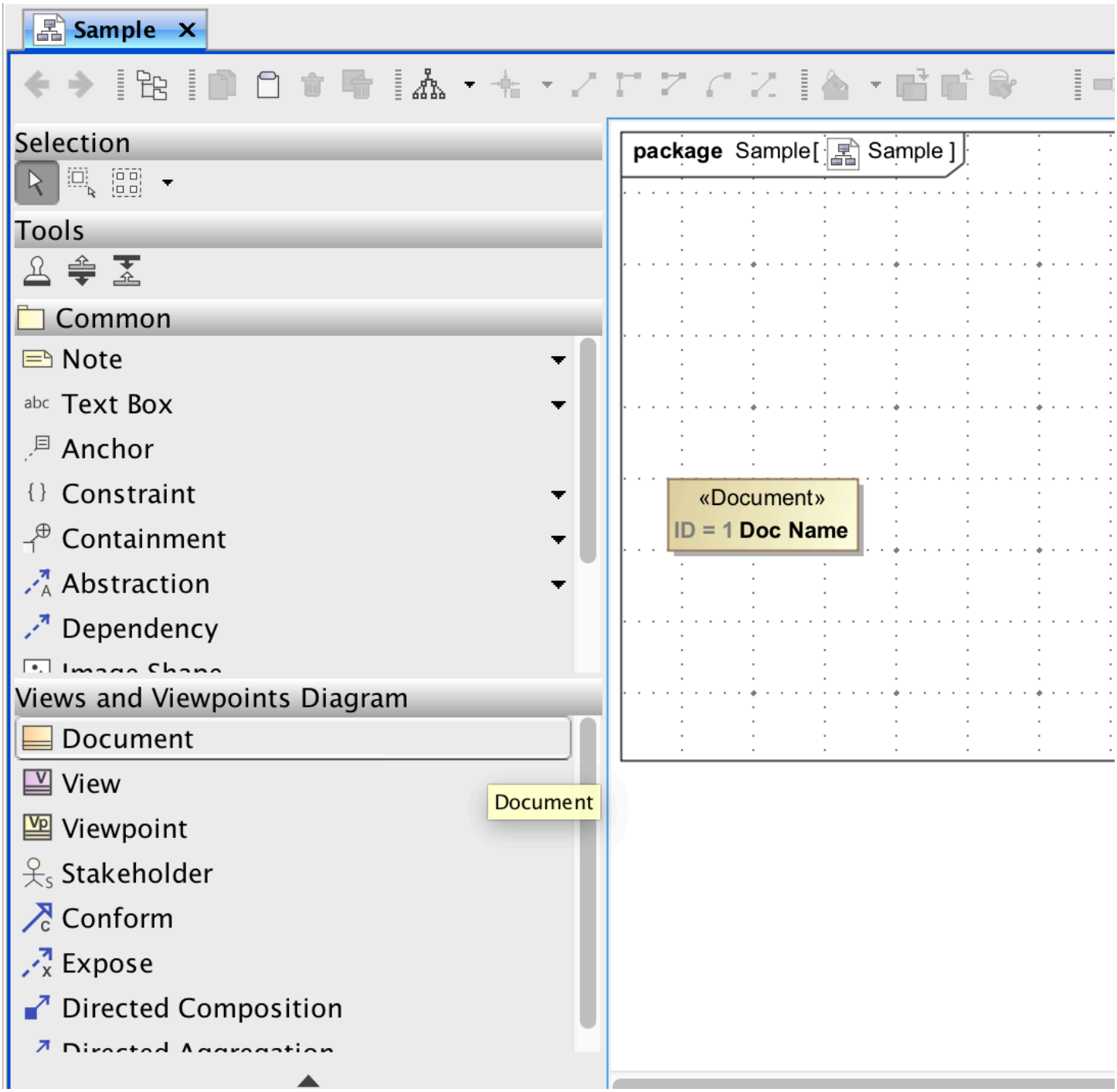
Setup: Model has been synced to it's own Alfresco site (See Section 1.1 for more information on this process) A new package is created for the purpose of simple organization. Each modeler should follow the modeling practices of his/her project.

Actions:

- Create a new View Diagram
    - Right click the package in the containment tree where you would like to create the view diagram > create diagram > select "view diagram" from the MDK section of the diagram list
    - New diagram is created and is displayed in the middle pane

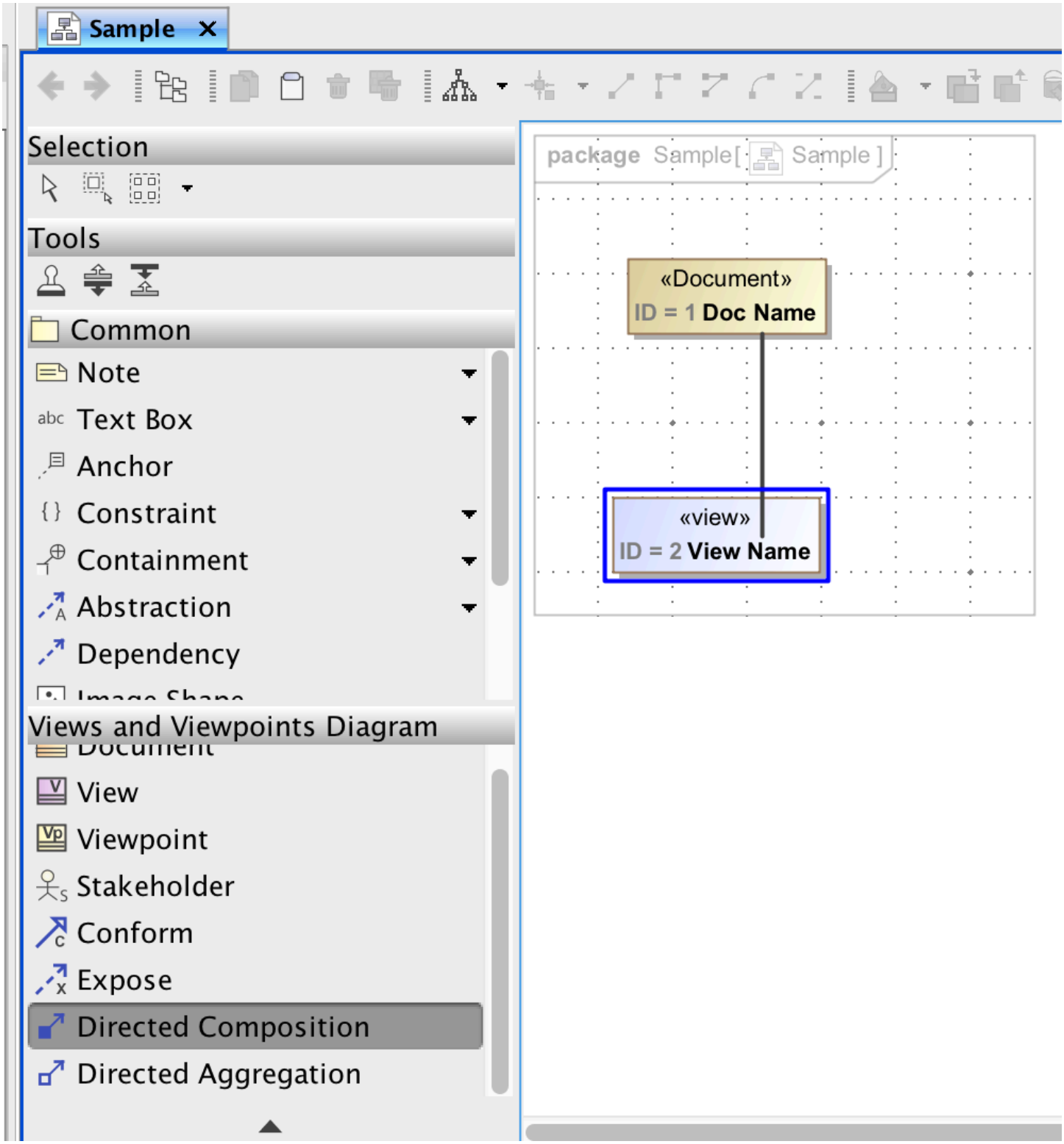


- Create new Document
  - Select "Document" from the menu to the bottom left of the diagram in the center pane
  - Click anywhere in the empty diagram to add the document
  - Double click on the newly created document or right click > specification to open the document specification
  - Enter a name in the name section of the specification



- Create new View
  - Select "View" from the menu to the bottom left of the diagram in the center pane
  - Click anywhere in the empty diagram to add the view
  - Double click the newly created document or right click > specification to open the view specification
  - Enter a name in the name section of the specification

- Create a "Directed Composition" relation from the new Document to the new View
  - Select "Directed Composition" from the menu on the bottom left of the diagram in the center pane
  - Click on the document, then click on the view to create a directed composition relationship between them
  - Relationship should appear between the two elements with a black diamond on the end of the document and an arrow on the end of the view



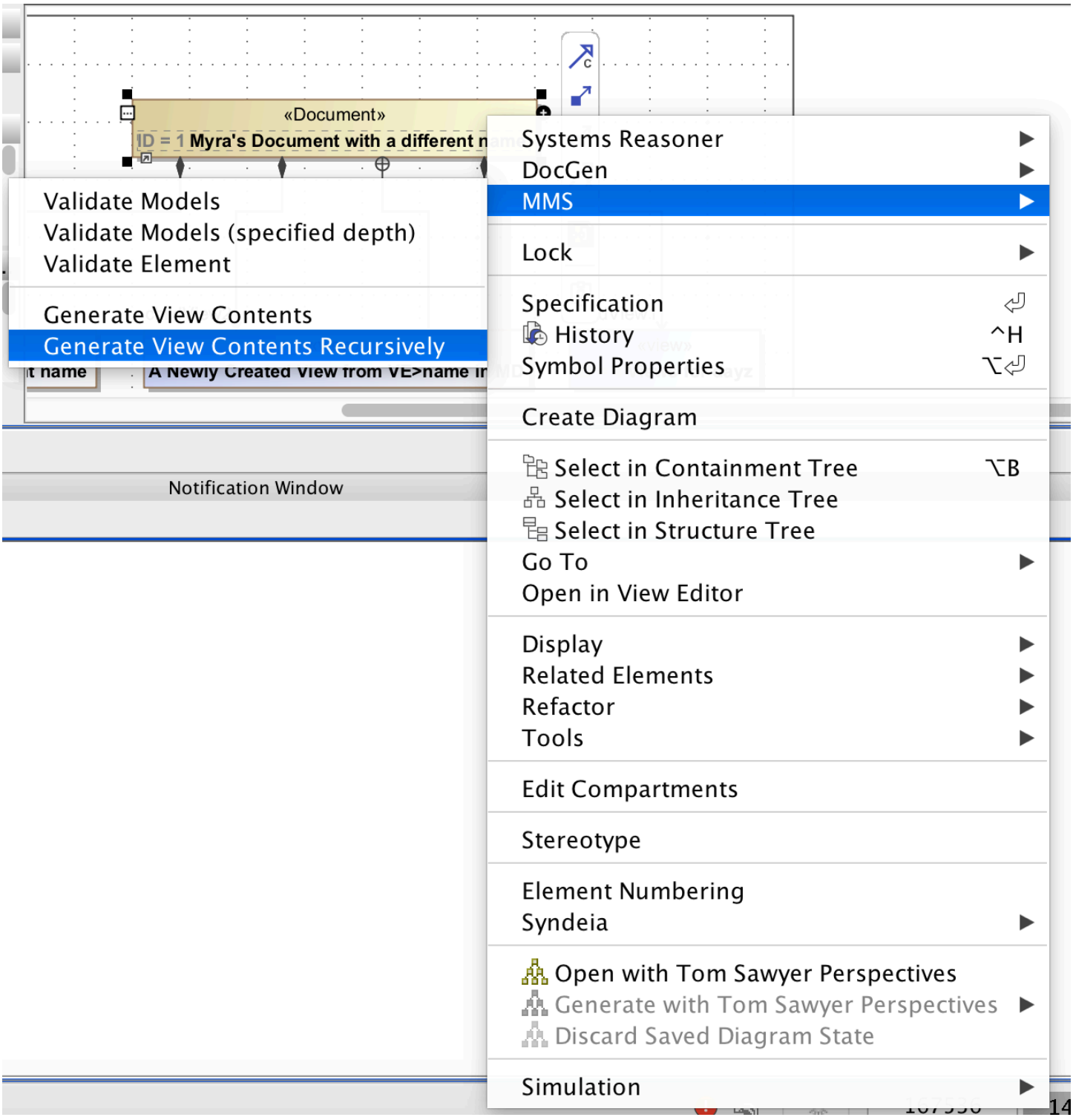
2.5.1.1.1 Generate Views and Sync with MMS

2.5.1.1.1.1 Generate Views

Actions:

Generate select documents:

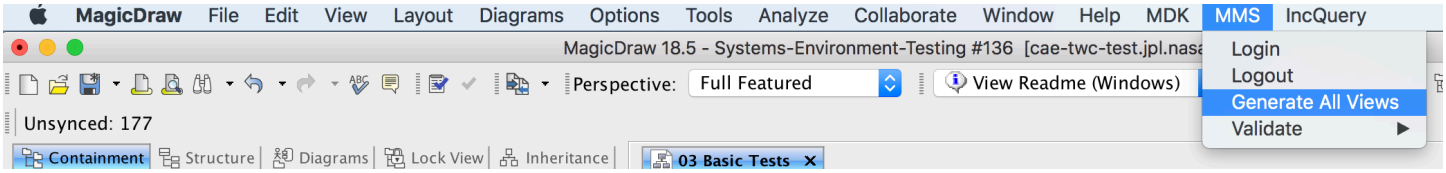
1. Generate Views by right clicking the document (in diagram or containment tree) > MMS > "Generate View Contents" or "Generate View Contents Recursively".
2. Results of generation appear in a message in the notification window
3. Navigate to document on view editor
4. Updated Document and views should be visible
5. Save changes in MagicDraw



Generate all documents in a model:

1. Select "MMS" from the top ribbon >> Generate all Views
2. Results of generation appear in a message in the notification window
3. Navigate to the project on View Editor

4. Updated documents and views should be visible
5. Save changes in MagicDraw

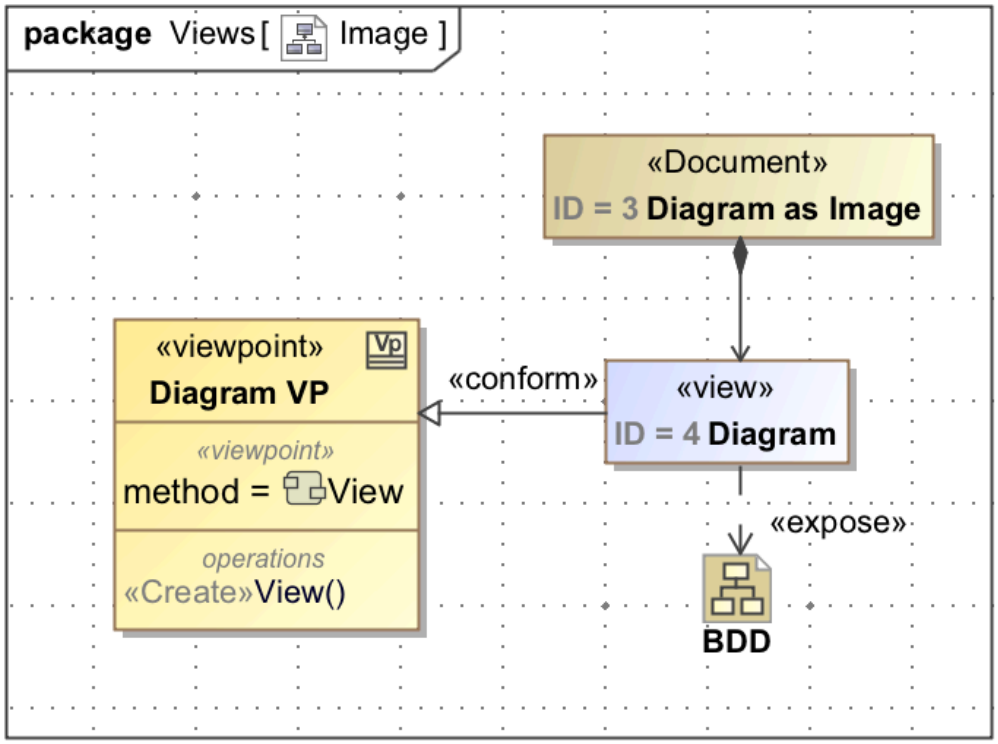


2.5.1.2 Insert Diagram as Image

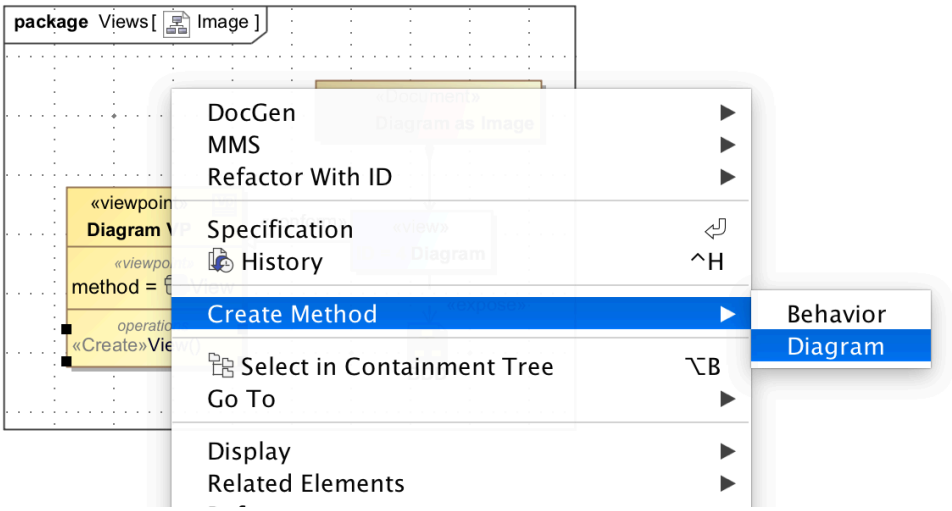
The following instructions show one of the most basic ways of using viewpoints and viewpoint methods to construct contents of a view. For more information about what viewpoints and viewpoint method are, refer to Section 2.4.2 . More detailed information about creating specific viewpoints and methods can be found in the DocGen .

Actions:

1. Create a new View as described in Section 2.5.1.1
2. Create new View Point by selecting "Viewpoint" from the menu to the lower left of the view diagram and clicking on open space in the diagram. Name the View Point.
3. Create a "Conforms" relation from the new View to the new View Point.
    - Select "Conform" relationship from the menu to the lower left of the view diagram.
    - Click the view, then the view point.
    - Conform relationship is displayed as a white arrow pointing to the view point.
4. Select a diagram from the containment tree (to insert as an image) and drag it into the view diagram
5. Create an "Expose" relation from the chosen view to the diagram
    - Select "Expose" relationship from the menu to the lower left of the view diagram
    - Click the view, then the diagram
    - Expose relationship is displayed as a dotted line pointing towards the diagram



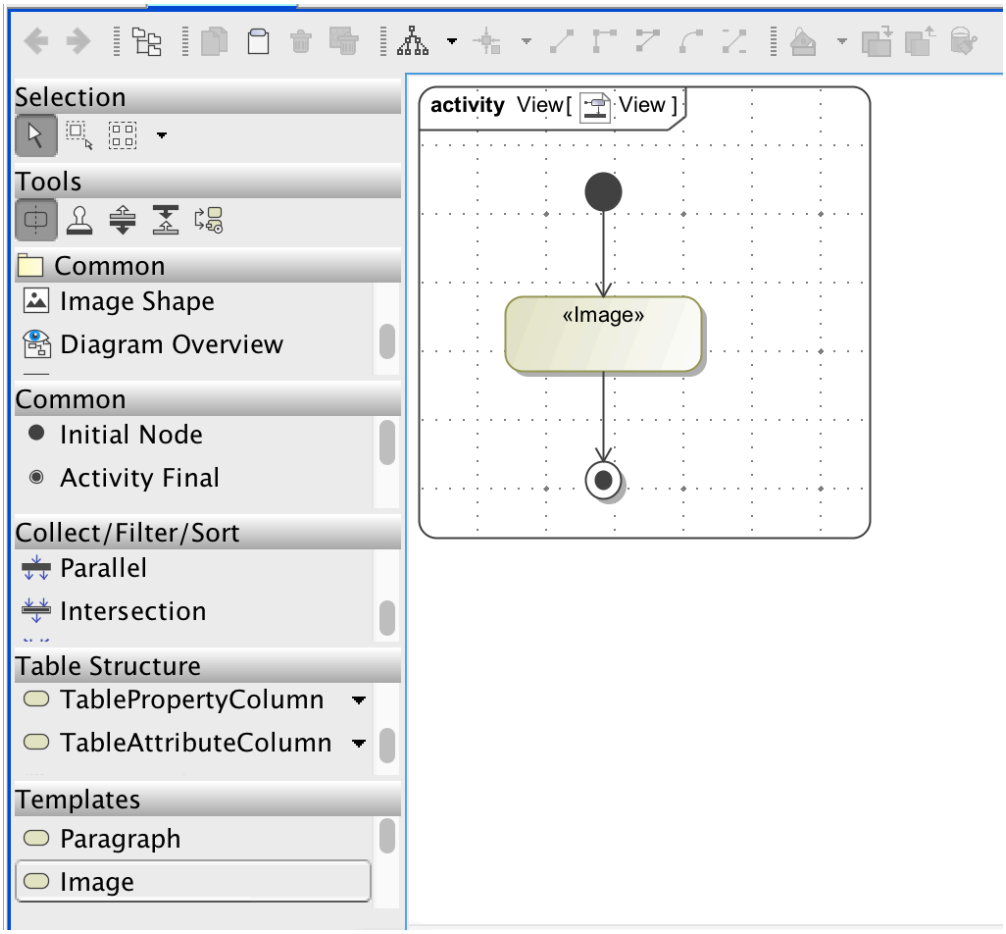
1. Right click the "operations" section of the Viewpoint > Create Method > Diagram > "Viewpoint Method Diagram" under the MDK section



NOTE: If the viewpoint method diagram is created by other methods (such as right clicking the viewpoint in the containment tree), an error may occur during document generation regarding a viewpoint method. This error may be resolved by right clicking the error in the notification window > Set Nested Behavior as Viewpoint Method. There are two ways to check for this error before view generation:

- Right click a View > DocGen > Validate View
    - Any not compliant views will appear in the notification window. Right click error > Set Nested Behavior as Viewpoint Method
- Select "MDK" from the top ribbon > Validate > Views
    - Any not compliant views will appear in the notification window. Right click error > Set Nested Behavior as Viewpoint Method

1. Create the View Point activity on the viewpoint method diagram
    - Insert "Initial Node" (from menu to left of diagram, under the Common section)
    - Insert ”Image" (from menu to left of diagram, under the Templates section)
    - Insert “Activity Final" (from menu to left of diagram, under the Common section)
    - Connect activity flow by clicking the element, selecting "Control Flow" from the popup menu (icon is an arrow), and clicking of the subsequent element



1. Commit Changes
2. From the View Diagram, right click on the document > MMS > “Generate Views Contents Recursively"
3. Diagram should now be visible on View Editor

1 Diagram

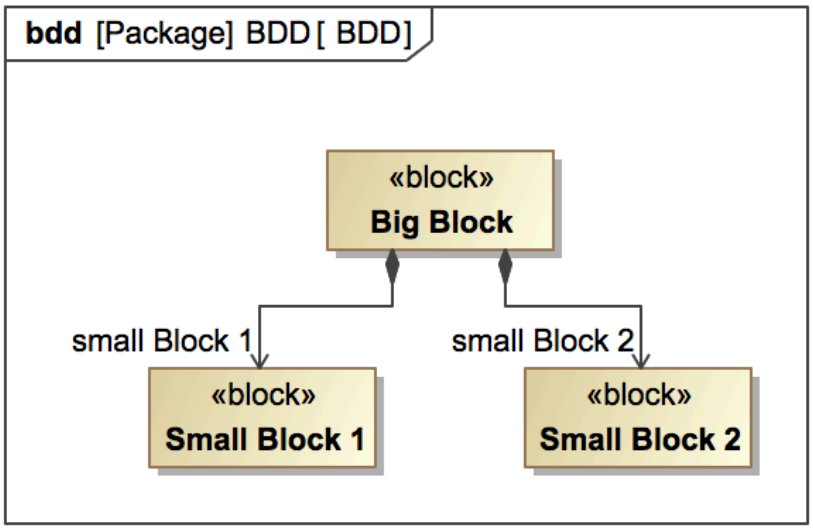


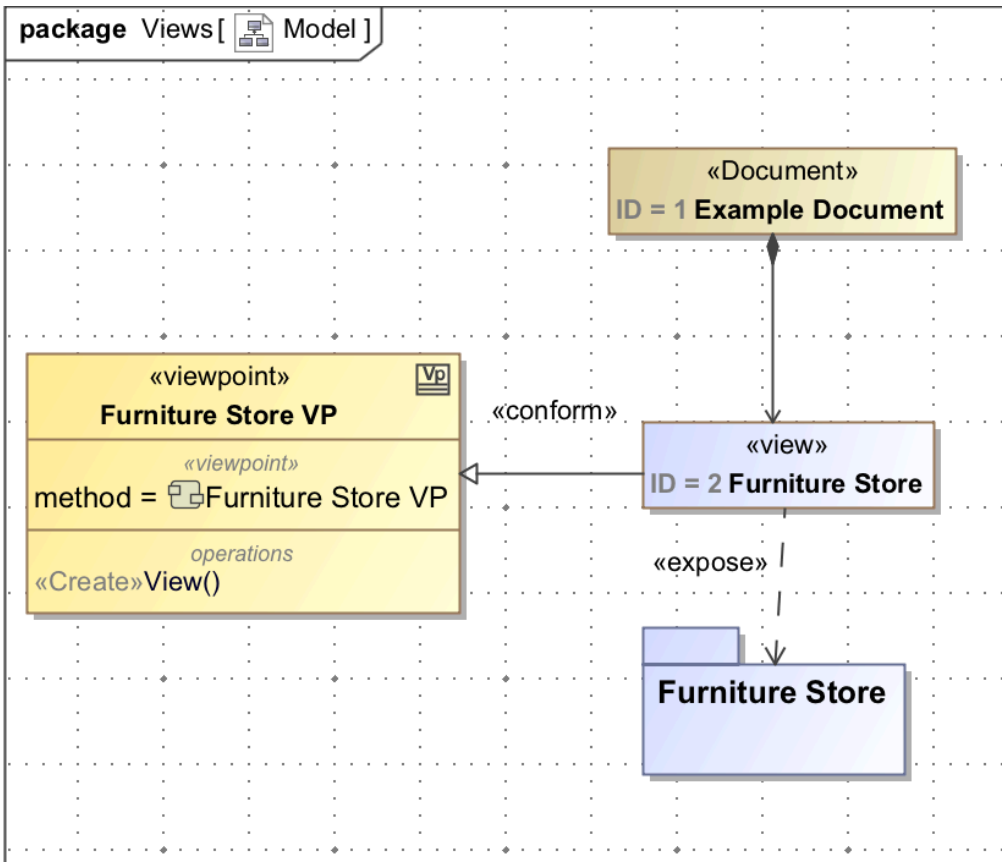
Figure 1. BDD

2.5.1.3 Create and Generate a Rapid Table

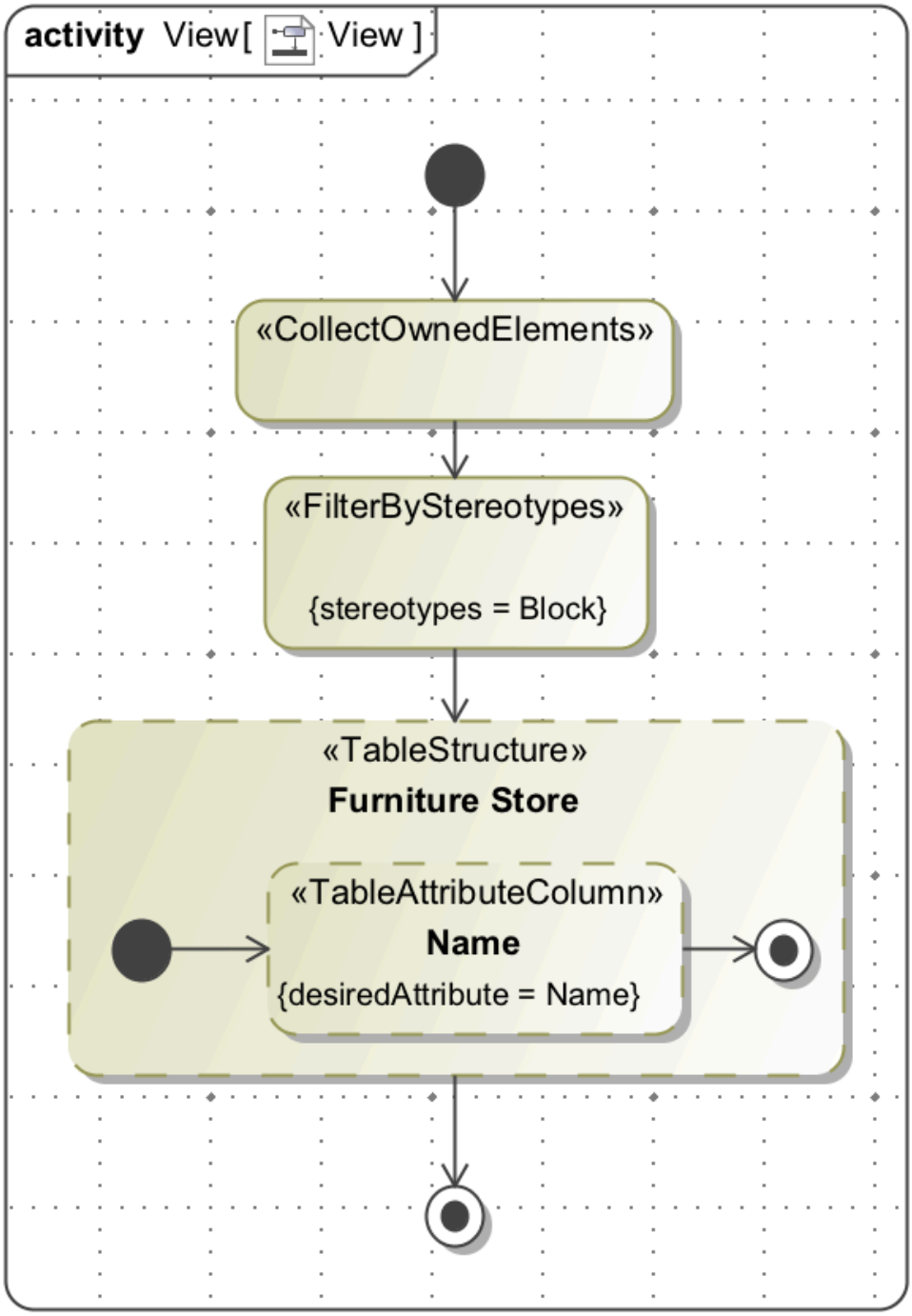
For more detailed information, about using viewpoints and viewpoint methods refer to Section 2.4.2 and the DocGen user guide.

Actions:

1. Create a new View in the Document for the table as described in Section 2.5.1.1
2. Create new Viewpoint as described in Section 2.5.1.2
    - Name new Viewpoint
    - Connect Viewpoint to new View via "Conforms" relationship
3. Drag existing package of blocks onto diagram
    - Connect package and view via "Expose" relationship
4. Create View Point Method Diagram
    - Right click the "operations" section of the Viewpoint > Create Method > Diagram > "Viewpoint Method Diagram" under the MDK section
    - NOTE: If the viewpoint method diagram is created by other methods (such as right clicking the viewpoint in the containment tree), an error may occur during document generation regarding a viewpoint method. This error may be resolved by right clicking the error in the notification window > Set Nested Behavior as Viewpoint Method



1. Start building the activity - the end result will create a table that has the name of the blocks.
    - Create “Initial Node” from left menu
    - Create “CollectOwnedElements” from left menu
    - Create “FilterByStereotypes” from left menu
        - Double click FilterByStereotypes or right click > specification
        - Find “Stereotypes” under StereotypeChoosable > search for "Block" and select Block[Class] Sysml::Blocks
    - Create “Table Structure” from left menu
        - Name the table. This name will display in the view
        - Create “Initial Node” inside the table.
        - Create “TableAttributeColumn” inside the table.
            - Name the column. This name will display in the view
            - Double click TableAttributeColumn or right click > specification
            - Find “Desired Attribute” > select desired attribute from options (ex. name)
        - Create “Activity Final” inside the table.
    - Create “Activity Final” outside the table, in the activity.
    - Connect all control flows



1. From the View Diagram, right click document > MMS > “Generate View Contents Recursively"
2. Table should be visible in view editor

Table 2. Furniture Store

Name
Bookshelf
Table
Chair
Customer1
Customer2
Customer3

Tables with additional columns may be created by adding more TableAttributeColumn elements into the Table Structure.

2.5.1.4 Generate Views Locally

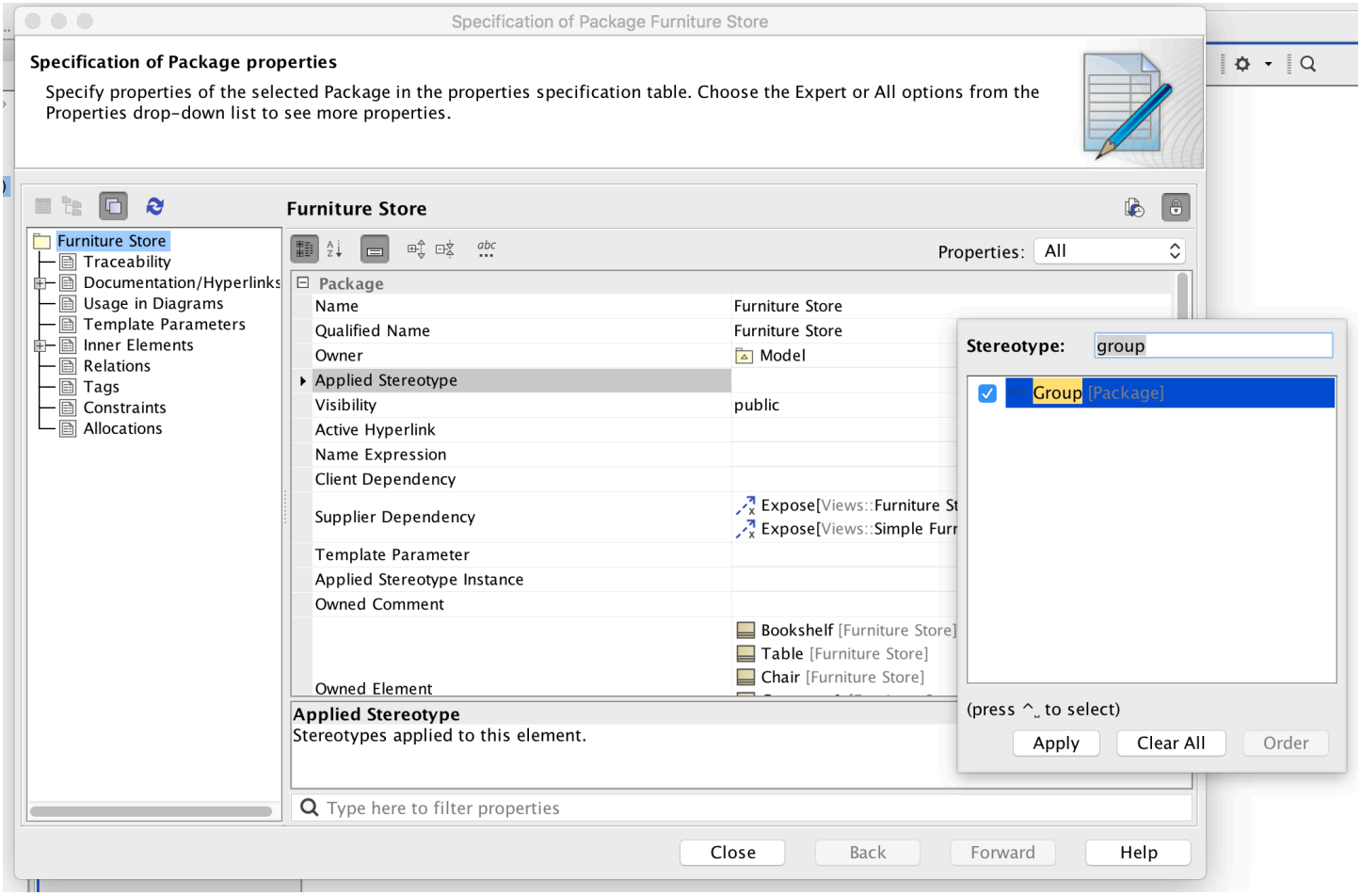
A user can generate Views locally as DocBook XML by right clicking a View > "DocGen" > "Generate Document". The DocBook XML can then be converted to other file formats, such as PDF, Word document, etc., with third-party tools. An example of such a tool is Oxygen XML Editor.

2.5.2 Create a Group of Documents

Groups offer users the ability to better organize projects by allowing Packages to be designated as containers of Documents. Tools that visualize Documents, such as View Editor, would display these Groups in navigation. Documents that are owned by the Group (recursively) would show up under that Group. This is for organizational purposes and applies no semantics.

Actions:

1. Double click a Package that will represent a group or Right click > Specification
2. Click "Stereotype" in the context menu.
3. Add "Group" stereotype in the popup menu and click "Apply".
4. Commit Changes

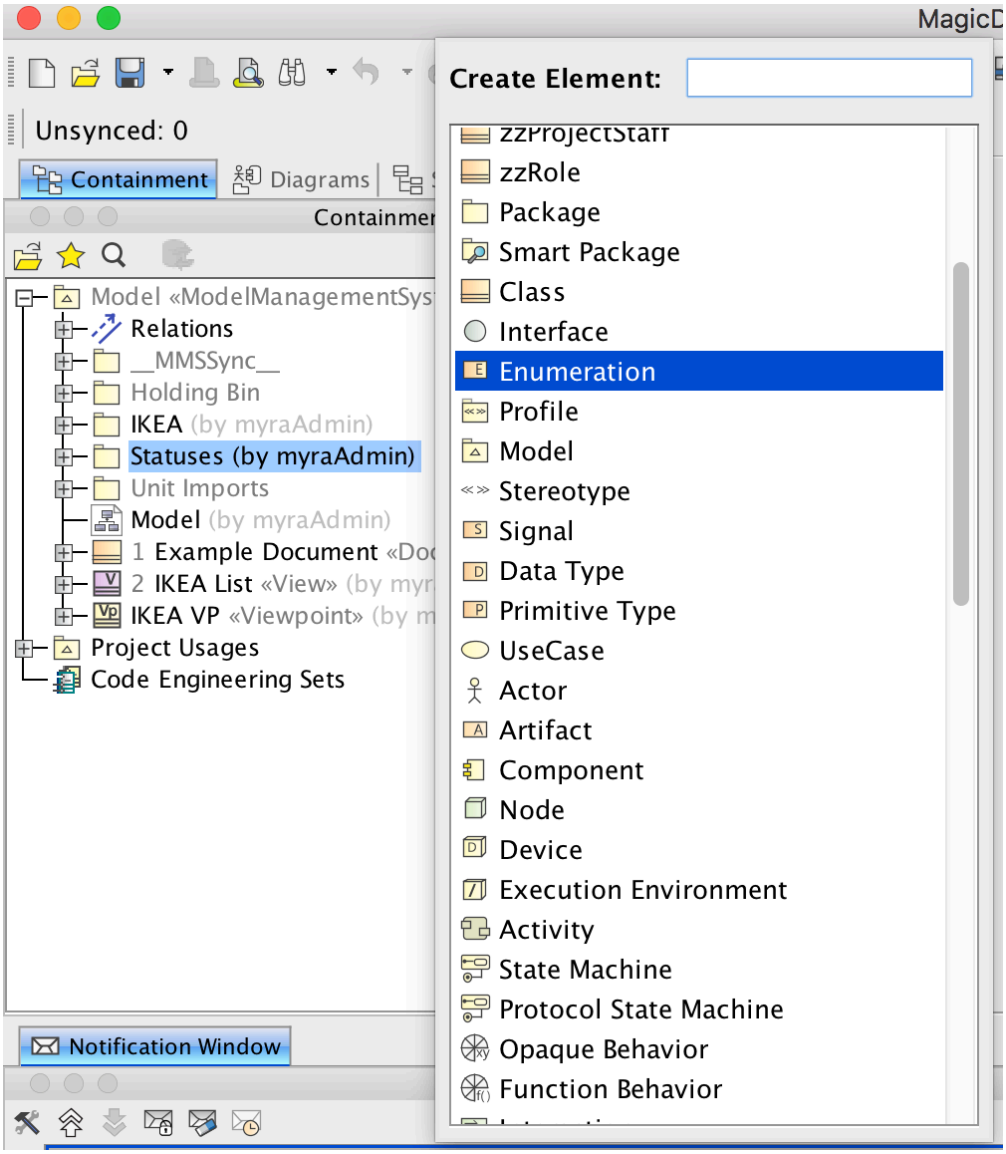


2.5.3 Create Enumerated Values

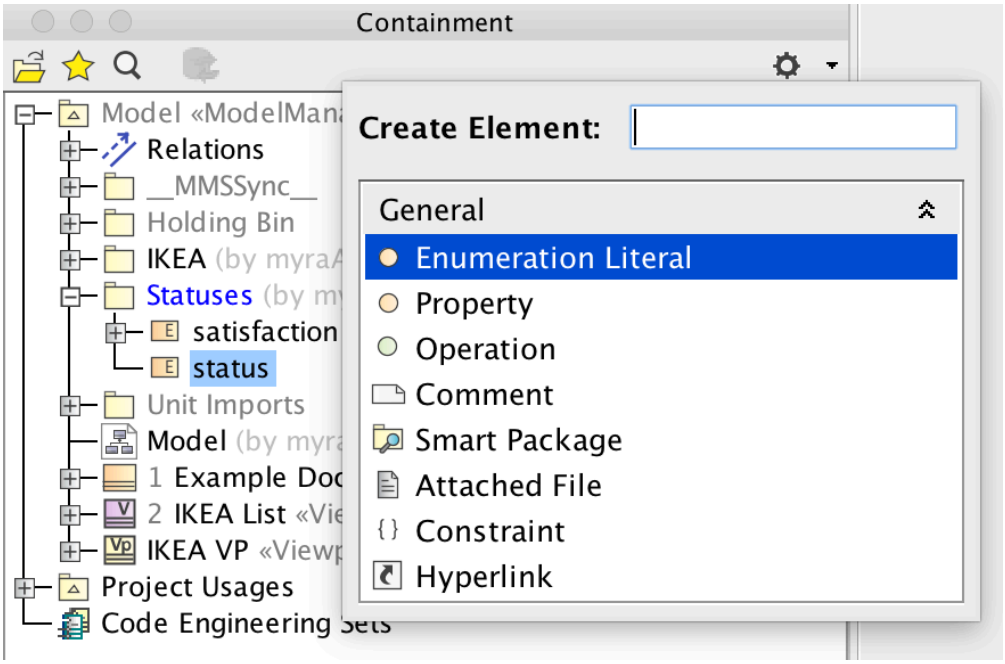
One of the more sophisticated features of View Editor is the option to have values be enumerated values. This means that a user will only be allowed to set an element's value to a specific set of choices. In View Editor, this is shown as a drop down list. This can be extremely useful for elements that are similar in makeup but have different properties and different values.The following instructions demonstrate how to create enumerated values so that they may be seen as drop down lists on View Editor.

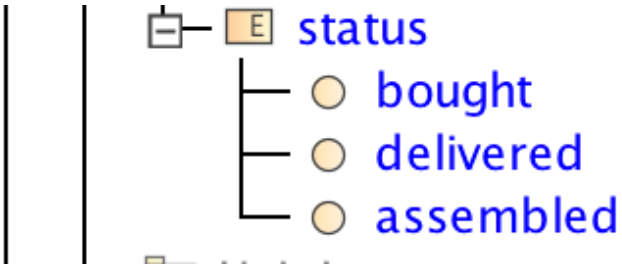
Creating the Enumeration

1. Create an "Enumeration" element
    - Right click on a package that will contain the enumeration > create element > ennumeration
    - Name the enumeration

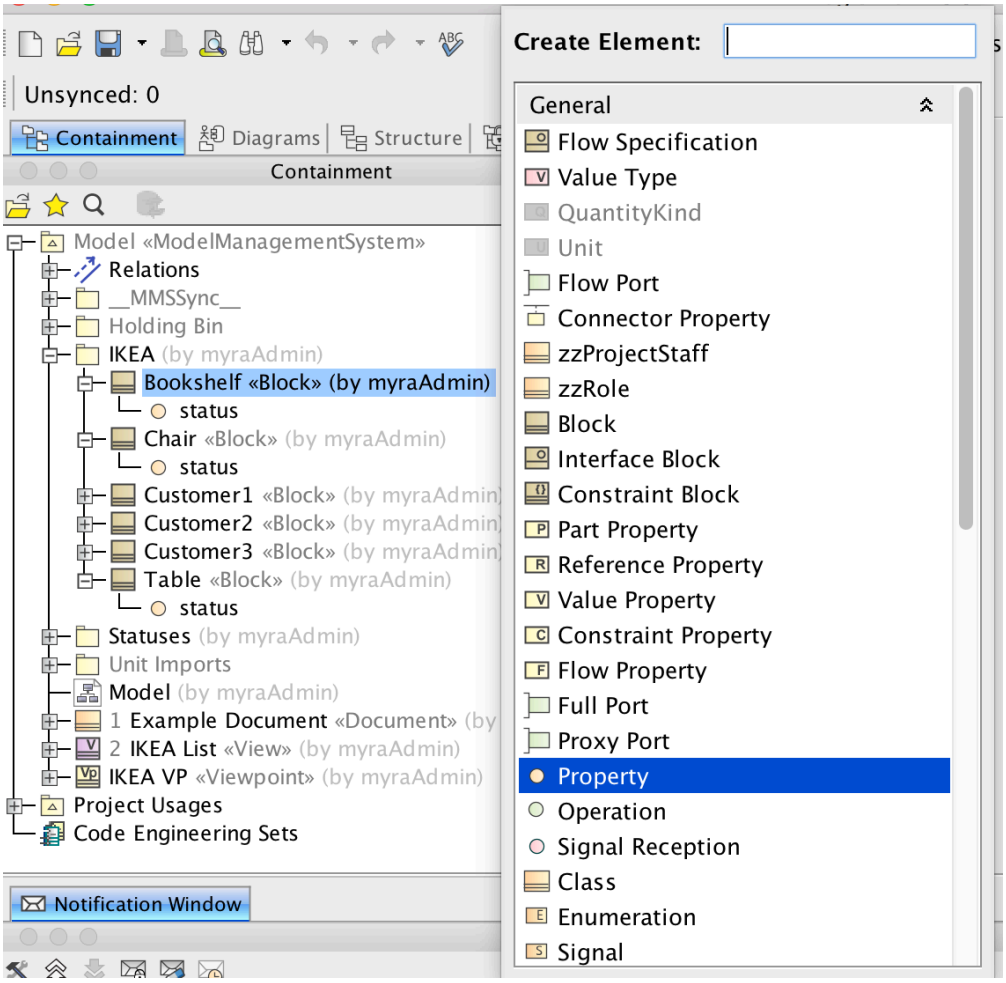


1. Create enumeration literals. Enumeration literals represent the values that the enumeration may hold
    - Right click on the newly created enumeration > create element > enumeration literal
    - Name the enumeration literal
    - Create multiple enumeration literals for each enumeration

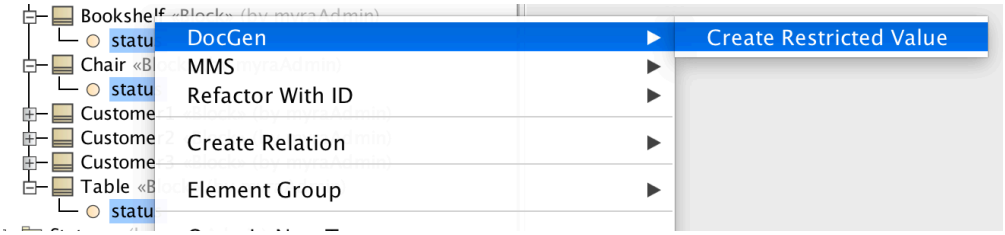




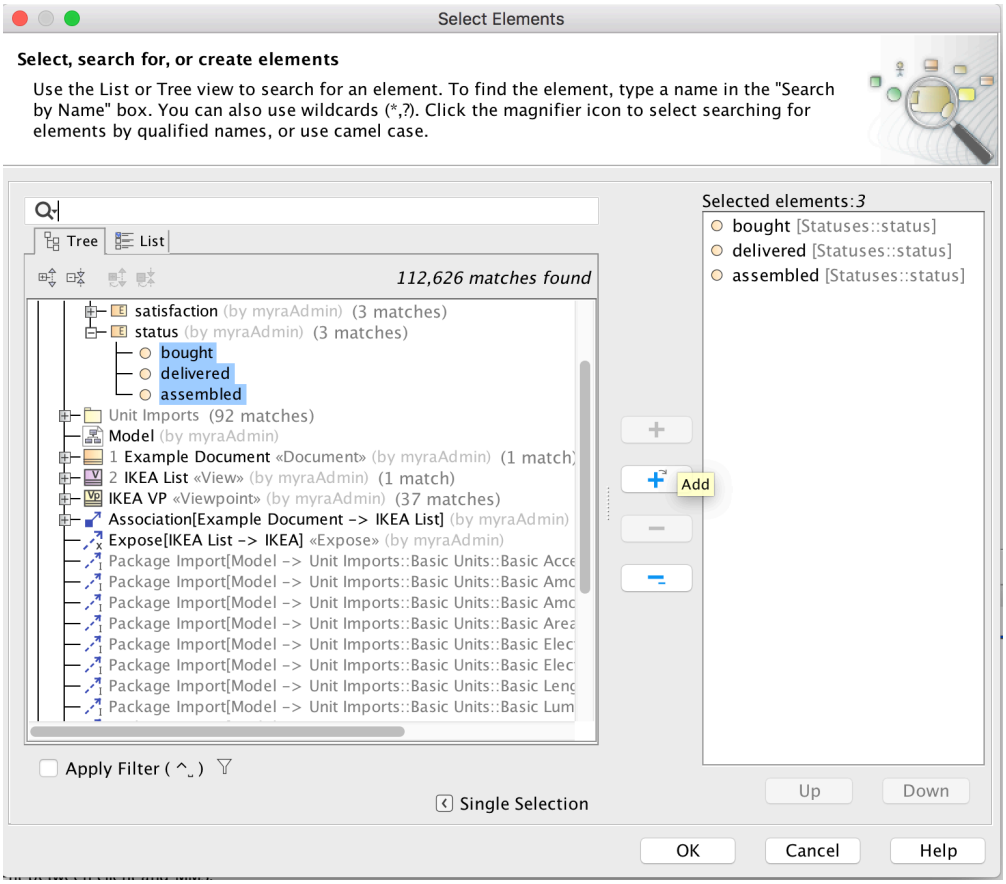
1. For each component that will utilize the enumeration, create a part property
    - Right click component > create element > Part Property
        - Optional Note: Once the part property has been typed by the enumeration, the part may be refactored as a value property and maintains the same functionality
    - Name the value property (preferably the name of the enumeration)



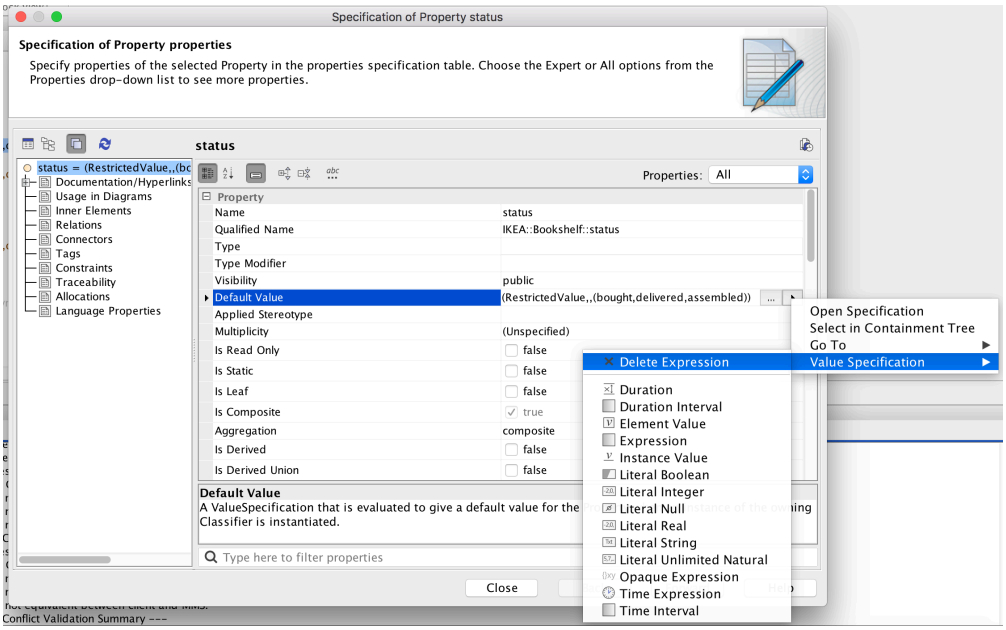
1. Right click the newly created value property > DocGen > Create Restricted Value



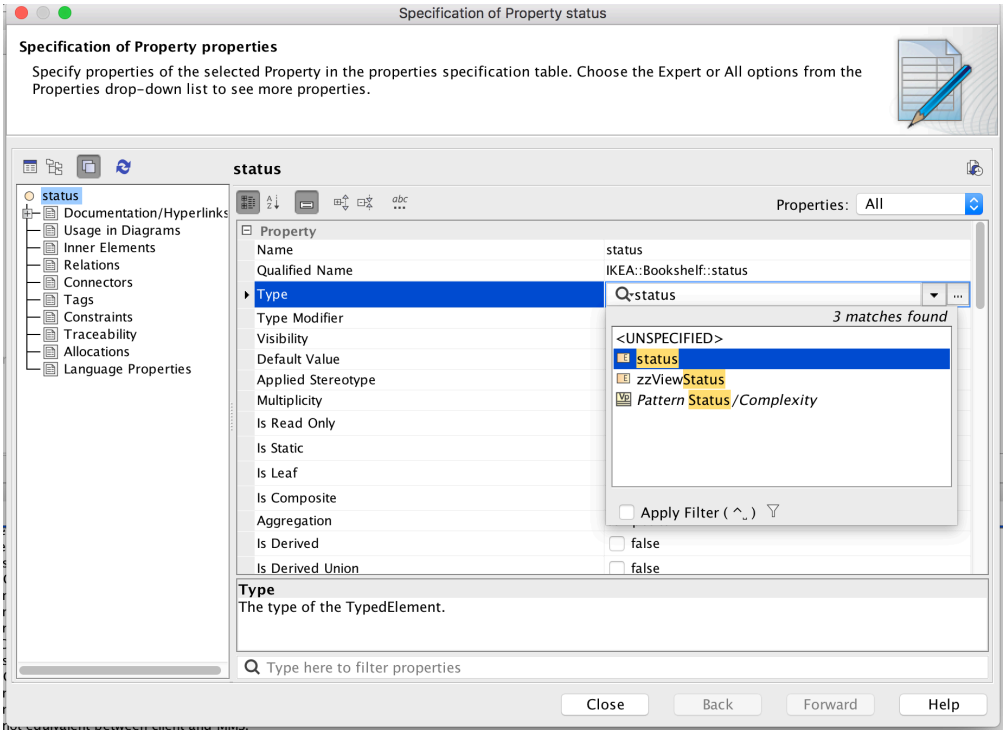
1. From the tree presented in the popup, select the enumeration literals from the enumeration. Select the "+" button to move them to the "selected elements" area. Select "okay"



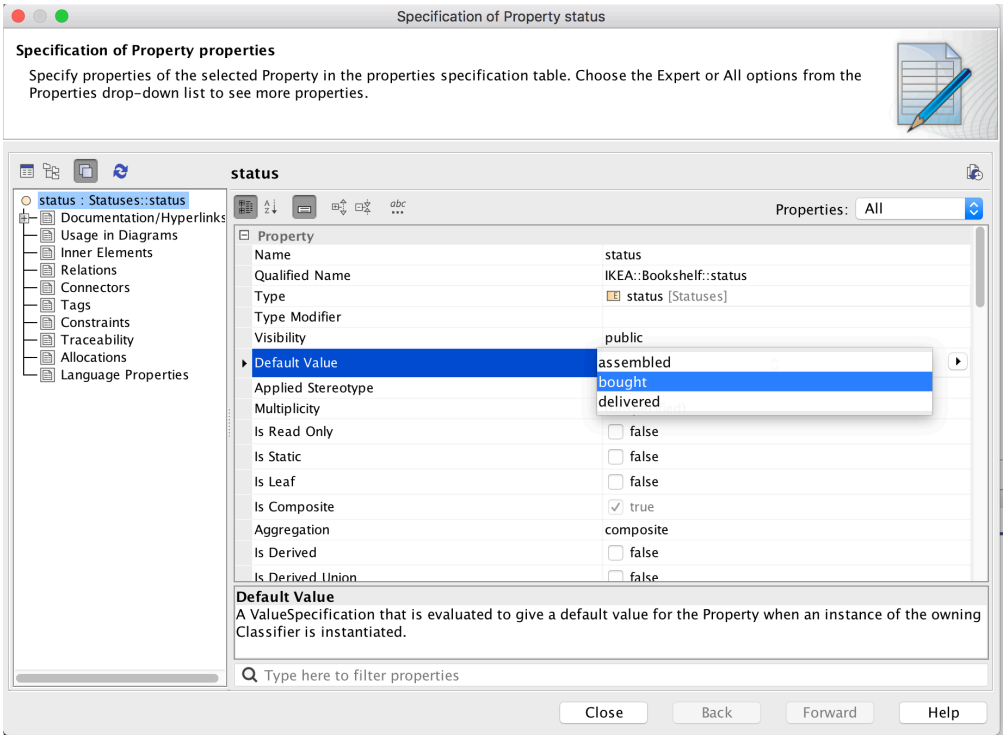
1. In the following popup, select "okay"
2. Open the specification of the value property by double clicking or right click > specification
    - Select the "default value" field > select the arrow to the right of the field > value specification > delete expression



1. Set the "Type" of the value property as the enumeration
    - Open the specification of the value property by either double clicking or right click > specification
    - Drag the enumeration from the containment tree to the "Type" field in the specification or browse for the enumeration



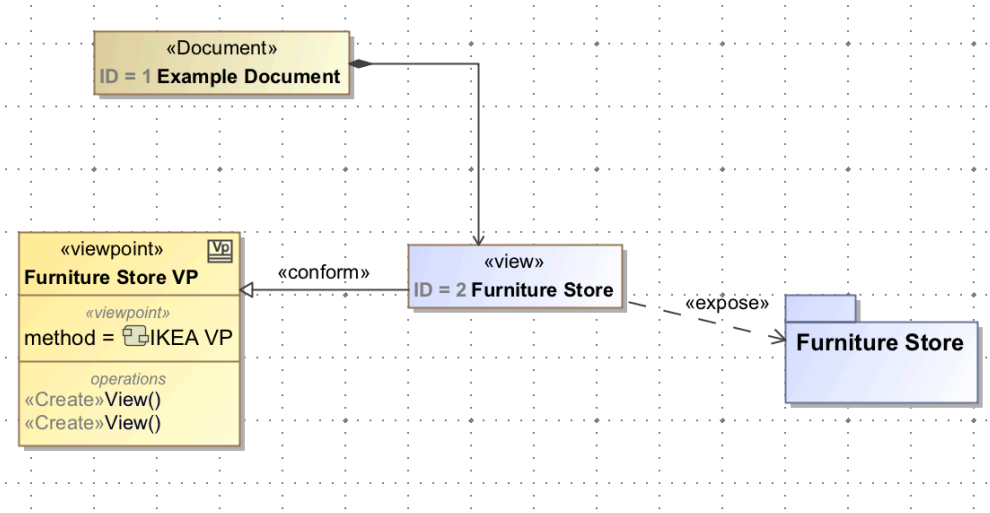
1. Under specification, default value may now be specified as any of the enumeration literals



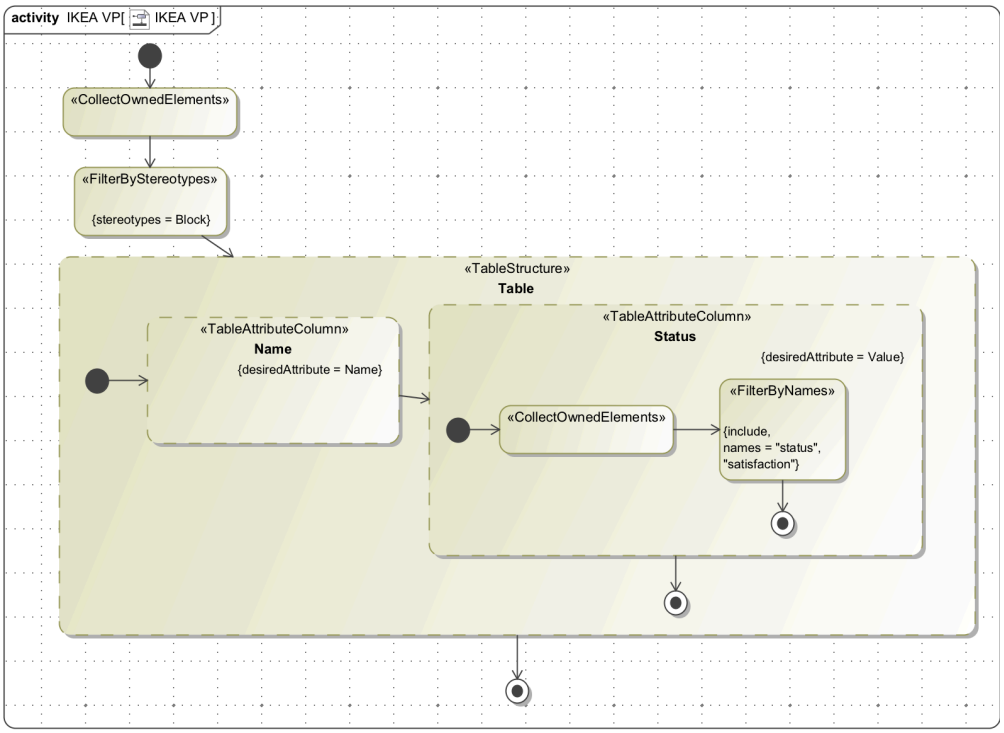
Implementation in the context of a table in View Editor

The following instructions demonstrate how to create enumerated values so that they may be seen as drop down lists in the column of a Rapid Table (Section 2.5.1.3) on View Editor

1. Create a View diagram, document, view, and view point method diagram as described in Section 2.5.1.1 and Section 2.5.1.2



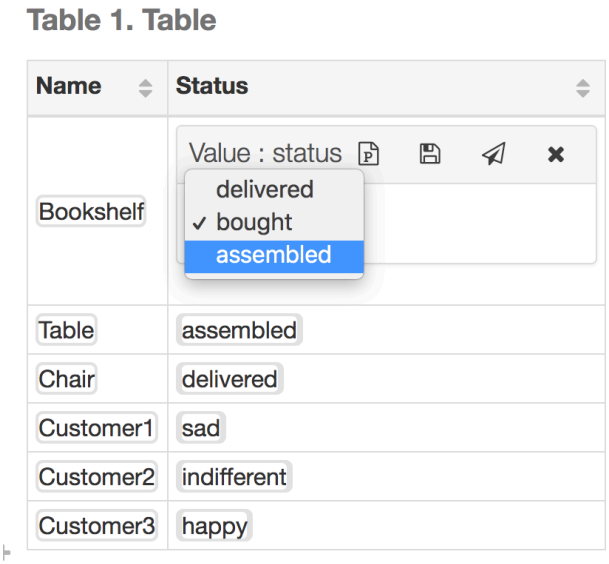
1. In the viewpoint method diagram, include the following elements:
    - Initial node
    - CollectOwnedElements
    - FilterByStereotypes with stereotypes field set to the element type containing properties with enumerations (ex. block)
    - Table Structure
    - Activity Final node
2. Within the Table structure, include the following elements:
    - Initial node
    - TableAttributeColumn with desired attribute (ex. name)
    - TableAttributeColumn with desired attribute set to "value"
    - Activity Final node
3. Within the TableAttributeColumn with the attribute set to "value", include the following elements:
    - Initial node
    - CollectOwnedElements
    - FilterByNames with the NameChoosable field set to the name of the part property with enumerated values
    - Activity Final node
4. Define control flow between all elements on the diagram.



1. Save changes
2. Generate document
    - Right click document > MMS > Generate View Contents Recursively
3. See that table is generated in View Editor that includes the name of the element and its value (one of the enumeration literals). When the value is edited, the list of enumeration literals is displayed as a dropdown menu.

Table 1. Table

Name	Status
Bookshelf	bought
Table	assembled
Chair	delivered
Customer1	sad
Customer2	indifferent
Customer3	happy



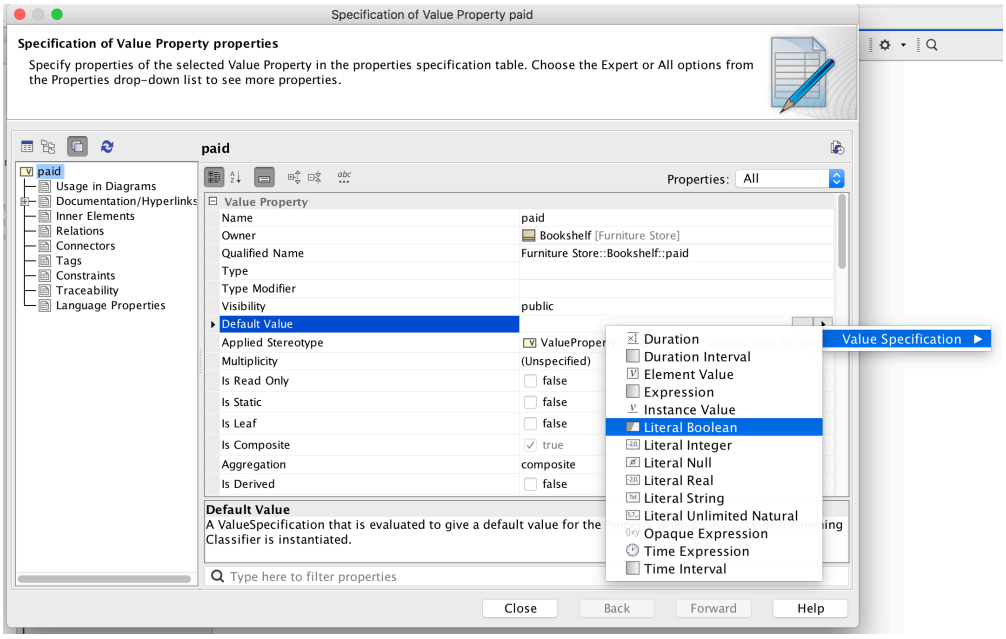
2.5.4 Create Togglable Boolean Values

Another relevant feature of View Editor is the option to have an element's value be Boolean. In View Editor, this is shown as a toggable check box indicating if a value is "True" or "False". This can be extremely useful for all elements that require a quick Boolean answer.

These instructions demonstrate how to create togglable Boolean values so that they may be seen as check boxes on View Editor. This example builds off of Section 2.5.1.3 for the first table column and Section 2.5.3 for the second table column.

Create Togglable Boolean

1. For each component that will utilize a togglable boolean, create a value property
    - Right click component > create element > value property
    - Name the value property
2. Set the default value of the value property to literal boolean
    - Double click the value property or right click > specification
    - In the default value field, select the arrow to the right of the field box > value specification > literal boolean
    - The default value is now a togglable true/false



Implementation in the context of a View Editor Table

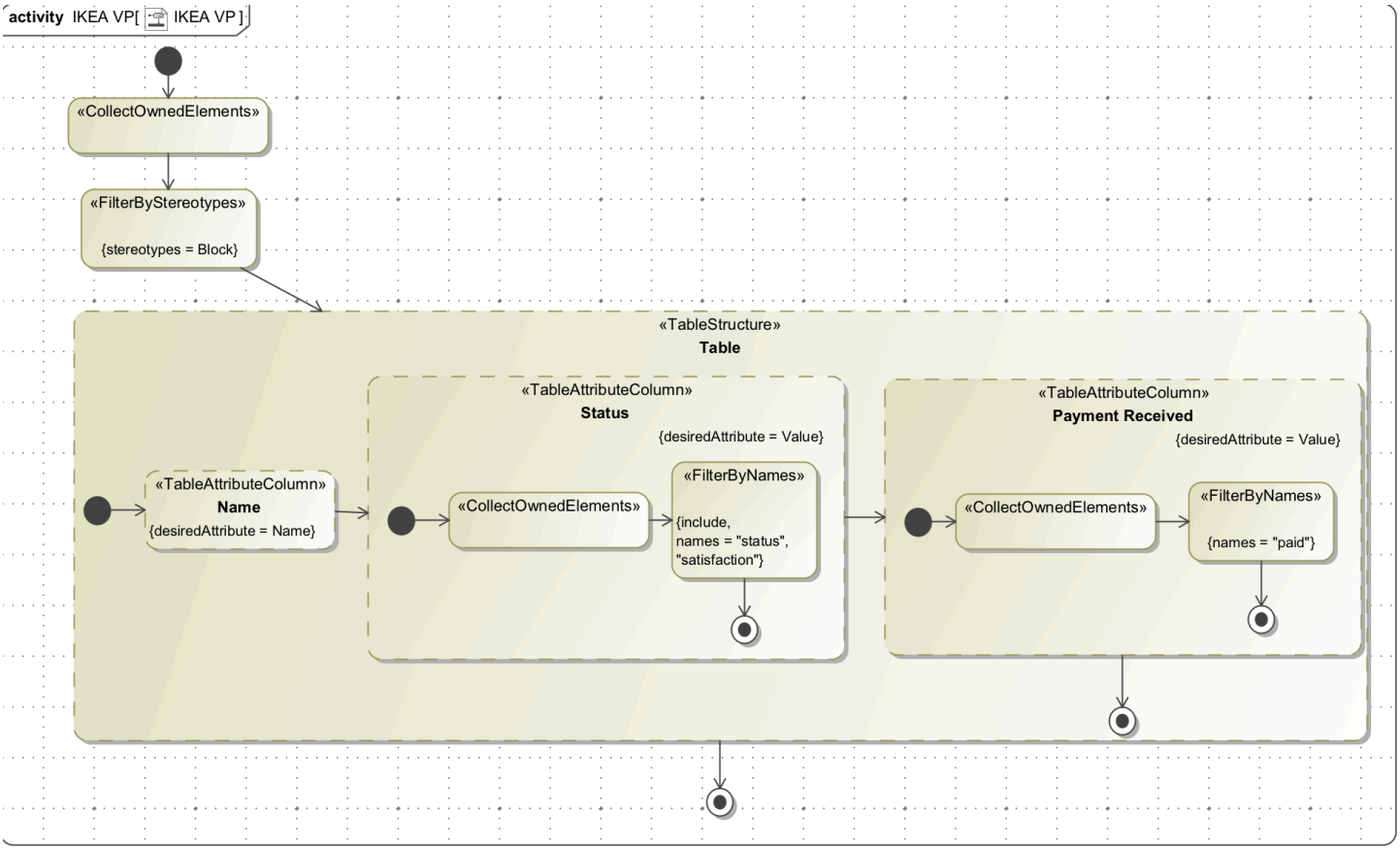
The following instructions demonstrate how to create enumerated values so that they may be seen as drop down lists in the column of a Rapid Table (Section 2.5.1.3) on View Editor.

1. Create a View diagram, document, view, and view point method diagram as described in Section 2.5.1.1 and Section 2.5.1.2

1. In the viewpoint method diagram, include the following elements:
    - Initial node
    - CollectOwnedElements
    - FilterByStereotypes with stereotypes field set to the element type containing value properties with literal boolean (ex. block)
    - Table Structure
    - Activity Final node
2. Within the Table structure, include the following elements:
    - Initial node
    - TableAttributeColumn with desired attribute (ex. name)
    - TableAttributeColumn with desired attribute set to "value"
    - Activity Final node
3. Within the TableAttributeColumn with the attribute set to "value", include the following elements:
    - Initial node

- CollectOwnedElements
- FilterByNames with the NameChoosable field set to the name of the value property with a literal boolean
- Activity Final node

4. Define control flow between all elements on the diagram. The figure below shows the viewpoint method diagram, including the TableAttributeColum from the Section 2.5.3 example.



1. Save Changes
2. Generate document
    - Right click document > MMS > Generate View Contents Recursively
3. See that table is generated in View Editor that includes the name of the element and its value (either true or false). When the value is edited, the user can toggle between true or false.

Table 1. Table

Name	Status	Payment Received
Bookshelf	assembled	true
Table	assembled	true
Chair	delivered	false
Customer1	sad	
Customer2	indifferent	
Customer3	happy	

Table 1. Table

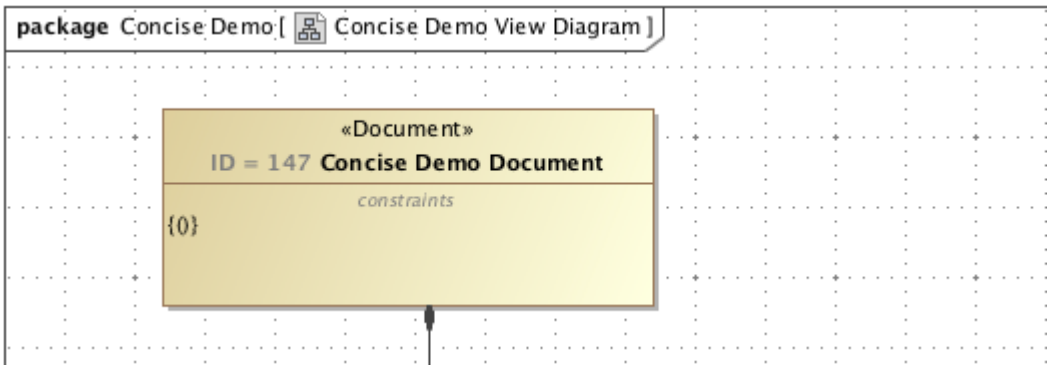
Name	Status	Payment Received
Bookshelf	assembled	Value : paid true
Table	assembled	true
Chair	delivered	false
Customer1	sad	
Customer2	indifferent	
Customer3	happy	

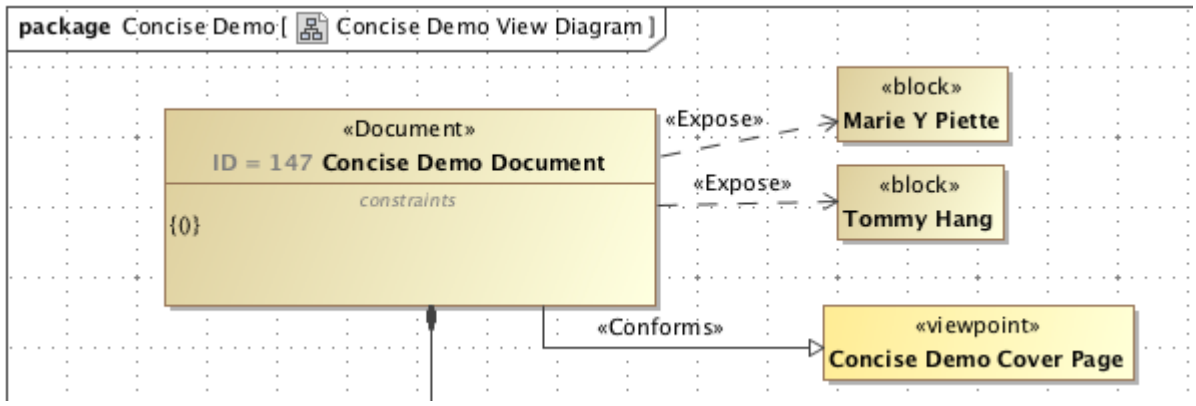
2.5.5 Create a Reusable Cover Page

A user can create a reusable cover page by constructing a ViewPoint with a ViewPoint method. Once the method is made, a user can apply it to any View, and specifically to the Cover Page of a document. See Section 2.4.2 for more information about using them in general and see DocGen for more detailed information.

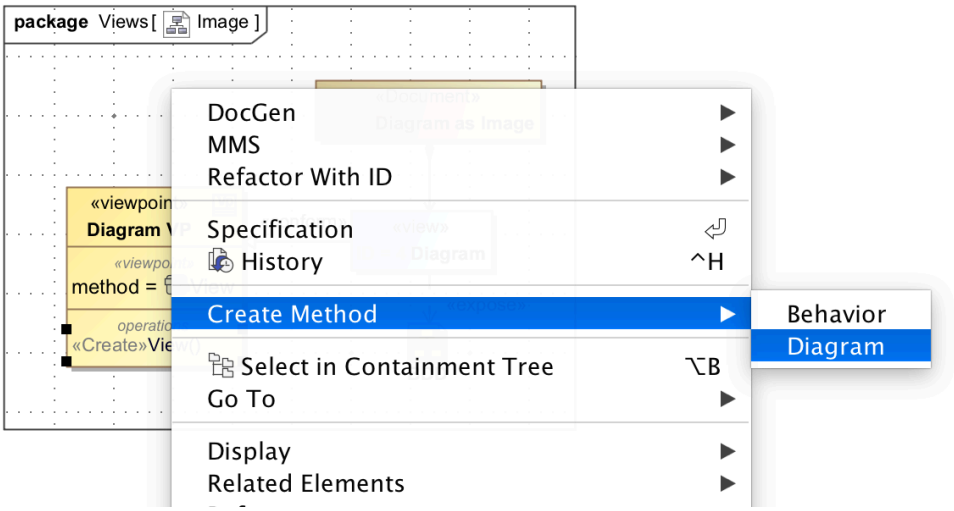
This model based cover page will be reflected when generating PDFs as well. See Save As for more information on saving a document (with said cover page) in different forms, including generating a PDF.

The following instructions is an overview about how to create a simple, reusable Cover Page for a document that already exists:

1. Find a Document that already exists (Concise Demo Document)
2. Open the document's View Diagram by double clicking
    1. If it has/belongs to a View Diagram, double click the Diagram and view in main window
        1. To find the View Diagram, Right Click the Document>>Go To>> Usage in Diagrams ....select the appropriate View Diagram
    2. If it does not have a View Diagram,
        1. Right click the containing package>>Create Diagram>>MDK>>View Diagram
        2. Name the View Diagram
        3. New Diagram should be shown in main window
        4. Drag the Document onto the View Diagram
3. 

3. Set Up Document Cover Page relationships
    1. Note: Although these instructions are specifically in regards to generating a Document Cover Page, the ViewPoint can be applied to any View and would appear the same.
2. 

3. Create a new Viewpoint
    1. While on the View Diagram, drag and drop the "Viewpoint" icon from the tools pane onto the Diagram
    2. Name the new Viewpoint ("Concise Demo Cover Page")
    3. Connect the Document to the new Viewpoint via a "Conforms" relationship
4. Drag wanted elements that will be used for constructing the Cover Page
    1. Connect Document to said elements via "Expose" relationship
4. Create View Point Method Diagram
    1. Right click the "operations" section of the Viewpoint > Create Method > Diagram > "Viewpoint Method Diagram" under the MDK section
    2. NOTE: If the viewpoint method diagram is created by other methods (such as right clicking the viewpoint in the containment tree), an error may occur during document generation regarding a viewpoint method. This error may be resolved by right clicking the error in the notification window > Set Nested Behavior as Viewpoint Method



1. Create ViewPoint Method Activity
    1. The following activity will create a Cover Page with several presentation elements including: a title, an image, a table of contributors, several paragraphs, and signature lines with badge numbers/names of the "owners", along with some stylistic. Note: in this example, the owners and contributors are the Person blocks exposed to the Document; the viewpoint method would change depending on the context of the model i.e. a user could navigate through the model to find real owners, roles, work packages, etc. if that is what is required.
    2. To keep this User Guide succinct, the activity will only be briefly described below. More information about the used functions can be found in DocGen and later there will be specific documentation about different reusable activities

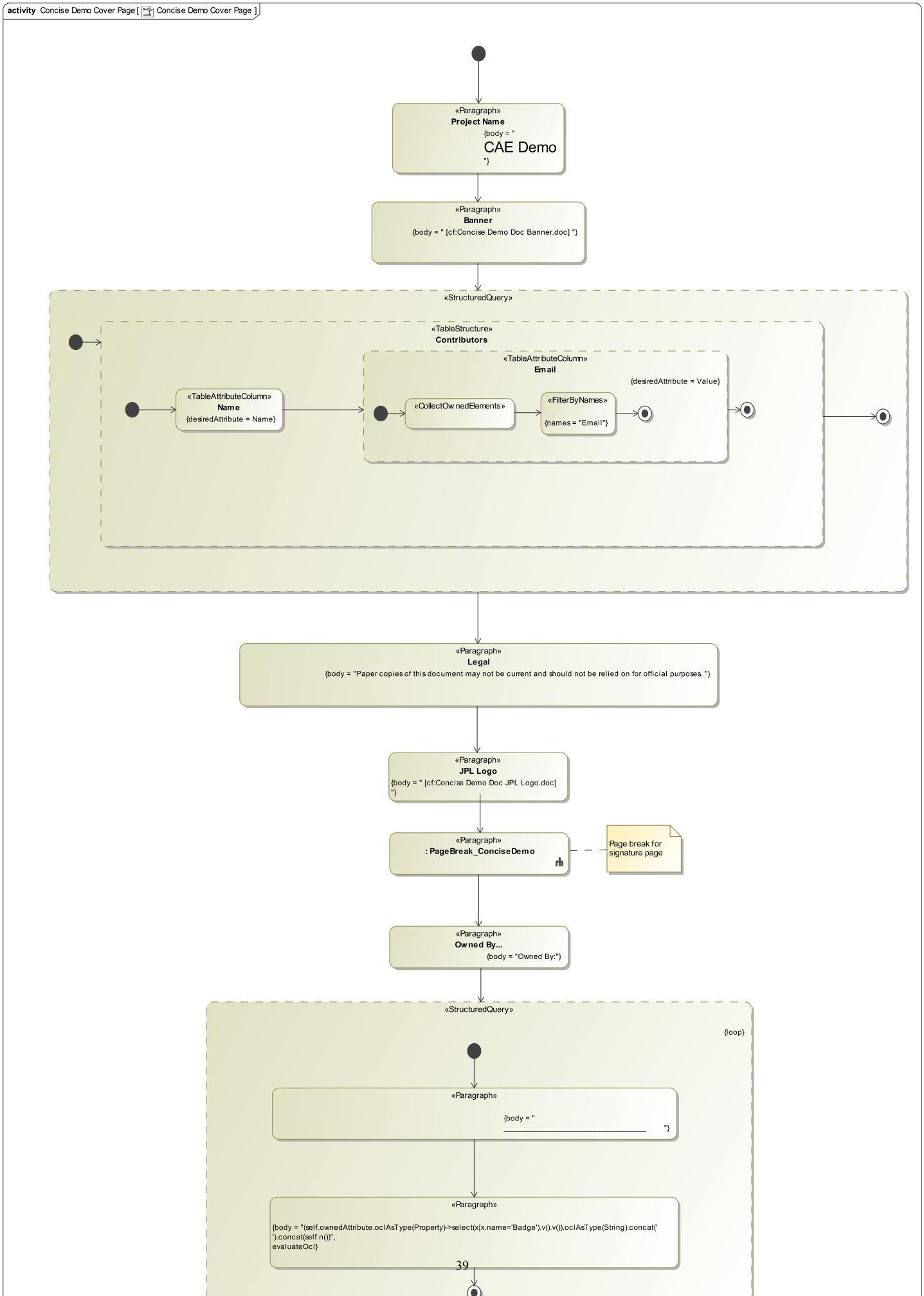

activity Concise Demo Cover Page [Concise Demo Cover Page]
«Paragraph»
Project Name
{body = "
CAE Demo
"}
«Paragraph»
Banner
{body = " [cf:Concise Demo Doc Banner.doc] "}
«StructuredQuery»
«TableStructure»
Contributors
«TableAttributeColumn»
Email
{desiredAttribute = Value}
«TableAttributeColumn»
Name
{desiredAttribute = Name}
«CollectOwnedElements»
«FilterByNames»
{names = "Email"}
«Paragraph»
Legal
{body = "Paper copies of this document may not be current and should not be relied on for official purposes. "}
«Paragraph»
JPL Logo
{body = " [cf:Concise Demo Doc JPL Logo.doc]
"}
«Paragraph»
: PageBreak_ConciseDemo
Page break for signature page
«Paragraph»
Owned By...
{body = "Owned By:"}
«StructuredQuery»
{loop}
«Paragraph»
{body = "
_____
"}
«Paragraph»
{body = "(self.ownedAttribute.oclAsType(Property)->select(x|x.name='Badge').v().v()).oclAsType(String).concat('
').concat(self.n())",
evaluateOcl}

Figure 6. Concise Demo Cover Page

The Concise Demo Cover Page is an example of a reusable cover page. It creates a cover page that includes the following:

1. A title - "CAE Demo" is displayed at the top of the page and centered, according to the Paragraph function and its internal html
2. An image - This Paragraph function includes a cross reference to a model element, whose documentation is an image.
3. A table of contributors - This Rapid Table uses the Exposed blocks (in this case, Personnel blocks) to create a table of their Name and Email attributes.
4. A "legal" paragraph - Similar to the title, this paragraph function creates and formats the written texts.
5. A logo - See #2, an image
6. A page break - this is an example of using an embedded reusable activity. In this case, there is an activity that creates a html page break and it is used here instead of creating a new one. This allows for multiple pages to be created as part of the "Cover Page"
7. A identification paragraph - "Owned by_", see #1 and #4,
8. Signature lines with badge numbers/names of the "owners" - this structured query has several aspects:
    1. It's in a separate structured query, not only for clean separation of functions, but also so that it can "loop", meaning that it will repeat the internal functions for as many times as intended. In this case, there are 2 "owners" that are exposed to the VP so it repeats twice.
    2. The first paragraph is simply the html for a line and provides a place for a signature
    3. The second paragraph is constructed of OCL that finds the Owner's badge number and name and concatenates them. This specific combination may not be what is most used for most formal documents, but it demonstrates how someone could use OCL to navigate through the model and get the desired attributes.

1. Commit to TW
    1. Collaborate>>Commit Changes to Server
2. Generate the Cover Page
    1. In the View Diagram or in the Containment Tree, Right Click Document>>MMS>>Generate View
3. See Cover Page on VE
4. To see how the Cover Page will be appear as a PDF, Generate PDF of View (note: other PDF generation options are in the provided view)
    1. Below is how the Cover Page of the Concise Demo Document appears according to the Concise Demo Cover Page above:

1.

Concise Demo Document

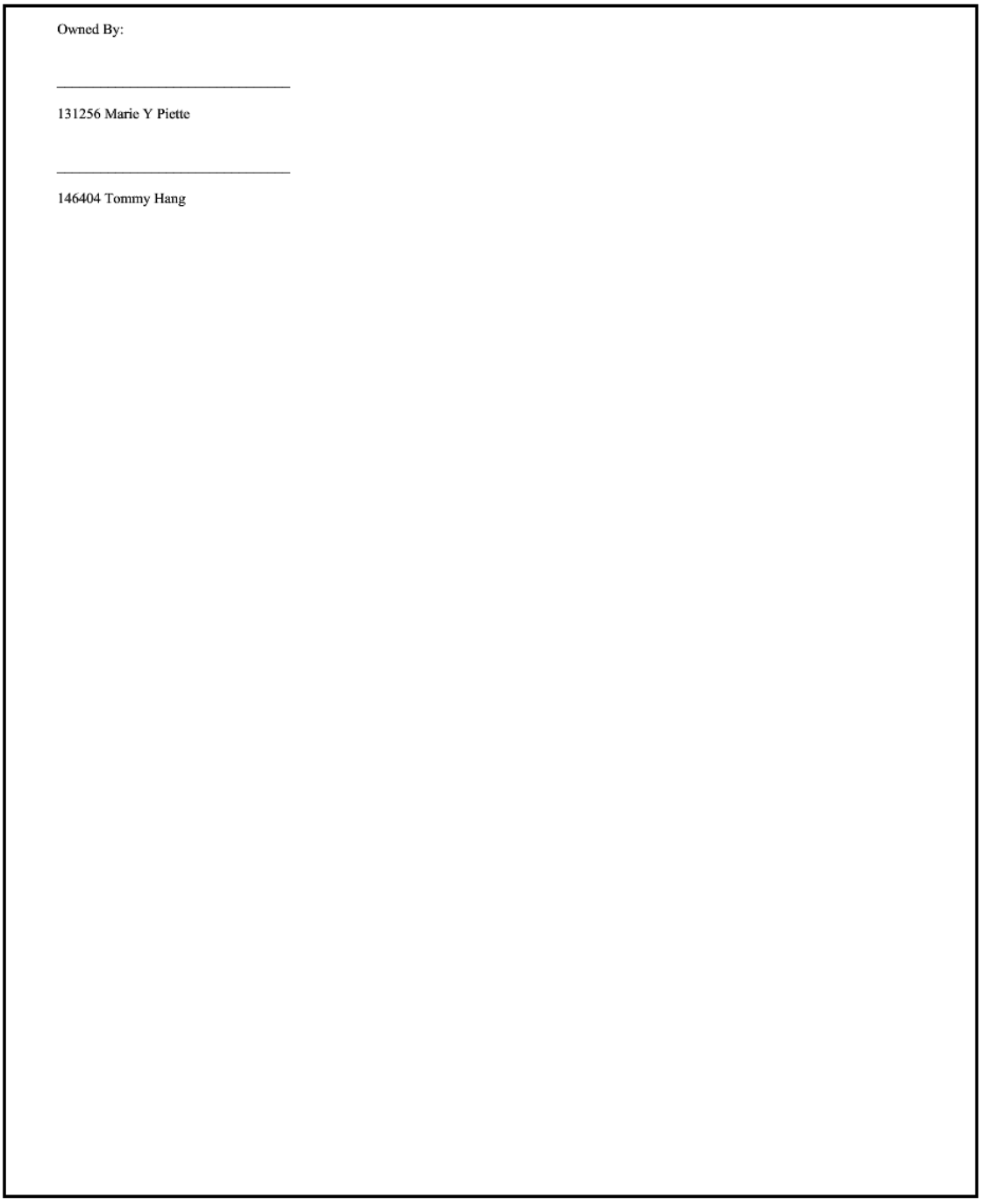
A playground for the demonstration of various MDK-to-View Editor features.

CAE Demo

Table 1. Contributors

Name	Email
Marie Y Piette	Marie.Y.Piette@jpl.nasa.gov
Tommy Hang	Tommy.Hang@jpl.nasa.gov

Paper copies of this document may not be current and should not be relied on for official purposes.

2. 

3 Developer Resources

3.1 API

The Javadoc for MDK is included in the plugin and can be found in the MagicDraw plugins folder after installation. Note that only the classes found in `gov.nasa.jpl.mbee.mdk.api` are intended to be used by other plugins/tools and that the rest of the code can change without any notification.

3.2 MDK Environment Options

Options for advanced users to customize certain behaviors or enable MDK functionality are available in MagicDraw's Environment Options.

- LOG_JSON: Enables logging of MMS request and response information, including sent and received JSON, to the MagicDraw log.

To access these settings, navigate to the "Options Menu" -> "Environment" dialog, and select the "MDK" section.

Several [DEVELOPER] options are also available, but are generally hidden from users. They are included here for completeness - we strongly recommend they not be modified unless for development purposes as they will likely result in data loss.

- PERSIST_CHANGELOG: [DEVELOPER] Enables persisting of the changelog in the _MMSSync_ package for uneditable model elements. Disabling this option will cause these changelogs to be lost after CSync and may cause loss of model parity.
- ENABLE_CHANGE_LISTENER: [DEVELOPER] Enables listeners for model changes in MagicDraw and MMS. Disabling this option will cause changes to not be tracked and may cause loss of model parity.
- ENABLE_COORDINATED_SYNC: [DEVELOPER] Enables Coordinated Sync on Teamwork Cloud project commit. Disabling this option will cause CSync to be skipped on Teamwork Cloud commit and may cause loss of model parity.