Papyrus User Guide Series

About UML profiling, version 1.0.0

This document is part of a series of documents called, Papyrus User Guides Series, and dedicated to assist the usage of Papyrus. The focus of this volume is to describe how to use the concept of profile within Papyrus. It handles both following aspects, firstly their design and definition and secondly their application and usage.

Sébastien Gérard

25/11/2011

Table of Contents

Historic of the document 2
Introduction 3
Profile Modeling and Definition 5
  Introduction to UML Profiles 5
  Profile Creation and Modeling 7
    Profile creation 7
    Profile authoring 12
  Profile Definition and Export 21
Profile Applying and Use 24
  Applying a profile 24
  Using the stereotypes of a profile 26
    Applying a stereotype 26
    Assigning values to properties of stereotypes 28
    Displaying options of a stereotype application 29
References 32

HISTORIC OF THE DOCUMENT

Version #	Contributors <Name>, <Affiliation>	Comments
1.0.0	Sébastien Gérard, CEA, LIST, Laboratory of model driven engineering for embedded systems	This is the initial version of the document. It is based on the Papyrus version 0.8.2.

INTRODUCTION

The complexity level reached today for the development of computer-intensive systems requires for new design paradigms. In this context, the most promising used principle is commonly referred to as "raising the level of abstraction" and is one of the main distinguishing features of model-based approaches.

The principle of separation of concerns is indeed widely used in engineering, including model-based engineering, to address the ever growing complexity of system designs. This principle implies specialization of technologies, and in the context of model-based engineering, it has led to the development of domain specific modeling languages (DSML). Those DSMLs provide constructs that are directly aligned with the concepts of the domain in question. Yet, the large majority of systems, including software-intensive systems, are still denoted using general purpose languages. The main reasons are probably the ready availability of tools, language documentation and training courses for these languages, as well as the ready availability of expert practitioners. Also, for many enterprises, using general-purpose languages appears as the less risky and less expensive alternative. Consequently, various domain-specific program libraries, written in a general purpose language, are often developed in place of DSLs. Main drawbacks of this approach are modeling tools are then not domain-oriented, and automated tools, such as compilers, cannot recognize and take advantage of domain-specific knowledge (e.g., for optimization). Behind those aforementioned disadvantages, the issue is that the positive impact of using model-based engineering for designing complex systems is then reduced: models are more difficult to be authoring, and their automatic exploitation is limited, and any way much more difficult to implement.

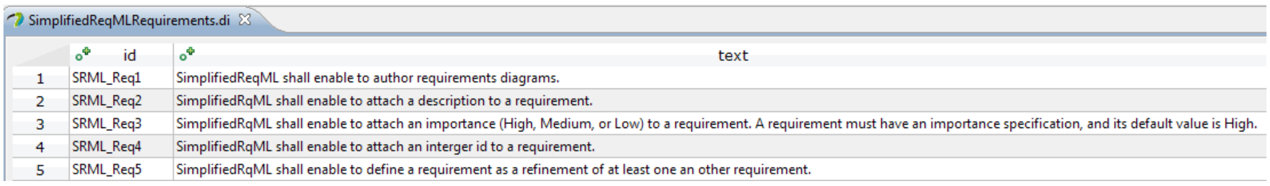
In addition, a key design principle for industry is to maximize reuse of technology, knowledge, and tools—an argument that usually resonates in industry, where reducing cost is a primary objective. Unless they serve core business purposes or provide major competitive advantages, enterprises generally prefer well-known standard technologies. The Object Management Group (OMG, www.omg.org) is one of the principal international organizations promoting standards supporting the usage of model-based software and systems development. The Unified Modeling Language (UML) standard (1) is probably the most representative of these and has had definitively significant successes in the software industry as well as in other domains, such as IT and financial systems. UML was designed as a general-purpose modeling language as well as a foundation for deriving different domain-specific languages, mainly through its profile mechanism. Consequently, when an industry needs a DSML, the best advice we can give is: "Don't reinvent the wheel, adopt UML instead, but tailor it to your needs!"

"Don't reinvent the wheel, adopt UML instead, but tailor it to your

Papyrus is the Eclipse graphical editing tool for UML2: www.eclipse.org/papyrus. In accordance with its primary goal to implement the complete standard specification of UML2, Papyrus provides an extensive support for UML profiles. It includes hence all the facilities for defining and applying UML profiles in a very rich and efficient manner. But, it also provides powerful tool customization capabilities similar to DSML-like meta-tools. This way, Papyrus is a tool enabling to gather the advantages of using a general-purpose language such as UML2, but also those of DSML-based approaches.

The purpose of this document is to detail all Papyrus features related to DSML-engineering, including information about UML profiles, and Papyrus customization facilities. To achieve this goal, the document is structured as follow: Next chapter is dedicated to the UML profile concept. It will outline the concept of UML profiles and then denote how to model, define and apply a profile within Papyrus. Then, the document describes all parts of the modeling tool that may be customized. Then, an example is given, and some conclusions are drawn before citing all the references cited all along the document.

In order to illustrate this document, we will use a very simple example consisting in engineering a very basic DSML for requirement engineering, named SimplifiedReqML. The requirements for this DSML are shown in Figure 1 which was authored using the Papyrus table editor of SysML requirements.



	id	text
1	SRML_Req1	SimplifiedReqML shall enable to author requirements diagrams.
2	SRML_Req2	SimplifiedRqML shall enable to attach a description to a requirement.
3	SRML_Req3	SimplifiedRqML shall enable to attach an importance (High, Medium, or Low) to a requirement. A requirement must have an importance specification, and its default value is High.
4	SRML_Req4	SimplifiedRqML shall enable to attach an interger id to a requirement.
5	SRML_Req5	SimplifiedRqML shall enable to define a requirement as a refinement of at least one an other requirement.

Figure 1 – Requirements description of SimplifiedReqML

PROFILE MODELING AND DEFINITION

The purpose of this chapter is to provide the Papyrus users all the documentation needed in order to be able to use UML profiles. It includes the information for modeling and defining a profile using the Papyrus UML profile editor, but also the information for the usage of a profile within a user application model.

Introduction to UML Profiles

As already mentioned previously, because of the diverse nature of the disciplines needed for designing real-time and embedded system, it is clear that a single modeling language will not be enough to cover all the various concerns involved in this specific area. Consequently, there has been much discussion about the suitability of UML for such domains relative to custom domain-specific modeling language designed from scratch (2). A custom language has the obvious advantage that it can be defined in a way that is optimally suited to a specific problem. At first glance, this may seem the ideal approach, but closer examination reveals that there it can have serious drawbacks. If each individual sub-domain of a complex system uses a different modeling language, the problem will be how to interface the various sub-models into a consistent integrated whole that can be verified, tested, or simply unambiguously understood. Furthermore, there is also the issue of designing, implementing, and maintaining suitable industrial-strength tools for each custom language as well as providing teaching materials and training, all of which can result in significant and recurring expenses.

Conversely, although UML was designed to eliminate the accidental complexity stemming from gratuitous diversity, it provides the profile concept for deriving domain-specific modeling languages from its set of general language concepts. An important advantage of this approach to DSML design is that it allows reuse of existing UML tools and widely available UML expertise. Note that we are not saying that UML profiles completely avoid DSML integration problems. However, many of the fragmentation issues[1] stemming from diversity of individual domains and their languages can be mitigated because all domain-specific modeling languages derived from UML share a common semantic and syntactic foundation. There is typically a lot of commonality between the various disciplines involved in system engineering. For instance, the concepts of package, composition, property and connector, which are provided by UML, are common to many disciplines, as are the basic notions of object, class, and interface.

The basic premise of profiles is that all domain-specific concepts are derived as extensions or refinements of existing UML concepts, called UML metaclasses. These extensions are called stereotypes. A stereotype definition must be consistent with the abstract syntax[2] and semantics of standard UML meta-classes it extends. Consequently,

[1] This is used to refer to the situation that occurs when different domain-specific languages are used to describe different aspects of a complex system. For example, one language might be used to describe the user interface function while a different one for the database management and access functions. The individual languages involved could have very different models of computation, which raises the question of how to meld the different specifications into a coherent and consistent whole.

[2] In the domain of language engineering, the abstract syntax of a language usually refers to the definition of the concepts of the language including their properties and

a profile-based model can be created and manipulated by any tool that supports standard UML. Moreover, because the concepts underlying a profile are specializations of existing UML concepts, it is more easily learned by anyone with knowledge of UML.

A stereotype is defined either as an extension of a UML base metaclass or as a specialization of an existing stereotype. The extension relationship of UML is not an association but a kind of association directed from the stereotype to the extended metaclass. Consequently, the metadata conveyed by the associated the attributes of the stereotype are associated to the extended metaclass in a transparent manner for the metaclass itself. This allows profiles owning the stereotypes to be applied and removed dynamically without modifying the underlying models—a fundamental feature of the profile mechanism.

A stereotype may have attributes and may be associated with other stereotypes or existing UML metaclasses. A stereotype may be required (i.e., its isRequired metaproperty must be set to true). In that case, the stereotype will be automatically applied to all instances of the extended metaclasses the user will model. This feature is very useful to impose a specific terminology.

Constraints, such as OCL constraints (3), can also be defined in a profile. They can apply to stereotypes defined in the profile or those imported by the profile. They can also be used to further constrain elements of the UML metamodel. For instance, one could define an OCL constraint that all instances of Class in a model are active, or that all instances of Class must have at least one Operation (regardless of whether the Class is extended by a stereotype or not). However, not all constraints can be written in OCL. In that case, it is common to denote those latter in natural language. The drawback is that such constraints are no more automatically interpretable and need to be first rewritten in some language the UML tool will understand. In the context of Papyrus, it is then usual to use Java.

As an alternative to aforementioned constraints for pruning UML, this letter provides a metaclass filtering function known as “isStrict”. The filtering rules allow selecting the kinds of metaclasses that are visible in the model (details on the filtering rules are defined in clause 18.3.6 of [1]). The actual filtering occurs when the profile is applied. This feature can be used as a means to define viewpoints that only filter out metaclasses.

Since “strict” filtering is merely a restricted view of a model, use of this feature does not change conformance to the underlying metamodel. However, no specific constraint is defined on how the subset is to be defined. Thus, the filtering mechanism does not prevent hiding of mandatory metamodel elements; for example, a poorly defined profile could specify that only Connector elements are visible, despite the fact that a Connector needs at least two ConnectorEnds. This means that a designer could create models through the view offered by the “strict” application of the profile that are not compatible with standard UML (cf. SelfFilterProtect definition).

PS: Let’s notice that this mechanism is not yet implemented within papyrus.

---

relationships. It is usual to denote such abstract syntax using a meta-model written as for example in MOF. The abstract syntax is not dedicated to be manipulated directly by the user. For that, the language engineer defines a concrete syntax, also called notation.

Warning: Given that the primary constraint for profiles is that they cannot contradict UML, filtering rules should always be defined to prune only optional constructs. This will ensure self filtered protection, edit forward compatibility, and edit filtered backward compatibility. However, again, defining a compatible pruning is necessarily a manual task that requires deep UML expertise.

From a notational viewpoint, stereotypes can also be used to adapt the concrete syntax of UML in order to provide a more domain oriented concrete syntax. Defining the concrete syntax associated with a stereotype may be done in three ways which difficulty to implement is growing:

- Use the pre-ordained UML notation of the base class and standard French guillemets, « MySterotype ». The development cost is null. In addition, it reuses a well-known UML notation so that the cost of ascending the learning curve is reduced. Reuse and compatibility are maximized.
- Attach icons to the standard notations. With such a minimal effort, the standard representation is illustrated with domain specific icons. The development cost is then limited to the graphical design of the related icons. It reuses the standard notation and thereby keeps a flat learning curve.
- Redesign the whole notation. With tools available today, the development and maintenance cost of a specific notation is of course higher. However, it has the advantage of providing a concrete syntax that domain experts may prefer, but it also reduces reuse of UML expertise.

The rest of the chapter will now be focused on the Papyrus implementation of the UML profile concept, and is mainly going to answer following questions:

- How to create and author a UML profile?
- How to define and export a UML profile?
- How to apply and use a UML profile?

Profile Creation and Modeling

Profile creation

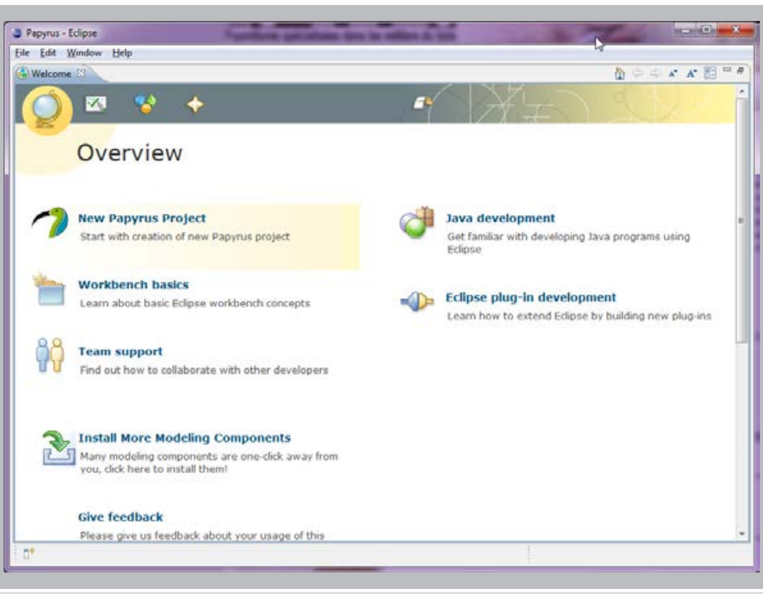
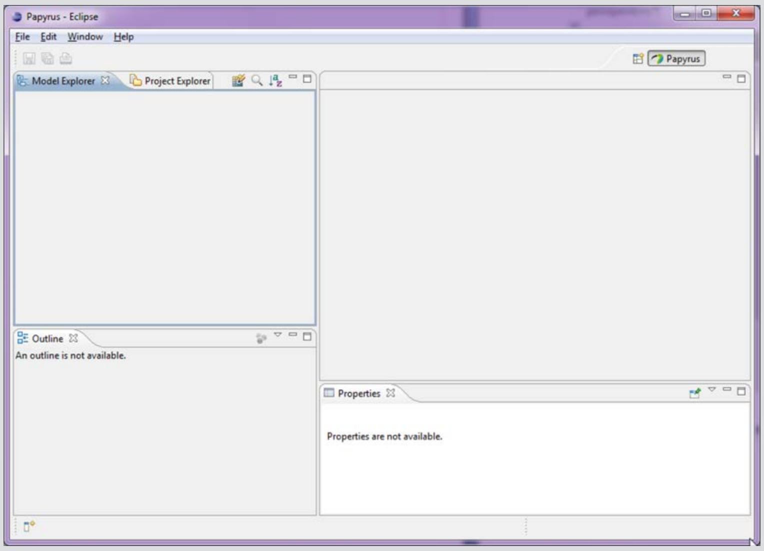
Table 1 denotes step-by-step the actions required to create a new UML profile with Papyrus. If it is the first time you are using Papyrus after installing it, you will first have the screen as shown in step 1 of Table 1, otherwise you will directly get Eclipse open with the Papyrus perspective[3] selected as shown in step 2 of Table 1.

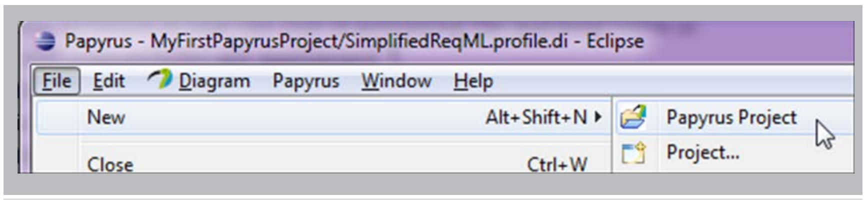
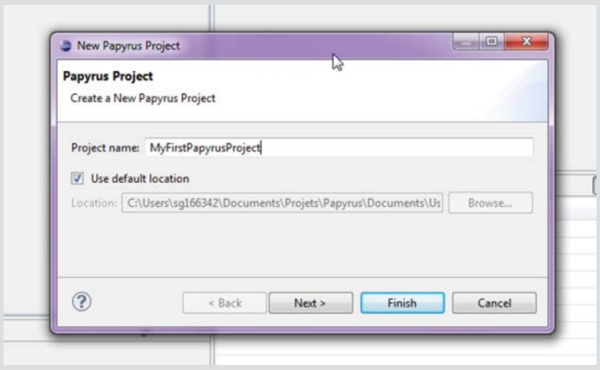
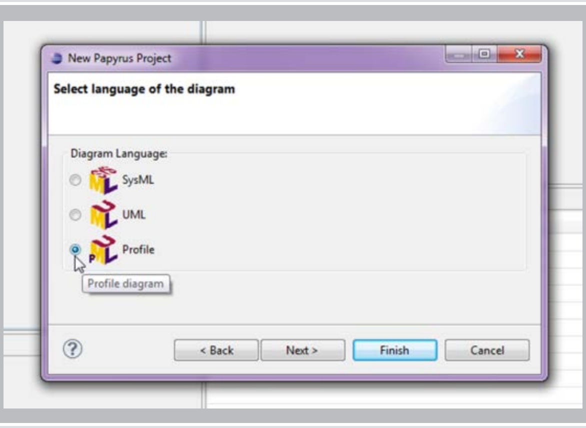
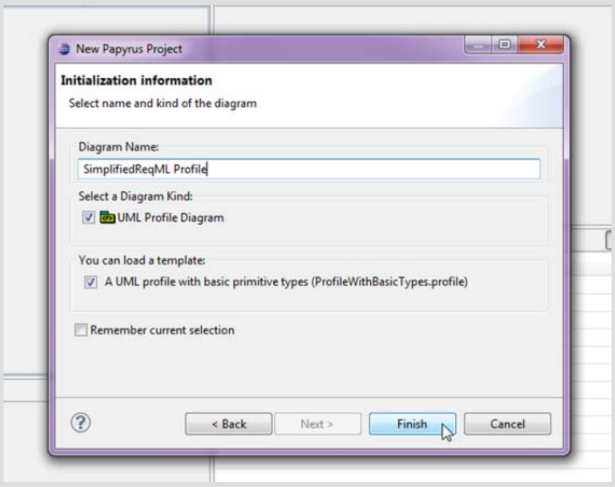
For creating a UML profile model, as it is also the case for a UML model, you need first to have created a Papyrus project which will be the folder for your profile model. One Papyrus project can consist of several UML models or/and profiles models. As shown in step 3 of Table 1, for creating a new Papyrus project, select in the menu bar: File > New > Papyrus Project. While you are creating a new Papyrus project, the project creation wizard will also enable you to create either a UML model or a UML profile as show in step 4 of Table 1.

During the process creation of a model or a profile, as shown in step 5 of Table 1, the creation wizard will ask you if you want to create a default profile diagram (if yes, let's set a name for this diagram), and if you want to import the basic primitive types package of UML. We advocate you to select at least this latter option that will be very useful later to design your profile. Hence, the basic types of UML are used for typing the stereotype's properties of the profile.

Finally, once the new model is created, you have to select the model explorer in order to start to model your application as shown in the figure shown at step 6 of the following table.

Table 1 – Steps for creating a new Papyrus project

Step Number	Screen Snapshots
Project Creation ——— Step 1	 Overview **New Papyrus Project** Start with creation of new Papyrus project **Workbench basics** Learn about basic Eclipse workbench concepts **Team support** Find out how to collaborate with other developers **Install More Modeling Components** Many modeling components are one-click away from you, click here to install them! **Give feedback** Please give us feedback about your usage of this **Java development** Get familiar with developing Java programs using Eclipse **Eclipse plug-in development** Learn how to extend Eclipse by building new plug-ins
Project Creation ——— Step 2	 Papyrus - Eclipse File Edit Window Help Papyrus Model Explorer Project Explorer Outline Properties An outline is not available. Properties are not available.

Project Creation ——— **Step 3**	
Project Creation ——— **Step 3**	
Project Creation ——— **Step 4**	
Project Creation ——— **Step 5**	

Papyrus - MyFirstPapyrusProject/SimplifiedReqML.profile.di - Eclipse

File Edit Diagram Papyrus Window Help

New Alt+Shift+N ▸ Papyrus Project

Close Ctrl+W Project...

New Papyrus Project

Papyrus Project

Create a New Papyrus Project

Project name: MyFirstPapyrusProject

☑ Use default location

Location: C:\Users\sg166342\Documents\Projets\Papyrus\Documents\Us Browse...

< Back Next > Finish Cancel

New Papyrus Project

Select language of the diagram

Diagram Language:

○ SysML

○ UML

● Profile

Profile diagram

< Back Next > Finish Cancel

New Papyrus Project

Initialization information

Select name and kind of the diagram

Diagram Name:

SimplifiedReqML Profile

Select a Diagram Kind:

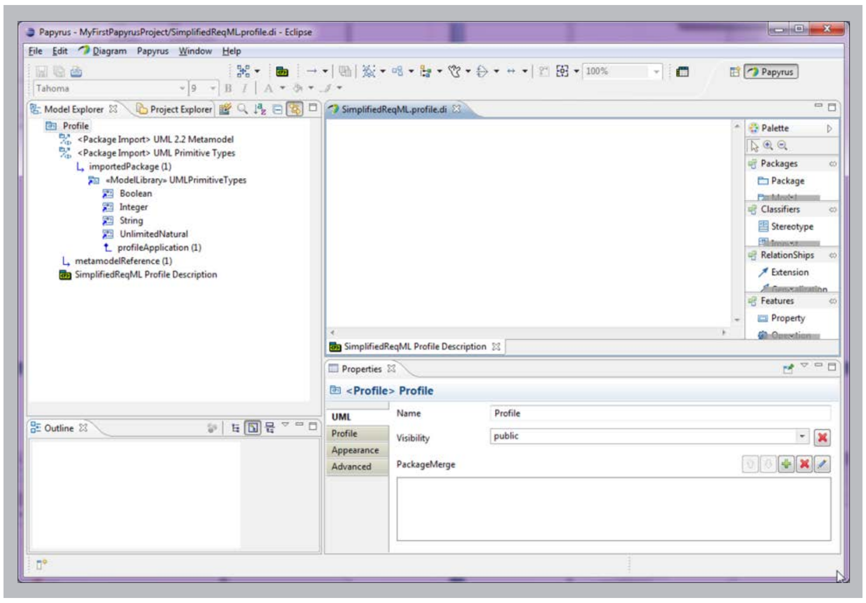
☑ UML Profile Diagram

You can load a template:

☑ A UML profile with basic primitive types (ProfileWithBasicTypes.profile)

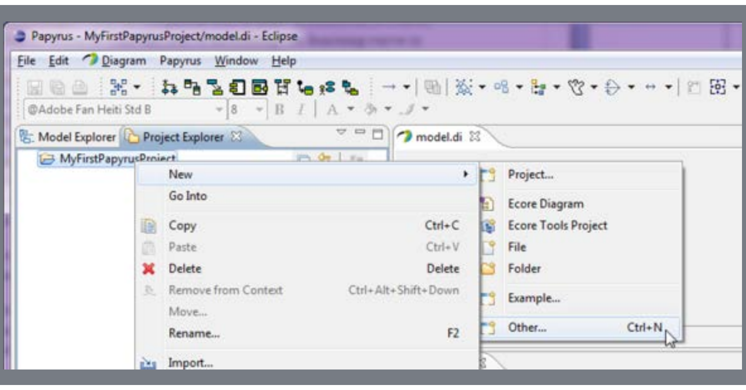
☐ Remember current selection

< Back Next > Finish Cancel

Project Creation ——— **Step 6**	

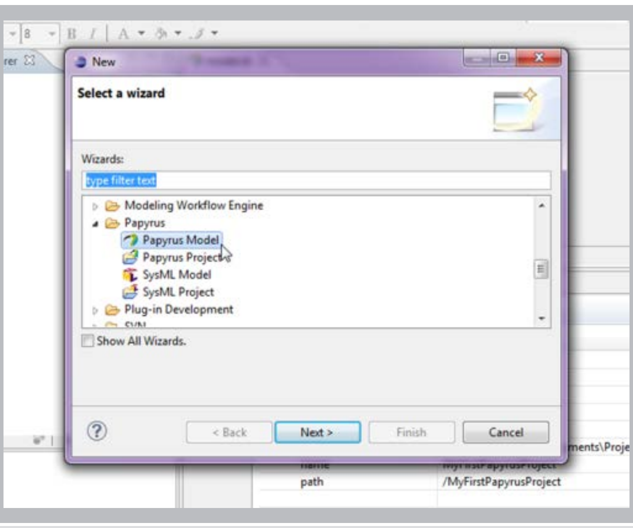
Next table denotes step-by-step the actions to create a new UML profile if you have already created a Papyrus project as previously denoted. For creating a new profile, you first have to select the project where you want to add your new profile and then right-click on the project folder and select Menu>New as shown in step1 of Table 2, and apply next steps as denoted in this table. You can also select the project folder and hit the keys, CTRL + N. Then, you have to choose Other and select Papyrus Model as shown in step 2 of Table 2. Once you are there, just let you drive by the wizard as illustrated by next steps shown in Table 2.

Table 2 - Steps for creating a new UML profile

Profile Creation ——— **Step 1**	

Profile Creation

Step 2


New

Select a wizard

Wizards:

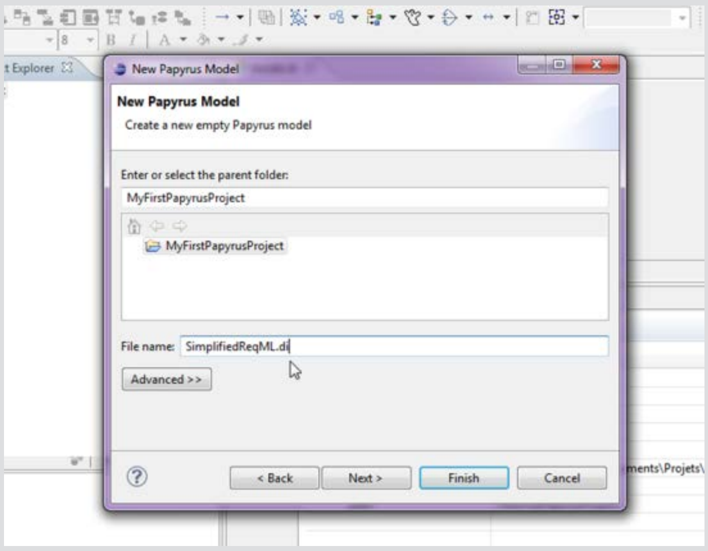
type filter text

- Modeling Workflow Engine
- Papyrus
  - Papyrus Model
  - Papyrus Project
  - SysML Model
  - SysML Project
- Plug-in Development

Show All Wizards.

Profile Creation

Step 3


New Papyrus Model

New Papyrus Model

Create a new empty Papyrus model

Enter or select the parent folder:

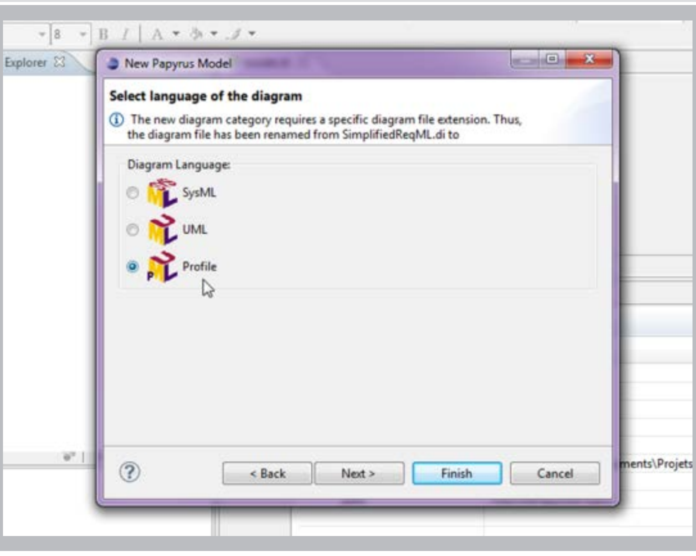
MyFirstPapyrusProject

MyFirstPapyrusProject

File name: SimplifiedReqML.di

Profile Creation

Step 4


New Papyrus Model

Select language of the diagram

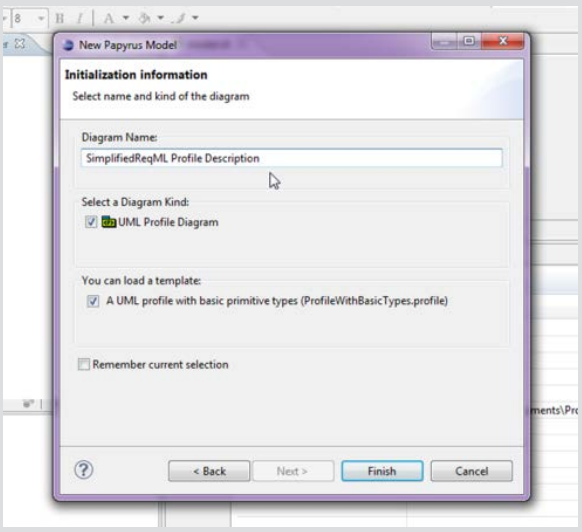
The new diagram category requires a specific diagram file extension. Thus, the diagram file has been renamed from SimplifiedReqML.di to

Diagram Language:

- SysML
- UML
- [✓ Profile]

Profile Creation

Step 5



Initialization information

Select name and kind of the diagram

Diagram Name: SimplifiedReqML Profile Description

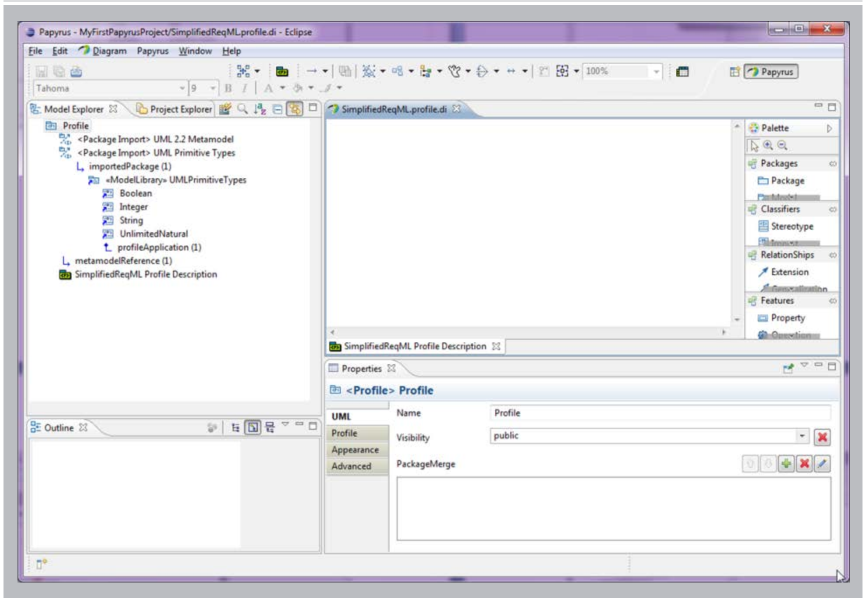
Select a Diagram Kind: [✓] UML Profile Diagram

You can load a template: [✓] A UML profile with basic primitive types (ProfileWithBasicTypes.profile)

[ ] Remember current selection

Profile Creation

Step 6



```
Profile
  <Package Import> UML 2.2 Metamodel
  <Package Import> UML Primitive Types
  importedPackage (1)
    «ModelLibrary» UMLPrimitiveTypes
      Boolean
      Integer
      String
      UnlimitedNatural
    profileApplication (1)
  metamodelReference (1)
  SimplifiedReqML Profile Description
```

Properties

<Profile> Profile

Name: Profile

Visibility: public

PackageMerge

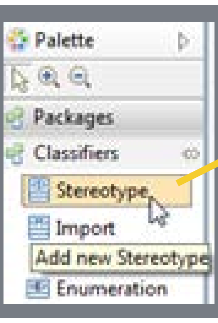
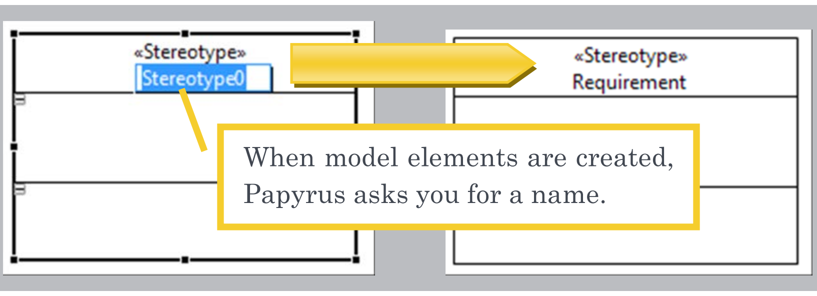
Stereotype definition

Once a profile has been created, it is now time to populate this latter with UML extensions, i.e. stereotypes, and their related concepts such as properties, extensions, and metaclasses.

Stereotype creation

A stereotype is created as any other UML model elements in Papyrus: select the related tool in the palette of the profile diagram editor, and then click in the place you want to create this element on the background of the diagram. If the palette is not open, just click on the small arrow on the upper right corner of the diagram editor. Papyrus will then ask you to provide a name. Once done, hit the return key and that all.

Table 3 - Steps for creating a stereotype

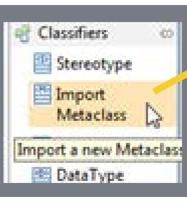
Stereotype Creation ——— Step 1	 Palette Packages Classifiers Stereotype Import Add new Stereotype Enumeration Selection of the palette tool for Stereotype modeling.
Stereotype Creation ——— Step 2	 When model elements are created, Papyrus asks you for a name.

Metaclass import

Once you have created a stereotype, you need to import the UML2 metaclasses you want to extend. Next table illustrate all the steps required to do it. First, you have to select the tool “Import Metaclass” within the palette of the profile diagram editor. Then, let’s click on the profile diagram where you want to drop the imported metaclass. A Papyrus dialog box is then opened in order to ask you to specify which metaclasses you want to import. Select the metaclasses in the left list and either drag and drop these latter in the right list or press the button with arrow directed from left to right and located between both aforementioned lists. Then, let’s press the button “ok” and it is done. The imported metaclasses are then shown in the diagram as illustrated in the step 3 of the Table 4.

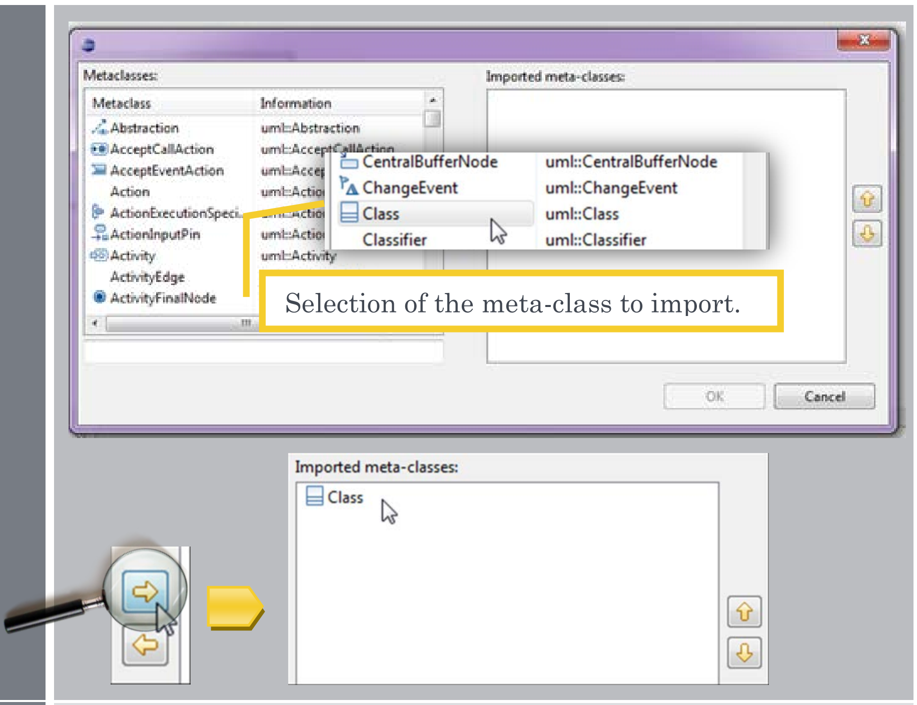
Note: When you import UML2 metaclasses, Papyrus is creating ImportElement model element referent to the metaclasses of the UML2 metamodel itself. Those so-called specific model elements, the ImportElement, are indeed a kind of proxy to the model elements contained in another model, in the UML2 metamodel. If a metaclass has already be imported, you do not need to import it again to use it in another context. You can select the imported meta-class from the model browser and drag and drop this latter on the diagram you want to use it.

Table 4 - Steps for importing UML2 metaclasses

Metaclass Import ——— Step 1	 Classifiers Stereotype Import Metaclass Import a new Metaclass DataType Selection of the palette tool for importing UML metaclass modeling.

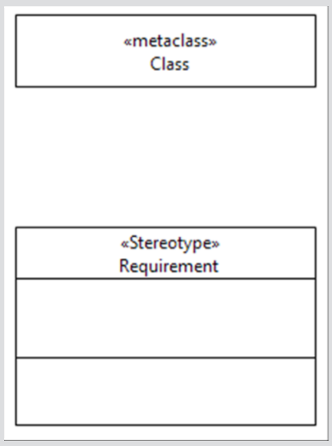
Metaclass Import

Step 2



Metaclass Import

Step 3



Extension creation

Once the stereotype is created and the meta-class is imported, you may then model the extension relation from the stereotype to the meta-class as shown in the following figure. The extension relationship is modeled using the extension tool in the profile diagram palette, . Within the diagram shown in Figure 2, both stereotypes «Requirement» and «Refinement» extend respectively both meta-classes Class and Dependency.

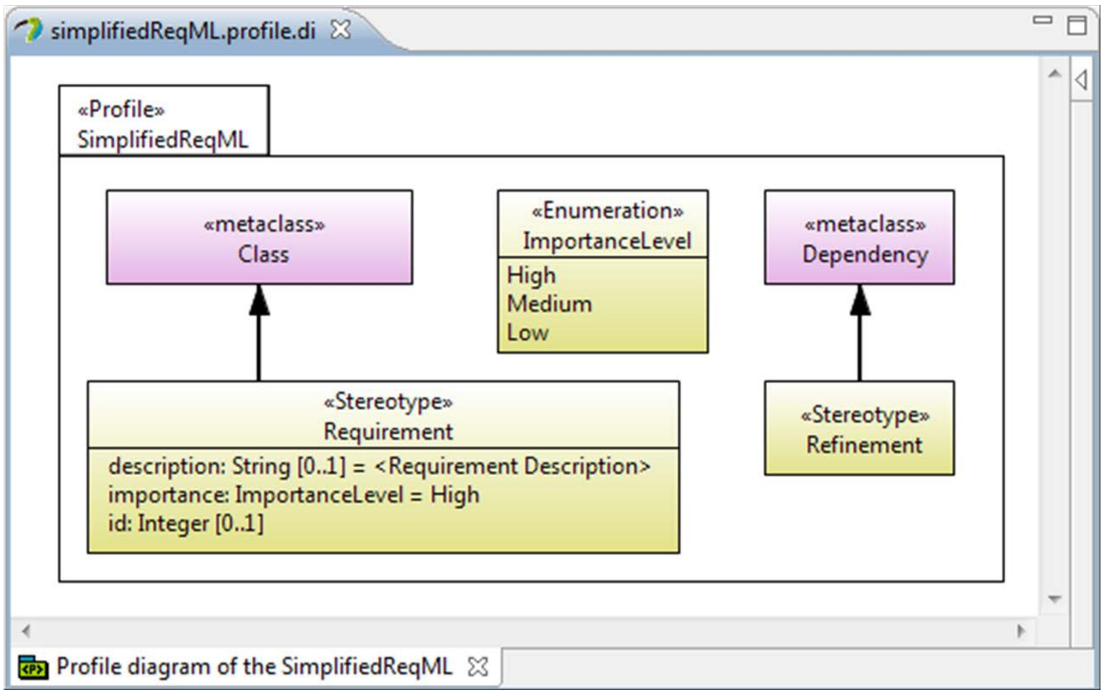


Figure 2 - Example of profile model

Subprofile creation

Some profile may be complex due as for example to their scope that may be large. In order to cope with this complexity, it is then possible to decompose a profile into a hierarchy of subprofiles. A subprofile behaves such as a sub-package; it is a container of stereotypes.

To create a sub profile, select the profile tool in the profile diagram palette ( ) and then click on the profile diagram where you want to create your new profile.

In the example denoted in Figure 3, we have created two sub-profiles of the SimplifiedReqML profile in order to gather in one hand the extensions that apply to node elements of the models and in the other hand the extension defined in the DSML for modeling relationships between those nodes.

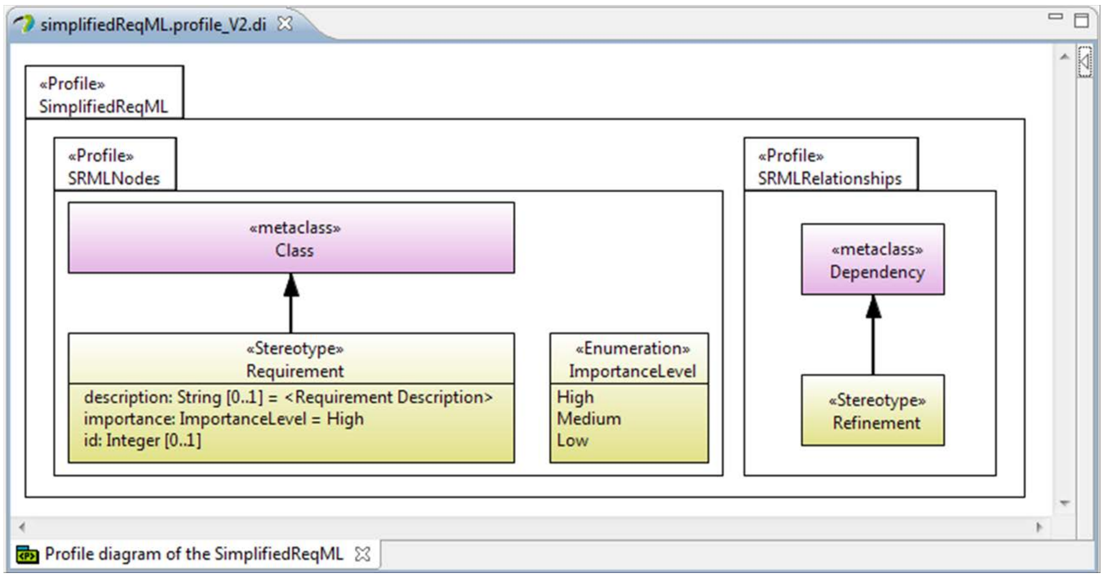


Figure 3 - Example of profile model with sub-profiles

Stereotype Generalization

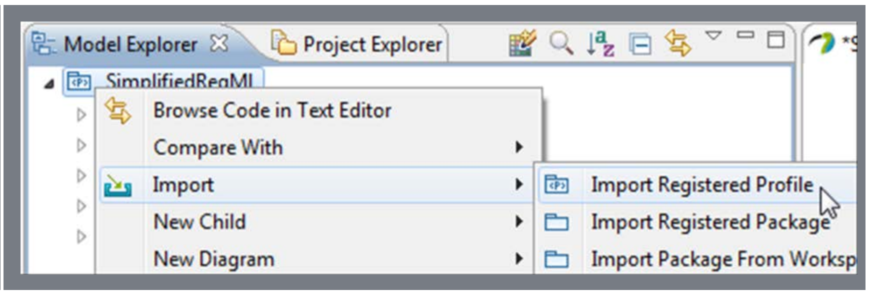
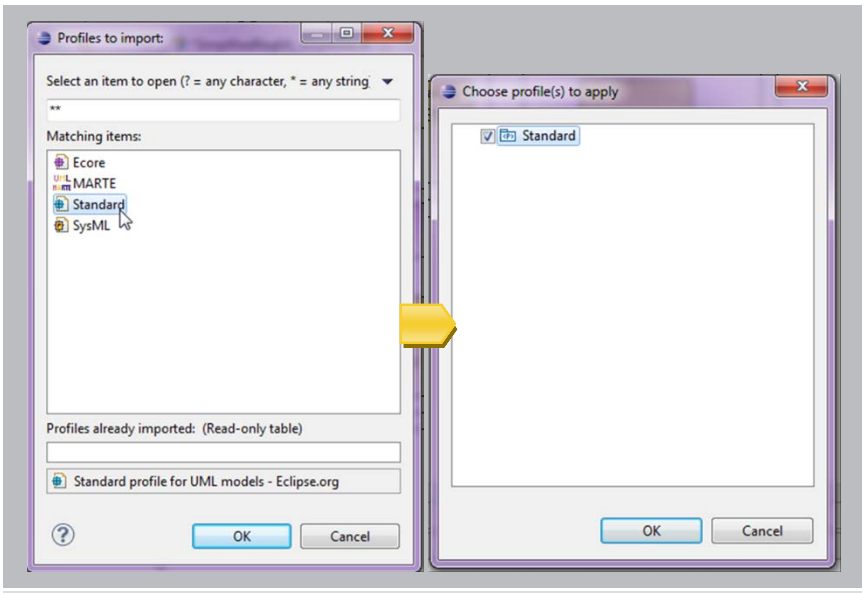
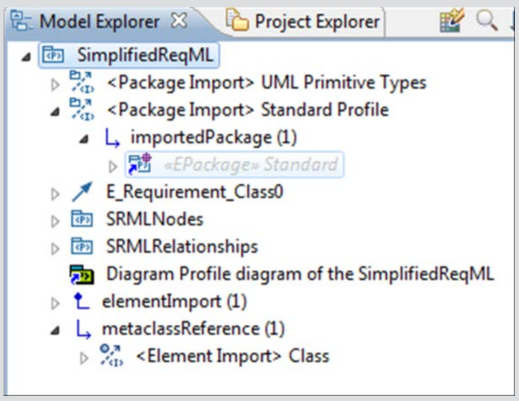
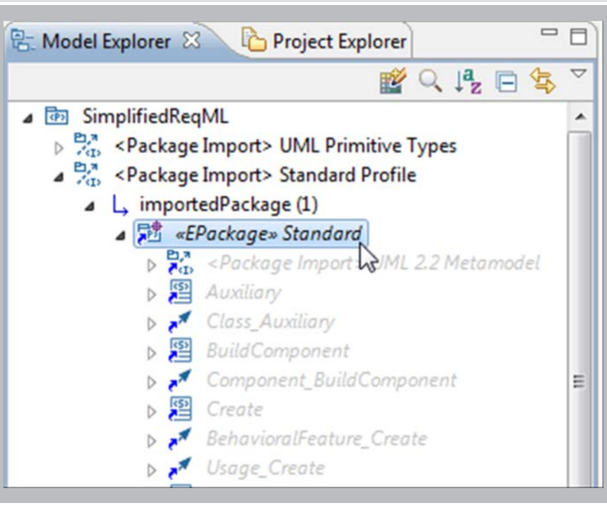
When designing a profile, it is possible to reuse existing stereotypes defined in other existing profiles. Stereotypes can indeed be generalized enabling to create child stereotypes that inherit features of one or more generalized stereotype define either locally in the profile or externally other profiles.

To create a stereotype generalization using the profile diagram editor, let's draw a generalization relationship using the tool ↗ from the palette. As any relationship within Papyrus, you need to click first on the source and then on the target of the relationship you want to model.

In our example, there exists in the UML predefined profile a stereotype extending the dependency relationship in order to introduce the concept of refinement: «Refine». Consequently, we will redesign our sub-profile SRMLRelationship in order our stereotype «Refinement» to be a specialization of this UML stereotype instead of directly extending the UML dependency meta-class.

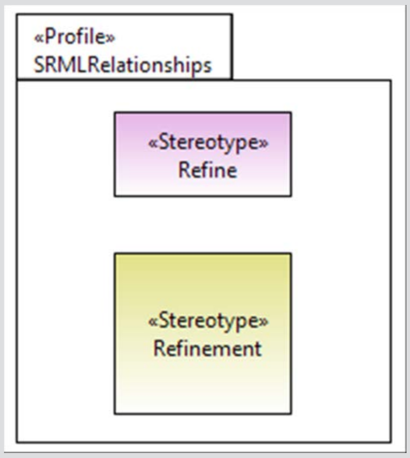
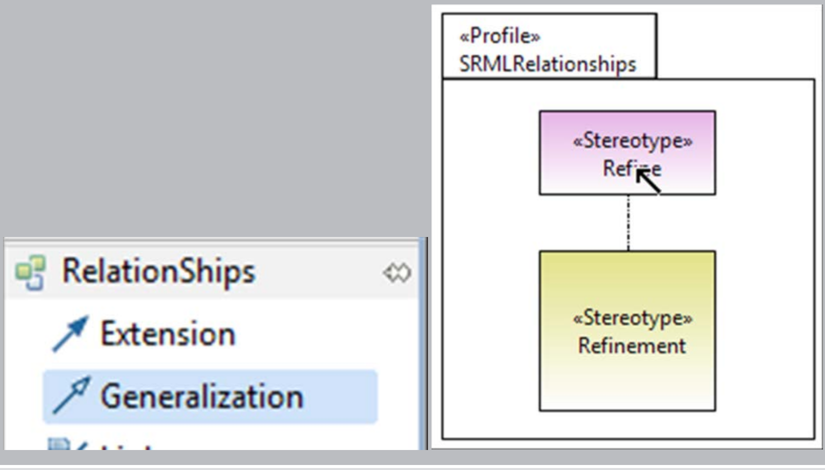
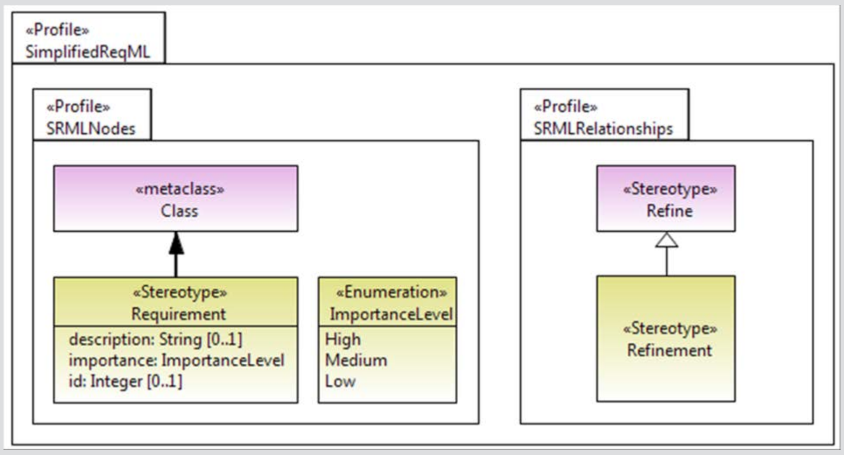
As previously mentioned, you may extend a stereotype defined in your profile or a stereotype defined in another external profile. In the latter case, the first thing to do is to import the profile where the stereotype has been defined as denoted in Table 5 from step 1 to step 4. Once the profile has been imported, you can select the stereotype you want to generalize from the imported profile and drop this latter in the diagram of profile description. Now, let's draw the generalization relationship from your stereotype (e.g., «Refinement» in figure shown at step 6) to the generalized stereotype (e.g., «Refine» in the figure shown in step 6). At this point, your diagram should look like something like the one shown in figure of step 7.

Table 5 - Steps for generalizing a stereotype

Stereotype Generalization ——— Step 1	
Stereotype Generalization ——— Step 2	
Stereotype Generalization ——— Step 3	
Stereotype Generalization ——— Step 4	

Model Explorer
Project Explorer
SimplifiedReqML
Browse Code in Text Editor
Compare With
Import
New Child
New Diagram
Import Registered Profile
Import Registered Package
Import Package From Worksp

Profiles to import:
Select an item to open (? = any character, * = any string
**
Matching items:
Ecore
MARTE
Standard
SysML
Profiles already imported: (Read-only table)
Standard profile for UML models - Eclipse.org
OK
Cancel
Choose profile(s) to apply
Standard
OK
Cancel

Stereotype Generalization ——— **Step 5**	 «Profile» SRMLRelationships «Stereotype» Refine «Stereotype» Refinement
Stereotype Generalization ——— **Step 6**	 RelationShips Extension Generalization «Profile» SRMLRelationships «Stereotype» Refine «Stereotype» Refinement
Stereotype Generalization ——— **Step 7**	 «Profile» SimplifiedReqML «Profile» SRMLNodes «metaclass» Class «Stereotype» Requirement description: String [0..1] importance: ImportanceLevel id: Integer [0..1] «Enumeration» ImportanceLevel High Medium Low «Profile» SRMLRelationships «Stereotype» Refine «Stereotype» Refinement

Stereotype display options

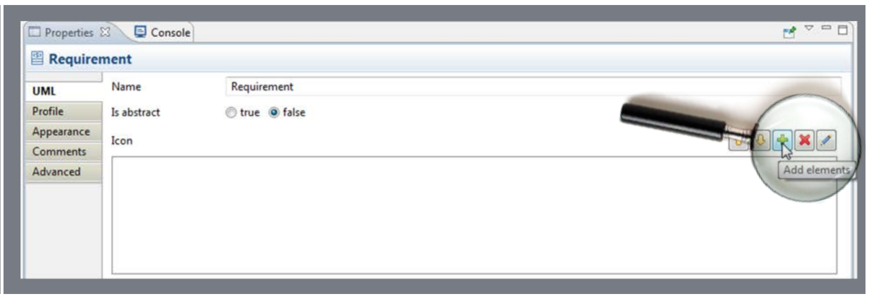
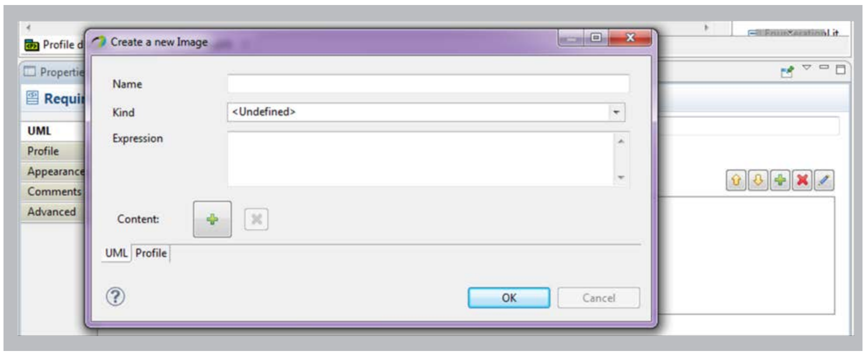
When a stereotype application is shown in a diagram, the by-default way to show it within diagrams is using a string where the name of the stereotype is shown within a pair of French guillemets above or before the name of the model element. However, it is also possible to modify the graphical appearance of the annotated model element using icons. If the graphical representation of the model element is something like a box (including ellipse of the use cases), the icons can be displayed inside and on top of the figures, or it may replace this latter. In that latter case, the property of the element cannot be shown and the name of the model element appears within a label displayed near the icon. If the model element is graphically denoted by a line, the icon is shown in front of the name of the link (let's see example in section "Displaying options of a stereotype application" in page 29).

In order to specify the icons you want to attach to a stereotype, select the stereotype and the tab UML of the property view. In the right-upper corner of the widget named "icons", press the button ✚ to add a new icon as shown in step 1 of Table 6. Once done, the dialog box shown at step 2 is open. Within this latter, let's fill in a name and select an image file using the button ✚ defining the content of the icon. Next, you have to define the kind value: icon or shape. Using this property, you can choose to associate the selected image either as an icon or as a shape.

It is possible to associate only one shape to a stereotype, but you can associate different icons to a same stereotype. In that latter case, there is a description property that is used to select which one to display. By default, if no expression is defined, the first one is the list is chosen for displaying. The expression has to be on a property of the stereotype which type is an enumeration type.

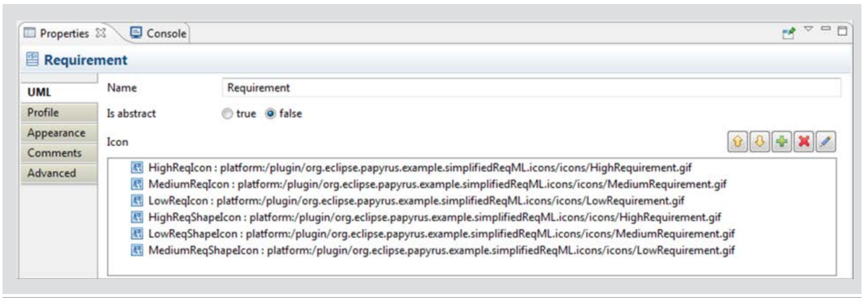
In our example, we can set different icons to the stereotype «Requirement» depending on the value of its importance property. This latter is indeed type by the ImportanceLevel enumeration which values may be High, Medium of Low (figure shown at step 7 of Table 7). For this example, we will then associate the three following images, 🟥, 🟧 and 🟩, to the stereotype «Requirement» and their related expression will be respectively importance=High (e.g. figure shown at step 6 of Table 6), importance=Medium and importance=Low.

Table 6 – Steps for attaching icons to stereotypes

Icons for Stereotypes ——— **Step 1**	 Properties Console Requirement UML Profile Appearance Comments Advanced Name Requirement Is abstract true false Icon Add elements
Icons for Stereotypes ——— **Step 2**	 Create a new Image Name Kind <Undefined> Expression Content: UML Profile OK Cancel

Icons for Stereotypes

Step 3



Requirement

Name: Requirement

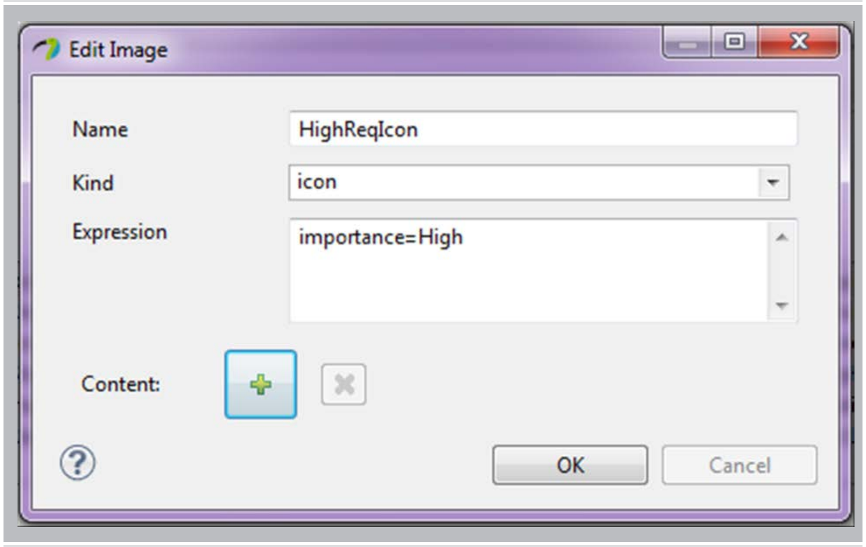
Is abstract: ( ) true (●) false

Icon:

- HighReqIcon : platform:/plugin/org.eclipse.papyrus.example.simplifiedReqML.icons/icons/HighRequirement.gif
- MediumReqIcon : platform:/plugin/org.eclipse.papyrus.example.simplifiedReqML.icons/icons/MediumRequirement.gif
- LowReqIcon : platform:/plugin/org.eclipse.papyrus.example.simplifiedReqML.icons/icons/LowRequirement.gif
- HighReqShapeIcon : platform:/plugin/org.eclipse.papyrus.example.simplifiedReqML.icons/icons/HighRequirement.gif
- LowReqShapeIcon : platform:/plugin/org.eclipse.papyrus.example.simplifiedReqML.icons/icons/MediumRequirement.gif
- MediumReqShapeIcon : platform:/plugin/org.eclipse.papyrus.example.simplifiedReqML.icons/icons/LowRequirement.gif

Icons for Stereotypes

Step 4



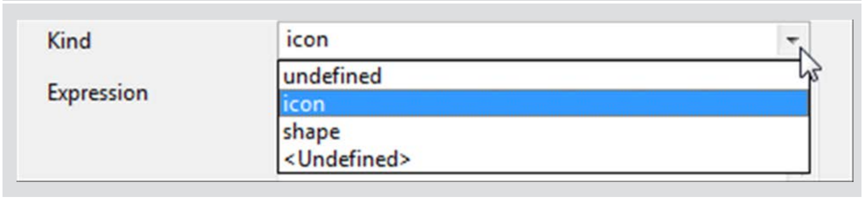
Name: HighReqIcon

Kind: icon

Expression: importance=High

Icons for Stereotypes

Step 4

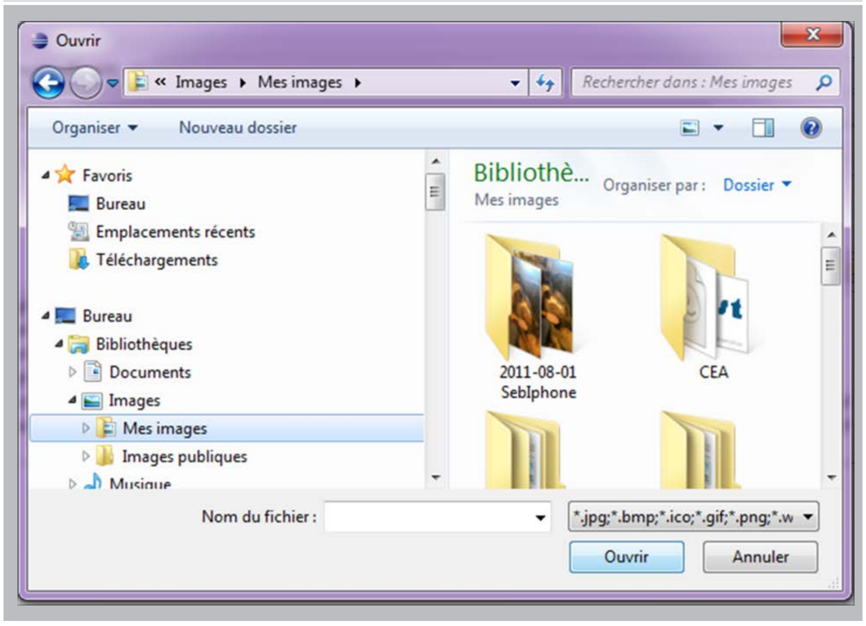


Kind: icon

Expression

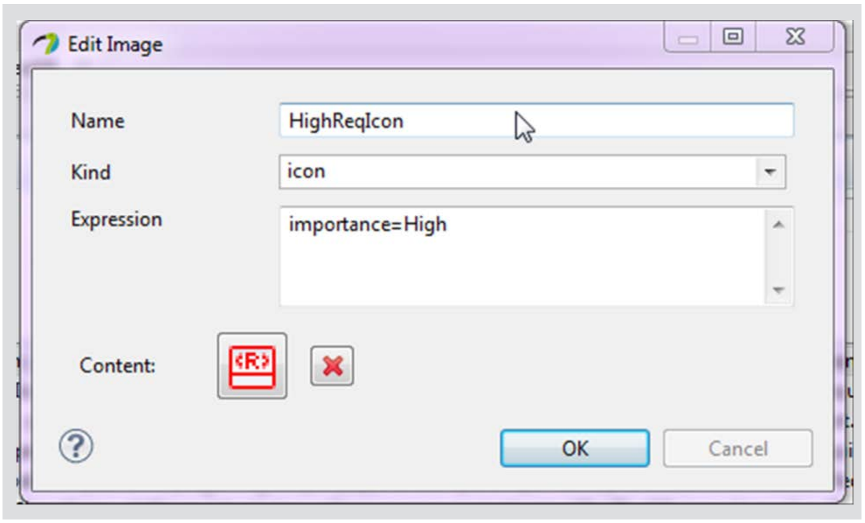
Icons for Stereotypes

Step 5



Icons for Stereotypes

Step 6



Name HighReqIcon
Kind icon
Expression importance=High
Content:

Profile Definition and Export

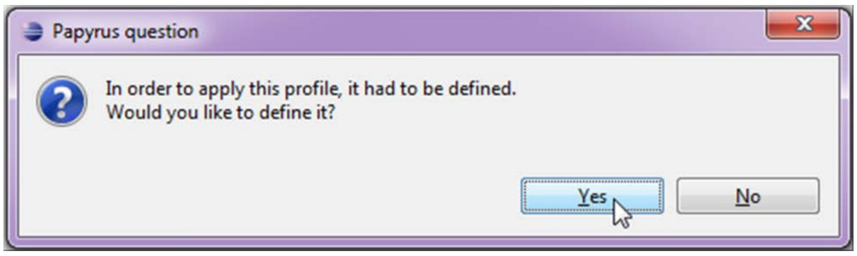
Once a profile has been modeled (stage we are now for our example as denoted in Figure 2), we need to define it before being able to apply it on user models. The definition of a profile consists in:

"When defining a dynamic profile representation, the contents of a profile are converted to an equivalent Ecore format that is stored as an annotation on the profile. Then, when a profile and its stereotypes are applied to a model and its elements, dynamic EMF (see the EMF book for details) is used to store property values for the stereotypes. For the most part, you can ignore this complexity, as long as you remember to define your profile before using it." (This definition has been extracted from http://wiki.eclipse.org/MDT/UML2/Introduction_to_UML2_Profiles)

PS: The implementation of profile support in the UML2 component of MDT supports defining both dynamic and static profile representations. In this the document, we will focus on dynamic profiles.

To define a profile within Papyrus, you just need to save it doing as for example following actions: either through the menu bar action File > Save or using the key shortcut "CTRL S".

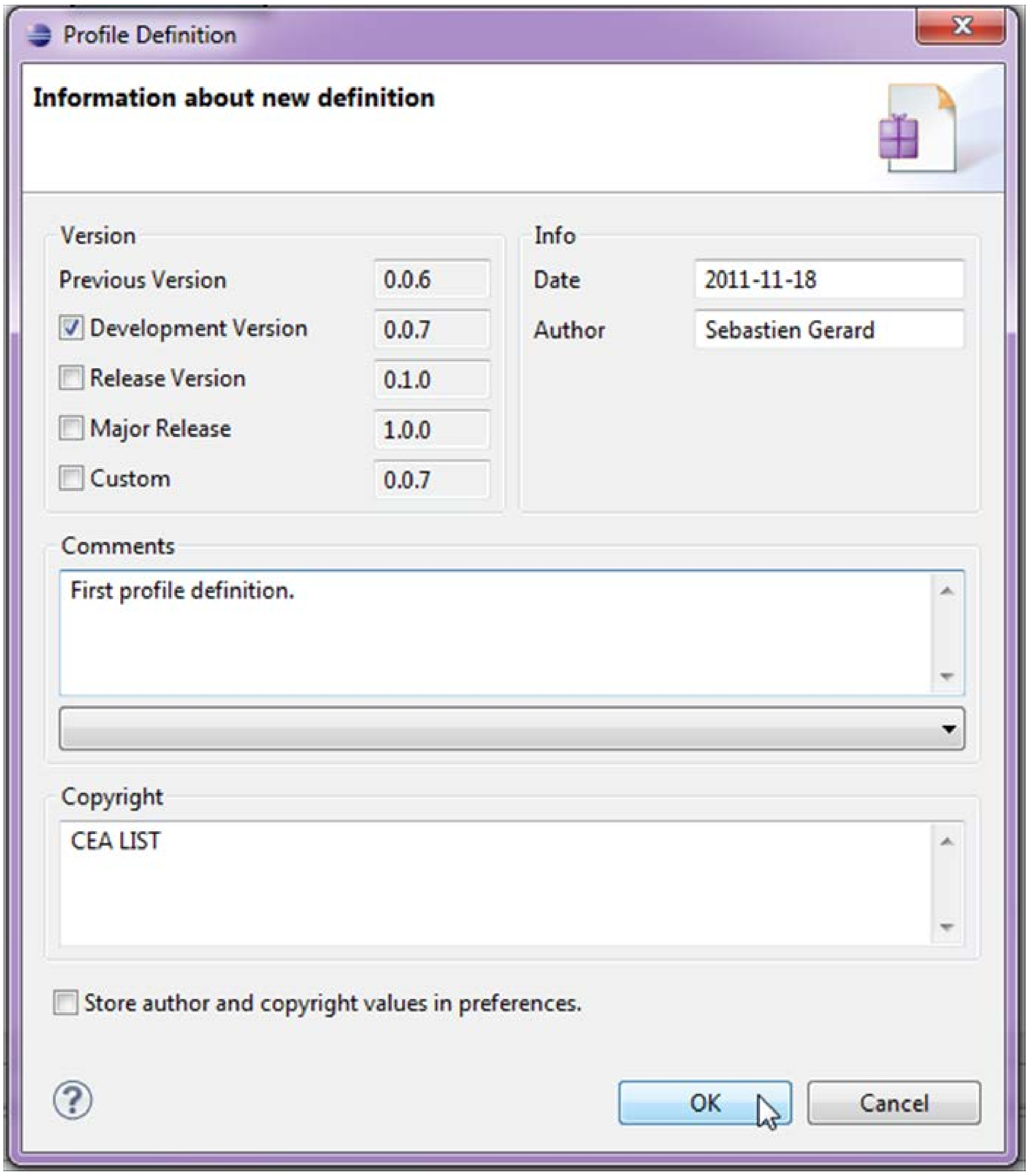
Let's notice, that it is not mandatory to define a profile each time you save it. If you do not want to define your profile when saving it, just answer no to the related question asked by Papyrus when saving profile modifications as shown in Figure 4. However, if you want to apply the modifications you have done on a given profile, this you have to define it again in order the modifications may be taken into account at the user model level.


Papyrus question

In order to apply this profile, it had to be defined.
Would you like to define it?

Figure 4. Dialog box about profile definition

Next, Papyrus will interact with you asking some additional information about your profile definition, mainly versioning information, as denoted in Figure 5. Once you have pressed the ok button your profile will be usable at user model level as denoted in the next chapter.


Profile Definition

Information about new definition

Version

Previous Version	0.0.6
[✓] Development Version	0.0.7
[ ] Release Version	0.1.0
[ ] Major Release	1.0.0
[ ] Custom	0.0.7

Info

Date	2011-11-18
Author	Sebastien Gerard

Comments

First profile definition.

Copyright

CEA LIST

[ ] Store author and copyright values in preferences.

Figure 5. Dialog box about meta-information for the profile versioning

Profile Applying and Use

Once a profile has been designed and defined as previously explained, it now times to use it. The first step to process is to apply the profile on your model or a part of the model. This is the purpose of the next section to explain this first step. Then, we will explain how to use the applied profile and mainly how to use its extensions, i.e. its stereotypes.

Applying a profile

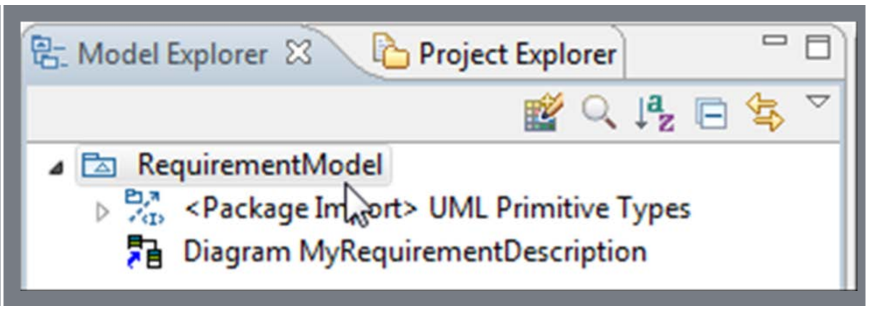
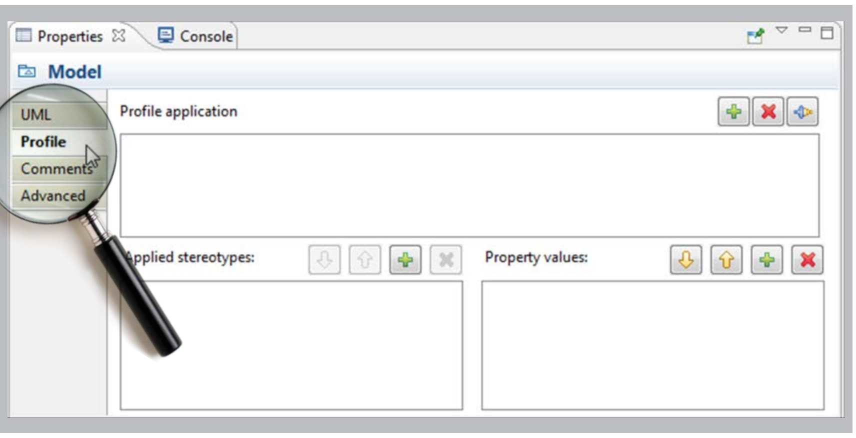
To apply a profile, first you need to open your model and then you can follow the steps denoted in the Table 6.

The first step consists in selecting the part of the model you want to apply the profile on[4]. Then to apply the profile on this part, let's select the profile tab in the property view as shown in the figure denoted in the step 2, and press the button ✚ as shown at step 3.

Then, Papyrus will ask you firstly to choose the profile to apply from your workspace (Step 4), and secondly to choose which part of the selected profile you want to apply. It is indeed possible that your profile may be composed of sub-profiles as explained in the previous chapter. It is then possible to apply partially a profile by applying one or more of its sub-profiles (Step 5).

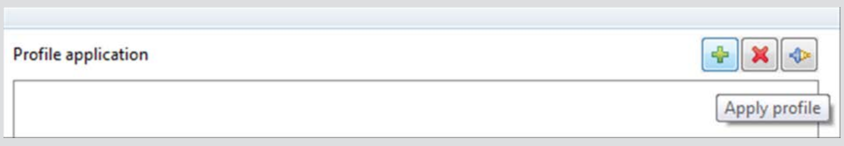
At this point, your Papyrus should looks like both figures shown at steps 6 and 6bis in the Table 6.

Table 7 – Steps for applying a profile

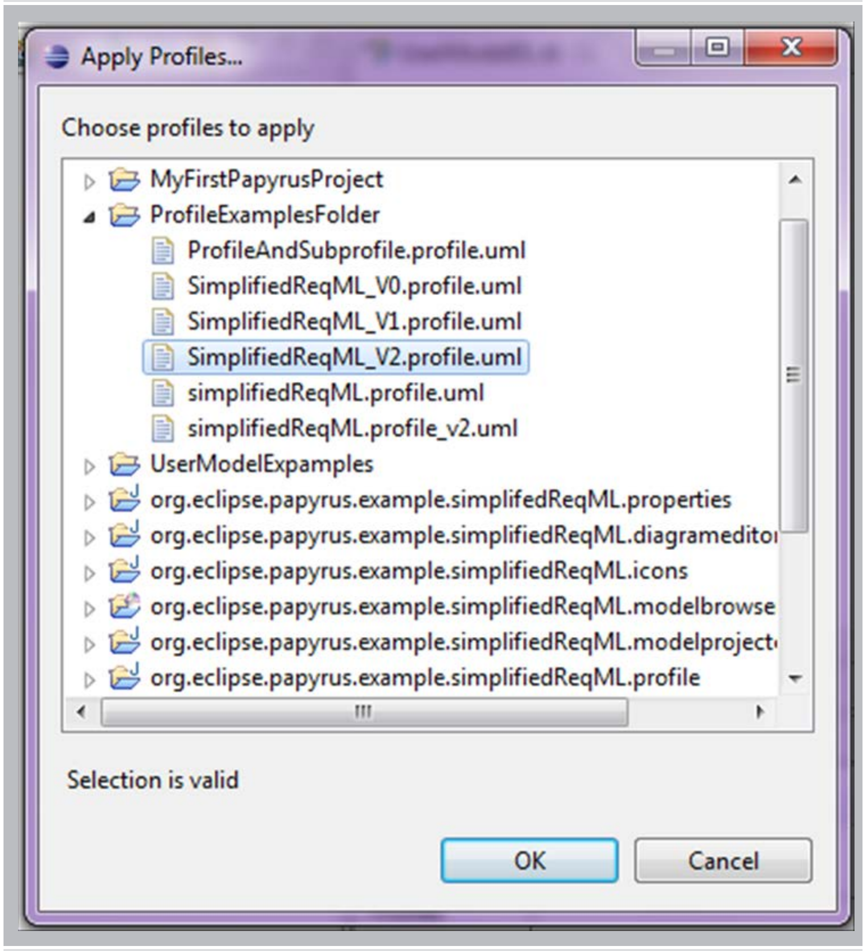
Profile Application ——— **Step 1**	 Model Explorer Project Explorer RequirementModel <Package Import> UML Primitive Types Diagram MyRequirementDescription
Profile Application ——— **Step 2**	 Properties Console Model UML Profile Comments Advanced Profile application Applied stereotypes: Property values:

[4] The parts of the model can be denoted in UML either as a package (🗀) or as a model (🗀), and UML profiles can be applied on both kinds of element.

Profile Application
Step 3


Profile application

Profile Application
Step 4

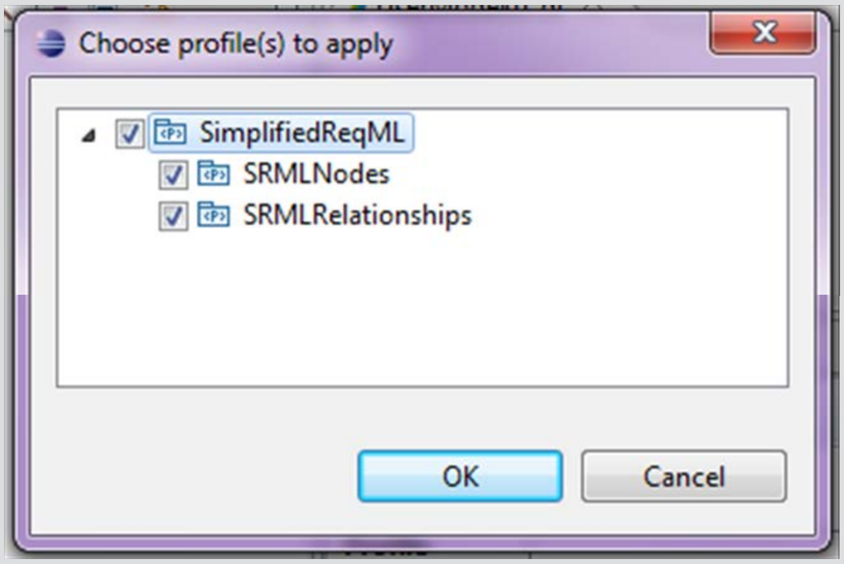


Choose profiles to apply

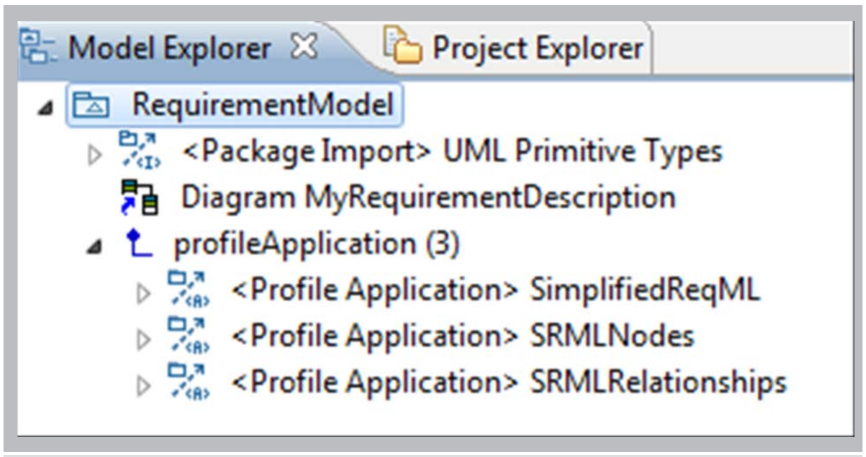
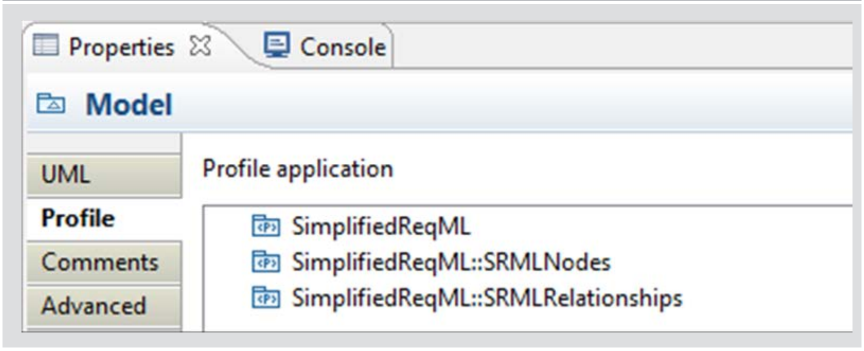
- MyFirstPapyrusProject
- ProfileExamplesFolder
  - ProfileAndSubprofile.profile.uml
  - SimplifiedReqML_V0.profile.uml
  - SimplifiedReqML_V1.profile.uml
  - SimplifiedReqML_V2.profile.uml
  - simplifiedReqML.profile.uml
  - simplifiedReqML.profile_v2.uml
- UserModelExpamples
- org.eclipse.papyrus.example.simplifedReqML.properties
- org.eclipse.papyrus.example.simplifiedReqML.diagrameditor
- org.eclipse.papyrus.example.simplifiedReqML.icons
- org.eclipse.papyrus.example.simplifiedReqML.modelbrowse
- org.eclipse.papyrus.example.simplifiedReqML.modelproject
- org.eclipse.papyrus.example.simplifiedReqML.profile

Selection is valid

Profile Application
Step 5



- ✓ SimplifiedReqML
  - ✓ SRMLNodes
  - ✓ SRMLRelationships

Profile Application ——— Step 6	 Model Explorer Project Explorer RequirementModel <Package Import> UML Primitive Types Diagram MyRequirementDescription profileApplication (3) <Profile Application> SimplifiedReqML <Profile Application> SRMLNodes <Profile Application> SRMLRelationships
Profile Application ——— Step 6 bis	 Properties Console Model UML Profile Comments Advanced Profile application SimplifiedReqML SimplifiedReqML::SRMLNodes SimplifiedReqML::SRMLRelationships

Using the stereotypes of a profile

Once the profile is applied on you model, its extensions, i.e. stereotype, are available in the modeling tool and can be used in your model to annotate it.

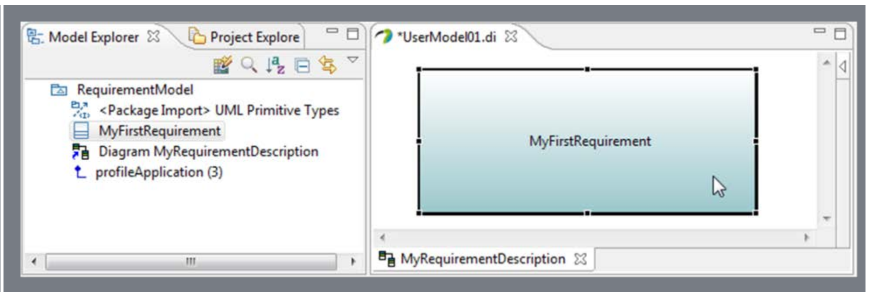
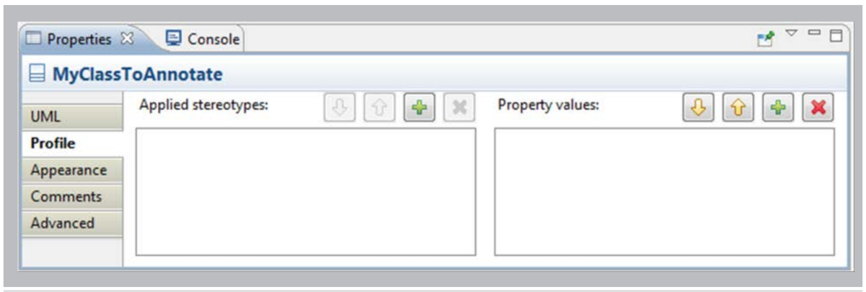
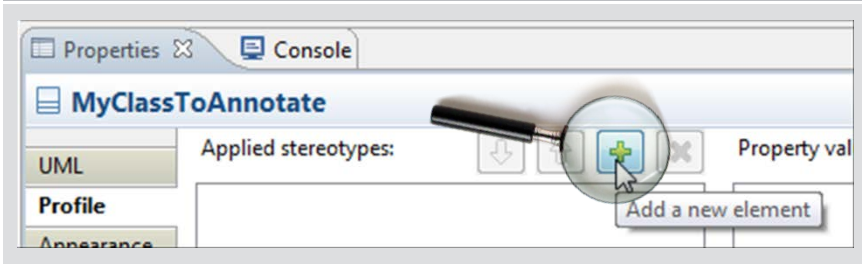
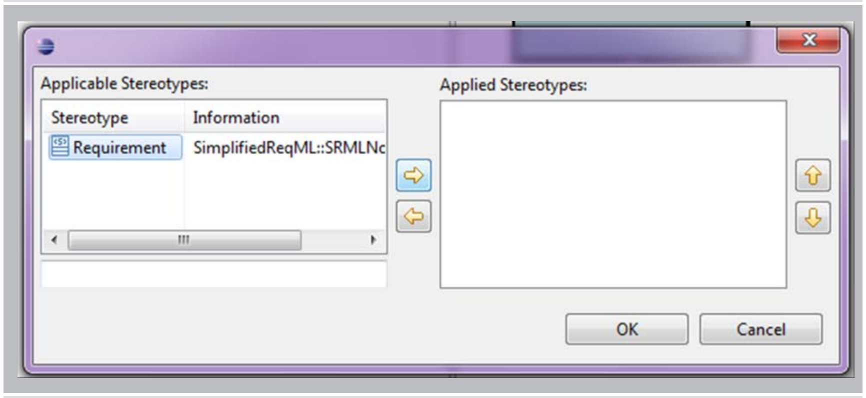
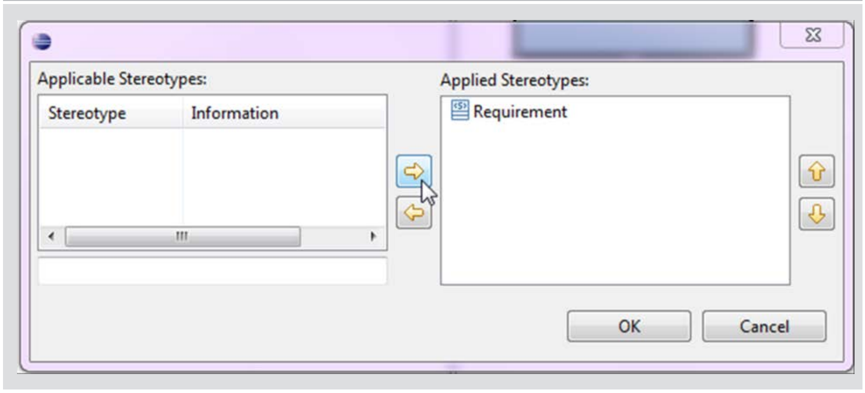
Applying a stereotype

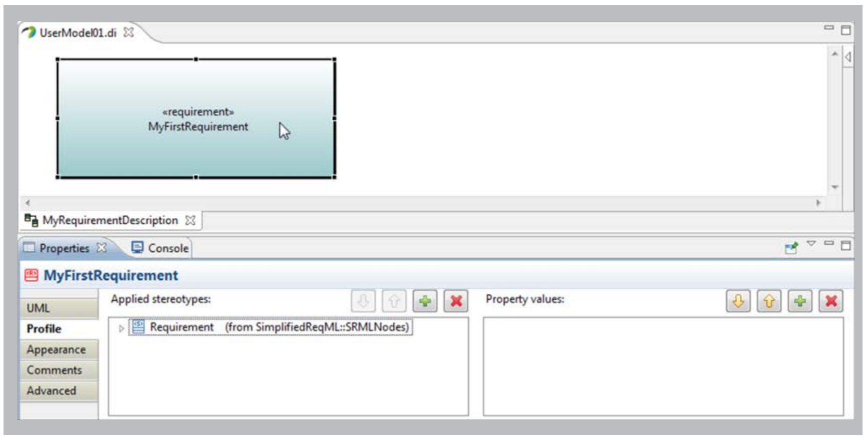
To annotate a model element, you first need to select it either through the model explorer or in one open diagram. Then, select the tab named “profile” in the property view as shown in the figure shown at step 2, and add your stereotype using the button ✚ of the widget named “Applied stereotypes”.

A dialog box enables you to select the stereotype(s) you want to apply (left part of the dialog box) and using the button ⇨ located in the middle of the dialog box enables to define which stereotypes have to be applies. The list located on the right of the dialog box denotes the list of applied stereotype. If you want to unapply stereotypes, you can select those latter from the right list and use the button ⇦ to unapply them.

At this point, your Papyrus should looks like the figure shown at steps 6 in the Table 7.

Table 8 – Steps to apply a stereotype

Stereotype Application ——— Step 1	 Model Explorer Project Explore RequirementModel <Package Import> UML Primitive Types MyFirstRequirement Diagram MyRequirementDescription profileApplication (3) *UserModel01.di MyFirstRequirement MyRequirementDescription
Stereotype Application ——— Step 2	 Properties Console MyClassToAnnotate UML Profile Appearance Comments Advanced Applied stereotypes: Property values:
Stereotype Application ——— Step 3	 Properties Console MyClassToAnnotate UML Profile Appearance Applied stereotypes: Property val Add a new element
Stereotype Application ——— Step 4	 Applicable Stereotypes: Stereotype Information Requirement SimplifiedReqML::SRMLNc Applied Stereotypes: OK Cancel
Stereotype Application ——— Step 5	 Applicable Stereotypes: Stereotype Information Applied Stereotypes: Requirement OK Cancel

Stereotype Application ——— Step 6	

MyFirstRequirement

Applied stereotypes:
Requirement (from SimplifiedReqML::SRMLNodes)

Property values:

Assigning values to properties of stereotypes

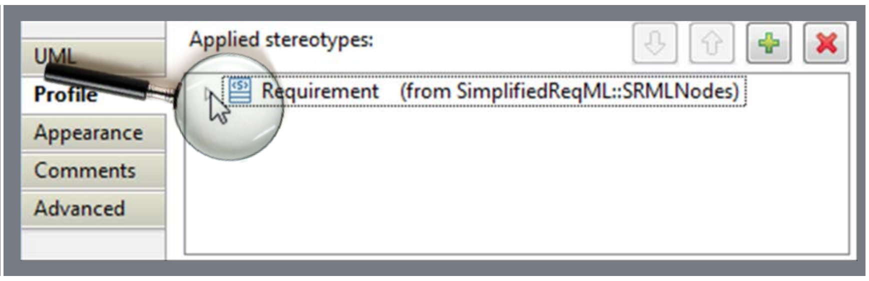
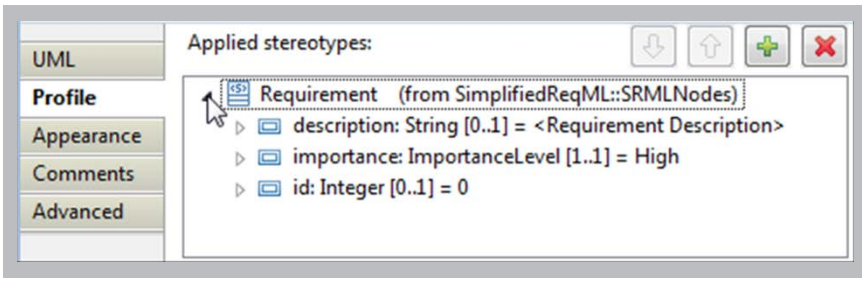
As already mentioned, Stereotypes may have properties. Consequently, when applying a stereotype to a model element, it may be necessary to set the values of those properties. For that, you will go to the profile tab of the property view and then you can unfold the stereotype application as shown in figures shown at step 1 and 2 of Table 8.

Once you have selected the property to edit as shown at setp3, its value (if already set, as for example if there is a default value defined in the profile for the property) appear in the right part of the property view. If the property has not yet been valued, you can add a value by using the button ✚ located on the top right of the widget named "Properties values". If there is a value, double-click on the value o edit this latter. A dialog box will then appear enabling you to change the value of the property as exemplified at steps 4 and 5.

At this point, your Papyrus should looks like the figure shown at steps 6 in the Table 8.

Ps: If you want to delete a value set to property, let's use the button ✖ located on the top right of the widget named "Properties values".

Table 9 – Steps for assigning values to stereotype properties

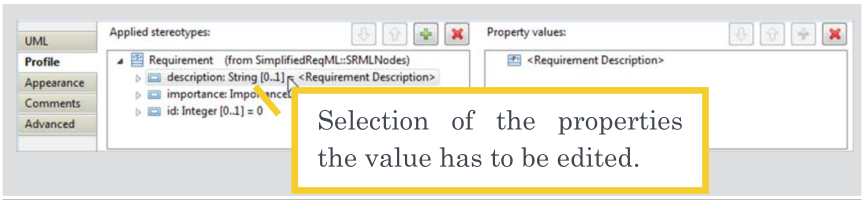
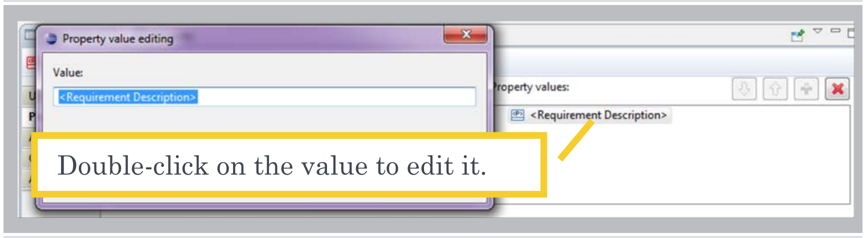
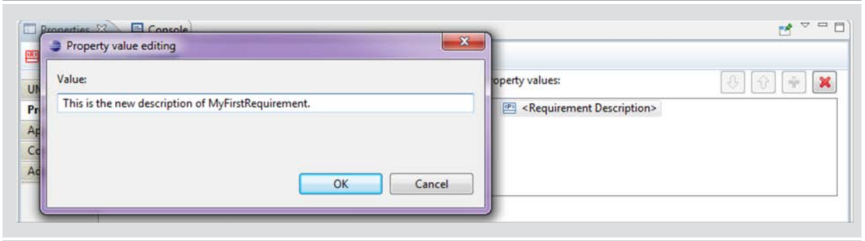
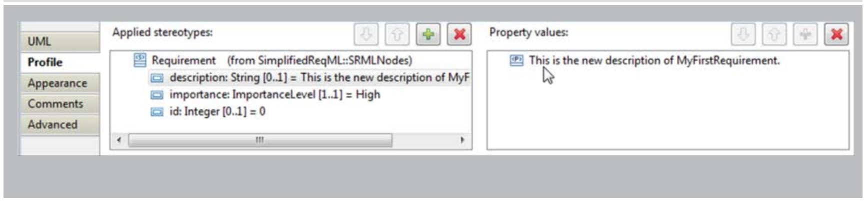
Stereotype Properties Assignment ——— Step 1	
Stereotype Properties Assignment ——— Step 2	

Step 1:

Applied stereotypes:
Requirement (from SimplifiedReqML::SRMLNodes)

Step 2:

Applied stereotypes:
Requirement (from SimplifiedReqML::SRMLNodes)
description: String [0..1] = <Requirement Description>
importance: ImportanceLevel [1..1] = High
id: Integer [0..1] = 0

Stereotype Properties Assignment ------- Step 3	 Selection of the properties the value has to be edited.
Stereotype Properties Assignment ------- Step 4	 Double-click on the value to edit it.
Stereotype Properties Assignment ------- Step 5	
Stereotype Properties Assignment ------- Step 6	

Displaying options of a stereotype application

As shown in the Figure 7, stereotype applications may be graphically rendered under different forms, either textually, or using specific icons. Details to specify those icons associated to a stereotype are given in section “Stereotype display options” in page 18.

Let’s remain that in UML, you can apply several stereotypes on a same model element. In Papyrus, it is then possible to select the one you want to show per diagram. For showing or hiding a stereotype application, you have to select the tab “appearance” in the property view. As shown in the Figure 6, there is one widget named “Applied stereotypes” that denotes the list of stereotypes applied on the current selected model element. On the example illustrated within this figure, you can see on the upper corner of the icon placed in front of the stereotype name Requirement” an overlay denoting that this stereotype application is shown.

Warning: Let’s notice that the appearance tab of the property view is only visible if you select a model element from one of the open diagram. In other case, if you select the model element from the model explorer, the concept of graphical does not make sense because the information specified within this view are only related to graphical information.

Consequently, the values set to the appearance properties of model element are valid only in the context of the diagram where the element is selected. It is then possible to show a stereotype on a diagram and hide it in another diagram depending on the concerns of the view realized by the diagram.

To hide or show stereotype applications, you have to select them from the list of applied stereotypes available within the appearance tab and then either press the button or

. The former is to be used if you want to display the stereotype application with its qualified name, and the latter is to be used if you want to show it without qualified name. Using one of both depends if you may have ambiguities or not on the origin of the stereotype when as for example applying several profiles defining similar stereotypes. As for example, both profiles, MARTE and SysML, define a stereotype named «FlowPort».

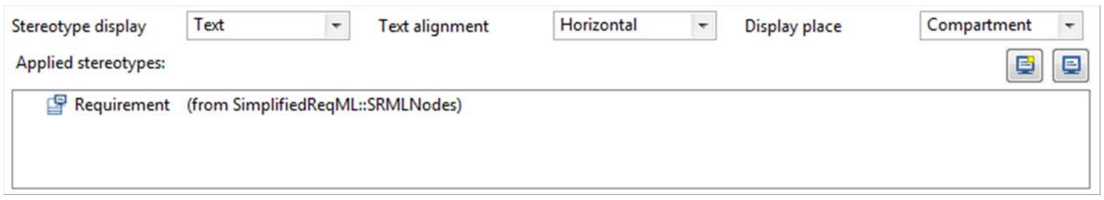

Stereotype display: Text
Text alignment: Horizontal
Display place: Compartment
Applied stereotypes:
Requirement (from SimplifiedReqML::SRMLNodes)

Figure 6. Extract of the tab appearance of the property view, stereotype display options

In addition, there are three other widgets dedicated to configure the stereotype display options (Figure 6):

- "Stereotype display" is an enumeration which values may be Text, Icon, Text and Icon or Shape. In UML, as explained previously, a stereotype may be denoted either as a string between a pair of French guillemets (e.g., «Requirement»), or as an icon embedded in the figure or as a shape with a label. In case of the shape option, it substitutes the normal graphical figure used to represent the element and its label denotes the name of the element.
- "Text alignment" is an enumeration which values may be Horizontal or Vertical. When several applied stereotypes are shown, it is rendered as a list of string separated by a comma and enclosed between a pair of French guillemets. By default, this string is shown horizontally. But some times for aesthetic reasons, it may be useful to show it vertically, that is to say showing one stereotype per line.
- "Display place" is an enumeration which values may be Compartment, Comment or With brace. This appearance property is used to set where to show the properties values of the applied stereotypes. In UML, those values can be shown either within a pair of braces located just near (above or on top) the name label of the model element ("With brace" option), or into a dedicated compartment, or in a text note associated with the annotated model element. This latter option is currently not implemented in papyrus.

PS: if you display several stereotype applications, and if you select the options to display them with icon or with icon and text, this is the icon of the first stereotype application in the list which is chosen to be displayed.

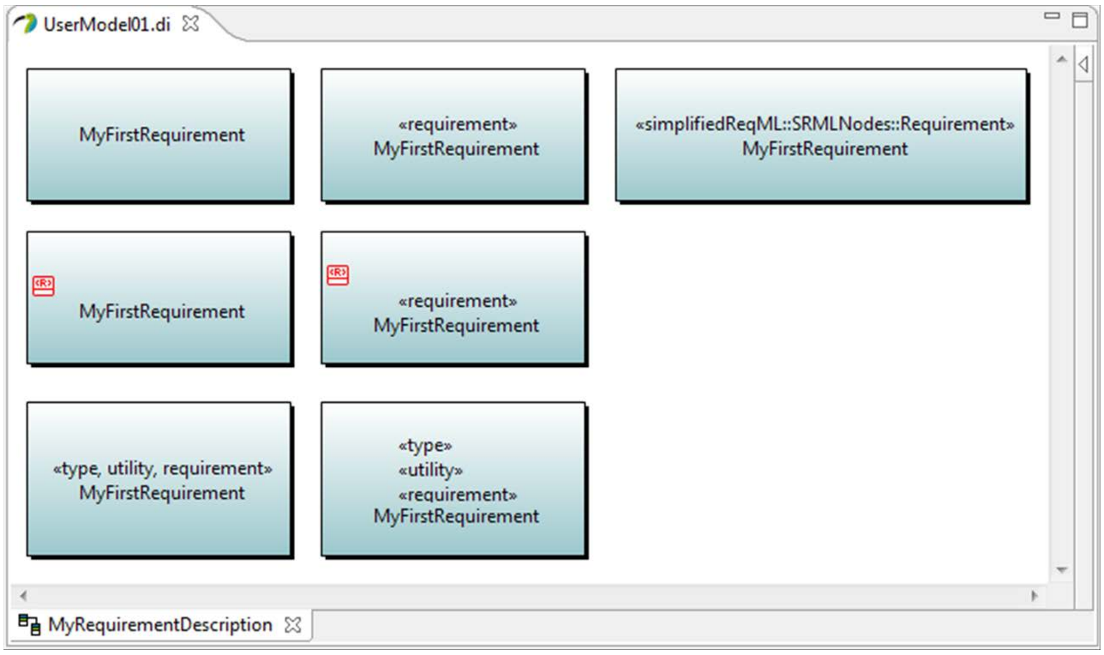


Figure 7. Examples of possible display option for stereotype applications

References

1. **OMG.** *OMG Unified Modeling LanguageTM (OMG UML), version 2.3.* s.l. : OMG, formal/2010-05-05, 2010.

2. *On the semantic foundations of standard UML 2.0.* **Selic, Bran.** 2004, SFM-RT, LNCS, pp. 181-199.

3. **OMG.** *Object Constraint Language (OCL) - Version 2.2 - formal/2010-02-01, http://www.omg.org/spec/OCL/2.2/.* 2010.