Éditeur d’argumentation de sûreté **Guide de développement**

[**Objectif de ce document**](#_mwko9u5pqs7t) **2**

[**Modification du profil**](#_7zmxxiqs3tfd) **3**

[**Elementtype sémantique**](#_q6ixs1tiylfc) **6**

[**Elementtype graphique**](#_7vybwfxtwkwp) **8**

[**Palette**](#_2hoe8m2np1po) **9**

[**New Child menu**](#_mxchtqfg0jbf) **11**

[**CSS et customization**](#_q3s8p1nu72f7) **12**

[**Génération des propriété**](#_nvc5df3qnyxu) **12**

# Objectif de ce document

Ce document décrit l’ensemble des procédures à suivre pour modifier ou ajouter un élément au profil et l’intégrer correctement au plug-in.

Du fait de l’existence de dépendances entre les différentes parties du plug-in, il est préférable de réaliser les différentes étapes dans l’ordre.

# Modification du profil

Certaines modifications ou ajouts peuvent nécessiter une modification du profil servant de base au plug-in.

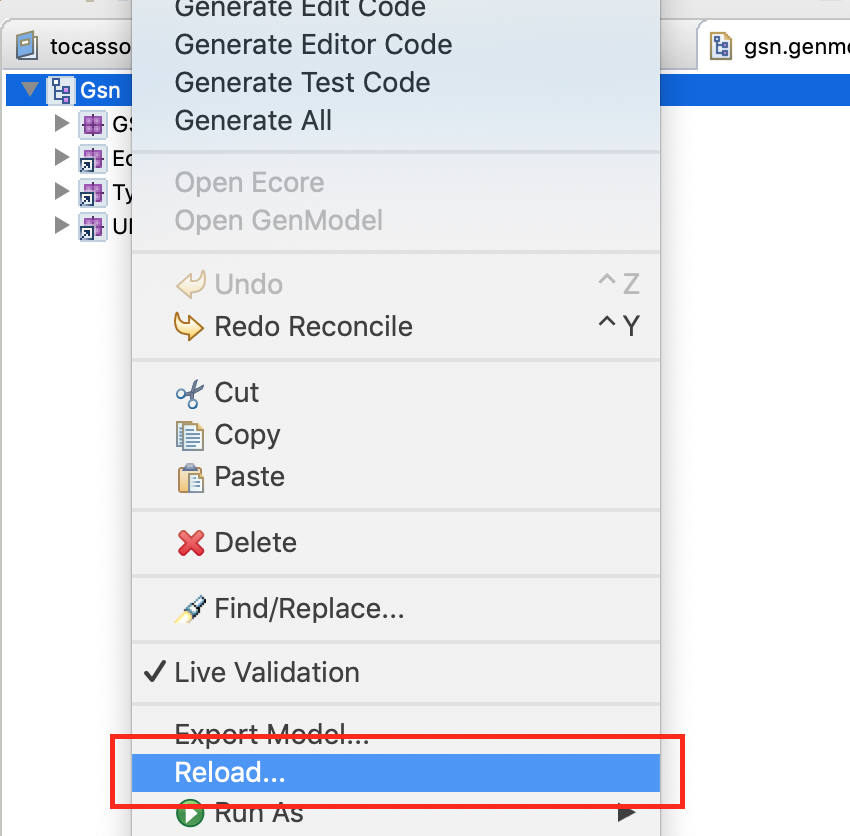
Le profil en lui-même se situe dans le fichier *model/gsn.profile* du plug-in *org.eclipse.papyrus.gsn.profile* et peut être édité graphiquement dans la vue *Papyrus Editor Core*.

Génération automatique du code du modèle

Après avoir modifié le profil, il est indispensable de re-générer le code source du modèle.

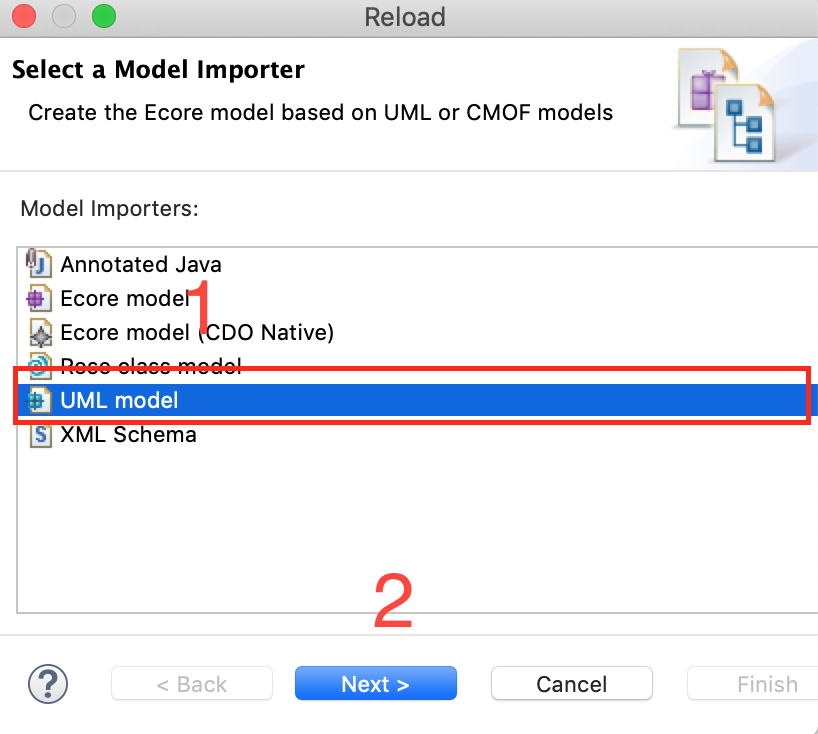
La procédure de génération du code est décrite ci-dessous:

1. Dans le fichier, *model/gsn.genmodel* , faire un clique-droit sur la racine du *genmodel* et sélectionner l’option *Reload.*

**

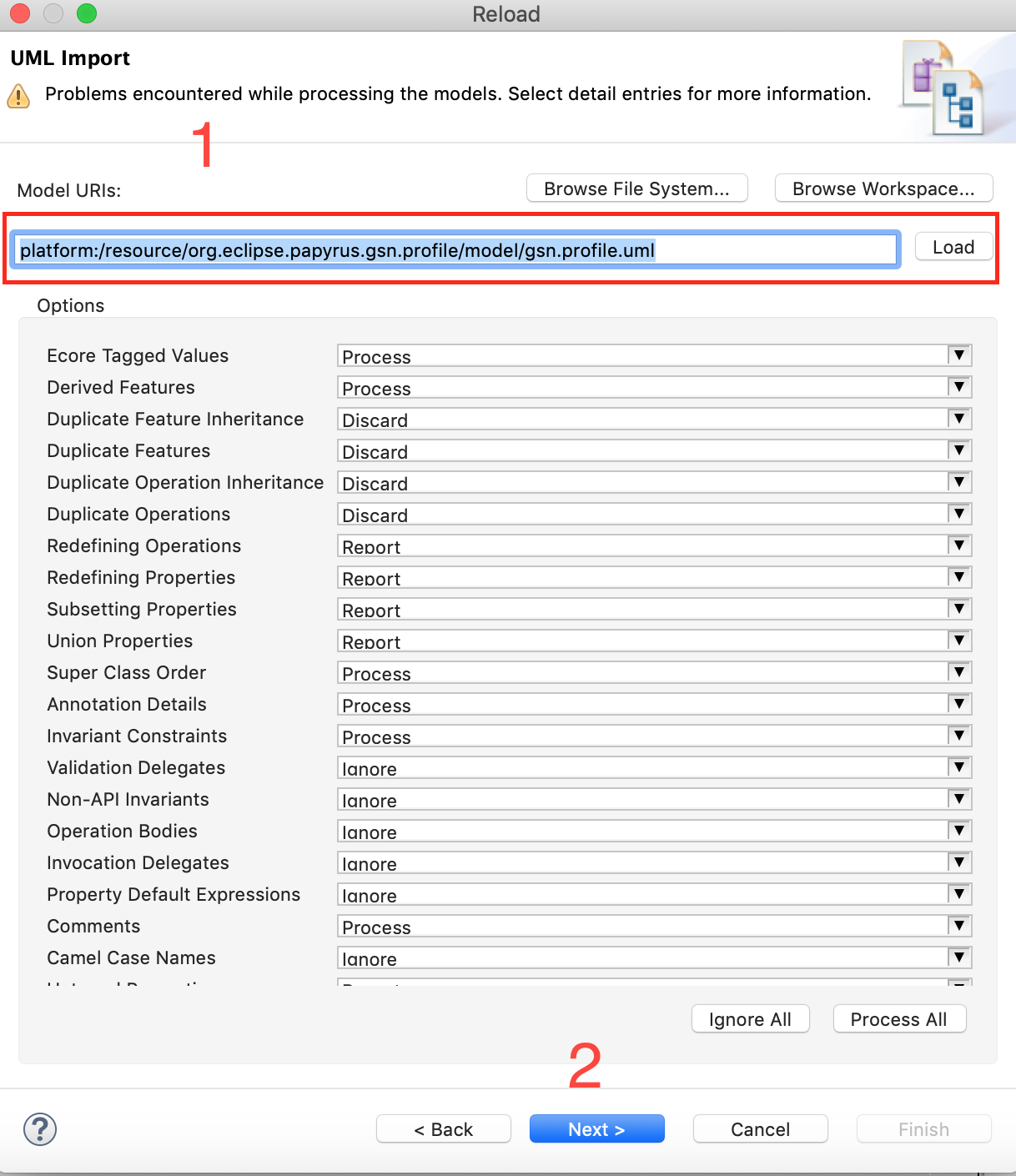
**Figure 1 : Reload du *Genmodel***

2. Dans la fenêtre de dialogue qui apparaît, sélectionner l’option *UML Model.*

**

**Figure 2 : Choix d’UML Model**

3. Dans la boîte de dialogue suivante, sélectionner le fichier UML correspondant au profil, et cliquer sur next sans modifier les options par défaut.



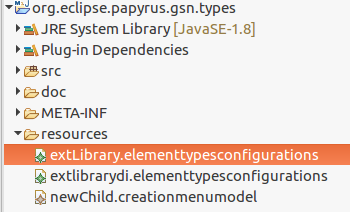
**Figure 3 : Importer le Modèle UML**

4. Dans la boîte de dialogue suivante, cliquer sur *Finish* sans modifier les options par défaut.

# Elementtype sémantique

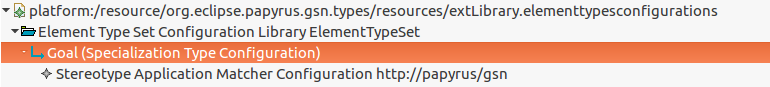
Ajout d’un nouvel *ElementType*

Une foisle profil modifié, il faut mettre en place sa partie sémantique dans le fichier *ressouces/extLibrary.elementtypesconfigurations* du plugin *org.eclipse.papyrus.gsn.types.*

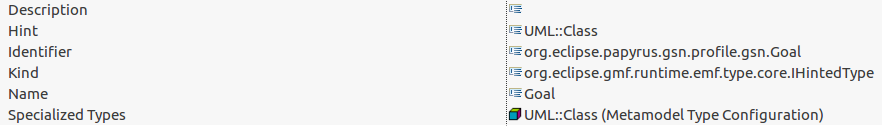


**Figure 4 : Le plugin “types”**

Afin de faire le binding entre le stéréotype et l’*Elementtype* , il faut créer une *Specialization Type Configuration* en indiquant les informations.



**Figure 5 : Création d’un ElementType**



**Figure 6 : Les propriétés d’un ElementType**

Il faut également créer un *Stereotype Matcher*



**Figure 7 : Les propriété d’un ElementType**

et un *Apply* *Stereotype Advice* *Configuration* pour appliquer le stéréotype au spécialisation type

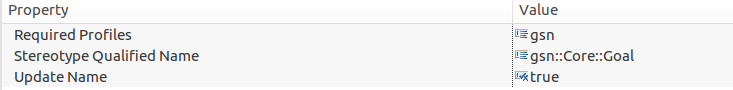


**Figure 8 : Appliquer le stéréotype à l’ElementType**

en indiquant le chemin du *specialization type*



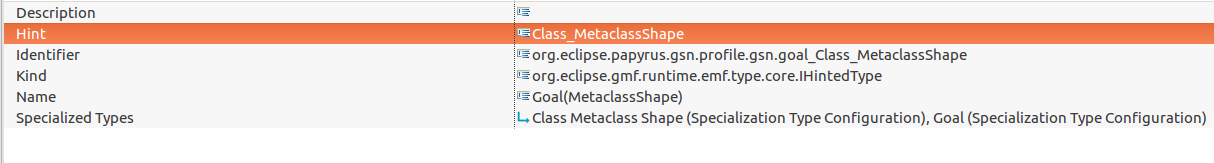
**Figure 9 : Appliquer la spécialization type configuration**



**Figure 10 : Le stéréotype à appliqué**

# Elementtype graphique

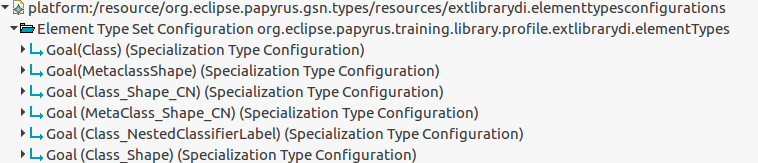
Pour chaque type de stéréotype étendant une *méta-classe* spécifique, il faut définir un ensemble de *Specialization Type*. Si cette étape n’est pas réalisée, l’élément n’apparaîtra pas dans le diagramme. pour chaque Elementtype , nous devons avoir l’elementtype sémantique et la spécialization type configuration qui correspond au type *(voir exemple)*



**Figure 11 : Elementtype graphique (Metaclass\_Shape)**

Pour le type *UML::Class :*

* *Class*
* *MetaclassShape*
* *MetaClass\_Shape\_CN*
* *Class\_Shape*
* *Class\_Shape\_CN*
* *Class\_NestedClassifierLabel*



**Figure 12 : Liste des ElementTypes à définir**

Pour le type *UML::Association*

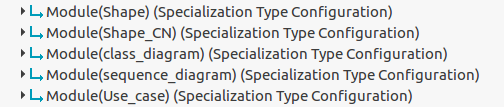
* *Edge*



**Figure 13 : Elementtype graphiques pour les associations**

Pour le type : *UML::Package*

* *Class\_Shape*
* *Class\_Shape\_CN*
* *class\_diagram*
* *Use\_case*
* package sequence diagram



**Figure 14 : Elementtype graphiques pour les packages**

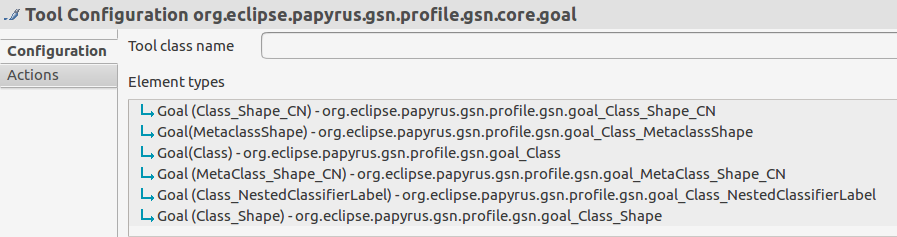
Dans le cas des associations, il faut définir des *Stereotype Advice Configuration* afin que les stéréotypes soient bien appliqués au niveau du diagramme.



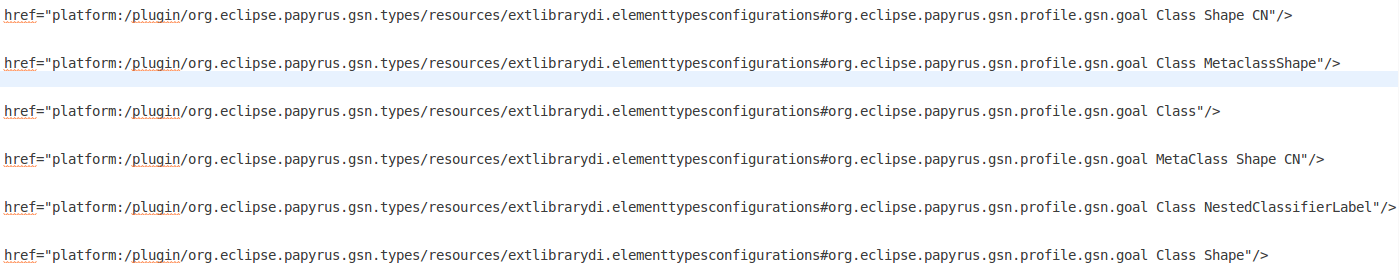
**Figure 15 : Elementtype graphiques pour les packages**

# Palette

Au niveau de la palette , il faut créer un *New* *Tool* et remplir les informations. Afin de pointer sur les *Elementtypes* graphiques , il faut modifier directement le fichier XML en ajoutant des références aux identifiants des *Elementtypes* graphiques.



**Figure 16 : Références aux elementTypes graphiques (1)**

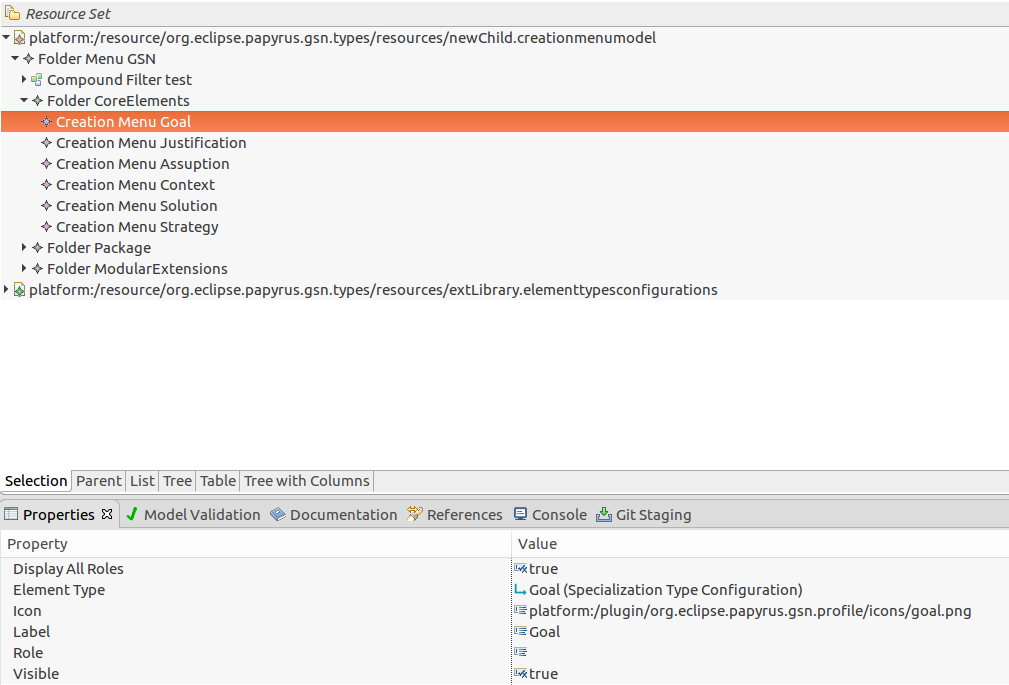


**Figure 17 : Références aux elementTypes graphiques (2)**

# New Child menu

Pour le menu *New Child* , il suffit de créer un *Creation Menu* référençant l’Elementtype sémantique pour que le nouvel élément apparaisse dans le menu contextuel.

⚠️: Si l’icône du nouvel élément n’est pas renseignée, ce dernier n’apparaîtra pas dans le menu.



**Figure 18 : Création d’un nouvel élément**

# CSS et customization

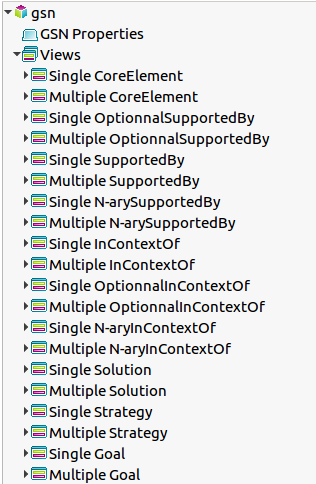
Il faut créer une image SVG et l’appliquer le stéréotype grâce au CSS .



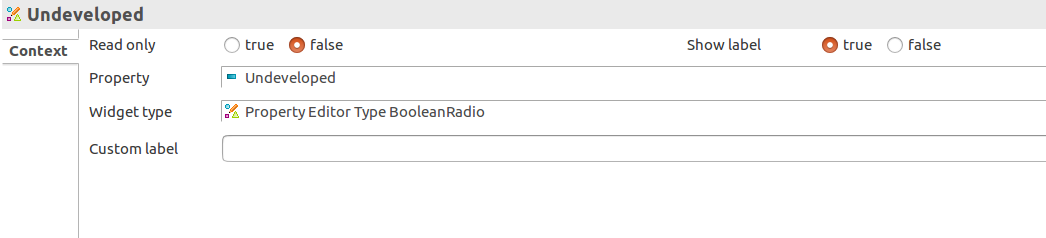
**Figure 19 : Appliquer une image SVG au stéréotype “Goal”**

# Génération des propriété

Afin d’ajouter un nouveau stéréotype , il faut régénérer les propriétés à partir du modèle et modifier par la suite le fichier .ctx généré . Il faut bien vérifier le widget type qui doit correspondre au type déjà défini dans



**Figure 20 : Génération des propriétés pour les éléments du langage**



**Figure 21 : Modification du widget type**