MetaModelAgent

for IBM Rational® Software Architect Designer and RealTime Edition
HCL RealTime Software Tooling and Eclipse Papyrus™

Modeling User Manual

MetaModelAgent version 4.4.1

AO-MMA-021 • 2020-01-07

ADOCUS

Contents

1 Introduction

This document is the main user manual for domain-specific modeling capabilities in MetaModelAgent. MetaModelAgent (MMA) is an Eclipse add-on that is installed on top of one of the supported host tool and is accessed by extended menus and views within the Eclipse workbench. MetaModelAgent supports the following host tools:

- IBM Rational® Software Architect Designer (RSAD)
- IBM Rational® Software Architect RealTime Edition (RSARTE)
- HCL RealTime Software Tooling (RTist).
- Eclipse Papyrus™

The main purpose of MetaModelAgent is to support the developers to create models which fulfill organization-specific UML-based domain-specific model languages (DSMLs). MetaModelAgent makes it easy to define and use DSMLs within the host tool. But even if you are using plain UML or UML-RT without any customization, you will find a lot of useful features in MetaModelAgent.

Besides the support for domain-specific modeling, MetaModelAgent provides extensive support for static model analysis. The model analysis capabilities and features are described in the separate *MetaModelAgent Model Analysis Manual*.

To use the MetaModelAgent functionality after installation, you must have access to a valid license. There are several different kind of licenses available with different levels of features. This user manual covers all kinds of licenses. Depending on the license you are using, some features described in this manual may not be applicable. See **MetaModelAgent License Management Manual** for details about the different licenses and how to order and manage licenses.

Beside this manual, there are three other manuals available in the Eclipse Help-system after installation:

MetaModelAgent Model Analysis Manual	Describes how to use MetaModelAgent for model analysis using the built-in metamodels or your own metamodel.
MetaModelAgent Metamodeling Manual	Described how to develop your own domain-specific metamodel for usage in MetaModelAgent.
MetaModelAgent License Management	Described how to require and install licenses to be able to use MetaModelAgent.

More information about MetaModelAgent is available on www.adocus.com and www.metamodelagent.com.

1.1 The concept of MetaModelAgent

Common domain-specific modeling guidelines (DSML-guidelines) within an organization are important in order for the developers to be able to read and understand each other's models. Unfortunately, it is rather difficult to understand modeling guidelines, as they consist of model structure, name conventions, restrictions in the use of relations, stereotype definitions, element properties etc.

MetaModelAgent offers an alternative and complementary user interface, making it simpler to comply with organization specific guidelines, thus saving time and money.

MetaModelAgent uses DSML-guidelines expressed in a so called metamodel. A MetaModelAgent metamodel is an UML-model which describes the DSML-guidelines. The metamodel must express all model constructions that should be valid in a model.

Model constructions that are not expressed explicitly in the metamodel are therefore not allowed in the model.

By expressing the DSML-guidelines as a metamodel, unambiguous guidelines are achieved and the risk for inconsistence is minimized. A metamodel is normally developed once, in a project or organization, and can thereafter be used by everyone that has access to MetaModelAgent.

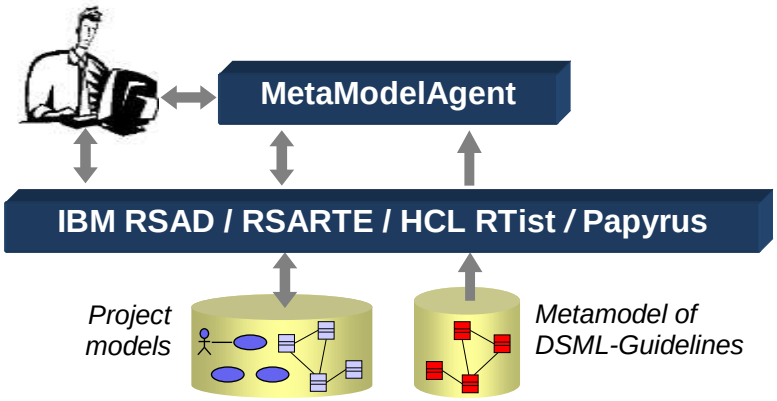


Figure 1: *MetaModelAgent supplements the host tools with an additional "guideline-sensitive" user interface*

The business value of using MetaModelAgent depends highly on the quality of the metamodels. Well defined and documented metamodels are the key to achieve a high business value of using MetaModelAgent.

If you don´t have ambition to develop any metamodel of your own, you may use MetaModelAgent with one of its included metamodels for standard UML and UML-RT.

MetaModelAgent does not leave any MetaModelAgent-unique footprints in your models, which means that models created by using MetaModelAgent features can continue to be developed and maintained using the original capabilities in the host tool.

1.2 Feature overview

MetaModelAgent offers additional views, decorators and wizards to the workbench that uses the information in the metamodels to better support the user in performing their modeling tasks.

MetaModelAgent offers the following capabilities:

- Automatically validate the model against the metamodel with support for quick-fixes.
- Add new correct model items and edit existing items according to the metamodel, focusing on significant properties only. The risk of violating the metamodel is eliminated, as only permitted items can be created.
- Adapt a pre-existing item according to the metamodel automatically, i.e. classification. A model can thus be designed without the use of a metamodel and subsequently be adapted to comply with the guidelines.
- Access to context sensitive integrated guidance based on metamodel documentation.
- Analyze one or several models simultaneous using property tables, trace matrixes, relationship trees and different kind of charts.
- Publish web based model documentation and guidelines based on the metamodel.

An important notice is that the MetaModelAgent functionality does not disable any built-in functionality in the host tool. MetaModelAgent only provides additional features to the tool. All original features are still available.

1.3 Typical usage scenarios

To which degree MetaModelAgent will be used to support the modeling effort is controlled by the end user. The usage may be divided into distinct levels:

1. **On Demand Validation only**
   The modeling work are being performed by using the standard UI in the host tool. MetaModelAgent is activated for the model only for validating the model against the metamodel when needed. Any correction needed of the model is being done by the standard UI or by using the MMA quick fix facility.
2. **Interactive Guidance**
   MetaModelAgent is being activated for the model when the model is opened. MetaModelAgent's additional UI wizards and views are used to support the end user in creating a correct the model according to the metamodel.
3. **Active Imposer**
   MetaModelAgent is being activated for the model when the model is opened and the preferences are set so that MetaModelAgent will prohibit any invalid element being added. Any trial to violate the metamodel will be rejected.
4. **Static Model Analysis**
   Models can be analyzed in a number of different ways to get an understanding of non-trivial complexity and behavior. Analysis may be performed out-of-the box or in context of your own metamodel and will be displayed in tables, matrixes, trees and graphs. Please refer to the *MetaModelAgent Model Analysis Manual* for details.

1.4 Usage without any customized metamodel

You will get maximum business value out of MetaModelAgent if you define your own customized metamodel according to your own DSML-metamodel.

But even if you are using plain UML or UML-RT[1] and do not define any customized metamodel, you will get significant business value by using the built-in metamodel for UML and UML-RT, especially when it comes to model analysis and web publishing.

By using one of these built-in metamodels you will never get any problems reported but you will make your modeling effort more efficient by using the wizards and views for creating and editing model items with access to integrated UML guidance, and you can analyze your model using the table and matrix views. Finally you can generate web-based model reports that can be published for anyone to read.

[1] UML-RT is only available in RSARTE and HCL RTist

1.5 Reading instruction

To be able to understand this user manual, basic knowledge of the Unified Modeling Language (UML) and the host tool being used, is required.

This manual is designed as a reference book. It includes the following chapters:

Chapter 1: *Introduction*, this chapter; includes an introduction to MetaModelAgent.

Chapter 2: *Terminology,* explains terms that are used widely in this manual.

Chapter 3: *The concept of metamodels*, explains the concept of metamodels and gives an overview of the functionality provided by MetaModelAgent.

Chapter 4: *Start using MetaModelAgent*, describes fundamental features for invoking the MetaModelAgent functionality.

Chapter 5: *Adding new items to a model*, describes how to use the MetaModelAgent features for adding items to a model.

Chapter 6: *Correcting* an existing item, describes how to use the MetaModelAgent features for correcting existing or newly added items.

Chapter 7: *Editing an existing item*, describes how to use the MetaModelAgent addition to the property view for editing item properties.

Chapter 8: *Web publishing*, describes how to generate a web-based model report using MetaModelAgent and how to generate web-based modeling guidelines from a metamodel.

Chapter 9: *Views*, *D*escribes the additional Eclipse workbench views provided by MetaModelAgent. The analysis view are only described briefly. For detailed description of the analysis views please refer to the *MetaModelAgent Model Analysis Manual*.

Chapter 10: *Decorators*, describes how to use the additional Eclipse workbench decorators provided by MetaModelAgent.

Chapter 11: *Preference settings,* describes the usage of the MetaModelAgent preference settings.

Chapter 12: *Utility functions*, describes utility functions for different tasks. Currently only one utility function is provided.

Chapter 13: *Headless validation,* describes in detail how to execute the model validation functionality without any interaction with the UI.

Chapter 14: *Public API,* describes in detail how to interact with MetaModelAgent using your own plug-in for activation and validation tasks.

Chapter 15: *Known limitations,* a list of known issues that could affect the expected behavior.

IMPORTANT: Most of the user interface elements provided by MetaModelAgent such as views, wizards, dialogs, menu entries and tool palette entries are dynamically adapted to the current metamodel being used. All figures in this manual showing user interface snapshots are therefore only illustrative examples and may differ from how there looks in your environment.

The figures are also a mixture from using MetaModelAgent together with the host tools on MS Windows. Some figures are also from elderly versions of MetaModelAgent. The figures can therefore look a bit different in your own environment.

1.6 What's new

From v4.2.0 to v4.2.1

- No changes. V4.2.1 is a pure bug-fix release. The list of fixed bugs is available at www.metamodelaget.com.
- This manual has been extended with a final chapter listing known limitations in the current release.

From v4.2.1 to v4.2.2

- Support for Papyrus 3.0: Support for the new Architectural framework and the two different variants of element documentation representation.
- Support for Papyrus RT 1.0: This will require a metamodel that also defines the internals of protocols, normally hidden in Papyrus RT. The built-in general metamodel can however be used also for RT-models.
- Web published model reports now also provides clickable hyperlinks for relationships and not only elements in the diagrams.
- Popup-dialog added to confirm that web publishing has finished successfully. The popup dialog provides some measurements of number of elements and diagrams published as well as elapsed time, which are useful for fine-tuning the publishing process.
- Several bugs have been fixed. The list of fixed bugs is available at www.metamodelaget.com.

From v4.2.2 to v4.3.0

- Brand new *Trace Tree View* for analyzing relationship chains within and between models, see chapter 9.7.
- Brand new *Chart View* for analyzing model information in bar charts and scatter charts, see chapter 9.8.
- Bulk editing capabilities of properties within the Property Table View, see chapter 9.5.
- Several bugs have been fixed. The list of fixed bugs is available at www.metamodelaget.com.

From v4.3.0 to v4.3.1

- Support for nested models which can be activated using their own separate metamodels.
- Improved behavior of MetaModelAgent diagram tool palette and support for state machine diagrams and activity diagrams.
- Extended options for populating the Trace Matrix View with incoming and outgoing relations as well as interconnected relationships, see chapter 9.6.
- New metamodel column in Problem View, see chapter 9.3.
- Support for non-enumerated properties in Property Bar Chart, see chapter 9.8.
- Export to CSV-file option in Trace Matrix View, see chapter 9.6.
- Support for showing transitions, object flows and control flows in the Trace Matrix View, see chapter 9.6.
- Unloaded fragments decorator in the model explorer showing the file path to the fragment, see chapter 10.2.

From v4.3.1 to v4.4.0

- Support for UML-RT. The support include a built-in UML-RT metamodel and specialized model analysis features for UML-RT capsules and protocols and state machines with awareness of UML-RT specific semantics.
  [RSARTE and HCL RTist only]
- Several new model analysis capabilities including analysis of implicit derived relations, transitive relations and static behaver analysis of state machines, capsule structures and activity flows.
- Built-in library models can be activated during loading, this is controlled by a new preference setting.
- Redesigned wizards where the guidance pane in each wizard page is initially collapsed and can be opened upon need.
- Improved layout of *Guidance View* when embedded web browser is missing, this might be the case in some Linux environments.
- Multi-selection supported in Explorer View and in Activation View.
- Improved customized tooltips when hovering over elements and other items in all MMA-specific views and wizards.
- Support for selecting preferred sort order in web publish output.
- Improved layout of web published source code using monospaced font.
- Overlapping GUIDs, that may lead to incorrect web publish result are detected during web publishing.
- Activation view is automatically filtered on the currently active editing domain.
  [Papyrus only]
- The previously used vague terms “Item Definition” and “Item Kind” has been replaced with the more precise term “Metaclass” in the user interface and in this user manual.
- The previous user manual has been split into three separate user manuals, focusing on modeling, model analysis and license management.

From v4.4.0 to v4.4.1

- Improved performance when activating a model using built-in metamodels for standard UML and UML-RT.
- Unloaded nested submodels, nor unloaded referenced elements are no longer reported as violations as long as their UML metaclass conforms to the expected metaclass´ UML metaclass.
- API added for prohibiting MetaModelAgent to react on model modification from a user-defined plugin. See chapter 14.3.
- API extended for making it possible to register metamodels in the workspace applicable for a specific kind of models. Se chapter 14.4.
- Tooltips for property values in the Property View representing one or several correct URLs displays hyperlinks for each URL. This makes it simple to navigate to the URL using the default web browser.
- Tooltips for property values representing unloaded elements displays the file path to the model file containing the unloaded element,
- Integer property values are correctly sorted in web publish search result.

1.7 Support

To obtain support, please use the support request form on the Adocus Website (www.adocus.com) or send an email to support@adocus.com.

2 Terminology

The following terms are used frequently in this user manual.

2.1 Modeling terms

Term	Explanation
Item	Item is the generic term for an element, a diagram or a relationship. Examples of items are: classes, dependencies and activity diagrams.
Property	A property is a predefined or user defined feature of an item. Examples of common properties are: name, stereotype, multiplicity, documentation.
Model	A model is a UML-model holding a set of items which represents a model as defined in UML, e.g. a semantically complete abstraction of a system. Examples of common models are Use Case Models, Analysis Models and Design Models.
Metaclass	A metaclass is a classification of items sharing the same characteristics. Examples of standard UML metaclasses are Use Case, Actor and Class. A metaclass holds property definitions.
Property Definition	A property definition defines the characteristics for a specific property on items classified by the same metaclass..
Metamodel	A *metamodel* defines the model guidelines, i.e. a domain-specific modeling language (DSML) in terms of metaclasses and their relationships, for a specific kind of models. A model is an instance of its metamodel.

2.2 Eclipse terms

Explorer View	The Explorer View is the name used in this manual for the standard view that shows the model structure of loaded models. • *Project Explorer View* in RSAD/RSARTE and HCL RTist. • *Model Explorer View* in Papyrus.
Diagram Editor	The Diagram Editor is the graphical editor view for editing UML-diagrams.
Property View	The Property View is used for viewing and editing item properties.

3 The concept of metamodels

To be able to use MetaModelAgent, a so called *metamodel* must be available. A *metamodel* is a UML-model that primarily contains a set of classes and relationships which describes the model guidelines for a specific kind of models, in a formal way.

There may be several user-defined metamodels available at the same time within the same workspace. To use the metamodel for a model, the metamodel must be assigned to the model (exemptions are given below). This is expressed by a simple dependency relationship from the model towards the metamodel.

Figure 2: *Example of a use-case model connected to a metamodel for use-case models.*

It is possible to assign several different metamodels to the same model. This can for example be useful if there is a need to have different metamodels focusing on separate kind of modeling rules.

The metamodels needed should not be created by everyone. Normally a few members of a project or an organization is responsible for the modeling guidelines and therefore also responsible for developing the metamodels.

As a user of MetaModelAgent, there is no need to read and understand the metamodels, just knowing they are the basis for the MetaModelAgent-functionality is enough.

How to create metamodels is therefore not included in this user manual. The UML-notation for metamodels is described as an UML-profile in the manual *MetaModelAgent – Meta Modeling Manual,* distributed together with MetaModelAgent.

3.1 Using the built in General UML Modeling Guidelines

MetaModelAgent comes with a set of general UML guidelines defined in a metamodel. These guidelines are available without a need for a dependency relationship, if no other guidelines are connected. These guidelines cover most part of UML and will not result in any validation error, however when used in MetaModelAgent they will provide useful guidance and comprehensive views for doing general UML-modeling.

The General UML Guidelines also covers the concepts used in Profiles which means that even profile editing can be made with support from MetaModelAgent.

3.2 Using MetaModelAgent for creating metamodels

MetaModelAgent comes with a set of guidelines for defining your own user-defined metamodels. These guidelines are referred to as the meta-metamodel.

There is no need to connect a user-defined metamodel to the meta-metamodel by using a dependency relationship, as described above. MetaModelAgent will understand that any metamodel should be able to use the meta-metamodel.

By using the meta-metamodel, the ones who are responsible for the model guidelines can use MetaModelAgent themselves when developing the metamodels.

4 Start using MetaModelAgent

4.1 Switching to the MetaModelAgent perspective

MetaModelAgent provides several additional views to the workbench which are useful when using MetaModelAgent. These views are available by switching to the MetaModelAgent perspective, which is an extension to the Modeling perspective.

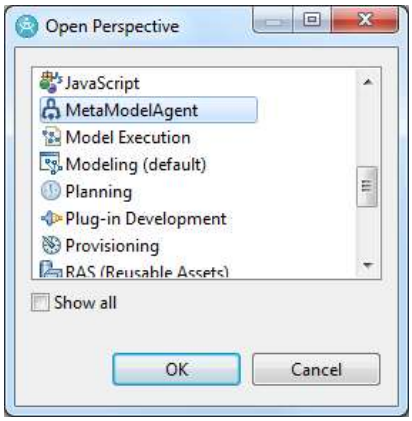


Figure 3: *The MetaModelAgent perspective in the Open Perspective dialog.*

Depending on the preference setting, switching to the MetaModelAgent perspective may automatically occur when activating MetaModelAgent for a model.

Optionally, you may open the MetaModelAgent views manually. To open the views:

1. Select *Window→Show View→Other* from the main menu to open the Show View dialog
2. Select the views beneath the *MetaModelAgent* entry in the Show View dialog and click OK.

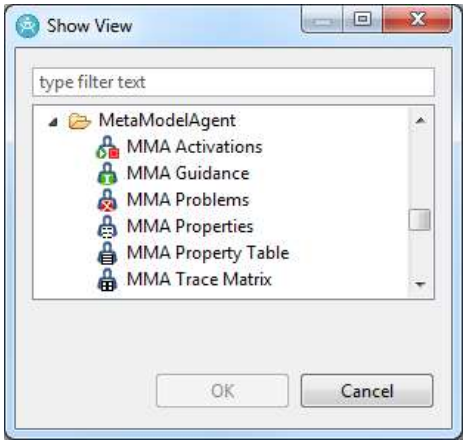


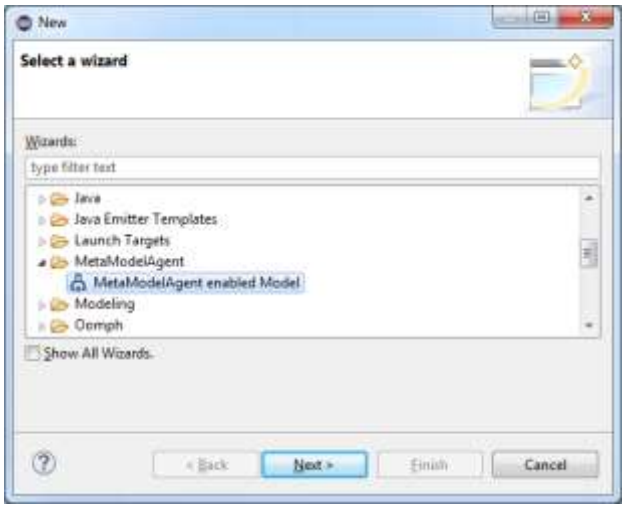
Figure 4: *The MetaModelAgent views in the Show View dialog.*

Instead of opening the *MMA Property View*, the ordinary tab-based property view can be used. There is a specific MetaModelAgent tab in the ordinary property view for any model item identified by the currently used metamodel.

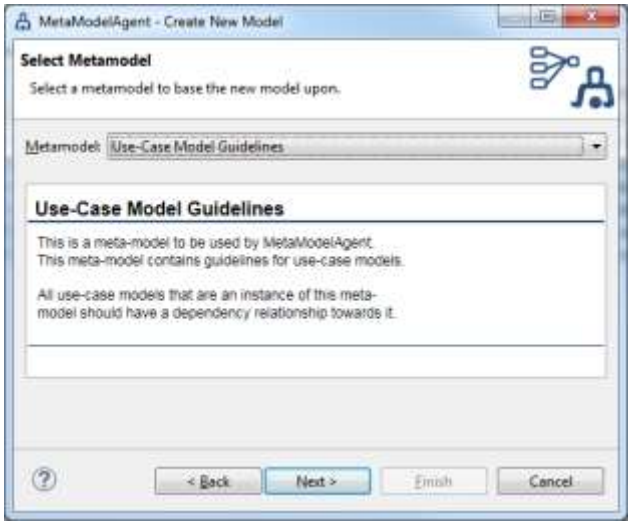
4.2 Create a new model based on a user-defined metamodel

A new model, based on the guidelines in a metamodel, can be automatically created by using MetaModelAgent's *Create New Model* wizard. You may notice that the dialog examples below are from a fictive metamodel for use-case models. The content of each dialog page may differ, depending on the metamodel used.

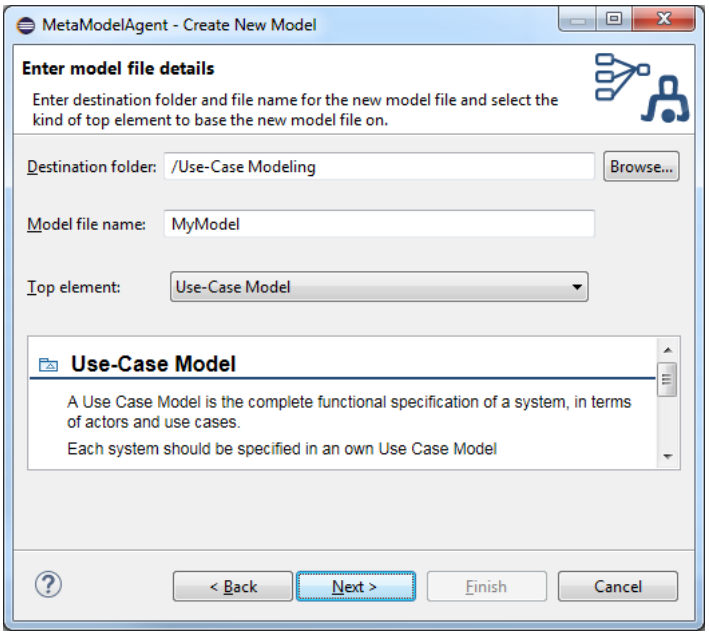
1. Open the wizard selection dialog by selecting *New→Other* from the main menu or from the project context menu in the Explorer View.
2. Select *MetaModelAgent / MetaModelAgent enabled Model* in the wizard selection dialog to bring up the *Create New Model* wizard.



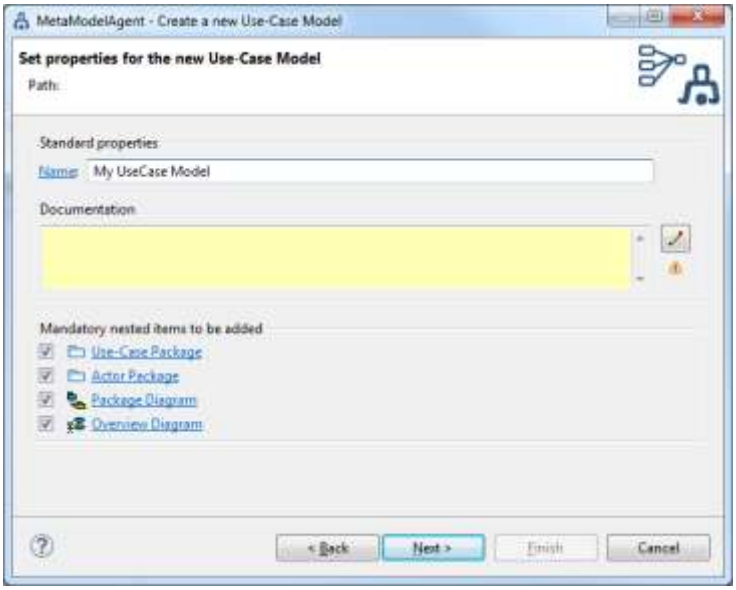
3. In the *Create New Model wizard*, select the preferred metamodels in the drop-down list. The last option in the drop-down list allows you to search for, and select a metamodel not yet loaded from your current workspace.
   A short description of the selected metamodel is displayed. Click *Next*.



4. In the next page, enter the destination folder and the file name of the new model. As some metamodels allows several different kinds of top-items, you may optionally select a top item of the new model itself. A short description of the top-item is displayed. Click *Next* to bring up the *MMA Add Wizard* for the selected top item



5. Enter valid values for the significant properties for the top item. Click *Finish* when ready. Depending on the metamodel used there may be consecutive pages to fill in for mandatory nested items.



These steps will create a new model in the selected project and connect the model to the metamodel by a dependency relation.

Depending on preference setting, activation of the new model is done automatically or a popup dialog may occur asking you to activate the MetaModelAgent functionality for the new model.

Papyrus 3.0 only: The new model will always have the UML architecture context selected as default. If the model is expected to use some other architectural context, an architectural switch must be made afterwards, using the *Switch Architecture Context…* menu entry in the Model Explorer´s context menu.

Papyrus 3.0 only: If the Papyrus settings indicates that comments used for documentation should have the «documentation» stereotype from the Papyrus Documentation profile, the Documentation profile will be applied automatically, enabling «documentation» stereotyped comments.

4.3 Connecting an existing model to a user-defined metamodel

If you want to start using MetaModelAgent for an already existing model, you need to establish a dependency relationship from the model to the metamodel.

To connect your model to a metamodel bring up the context menu in the Explorer View or in the Diagram Editor and select *MetaModelAgent→Activate→Connect to Metamodel*. This will bring up a dialog where you may connect the selected model to any metamodel available in the workspace or deployed in a plug-in.

Papyrus only: unloaded models in the workspace will not be selectable in the Connection Wizard. To connect to an unloaded metamodel, you must first load it. From the context menu in the Model Explorer select *Import→Import Package From User Model*. You will then be able to select the metamodel in the Connection Wizard.

Support for Marking Models (RSAD/RSARTE and HCL RTist only)

If the concept of marking models is used, you may assign a metamodel to the marking model instead of assigning it to the target of the marking model. If the marking model is opened when activating MetaModelAgent for the target model, the metamodel is available, otherwise not.

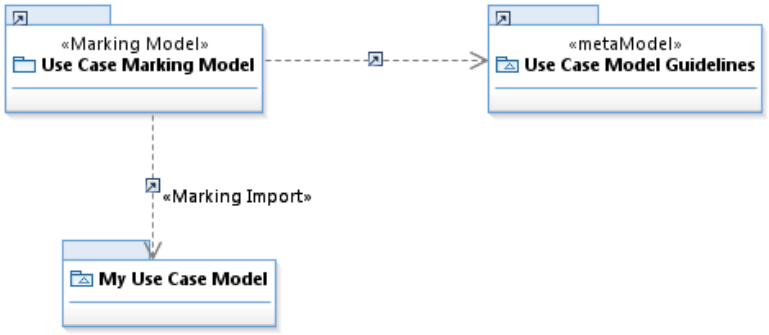


Figure 5: *The metamodel is assigned to a marking model and not to the target model.*

4.4 Using a predefined built-in metamodel for your model

You may use MetaModelAgent with one of the built-in metamodels provided by MetaModelAgent. Currently MetaModelAgent is delivered with a general UML metamodel and a general UML-RT metamodel (RSAD/RSARTE and HCL RTist only).

You don´t need to establish any connection to be able to use the built-in metamodels, MetaModelAgent recognizes automatically which metamodel that is applicable for your current model.

4.5 Activating MetaModelAgent for a model

MetaModelAgent must first be activated for a model, or part[2] of a model, to be able to use the functionality for that model. That can be automated upon open a model, or be performed manually on an already opened model using a menu entry.

[2] The scope of an activation may be a complete model or a part of a model, the activation scope may actually be one single UML-element as well

Automatic Activation

Depending on a preference setting, see chapter 11.1, activation may occur automatically when a model is opened, or a popup-dialog may prompt you for activation.

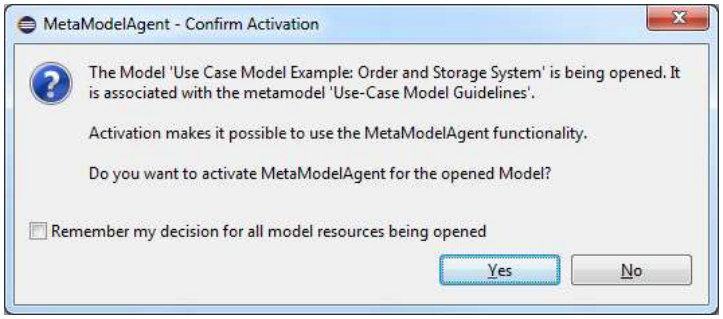


Figure 6: *Popup dialog for activation confirmation.*

Manual activation using the context menu

If a model is already opened, activation is done by following the steps below:

1. Select the item representing the model, or part of a model, in the Explorer View or in the Diagram Editor.
2. Select *MetaModelAgent→Activate* in the context menu
3. Select one of the available[3] metamodel for the model.

Alternatively, select *MetaModelAgent→Activate→Guided Activation* to bring up the *Guided Activation Wizard*, where you are able to read about all current metamodels that the selected model may be activated against.

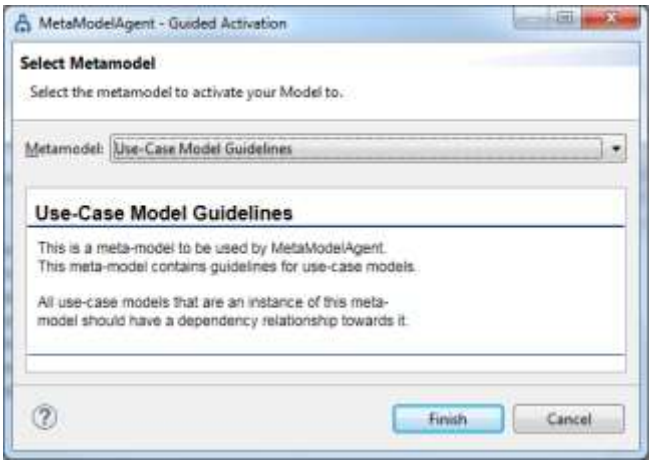


Figure 7: *The Guided Activation wizard.*

Manual activating using the tool bar

You may also activate MetaModelAgent by using the MetaModelAgent dropdown tool in the main toolbar and select the appropriate metamodel.

[3]The metamodel must be available in the workspace or deployed in a plug-in.

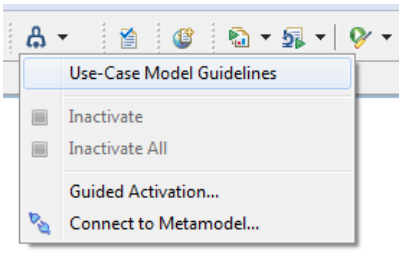


Figure 8: *The MetaModelAgent Tool in the main tool bar.*

If a model already has been activated for a certain metamodel, you can reactivate it by just pressing the MMA Tool in the tool bar.

Manual activation of several models in a single operation

Several models may be activated in the same single operation by selecting them all in the explorer view and then selecting *MetaModelAgent→Activate* from the context menu. This require that each model being activated is connected to at most one metamodel each.

Manual activation of nested models

If a model contains nested models connected to their own metamodels, those models will be activated automatically, when their owning model is getting activated.

Activation of built-in library models

Built-in library models provided by other plugins can be activated automatically. This is controlled by a preference setting. Those kinds of library models cannot be activated manually.

Automatic Perspective Switch

Depending on preference setting, see chapter 11.1, activation may lead to an automatic switch of perspective to the *MetaModelAgent Perspective* or to a popup dialog appear where you are prompt to switch perspective.

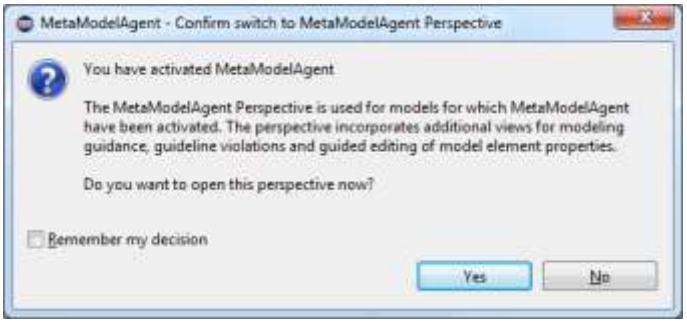


Figure 9: *Popup dialog, prompting for a perspective switch.*

4.6 Viewing the validation result

When MetaModelAgent is activated for an item, the item and all nested items are parsed and compared with the metamodel. All problems found are displayed in the *MMA Problem View*. As long as the activation for the item is still on, the *MMA Problem View* is automatically updated in the background when model changes occur.

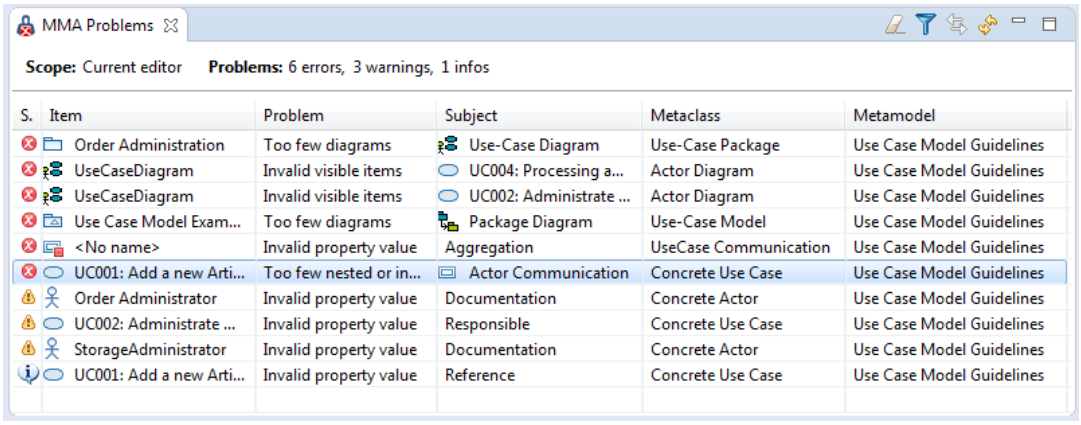


Figure 10: *The MMA Problem View displaying three errors, one warning and two information issues.*

There are some exemptions on how MetaModelAgent displays problem in relation to the UML specification. See chapter 9.3 for more information.

4.7 Activation management

There can be several activations of MetaModelAgent in the same time, just follow the steps above for each model that should be activated.

To be able to manage the current activations, you may use the *MMA Activation View*. This view will display all current activations and provide a context menu for deactivation of one or all activations.

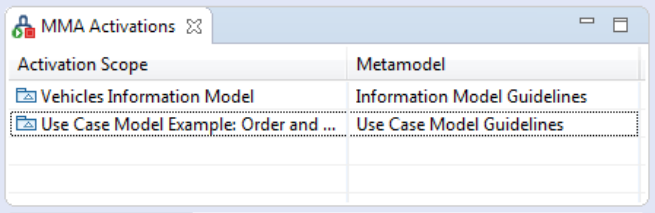


Figure 11: *The MMA Activation View after MetaModelAgent has been activated for two different models.*

The view gives information on the scope and metamodel for each activation.

4.8 Suppress validation for part of a model

Sometimes you may need to use constructions in the model that violates the metamodel. If you do so, MetaModelAgent will indicate problems for those constructions.

To suppress those problems, you can mark an item as invisible for MetaModelAgent. This is done by selecting the item in the Explorer View or in a diagram and select *MetaModelAgent→Suppress Validation* from the context menu.

When an item is marked as suppressed, MetaModelAgent will not consider that item or its nested items when it validates the model.

To unmark the item, making it able to validate again, just select *MetaModelAgent→Suppress Validation* once more.

The option to mark an item as invisible may also occur as a correction suggestion in the context menu of the problem view.

IMPORTANT: The information that an item has been marked as invisible is stored in a UML Constraint named “MetaModelAgent” on the closest namespace element in the

model. That information is therefore modifying the model and is persevered in the model when saving.

4.9 Inactivating MetaModelAgent for a model

An activation of MetaModelAgent can be manually disabled by the following steps:

1. Select the root item representing the activation in the Explorer View or in the Diagram Editor.
2. Select *MetaModelAgent→Activate→Inactivate* in the context menu

Alternatively, you can inactivate several activation by selecting them in the Activation View and then select *Inactivate* in the context menu.

All problems in the *MMA Problem View* concerning the selected model will automatically be removed and all MetaModelAgent functionality for the model will be disabled. The modeling work can thereafter continue without any presence of MetaModelAgent.

Inactivation of a model is also available in the context menu of the *MMA Activation View* and in the menu of the MetaModelAgent tool in the main tool bar.

Inactivation of all models

You may inactivate the MetaModelAgent functionality for all currently activated models simultaneously by one of the following operations:

- Selecting *MetaModelAgent→Activate→Inactivate All* in the context menu of the Explorer View or in the Diagram Editor.
- Selecting *Inactivate All* in the context menu of the *MMA Activation View*.
- Pressing the clear button in the *MMA Problem View*.

4.10 Guidance support

MetaModelAgent provides textual guidance for all items in the model, based on the documentation available in the metamodel and the result after validating the model.

The guidance is displayed in the MMA Guidance View and includes:

- Guidance on selected items in the Explorer View or Diagram Editor.
- Guidance on selected properties in the *MMA Property View* or in the *MetaModelAgent tab* of the ordinary *Property View*
- Guidance on selected problems in the *MMA Problem View*

To view the guidance, follow these steps:

1. Make sure that the *MMA Property View* is visible.
2. Select an item in the Explorer View or Diagram Editor, or a property in the *MMA Property View* or a problem in the *MMA Problem View*.
3. The guidance for the current item, property or problem will be visible in the *MMA Guidance View*.

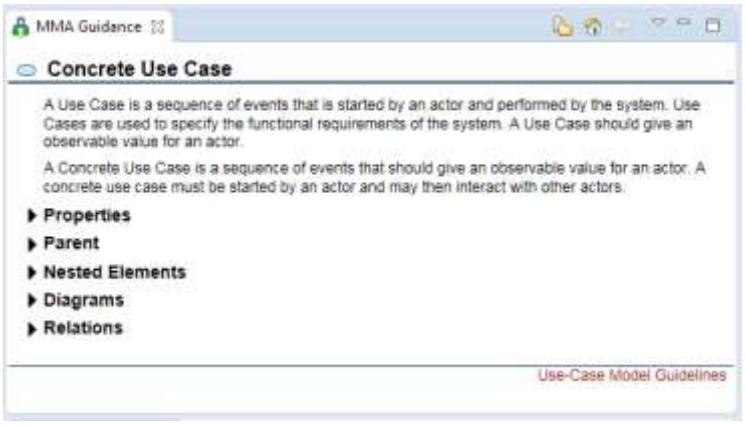


Figure 12: *The MMA Guidance View showing guidance for a use-case package.*

IMPORTANT: Parts of the guidelines displayed in the MMA Guidance View for selected items and item properties are based on documentation added to the metamodel. The quality of the guidance does therefor depend of the effort put into the documentation of the metamodel.

IMPORTANT: The Guidance View relies on web browser capabilities which normally is available in an Eclipse installation. However, in some Linux-environment the web-browser capabilities may be unavailable. If so, a simpler fall-back content layout of the Guidance View will be displayed.

5 Adding new items to a model

5.1 Using the Add Wizard

The built-in Add function in the host tools allows creation of almost any model item at any place in the model, as far as UML allows.

MetaModelAgent provides a powerful alternative to the built-in Add function which respects the metamodel and the current context.

The Add function provided by MetaModelAgent only allows valid items to be created and only allows the valid numbers of them, if there are such limitations defined in the metamodel.

Add a new item to the model by following these steps:

1. Select the item for which a new nested item should be created in either the Explorer View or in the Diagram Editor.
2. Select *MetaModelAgent→Add* from the context menu to bring up the valid alternatives. All valid metaclasses, according to the metamodel, are listed in the sub menu.
3. Select one of the listed metaclasses to bring up the *MMA Add Wizard* where you are able to add an item of that metaclass and edit significant item properties as well.
4. Optionally, click on the -button in bottom left corner to open the guidance pane of the wizard. The guidance pane will display guidance from the metamodel for each field in the wizard.

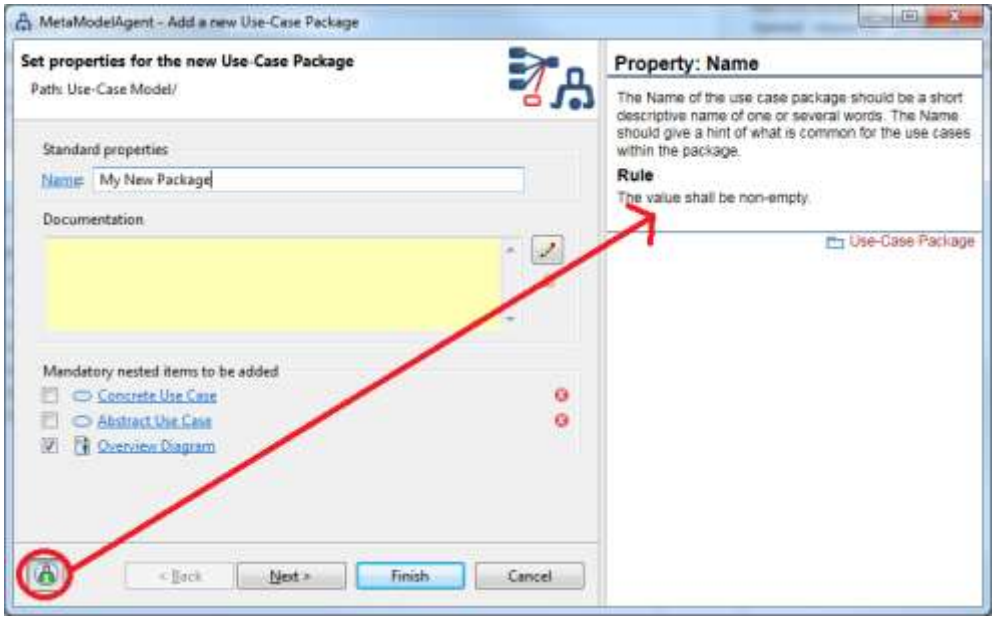


Figure 13: *The MMA Add Wizard* with the Guidance pane opened

5. Fill in the values of the significant properties for the item in the Add wizard. Any invalid value will be displayed with a coloured background and a problem indicator to the right. Invalid values of mandatory properties may stop you from continuing or finishing the wizard, this will be indicated by an error message in the upper part of the wizard page.
6. Click *Finish* to create the new item with the given values of the properties.

Papyrus 3.0 only: The representation of the documentation will be the same as the Papyrus/Documentation View settings. If the settings indicates that comments used for documentation should have the «documentation»-stereotype from the Papyrus. MetaModelAgent will apply that stereotype to the comment upon creation.

Entering element reference property values

For properties where the valid values are model element references, a button labelled "…" are provided, pressing the button will result in different kind of dialogs depending on the kind of property:

- Single-element reference properties will result in tree-selection dialogs where only valid elements are displayed. The search scope in that dialog will be all loaded[4] models as default. The search scope in the dialog could be narrowed to include elements from included packages only or any element in any opened model within the workspace.

  IMPORTANT: Unloaded models are not available for selection, except when the metamodel explicitly refer to a deployed model.

- Multiple-element reference properties will result in reference list dialogs where valid number of element references may be added by bringing up a nested tree-selection dialog. Based on the current metamodel, reordering is possible or not and duplicates are allowed or not. For properties such as slot value, the properties of the corresponding define feature is also considered to ensure a semantically correct model.

Creating mandatory nested items

If the metamodel stipulates mandatory nested items, there will be a corresponding section in the left bottom of the wizard page with checkboxes representing each kind of mandatory nested items. If at least one checkbox is checked, you will be able, or even forced, to click **Next** to go to the corresponding wizard page for the nested item.

Any invalid value will be displayed with a coloured background and a problem indicator to the right.

This works recursively so you can in one single wizard add a complex model item structure at the same time.

On the upper part of the wizard page, you will see the current position in the model structure that will be created.

IMPORTANT: The elements created using the Add-wizard are created at the time where the Finish button is pressed. In a multi-page wizard that means that elements specified in one page are not available as property values on the other pages.

Guidance

 By clicking on the Guidance button in the bottom left corner of the wizard, the guidance section of the wizard is displayed. Section shows context sensitive guidance from the metamodel.

- To learn more about each property, click on the property label.
- To learn more about each mandatory nested item, click on the item label.

[4] In Papyrus this mean loaded in the same editing domain.

Customization

There is a preference setting that can be used to automatically omit any fixed value property that are not allowed to change, see chapter 11.1.

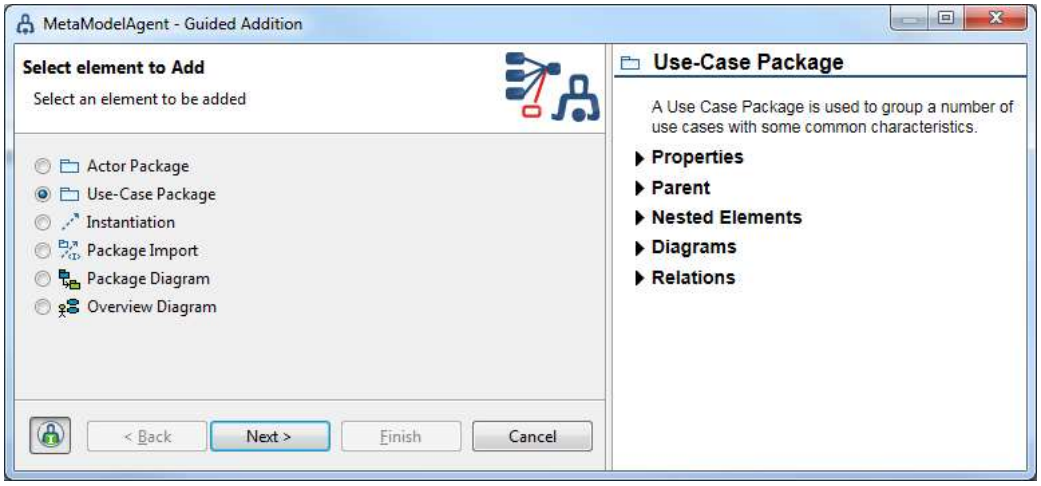
There is also a preference setting that can be used to automatically omit any wizard pages representing items with no editable properties, see chapter 11.1.

5.2 Using Guided Addition

If you are not familiar with the kind of items that are available in the *MetaModelAgent→Add* submenu, you may use the *Guided Addition Wizard* instead, which will bring up a wizard where you have the option to learn more about all available items kind prior to creating them in your model.

To use the *Guided Addition Wizard* follow these steps:

1. Select *MetaModelAgent→Add→Guided Addition* in the context menu, which will bring up the *Guided Addition Wizard*.
2. In the wizard, you are able to learn more about the different metaclasses that may be created by selecting the radio buttons and see the corresponding guidelines from the metaclass in the right pane.



3. Click on the Guidance Button in the bottom left corner to bring up the Guidance section of the wizard that displays an explanation on the alternatives available.
4. Select the item to be created and click *Next*. This will bring up the ordinary *MMA Add Wizard* for the selected item as described above.

5.3 Using the Diagram Palette

MetaModelAgent extends the built-in diagram palette with a new context sensitive section based on the rules of diagram content in the metamodel.

The MetaModelAgent palette is available for all diagrams of the following kinds:

- Class Diagram, Component Diagram, Use-Case Diagram, Deployment Diagram, StateMachine Diagram and Activity Diagram.
- Object Diagram and Free form diagram (RSAD/RSARTE and HCL RTIst only).
- Package Diagram and Profile Diagram (Papyrus Only).

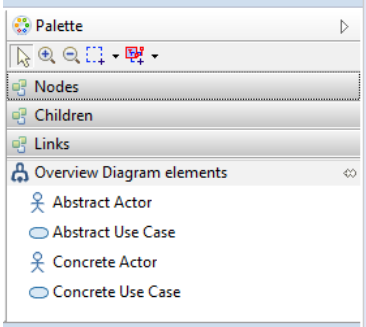


Figure 14: *The MetaModelAgent section of the Diagram Palette.*

The new section provides creation tools for all kind of classifiers and packages that are valid in the diagram. If some classifiers are not allowed in the same position as the diagram within the model, their palette entries will be separated from the others by a separator line in the palette.

IMPORTANT: Drag & drop from the MetaModelAgent palette is performed by clicking on a palette entry and then somewhere in the diagram area.

IMPORTANT: By selecting and dragging a tool to the editor area, *MMA Add Wizard* will appear. After the wizard has been completed, the new item is added to the model and to the diagram.

5.4 Adding relationships

Relationships can be added by selecting the source element and thereafter holding down the Ctrl-key and selecting the target element. The Add-submenu in the context menu will then display valid relationships between the two elements and also provide the Guided Addition menu entry, as described above.

If the elements are selected in a diagram the added relationship will automatically be added to the diagram.

When adding an association there will be two successive wizard pages if there are several valid combinations of association ends. On the first wizard page the combination of association ends will be selected. On the second page the properties for the association element and the two selected association end will be edited. If there are only one valid combination of association ends, only the second wizard page will appear.

IMPORTANT: The following kind of connectors cannot be added using the method above. For these transitions, please use the standard palette available

- Transitions in State Machine Diagrams
- Control Flows and Object Flows in Activity Diagrams
- Messages in Sequence Diagrams
- Connectors in Composite Structure diagrams

IMPORTANT: Drawing an Association between elements within two separate activations using the gesture described above is only possible if both elements are activated towards the same metamodel.

6 Correcting an existing item

You may use the built-in functionality in your host tool for adding new items, and still make use of MetaModelAgent-enabled wizards to ensure that any added item will have the significant properties set to correct values. This chapter describes the facilities for correcting invalid items.

6.1 Using the Change to Wizard

If an item has been added using the built-in facilities for adding items in the host tool, there might be impossible for MetaModelAgent to identify which kind of item, according to the metamodel that the item adhere to. In some situations there might even be more than one possible kind to choose between. In such situations there will be a need to change the invalid properties fulfilling one of the possible metaclasses.

MetaModelAgent provides the *Change To* feature and the corresponding *Change To Wizard* to simply the task of making sure that all significant properties are being valid.

To use the Change To feature, follow these steps:

1. Select the item to be changed in either the Explorer View or in the Diagram Editor.
2. Select *MetaModelAgent→Change to* from the context menu to bring up a sub-menu of the valid alternatives. All valid metaclasses, according to the metamodel, are listed in the sub menu. This submenu is only available, when there are some valid alternatives defined in the metamodel.
3. Select one of the listed metaclasses to create an instance of that item. If the change can be performed without manual property editing it will be done automatically.
   If the change requires some manual editing of one or more properties, *MMA Change Wizard* will be displayed.
4. Fill in the values of the significant properties for the item in the *MMA Change Wizard*.
5. Click **Finish** to apply the change of the item properties.

There are often several valid kinds of items based on the same kind of UML item, one example are the three variants of analysis classes when using RUP; boundary, control and entity classes, which only differ in the applied stereotype.

In such situations there will be a need to change from one kind to another in a convenient way without manually have to edit the properties making each variant unique.

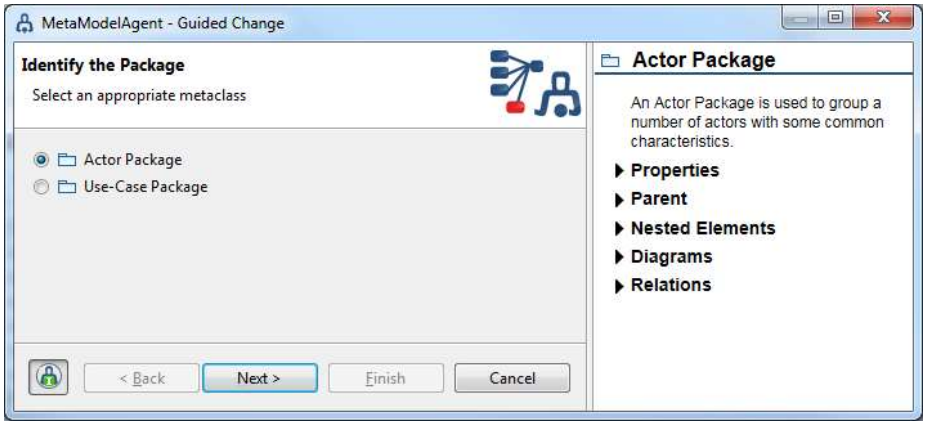
6.2 Using Guided Change

If you are not familiar with the kind of items that are available in the *MetaModelAgent→Change To* submenu, you may use the *MMA Guided Change Wizard* instead, which will bring up a wizard where you have the option to learn more about all available items kind that the selected item may be changed to.

To use the *MMA Guided Change Wizard* follow these steps:

1. Select the item to be changed in either the Explorer View or in the Diagram Editor.
2. Select *MetaModelAgent→Change to→Guided Change* from the context menu to bring up the *MMA Guided Change Wizard*.

3. In the wizard, you are able to learn more about the different kind of items that the selected item may be changed to, by selecting the radio buttons and see the corresponding guidelines from the metamodel in the right pane.



4. Finally, select the metaclass to change the current item into, and click *Next*. This will result in a *MMA Change Wizard* will be displayed, as described above.

6.3 Using Automated Change

MetaModelAgent can be configured to automatically bring up a wizard whenever you add a new item using the standard built-in UI functionality, or optionally only when needed to enforce correct property values on added items.

The corresponding preference setting is described in chapter 11.1.

When the feature is enabled, a *MMA Change Wizard*, as described above, occurs whenever an item is added to a model, by using the context add menu, the Diagram Editor palette or using the in diagram popup menus for elements and relationships.

If the item being created can be uniquely identified in the metamodel, a *MMA Change Wizard* for that item will be presented which is similar to the *MMA Add Wizard* described above, but where property values generated by the host tool are prefilled.

If there are several possible metaclasses in the metamodel that matches the item being added, a *MMA Guided Change Wizard*, as described above, will be displayed. The wizard will first let you select which kind of item, according to the metamodel, that you are creating. After that, a *MMA Change Wizard* is presented which let you edit the significant properties for the new item.

This feature is especially useful when adding relationships and associations. When adding an association, the *MMA Change Wizard* normally will have three pages where you in the first page, besides entering the property values for the association item, also select which kind of items the two ends of the association represents.

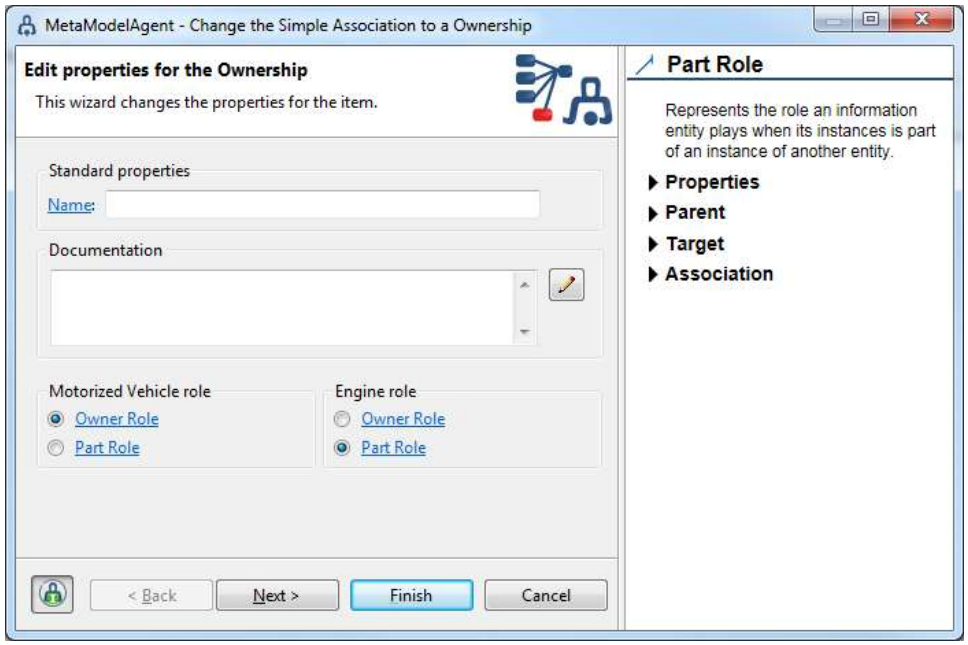


Figure 15: *The Change Wizard for a composition that lets you selected the metaclass for both association ends.*

Click *Next* to display the wizard page for the first association end, clicking *Next* once more will display the wizard page for the other association end.

6.4 Using Quick Fix in the Problem View

MetaModelAgent can assist in correcting the identified problems in the model. It can suggest possible solutions of a problem, if there are any, and often also perform automatic corrections of the problem, based on the suggestions.

To use this feature, follow these steps:

1. Make sure that the *MMA Property View* is visible.
2. Select the problem to be corrected in the problem view
3. Right-click on a problem in the *MMA Problem View*, to open the context menu
4. MetaModelAgent will provide a context menu item for each possible solution of the problem.
5. Select the solution to apply on the model
6. MetaModelAgent will automatically make the change. If the solution need some additional user input, an appropriate dialog or wizard will appear.

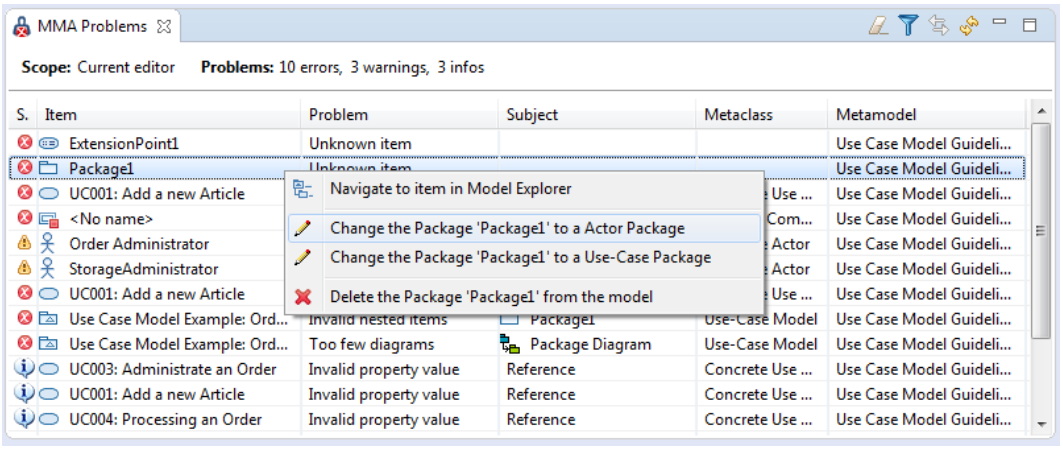


Figure 16: *Automatic correction of a problem in the MMA Problem View, MetaModelAgent provides three suggestions for the specific problem.*

There are some limitation in the quick fix when using Papyrus.

- Diagrams cannot be moved to another location in the model by using the quick-fix menu. Invoking a quick-fix Move-operation will have no effect.
- When a diagram is deleted using the quick-fix menu, the diagram will be deleted but any open editor for the diagram will not be closed automatically.

7 Editing an existing item

7.1 Using the Property View

MetaModelAgent provides the possibility to focus on only those properties that are defined as significant in the metamodel, hiding the other ones.

Either the standard tab-based property view or the *MMA Property View* can be used to edit the significant properties.

Edit the significant properties for an item by following these steps:

1. Make sure that either the standard property view or the *MMA Property View* is displayed in the workbench. The MMA Property View will be displayed when switching to the *MetaModelAgent Perspective*.
2. Select the item to edit, either in the Explorer View or in the Diagram Editor.
3. The significant properties for the selected item, according to the metamodel together with their current values and eventual problems, will be shown in the *MMA Property View* or in the standard property view.
4. Edit the properties, the new values are automatically saved as soon as you move your selection to something else. Problem icons are updated simultaneously. Properties holding element references uses popup-dialogs, please see chapter 5.1 for more information.

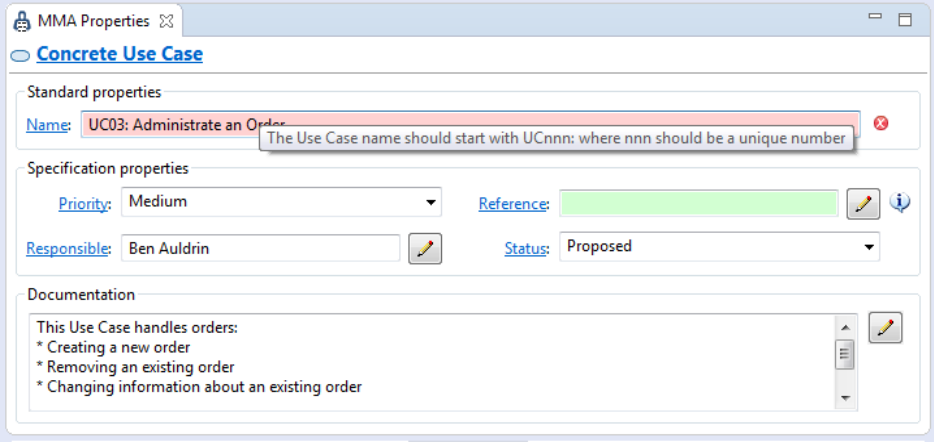


Figure 17: *The MMA Property View for a Use-Case, displaying the significant properties according to the metamodel.*

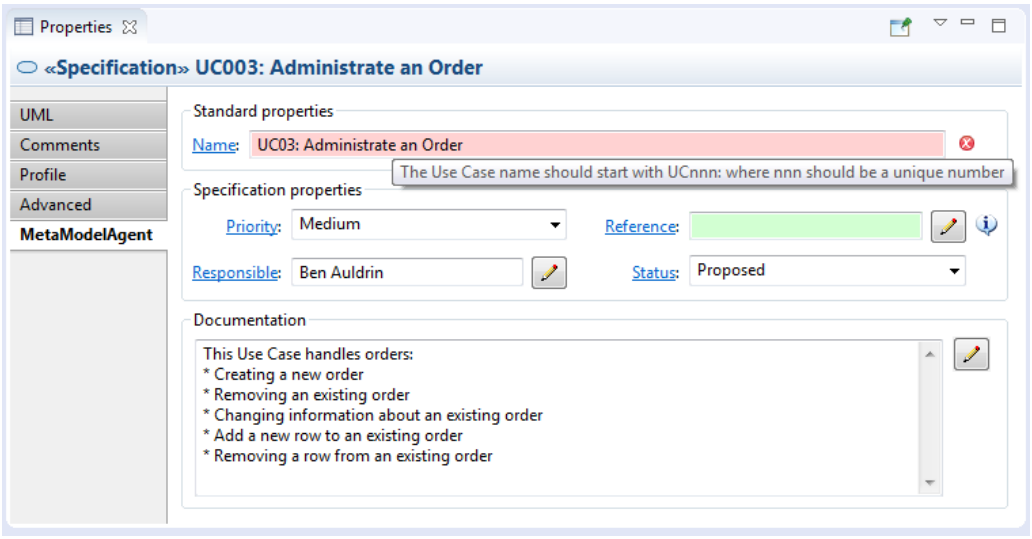


Figure 18: *The same Use Case displayed in the MetaModelAgent tab in the standard Property View*

If you have the *MMA Guidance View* displayed in the workbench, that view will display guidelines from the metamodel for each property, when clicking on the corresponding property label.

If the current metamodel contain alternative valid combination of properties for the item, the visible properties and their valid values may dynamically shift during the editing, to reflect valid combinations of the properties.

7.2 Papyrus only: Enabling rich-text documentation

MetaModelAgent supports rich-text documentation of elements in Papyrus. To enable rich text editing, change the Papyrus preference setting for rich-text support.

1. Open the preference setting dialog by selecting *Window→Preferences* in the main menu.
2. Navigate to the *Papyrus/Rich text* preference page.
3. Check the *Use advanced rich text editor* checkbox.
4. Save the settings and close the Preference Setting dialog.

When rich-text editing has been enabled, the documentation field in the MMA Property View and MMA Add Wizard will be a read-only web browser. To edit the documentation click on the edit button, which will bring up a popup-dialog with the same web-based rich-text editor as Papyrus itself is using.

8 Web publishing

8.1 Publish model documentation

MetaModelAgent makes it possible to generate and publish model documentation for one or several models or part of models. MetaModelAgent uses the information in the related metamodels to provide excellent model documentation with extensive inter-model cross-reference information and also with references to modeling guidelines from the used metamodels.

To publish model documentation, follow these steps:

1. Select one or several models or parts of models for which MetaModelAgent have been activated.
2. Right click to open the context menu and select *MetaModelAgent→Publish Documentation*.
3. Select the appropriate settings and provide the output folder, see detailed description of each setting below.
4. Click *Finish* to start the publishing process.

A progress dialog will indicate the progress of publishing process. When the progress meter is closed, the publishing is complete.

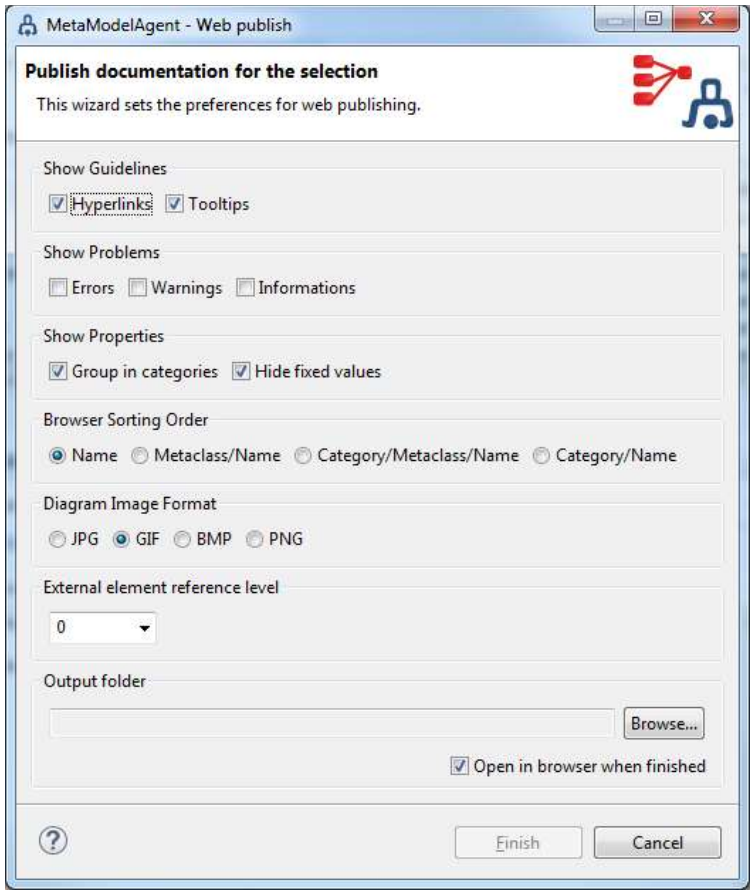


Figure 19: *The user dialog for model web publish settings*

Setting	Description
Show guidelines	Controls if and how guidelines should be displayed.

If **Hyperlinks** is checked, hyperlinks will be provided to an integrated guidelines website.

If **Tooltips** is checked there will be plain-text tooltips showing guidelines when hovering over an element or a property. The tooltips may be hard to read if guidelines include formatted text.

Show problems Controls which kind of problems that should be displayed. *This feature requires an Enterprise license*.

Show properties Controls the details of properties to be displayed.

Group in categories will organize the properties according to the stereotype categories they belong to.

Hide fixed values will hide any property that have a mandatory fixed value.

Browser Sorting Order Select the sort order of the items in the browser view in the web-publish output.

Available alternatives are:

- *Name* only
- 1:st: *Metaclass*, 2:nd: *Name*
- 1:st: *Item Kind Category*, 2:nd: *Metaclass*, 3:rd: *Name*
- 1:st: *Item Kind Category*, 2:nd: *Name*

Where *Item Kind Category* is the built-in categorization of items, e.g. packages, classifiers, features etc.

Diagram image format Select the preferred image format for published diagrams, normally GIF or PNG will give the best result.

Controls if a property table should be used for nested elements instead of a plain list of elements. Separate settings are available for different categories of elements.

External element reference level Select the number of levels of referenced external elements that will be published.

- '*0*' means that only elements within the selected (sub)models are published,
- '*1*' means that all elements directly referenced from an element or diagram within the selected (sub)models are published
- '*All levels*' means that all external referenced elements (directly or indirectly) from any elements or diagrams in the selected (sub)models are published

You may notice that selecting something other than ´*0*´ in combination with checking the *Include guidelines* checkbox will publish the guidelines of all activated models that contain referenced elements.

Output folder Select the output folder for the published report, and optionally any published guidelines.

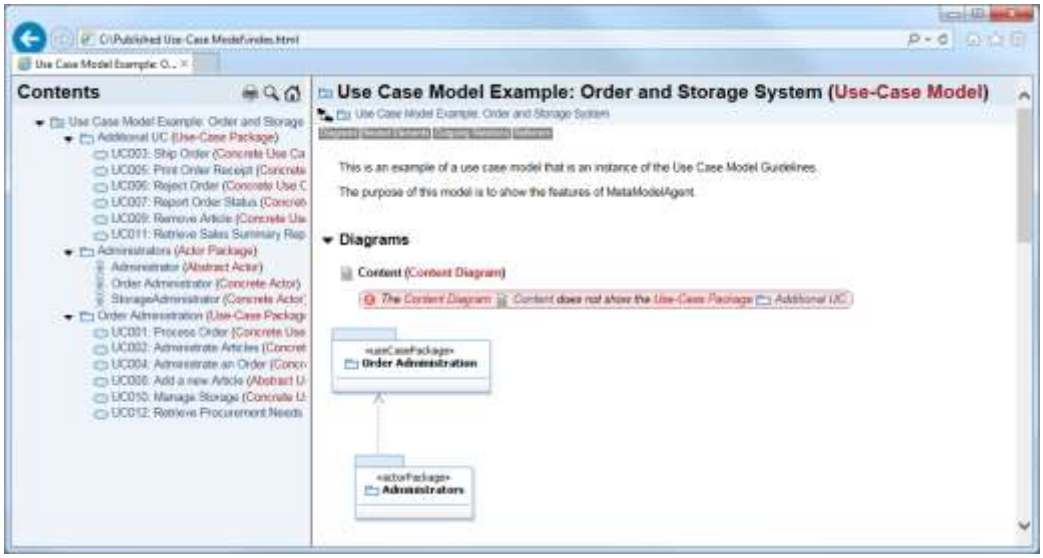


Figure 20: *The resulting web site of a published model*

8.2 Publish model guidelines

MetaModelAgent can publish modeling guidelines as part of the published model documentation, as mentioned above.

However MetaModelAgent can also publish modeling guidelines directly from a metamodel. To publish modeling guidelines, follow these steps:

1. Select the metamodel holding the guidelines that should be published.
2. Right click to open the context menu and select *MetaModelAgent→Publish Guidelines*.
3. Provide the output folder and click *Finish* to start the publishing process.

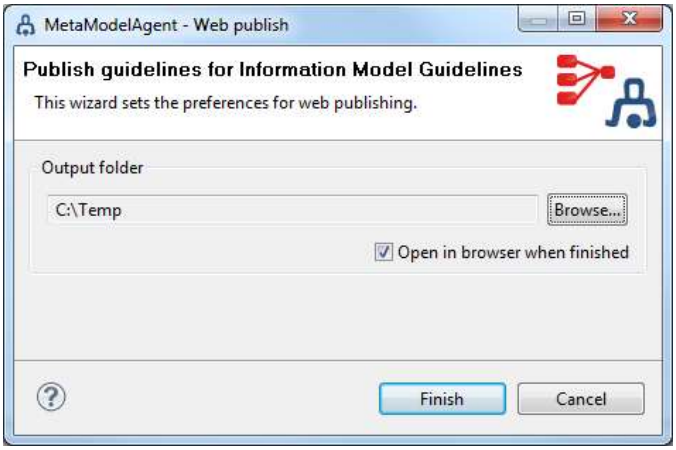


Figure 21: *The user dialog for guidelines publishing settings*

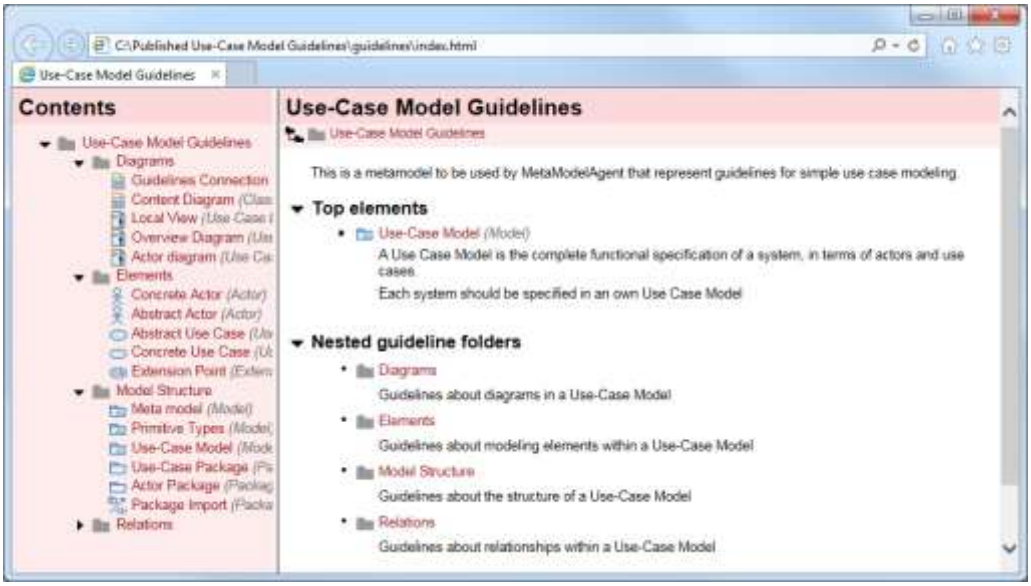


Figure 22: *The resulting web site of a published set of guidelines.*

9 Views

MetaModelAgent provides several additional views to the workbench which can be used for a model where MetaModelAgent is activated. This chapter describes each view in detail.

9.1 Activation View

The *MMA Activation View* is a table view that displays all current activations of MetaModelAgent. For each activation, the activation scope and the metamodel used are displayed.

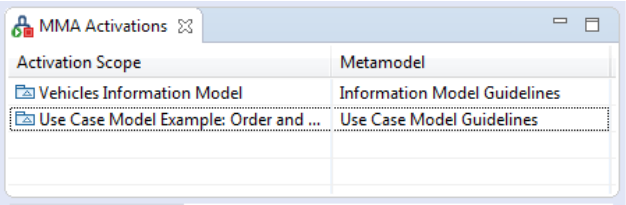


Figure 23: *The MMA Activation View*

Built-in library models that have been activated is displayed in grey.

By selecting one or several activations in the table, you may bring up a context menu where you are able to:

- Inactivate MetaModelAgent for the selected activations or for all activations
- Navigate to the corresponding model or metamodel in the Explorer View. To be able to do this, the metamodel must be available in the Explorer View.
- Apply and disable a filter in the MMA Problem so that only the problems identified in the selected activation are displayed.
- Populate the analysis views with content from the selected models.
- Generate a web publish report covering the selected model(s).

9.2 Guidance View

The *MMA Guidance View* is a read only browser view that always displays accurate guidelines from the metamodel for the model. This view shows guidelines from the metamodel in the following situations:

- When an item is selected in the Explorer View, Diagram Editor or in the *MMA Property Table View*, the guidance view presents the guidelines for that item including its significant properties, valid parents, nested elements and relations.
- When a problem is selected in the *MMA Problem View*, The guidance view presents detailed information of that problem. See also Exemptions from UML Language specification in 9.3 which also affects the problem specification in the Guidance View.
- If a property is selected in the *MMA Property View* or in the *MMA Property Table View*, the guidance view presents the guidelines for that property.

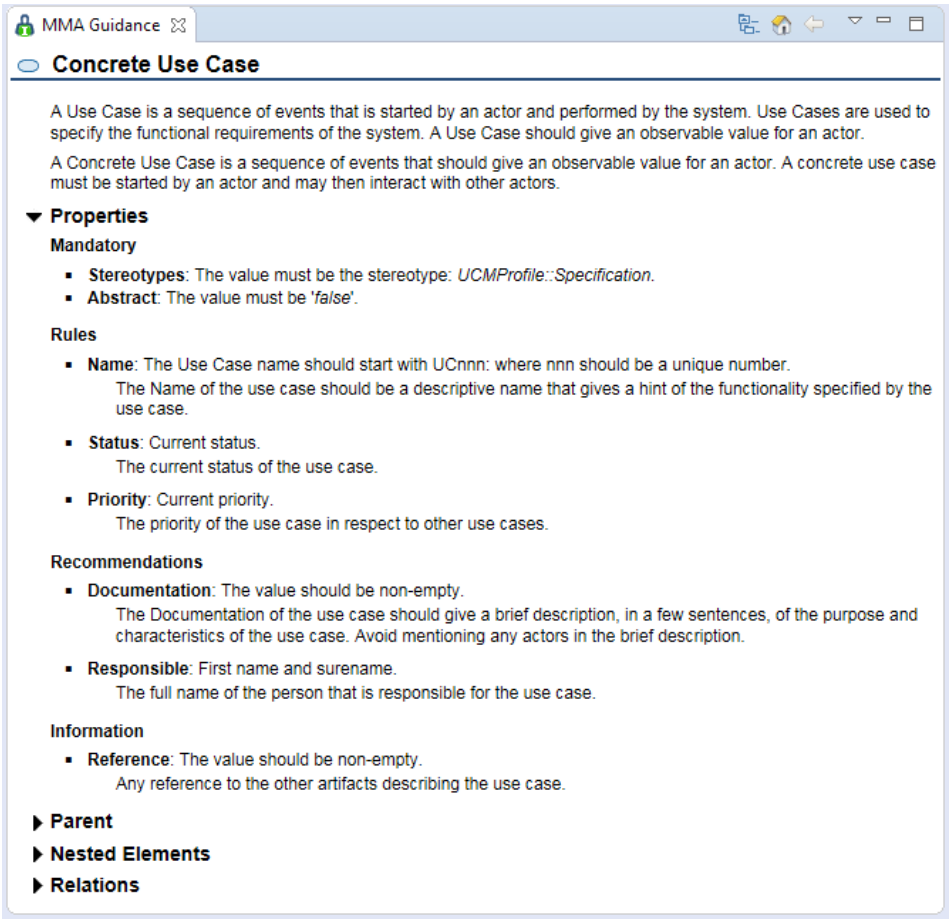


Figure 24: *The MMA Guidance View showing guidance for a concrete use case from a demo metamodel.*

If documentation is not provided in the metamodel for an item or a property, the guidance view is left empty.

There are four icon tools in the upper right part of the MMA Guidance View:


or

If the current metamodel are available in the Explorer View, this button navigates you to the corresponding metaclass, property definition or violated modeling rule in the metamodel. This is very useful feature when developing and debugging a metamodel.


Brings up the ”home page” of the guidance view which displays the current version of MetaModelAgent.


Return to the previous displayed page.

Brings up a view menu where you can select to populate the Table View with all occurrences of the current metaclass in all activated models. If the metaclass represents a relationship or other connection, there will be a menu entry for populating the Traceability Matrix View with all occurrences.

9.3 Problem View

The *MMA Problem View* is a table view that displays all detected problems in the model for which MetaModelAgent is activated.

In Papyrus the Problem View is automatically filtered to reflect the problems related to the models in the active editor.

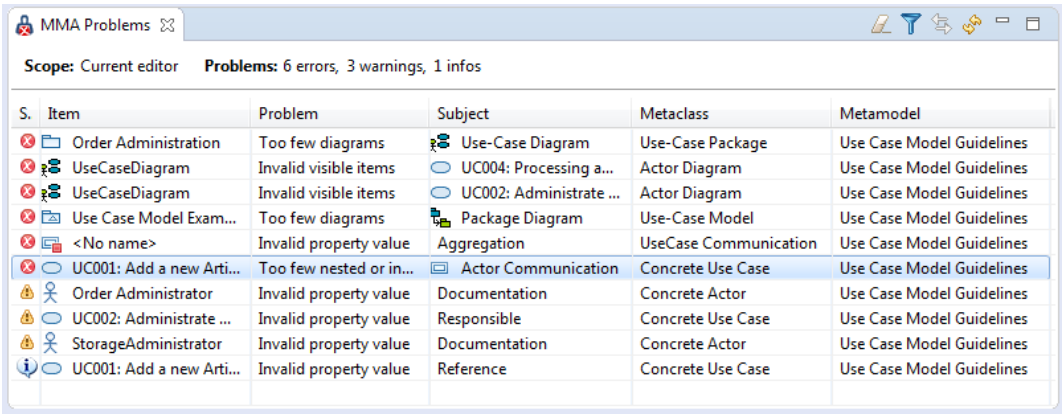


Figure 25: *Example of the MMA Problem View*

For each problem the following columns are displayed (from left to right):

Severity	An icon indicating the severity of the problem; - Error - Warning - Information *(If the metamodel contains errors, an ✘-icon may be shown, in that case, please contact the one responsible for your metamodel).*
Item	the item icon and the name of the item that holds the problem.
Problem	A short description of the problem.
Subject	The subject of the problem, the kind of subject displayed depends on the kind of problem.
Metaclass	The metaclass in the metamodel for the item holding the problem.
Metamodel	The metamodel used when detecting the problem.

The table view may be sorted on any column by selecting the column header.

If a problem is selected in the *MMA Problem View* a detailed explanation of the problem is presented in the *MMA Guidance View*.

- By double-clicking on a problem, the item holding the problem is highlighted in the Explorer View.

In the context menu for a problem, correction suggestions for the problem are presented, if available. By selecting a specific correction, the problem will be automatically corrected. However some corrections may need user input before they can be applied. If so, a wizard or a simple dialog will be presented.

You may apply a filter of the problem view by one of the following operations:

- In the Explorer View or Diagram Editor, right click on an item and select *MetaModelAgent→Show in Problem View→Selected Item*, which filter out all problems, except the ones that belong to the selected item.
- In the *MMA Activation View*, right click on an activation and select *Show in Problem View→Selected Item and its children*. Which filter out of problems except the ones that belong to the selected activation.

There are four icon buttons in the upper right corner of the *MMA Problem View*:

Disables any applied filter in the *MMA Problem View*.

Refresh all activations and updates the *MMA Problem View*.

Inactivate all activations and clears the *MMA Problem View*.

Link the content of the MMA Problem View to the element selected in the Explorer View or in the Diagram Editor. This means that any applied filter remains but the scope is changed.

Exemptions from the UML language specification

There are a few exemptions in the presentation of problems according to the UML-language specification.

- Dependencies and variants of dependencies are regarded as being owned by the source element. That means that if you have problems indicating that a nested dependency relationship to an element is missing or is incorrect, it actually means that the source element have problem with the dependency
- Association ends (e.g. property-elements with the association property set) are regarded as being owned by the classifier it “belongs to”, even if they actually are owned by the association itself. Therefor if the Problem view indicates that a nested association end to a classifier has a problem, the association element may be nested to the association instead.

9.4 Property View

MetaModelAgent extends the standard tabbed property view with a new tab and also provides an own *MMA Property View*.

Both the new tab in the standard property view and the *MMA Property View* provides the opportunity to focus on only the significant properties for an item. As a user you can therefore choose which one of the views that you want to use.

The standard property view in the host tool contains all possible properties for an item, organized in several tabs. The new tab and the *MMA Property View* only show the significant subset of properties according to the metamodel, making it easier to handle those properties.

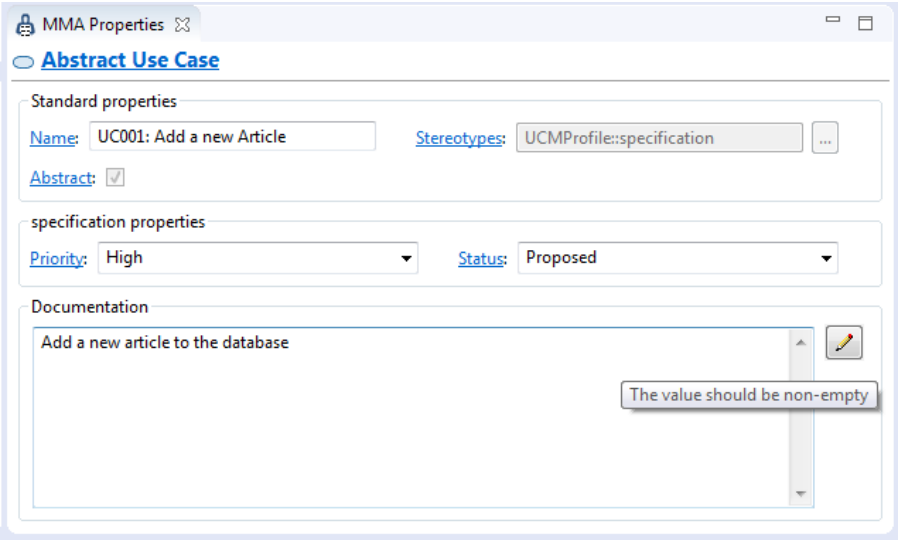


Figure 26: *The MMA Property View for an abstract use case using guidelines from a demo metamodel*

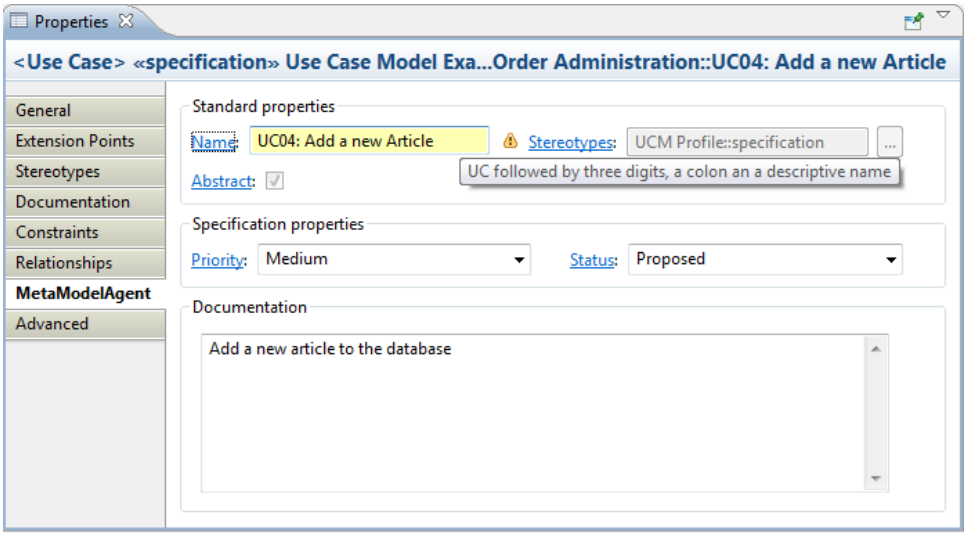


Figure 27: *T the MetaModelAgent-tab in the standard property view*

If there are any problems associated with a specific property, an icon representing the problem is visible to the right of the property field.

If there is a limitation of valid values for the property, only the valid values are available to set for the property. If there is only one valid value for the property, and that value already is set, the possibility to edit that property is disabled.

If the current metamodel contain alternative valid combination of properties for the item, the visible properties and their valid values may dynamically shift during the editing, to reflect valid combinations of the properties.

When selecting a property label in one of the property views, the corresponding guidelines of that property is presented in the *MMA Guidance View*.

Depending on a preference setting, see chapter 11.2, properties holding valid values that should not be changed according to the metamodel may be left out from the property views.

9.5 Property Table View

The *MMA Property Table View* is a unique spread-sheet table view that displays an overview of significant properties for all items of the same kind within a selected scope.

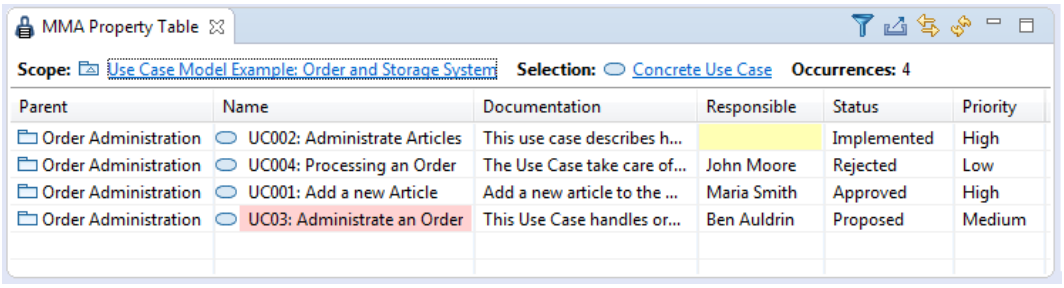


Figure 28: *MMA Property Table View showing significant properties for all concrete use cases within the package "Order Administration".*

The *MMA Property Table View* displays all items of the same metaclass in within a model or part of a model in a table view, one row for each item and one column for each significant property. Each column display a property that is significant for the chosen metaclass.

The columns representing item properties are editable, bulk editing of the same property for several selected items are available in the view´s context menu, as well as possibility to delete selected items.

This view is part of the model analysis capabilities in MetaModelAgent and is described in detail in the *MetaModelAgent Model Analysis User Manual*.

9.6 Trace Matrix View

The *MMA Trace Matrix View* displays the existence of relationships or relationship chains that fulfills the chosen criteria in a grid. The source elements are listed vertically and the target elements are listed horizontally.

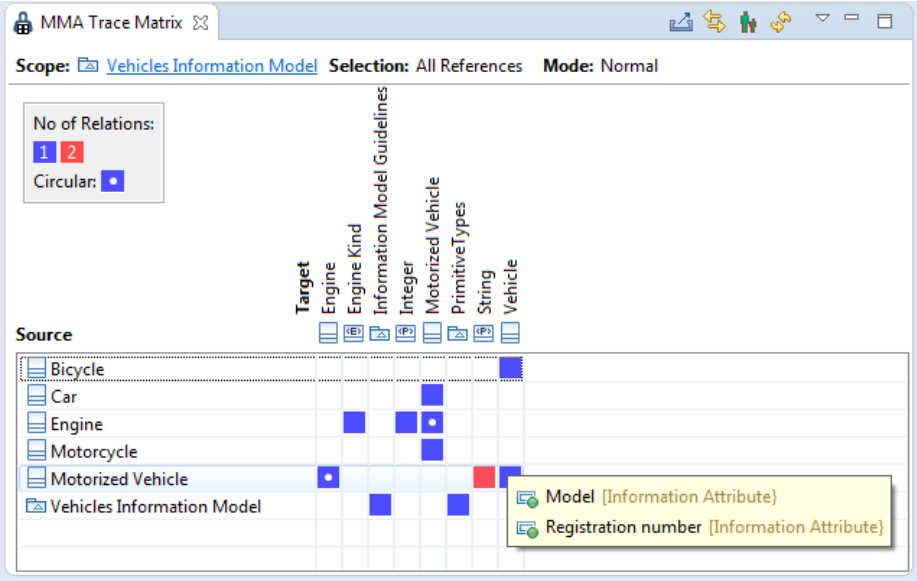


Figure 29: *MMA Trace Matrix View showing all relations and attributes within the Vehicle Information Model.*

Within the intersection cell of each source/target different variants of filled cell indicates that there is relationships or relationship chains between the source and the target element, depending how the Trace Matrix View was invoked.

This view is part of the model analysis capabilities in MetaModelAgent and is described in detail in the *MetaModelAgent Model Analysis User Manual*.

9.7 Trace Tree View

The *MMA Trace Tree View* displays the relationship chain starting from, or ending at, a specific element. Within the view, several features support your analysis of complex relationship chains.

All kind of UML relationships can be traced in this view including navigable associations, object flows, control flows and transitions.

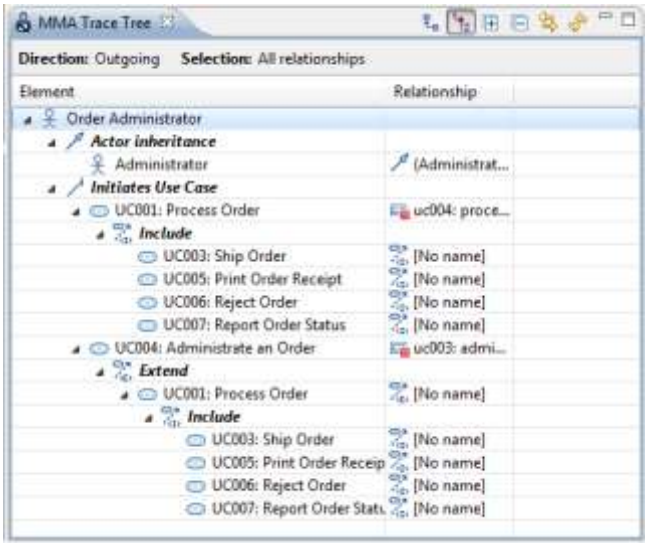


Figure 30: *MMA Trace Tree View displaying all relations starting from the actor Order Administrator.*

This view is part of the model analysis capabilities in MetaModelAgent and is described in detail in the *MetaModelAgent Model Analysis User Manual*.

9.8 Chart View

The *MMA Chart View* will provide you with different kind of bar charts and scatter charts that will help you analyze your models.

Each kind of chart can be initiated by selecting an element in the Explorer View, Diagram Editor or Property Table View.

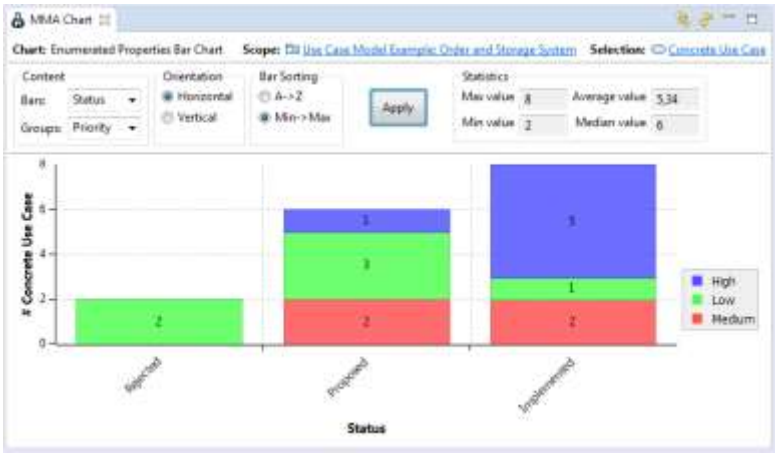


Figure 31: *MMA Chart View displaying an Enumerated Properties Bar Chart of Use Cases organized around status and priority.*

This view is part of the model analysis capabilities in MetaModelAgent and is described in detail in the *MetaModelAgent Model Analysis User Manual*.

9.9 Console View

The standard Console View is extended with a MetaModelAgent console tab which displays details about activations/deactivations and any related problems.

10 Decorators

A decorator is an adornment in an Eclipse view or editor which shows the state of an item.

MetaModelAgent provides three kinds of decorators, which are described in this chapter.

To enable or disable the decorators:

1. Select *Window→Preferences* from the *main* menu to open the *Preferences Dialog*.
2. Select the *General/Appearance/Label Decorations* entry in the *Preferences Dialog*.
3. Check the check boxes for *MetaModelAgent Problem Decorator* and *MetaModelAgent Metaclass Decorator* in the list of decorators.
4. Click OK to close the dialog.

10.1 Problem decorator

A *Problem Decorator* is an icon adornment to an item in the Explorer View or in the Diagram Editor which indicates that there are some problems associated with the item.

Figure 32: *Example of Problem Decorators on items in the Explorer View.*

Figure 33: *Example of Problem Decorators on items in the Diagram Editor*

The problem decorators will automatically be updated or removed in the *Explorer View* when the model is changed. Due to limitations in the Diagram Editor, a diagram needs to be closed and reopened, for the changes to occur.

To view the problems associated with an item, select *MetaModelAgent→Show in Problem View→Selected Item* in the context menu for the item. That will bring up the *MMA Problem View* and highlight all problems associated with the item.

10.2 Metaclass decorator

A *Metaclass Decorator* is a textual adornment at the end of an item in the Explorer View and some other view and dialogs which shows the item's metaclass.

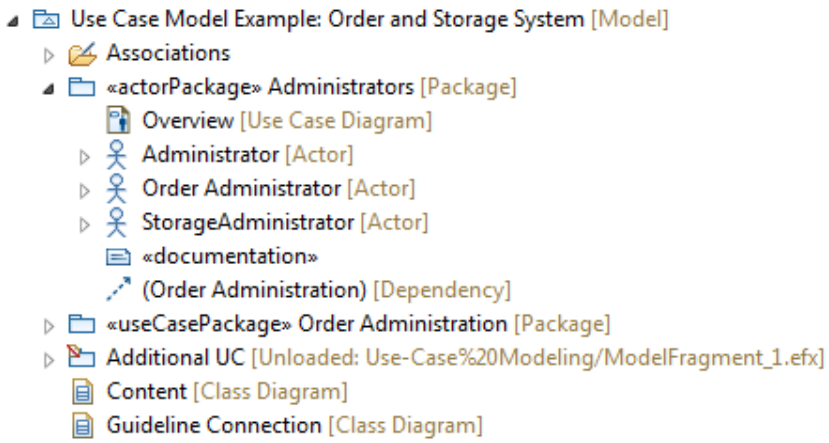


Figure 34: *Example of Metaclass Decorators within [...] on items in the Explorer View*

To view the guidelines associated with the metaclass, just select the Item in the Explorer View and the corresponding guidelines from the metamodel will be presented in the *MMA Guidance View*.

If there are any unloaded model fragments, these are indicated with a decorator that shows the relative file path to the fragment, see the “Additional IC” package in the figure above. .

10.3 Suppress validation decorator

A suppress validation decorator is a small black cross icon adornment to an element in the Explorer View which indicates that the element has been suppressed from validation.

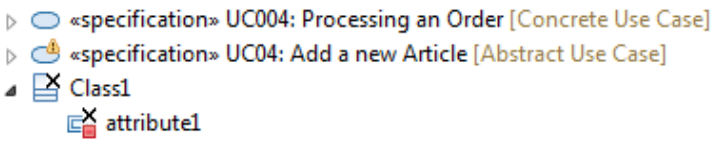


Figure 35: *Example of Suppress Validation Decorator in the Explorer View*

See chapter 4.8 for more information on how to manage suppress validations.

11 Preference settings

Select *Windows→Preferences* from the main menu to bring up the Eclipse preference settings dialog. Select *MetaModelAgent* in the tree view to display the MetaModelAgent preference page.

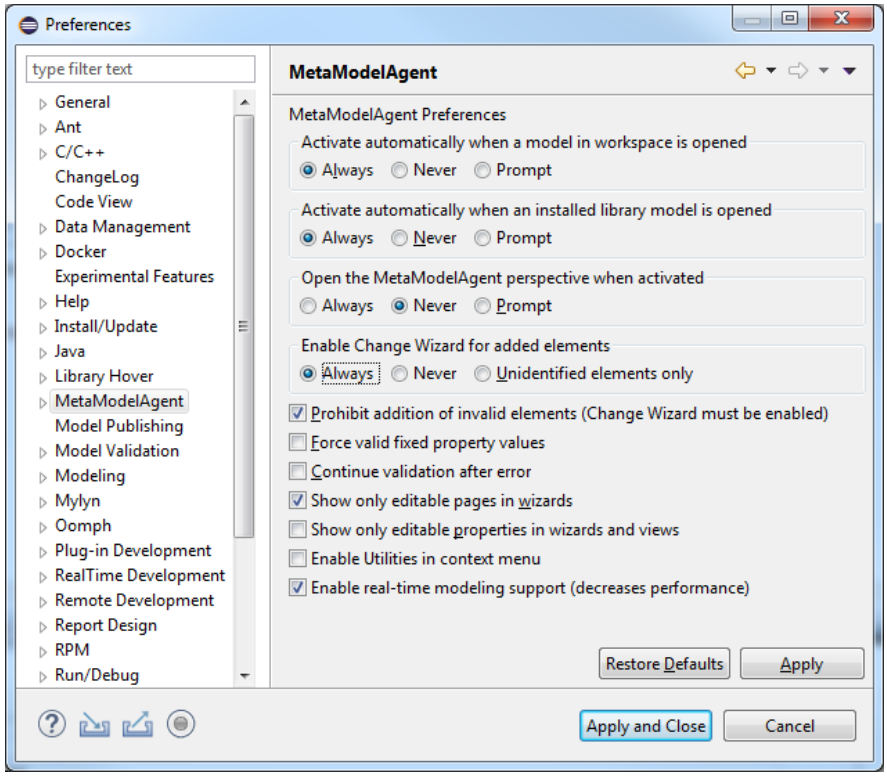


Figure 36: *The MetaModelAgent preference settings page. This picture may not look identical in your MetaModelAgent installation.*

The nested preference page for License Management is described in the **MetaModelAgent License Management Manual**.

11.1 Automation settings

Activate MetaModelAgent automatically when a model in workspace is opened

This setting controls if MetaModelAgent automatically shall be activated when a model is opened. The preconditions for activation is that there is only one metamodel associated to the model and the model itself is not a metamodel. If no associated metamodel exist, the built in General UML Guidelines or General UML-RT Guidelines metamodel will be used.

There are three different values of this setting:

Always MetaModelAgent is automatically activated when a model is opened.

Never MetaModelAgent is not automatically activated when a model is opened. Activation must be done manually by selecting *MetaModelAgent→Activate* from the context menu.

Prompt When opening a model, a pop-up dialog occurs letting you select to activate MetaModelAgent for the model.

Activate MetaModelAgent automatically when a installed library model is opened

This setting controls if MetaModelAgent automatically shall be activated when an installed library model, deployed in a plugin, is opened. The preconditions for activation is that there is only one metamodel associated to the model and the model itself is not a metamodel. If no associated metamodel exist, the built in General UML Guidelines or General UML-RT Guidelines metamodel will be used.

There are three different values of this setting:

Always	MetaModelAgent is automatically activated when an installed library model is opened.
Never	MetaModelAgent is not automatically activated when a model is opened.
Prompt	When opening a model, a pop-up dialog occurs letting you select to activate MetaModelAgent for the installed library model.

An installed library model is normally opened automatically as soon as any model is in need of it, e.g. refers to any element in the library model.

You may observe that installed library models cannot be activated manually after they are being opened. If you want to activate an installed model that has already have been opened, you must restart the workbench.

Open the MetaModelAgent perspective when activated

This setting controls if the *MetaModelAgent Perspective* should be automatically opened when MetaModelAgent is activated for a model.

There are three different values of this setting:

Always	The *MetaModelAgent Perspective* is automatically opened when MetaModelAgent is activated for a model.
Never	No perspective switch occurs when MetaModelAgent is activated for a model.
Prompt	When activating MetaModelAgent for a model, a pop-up dialog prompts you for switching the perspective to the *MetaModelAgent Perspective*.

Enable Change Wizard for added items

This setting controls if the MMA *Change Wizard* should be automatically invoked when a model item is added using standard UI-functions. The *MMA Change Wizard* will automatically set the values of mandatory properties with fixed values. If there are several metaclasses in the metamodel matching the item to be added, there will be an initial *MMA Guided Change Wizard* page where you may select which metaclass to apply on the new item.

Always	The *MMA Change Wizard* is automatically invoked when an item is added.
Never	No *MMA Change Wizard* will automatically appear. The host tool behaves as normal when model items are added using built-in UI-functions.
Unidentified elements only	When activating MetaModelAgent for a model, a pop-up dialog prompts you for switching the perspective to the *MetaModelAgent Perspective*.

You may notice that the item is being added to the model even if the *MMA Change Wizard* is cancelled.

IMPORTANT This setting should be turned off if there are any other plug-in extensions that automatically add model items to an activated model.

Force valid fixed property values

This setting controls if MetaModelAgent automatically should fix invalid property values which otherwise would be indicated as errors. Only properties that have a fixed valid value will be automatically fixed.

11.2 Other settings

Prohibit addition of invalid elements

This setting controls if MetaModelAgent should allow the addition of invalid elements or not, when using the built-in functionality.

- If checked, MetaModelAgent will present a confirmation dialog that the addition will not be performed.
- If unchecked, MetaModelAgent will present a selection dialog where the user can select to fulfil the addition of the invalid element, or to cancel the addition operation.

Continue validation after error

This setting controls if MetaModelAgent should try to continue validating a model beneath an unidentified model item. If this setting is checked, there might occur incorrect problems in the *MMA Problem View* due to bad assumptions done during validation

Show only editable pages in wizards

This setting controls if wizard pages in the *MMA Add Wizard* containing no editable properties should be displayed or bypassed.

Show only editable properties in wizards and views

This setting controls if properties containing valid fixed values should be displayed or not in the wizard or views. If the used metamodel defines a lot of fixed value properties, this setting may reduce the number of displayed properties a lot, letting you focus on the ones that you should edit.

Enable Utilities in context menu

Checking this setting enables the Utility submenu in the MetaModelAgent context menu in the Explorer View and in the Diagram editor. Currently there is only one utility function available that adjusts the representation of element documentation.

Enable real-time modeling support (RSARTE and HCL RTist only)

This setting must be checked if MMA is to be used for UML real-time models in RSARTE or HCL RTist. If you are using standard UML this setting can be unchecked for improved performance in large and/or multiple models.

12 Utility functions

12.1 Documentation Fix

MetaModelAgent prior to v4.1.3 did not set the 'annotated element' reference properly for UML Comments representing element documentation.

In RSAD/RSARTE the effect of this is only visible on the .emx file level. No effect can be observed in the user interface.

In Papyrus the effect is visible if there is other comments that are added to an element prior to the documentation being added.

If you have not used MetaModelAgent v4.1.2 or earlier no action is needed to be taken.

If you have used MetaModelAgent v4.1.2 and want to correct the representation of element documentation for your model, you will have access to a utility function that automatically correct your models added. The behavior of the function is slightly different between RSAD/RSARTE and Papyrus.

To be able to use the utility function:

1. Check the preference setting "*Enable Utilities in context menu*" on the MetaModelAgent preference page to enable the utility sub menu.
2. Select the top element in the model to be corrected and select *MetaModelAgent→Utilities →Fix documentation references…* from the context menu.
3. In the popup-dialog select to do a dry run (nothing will change) or run to perform the fix of the documentation
4. Brief information of how many elements for which documentation will be adjusted/has been adjust will be presented in a new popup-dialog. Detailed information of which elements it concerns will be listed in the MetaModelAgent tab of the Console View.
5. If changes have been made to your model. Save your model.
6. Change the preference setting back by unchecking the "*Enable Utilities in context menu*" preference setting.

If you want more information about this, please contact support@adocus.com.

13 Headless validation

The model validation feature in MetaModelAgent can be started and executed from a command prompt, without any interaction with the graphical UI. The result of the validation will be written to a semicolon-separated text-file which can be post-processed in some other tool, for example MS Excel.

13.1 Running headless validation from command prompt

To execute the MetaModelAgent model validation from the command prompt when using RSAD or RSARTE, enter the following:

```
<INSTALL_DIR>\eclipse.exe
       -data <WORKSPACE_DIR>
       -nosplash
       -product com.ibm.rational.rsa.product.ide
       -application com.adocus.mma.headless.validation
       [-model <MODEL_FILE_PATH>]
       [-metamodel <METAMODEL_FILE_PATH>]
       [-input <INPUT FILE_PATH>]
       -output <OUTPUT_FILE_PATH>
       [-format <FORMAT>]
       [-verbose]
```

Where <INSTALL_DIR> Is the directory where RSAD/RSARTE or Papyrus is installed.

Argument	Description	Comment
`-data`	The full path to the workspace to be used	Mandatory
`-nosplash`	Indicates that no splash screen should appear.	Mandatory
`-product`	The identity of Eclipse-based product that should be run.	Should be omitted when using Papyrus.
`-application`	The identity of the MetaModelAgent headless validation function.	Mandatory
`-model`	The path to the model file to be validated.	Should be omitted if several models should be validated in the same session using the `-input` argument.
`-metamodel`	The path to the metamodel file to be used in the validated.	Should be omitted if several models should be validated in the same session using the `-input` argument. Can be omitted when using RSAD/RSARTE if the model is connected to a single metamodel.
`-input`	The path to a csv-file holding paths to models to be validated and the metamodels to be used for	Should be omitted if a single model is to be validated using the `-model` argument

	validation (see below for details).	
`-output`	The path to the file where the result of the validation is written (see below for details).	Mandatory
`-format`	The format to be used in the output file	Should be omitted if the old format in MMA 4.1.1 or older versions is preferred
`-verbose`	Writes progress info to the console	Optional

Output format

If the `-format` argument is used, the output file will be a semicolon-separated text file. The valid values of the `-format` argument are:

Value	Description
`csv`	The output file will contain a list of all problems found The first row will be a row of headings
`csv_summary`	The output file will contain a summary of the result of the validation only and not any list of problems.
`csv_full`	The output file will contain both the summary and the list of all problems found

The summary is a section of semicolon separated attribute/value pairs. The following attributes are included:

Attribute	Description
`MetaModelAgent version`	The current build of MetaModelAgent
`Model`	The name of the model element being validated
`Metamodel`	The name of the metamodel being used in the validation
`Total number of elements`	Total number of elements in the part of the model being validated. Only element kinds that are supported by MetaModelAgent are included
`Validated elements`	Number of elements that have been validated
`Suppressed elements`	Number of elements that are suppressed from validation
`Errors`	Number of errors in the model
`Warnings`	Number of warnings in the model
`Infos`	Number of information issues in the model

The number of attributes and the order of them may be changed in future versions of MetaModelAgent.

The fields in the problem list section of the validation output are:

Field name	Description
`Item Name`	The name of the element holding the problem, same as in the MMA Problem View
`Full Qualified Item Name`	The full qualified name of the element holding the problem
`Problem Description`	The description of the problem, same as in the MMA

	Problem View
`Subject`	The subject of the problem, if appropriate, same as in the MMA Problem View
`Full Qualified Subject Name`	If the subject is an element, the full qualified name of the subject element
`Item Definition`	The metaclass in the metamodel that the element holding the problem has been associated to
`UML Metaclass`	The UML metaclass for the element holding the problem

IMPORTANT when using Papyrus: if there is a metamodel connected to the model (dependency relationship), there will always be one invalid error reported that says that the dependency target has an invalid property value. That is because MMA cannot resolve the dependency target because of the lack of package import of the metamodel.

Validating more than one model

When validating several models in the same session, the `-input` argument should be used instead of the `-model` and the `-metamodel` argument.

The value of the `-input` argument should be the path of a CSV-file that contains the paths to the models and the metamodels to be used in validation. The metamodel is mandatory in Papyrus but optionally in RSAD/RSARTE and HCL RTist if there is a single metamodel connected to the model.

The format of each line in the input-file should be:

```
<MODEL_FILE_PATH>;<METAMODEL_FILE_PATH>
```

When validating several models the output file will contain the result from all validations in the same order as the model file paths appear in the input file.

13.2 Running headless validation using ANT-script

To simplify batch-validation, an ANT-script (ant.apache.org) may be used. Below is a template for an ANT-script that may be used (see above for details of the arguments).

```
<?xml version="1.0" encoding="UTF-8"?>
<!This is a ANT-script template for validating a model in MetaModelAgent>
<project name="mma.headless" default="MMAheadless" basedir=".">
   <target name="MMAheadless">
      <property name="eclipse_dir" value="<INSTALL_DIR>"/>
      <property name="execution_dir" value="<EXECUTION_DIR>"/>
      <property name="workspace_dir" value="<WORKSPACE_DIR>"/>
      <exec executable="${eclipse_dir}\eclipse.exe" dir="${execution_dir}"
            vmlauncher="false">
         <arg value="-data '${workspace_dir}'">
         <arg value="-nosplash">
         <arg value="-product com.ibm.rational.rsa.product.ide">
         <arg value="-application com.adocus.mma.headless.validation">
         <arg value="-model '<FILE_PATH_FOR_MODEL_FILE>'">
         <arg value="-metamodel '<FILE_PATH_FOR_METAMODEL_FILE>'”>
         <arg value="-output '<FILE_PATH_FOR_OUTPUT_FILE>'">
         <arg value="-format <FORMAT>">
      </exec>
   </target>
</project>
```

The `<arg value="-product com.ibm.rational.rsa.product.ide">` line should be omitted if Papyrus is used.

14 Public API

There is a limited public API available that currently supports UI activation of a model and Non-UI validation. The API is provided in the package **`com.adocus.mma.api`** which is provided by the **`org.adocus.mma.core`** plug-in.

The methods in the API can be used in your own plug-ins to interact with MetaModelAgent

14.1 Activating MetaModelAgent for a model element

The following public static method can be used to activate MetaModelAgent for a given model.

```
void com.adocus.mma.api.UiActivation.run(
   final org.eclipse.emf.ecore.EModelElement umlMetaModel,
   final org.eclipse.emf.ecore.EModelElement activationItem)
```

Argument	Description
`umlMetaModel`	A reference to the metamodel to be used in the activation
`activationItem`	A reference to the model element to be activated

The result will be the same as if the activation has been performed using an activation operation in the MetaModelAgent UI.

The method will throw an exception if the activation fails of some reason.

14.2 Validating a model

The following public static method can be used to perform a validation of a single model. The result of the validation will be written to a file with the same format as the result of the headless validation, see previous chapter.

The method will return a summary of the validation with the same content as the output summary from the headless validation, see previous chapter.

```
java.util.Map<String,String> com.adocus.mma.api.NonUiValidation.run(
   final org.eclipse.emf.ecore.EModelElement umlMetaModel,
   final org.eclipse.emf.ecore.EModelElement itemToBeValidated,
   final int format,
   final java.io.PrintWriter out,
   final java.io.PrintStream statusOut)
```

Argument	Description
`umlMetaModel`	The metamodel object to be used in the validation
`itemToBeValidated`	The model element object to be validated
`format`	An integer specifying the format to be used for the output. There are public integer constants starting with `FORMAT_` defined in the `NonUiValidation` class that represents the supported formats. For details see the format argument in the previous chapter.
`out`	The PrintWriter object to be used as output
`statusOut`	The PrintStream object to be used for status report

The method results in a map that contains a summary of the validation. The keys in the summary map will be the same as the attributes in the output summary of the headless validation, see previous chapter. There are defined public string constants starting with `KEY_` for all keys in the **`NonUiValidation`** class.

The method will throw an exception if the validation fails for some reason.

The old API-method **`com.adocus.mma.api.BatchActivation.run()`** is still available for backward compatibility, but should be regarded as deprecated.

14.3 Model modification without validation

When doing model modification using an EMF transaction from a plugin, MetaModelAgent will automatically react on the operation and invoke validation of the effected element(s) if they are part of an activation. In some situations there might be a need to prohibit MetaModelAgent from reacting on these kind of modifications.

To achieve that behavior, let the EMF command implement the **com.adocus.mma.api.IReactCommand** interface. The implementation of the interface´s **`react()`** method should return false for MetaModelAgent to not react on the modification operation.

14.4 Registering a metamodel for a specific kind of model

Metamodels deployed in a plugin or available in the workspace, can be registered in MetaModelAgent together with a method that checks if one of the metamodels matches the current model. This means that there will be no need for a dependency relationship from model to metamodel for MetaModelAgent to use for activation.

This is especially useful if a model is split into several fragments/submodels that should be possible to activate independently.

More information of this API is available in the *MetaModelAgent Metamodeling Manual*.

15 Known limitations

There are a few known limitations which may affect the expected behavior:

- Elements from imported libraries cannot be used as property values in the Create New Model Wizard. It can be done in RSAD/RSARTE and HCL RTist if the model library has already been loaded by some other model, but it can never be done in Papyrus. Workaround is to fill in these properties after the Create New Model wizard have been finished.
- Sub-classifiers are not automatically revalidated when inherited features from super-classifiers have been changed. Manual revalidation must be performed, preferable by using the refresh button in MMA Problem View.
- Unavailable mandatory association ends give unexpected errors. Mandatory association ends not owned by the element gives invalid errors if the association end and its owning association is stored in an unloaded model.
- Quick-fixes involving adding object flows, control flow, transitions and messages are not available.
- Dependencies and its variants are handled by MetaModelAgent as owned by the source element, even thus dependencies always are owned by a package.