

1. Utilisation de TopCased_UML

Topcased UML est un **outil UML open source** qui fonctionne sous forme de *plugin pour l'environnement de développement ECLIPSE* (très utilisé pour le développement d'applications en JAVA). Le tout étant basé sur JAVA, **Topcased_UML** est "**multi-plateforme**" et fonctionne entre autres sur **Windows** , **Unix/Linux** , Mac .

L'ergonomie de Topcased UML est "correcte/bonne" . Il y a mieux et moins bien dans des produits UML concurrents. Après environ une journée de "prise en main", on peut commencer à utiliser efficacement le produit ("*clic,clic*" et "*hop,hop*" plutôt que "*clic,undo*").

Soutenu entre autres par Airbus et Atos-Origin le projet Topcased UML est assez dynamique. Une nouvelle version apparaît régulièrement pour suivre l'évolution du standard UML et de la plateforme ECLIPSE.

L'un des principaux intérêts de l'outil Topcased UML tient dans le fait qu'en tant que plugin bien intégré à eclipse , on peut l'utiliser conjointement avec d'autres plugins importants (ex: accéléo_M2T , gendoc2, ...).

Ainsi à partir d'un même outil ECLIPSE, on peut:

- **créer/paramétrer des modèles UML** (diagrammes avec stéréotypes)
- **(re-)générer automatiquement de la documentation au format ".doc" ou ".odt"** (avec le plugin intégré gendoc2) pour produire des spécifications
- **(re-)générer automatiquement une bonne partie du code de l'application** (avec le plugin "accéléo_M2T" / MDA).

Ceci permet de travailler efficacement avec de bon atouts pour obtenir des éléments produits (spécifications , code , tests) bien **cohérents** entre eux.

NB: Pour bien utiliser cette plateforme de développement (et ses différents plugins) , il faut savoir effectuer les **bons paramétrages** à chaque niveau:

- bien paramétrer les modèles UML (avec des stéréotypes adéquats)
- bien paramétrer la génération de documentation (avec des "templates" de "docs")
- bien paramétrer la génération de code (avec des "templates" de code)

Tout ceci correspond au final à **un investissement en "temps de mise au point"** qui peut se rentabiliser sur des projets importants.

URL pour télécharger l'outil "Topcased UML":

<http://www.topcased.org/>



1.1. Intégration/installation de Topcased UML

Préalable : une machine virtuelle JAVA (idéalement JDK 1.6) doit être installée sur le poste de développement.

Il y a au final deux grands modes d'intégration de "Topcased UML":

- intégration pré-établie (prête à être téléchargée)
---> **Topcased RCP** (*Rich Client Platform*)
- intégration spécifique (à construire soit même en partant d'ECLIPSE et en y ajoutant un à un tous les plugins jugés utiles : Topcased_UML , accéléo_M2T , Maven , ...)

NB:

- Avec la version "pré-intégrée" dite "RCP" , on a déjà de quoi construire les modèles UML et générer la documentation associée (spécifications).
- Pour en plus pouvoir générer du code utile et le tester , il faut tout un tas de plugins supplémentaires (accéléo_M2T, WTP/JEE , Maven , SVN, ...) qui peuvent soit être placés dans un même et seul "gros IDE / super Eclipse customisé" ou bien être placés dans un second ECLIPSE . Si plusieurs "IDE ECLIPSE" sont utilisés, il suffit de recopier les fichiers ".uml" (et facultativement ".umldi") d'un eclipse à l'autre pour récupérer une copie des modèles UML à partir desquels du code doit être généré en mode MDA.

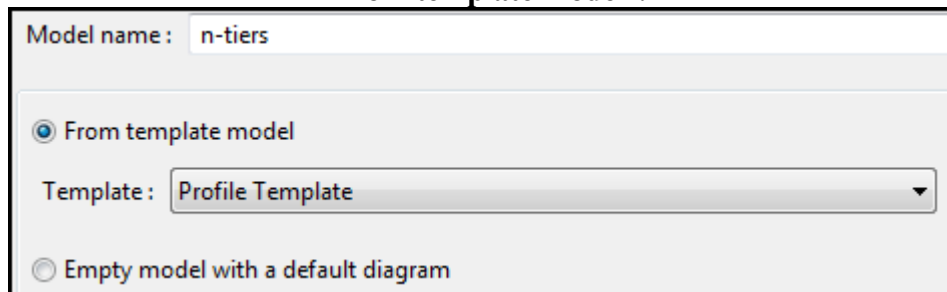
Dans tous les cas, le lancement de Topcased UML s'effectue en démarrant "**eclipse.exe**" (de la version RCP téléchargée ou bien spécifiquement construite).

1.2. Création d'un profil UML (avec stéréotypes)

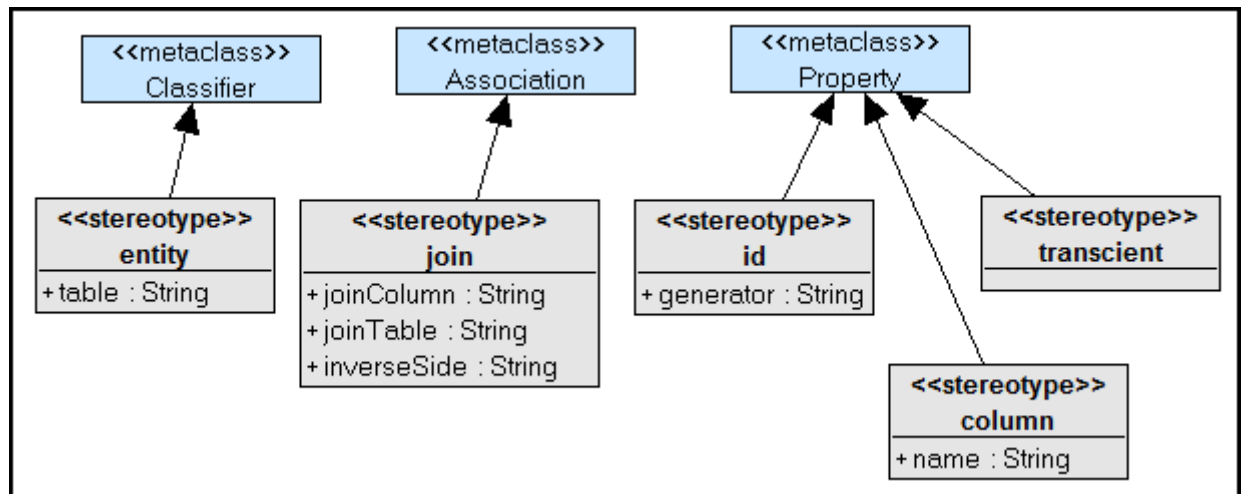
Pour placer des stéréotypes (ex: <<entity>> , <<id>>) dans un modèle UML , il faut d'abord les créer dans un fichier de type "**UML profile**".

Mode opératoire:

- Se placer dans la partie "**Models**" d'un projet eclipse de type "**Topcased project**"
- Créer un nouveau fichier via le menu "**New / UML model with Topcased**" en choisissant bien le nom du profil (ex: "n-tiers" ou "orm") et en sélectionnant "**Profile Template**" de "**From template model**":



- Créer de nouveaux **stéréotypes** (depuis la palette et en renseignant leurs **noms**).
- Placer et paramétrer des "**metaclass**" (ex: "Classifier" , "Property" , ...) .
- Relier les "**stéréotypes**" aux "**metaclass**" via des flèches d'extension pour indiquer "**sur quoi les stéréotypes seront applicables**".
- Bien sauvegarder le fichier généré.



NB: un stéréotype peut éventuellement comporter des propriétés.

Quelques idées de stéréotypes:

<<entity>>	Entité persistante
<<id>>	Identifiant (proche de "clef primaire")
<<service>>	Service métier
<<requestScope>> , <<sessionScope>> , <<applicationScope>>	Scope / portée des objets de la partie "IHM_WEB"
<<stateful>> , <<stateless>>	Avec ou sans état (traitements ré-entrants et partagés ?)
<<facade>> , <<dao>> , <<dto>> , ...	Design pattern / pour la conception
<<in>> , <<out>> , <<inout>> , <<select>>	Pour paramétrer les fonctionnalités souhaitées coté IHM (entrée/saisie , sortie/affichage , sélection , ...)
<<transactional>> , <<CRUD>>	Fonctionnalités diverses attendues (transactionnel , ...)
<<module_web>> , <<module_services>> , <<database>>	Types de composants
...	
...	

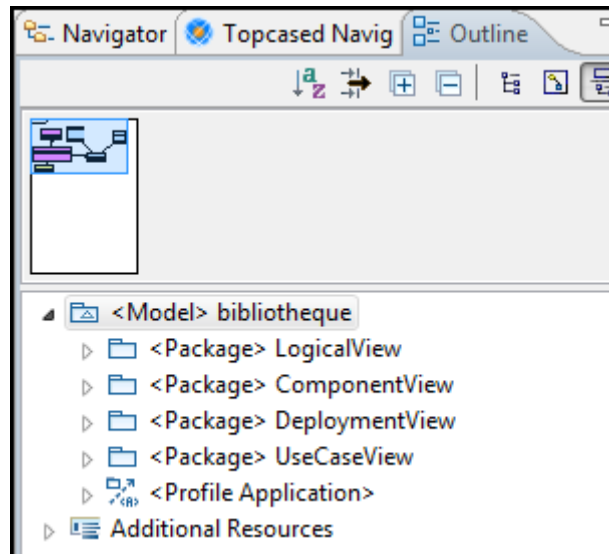
1.3. Création et initialisation d'un modèle UML (pour java)

Mode opératoire:

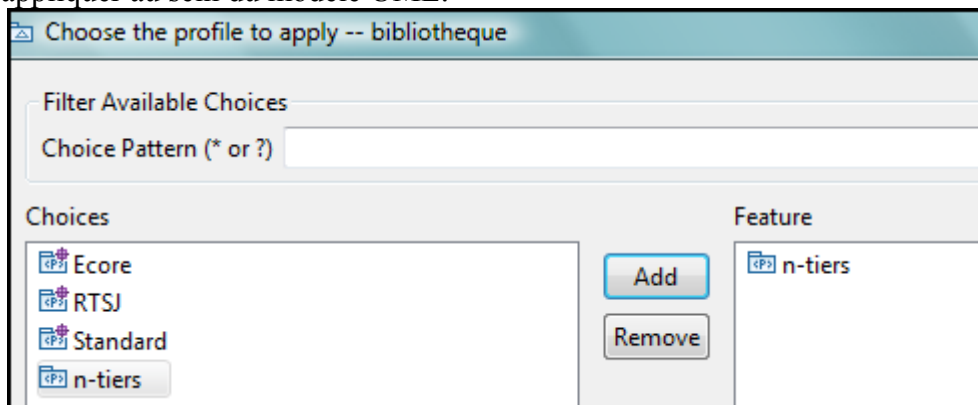
- Se placer dans la partie "**Models**" d'un projet eclipse de type "**Topcased project**"
- Créer un nouveau fichier via le menu "**New / UML model with Topcased**" en choisissant bien le nom du modèle (ex: "bibliotheque" ou "devises") et en sélectionnant "**Common Approach**" de "**From template model**":



- Se placer à la racine du modèle UML dans la vue "**outline**" :



- Activer le menu contextuel "**Load resources...**" et sélectionner/intégrer le (ou les) **fichier(s) de profile(s) UML** (ex: n-tiers.uml). Ceci permet d'enregistrer un chemin relatif entre le fichier du modèle UML et le fichier "profil" d'extension.
- Activer le menu contextuel "**Apply Profile...**" et sélectionner le (ou les) profile(s) à utiliser/appliquer au sein du modèle UML:



NB: "**Apply Profile**" sera éventuellement à re-déclencher ultérieurement s'il faut tenir compte d'une nouvelle version enrichie d'un fichier "UML profile".

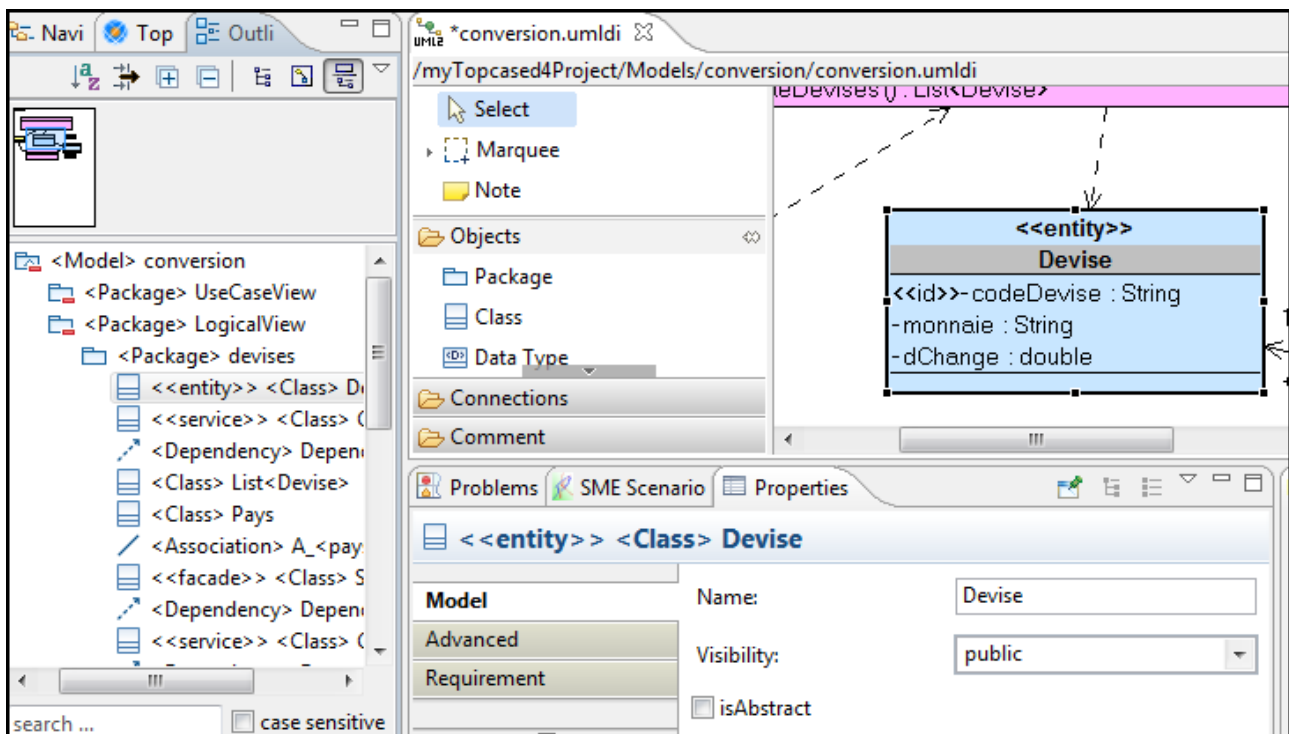
- Activer le menu contextuel "**Generate Primitive Type / Java**" pour générer si nécessaire les types de données élémentaires du langage Java (ex: double , int , ...).
- Bien (re-)sauvegarder le fichier du modèle UML.

1.4. Utilisation générale de l'outil TopCased UML

Chaque modèle UML est sauvegardé dans deux fichiers complémentaires:

- le fichier **".uml"** comporte tous les éléments significatifs du modèle UML (packages , classes ,). C'est à partir de ce fichier que l'on peut extraire les informations essentielles du modèle UML pour générer du code (via un outil MDA tel qu'accéléo_m2t par exemple)
- le fichier **".umldi"** comporte les coordonnées (x,y,...) des éléments internes des diagrammes UML.

==> pour éditer graphiquement un modèle UML, il faut ouvrir (via un double-clic) le fichier ".umldi". Le fichier ".uml" de même nom sera alors automatiquement pris en charge et mis à jour.



De façon intuitive, la fenêtre "Outline" permet de naviguer dans l'arborescence du modèle UML, la fenêtre "Properties" permet de fixer les propriétés de l'élément sélectionné, une palette permet de choisir le type d'élément à ajouter au modèle.

Comme dans la plupart des outils UML:

- il y a une distinction importante entre les menus contextuels **"Delete from Model"** (*Shift+Del*) et **"Delete from diagram"** (*Del*).
- La palette est constituée de sous palettes (qui se cachent quelquefois mutuellement)
- Le "glisser/poser" est quelquefois possible.
- Ctrl-Z pour "undo"

Spécifiquement à l'outil "Topcased UML" :

- pas de "clic/glisser/lâcher" mais "clic pour sélectionner" dans la palette puis un "autre clic" pour créer un nouvel élément dans le diagramme à la position voulue.
- pas de menu contextuel "add attribute/property" ni "add operation/method" mais des éléments **"Property"** et **"Operation"** à récupérer dans la palette et à ajouter aux classes du diagramme.
- Souvent besoin de cacher (via Properties/**graphics/hide**) certains éléments secondaires.

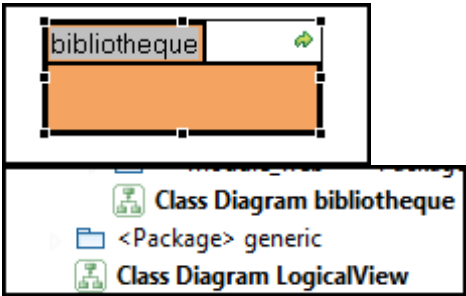
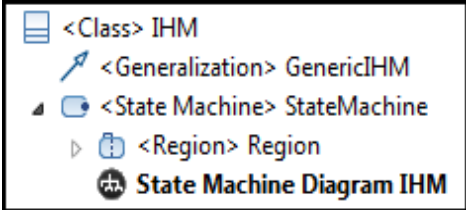
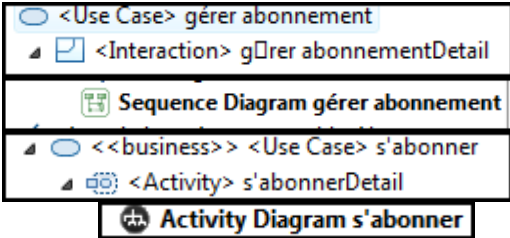
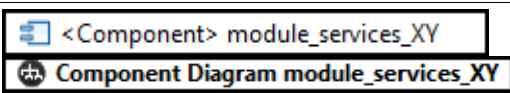
1.5. Organisation conseillée des packages et des diagrammes

L'un des principaux "points forts" de l'outil "Topcased UML" c'est de bien gérer la **cohérence** entre les **packages** et les **diagrammes** de façon à pouvoir naviguer de façon efficace dans l'arborescence d'un modèle UML.

Mode opératoire conseillé:

- Créer un élément UML (ex: package , classe, ...) dans un diagramme et le paramétrer.
- Effectuer un "double-clic" sur cet élément et **choisir le type de "sous diagramme" à générer**.
- Par la suite (une fois la création/association effectuée) , un **double-clic** ultérieur permettra de **naviguer d'un niveau vers un sous niveau** (de détails).

Sous diagrammes classiques pour indiquer les détails:

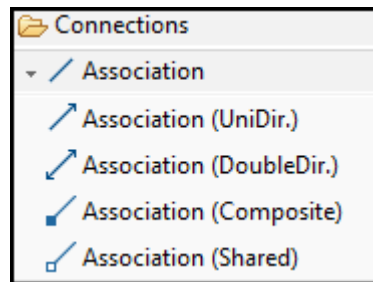
Eléments du modèle UML	diagramme(s) classique(s) associé(s) pour les détails	Exemple(s)
Package	Diagramme de classes (ou ...)	
Classe	Diagramme d'états (state machine) ou diag. de structure composite	
Use Case	diagramme d'activités et/ou diagrammes de séquences	
Composant	Diagramme de (sous)composants ou de structure composite , ...	

Remarques importantes:

- Pour bien organiser un modèle UML, il faut réfléchir le plus tôt possible à la décomposition en différents packages.
- Il est toujours possible de renommer ou déplacer un package (après coup / par la suite).

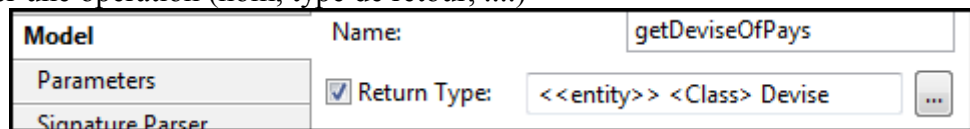
1.6. Edition d'un diagramme de classes (spécificités)

Sous palette "connections" avec **plusieurs variantes pour les associations**:

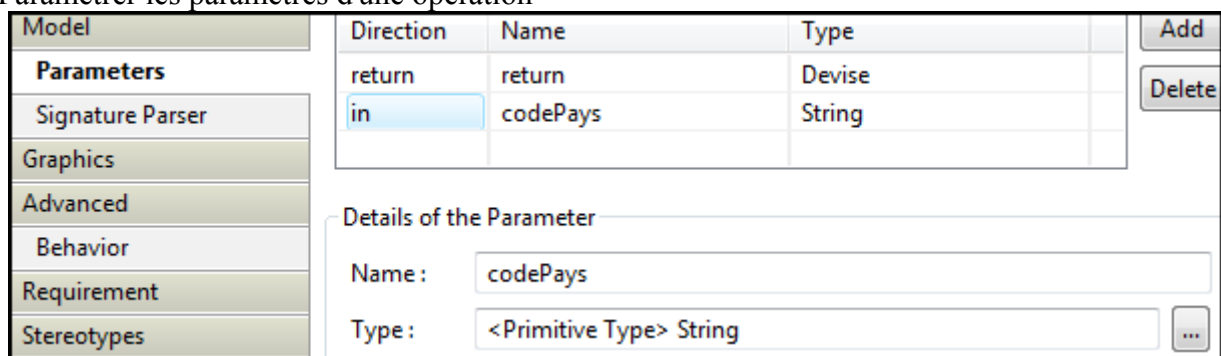


Dans la fenêtre des propriétés:

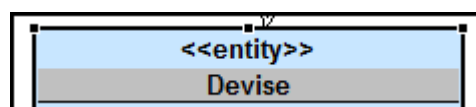
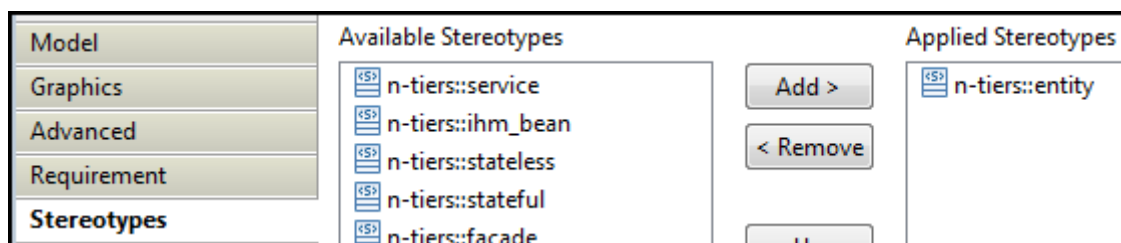
* Paramétrer une opération (nom, type de retour,)



* Paramétrer les paramètres d'une opération

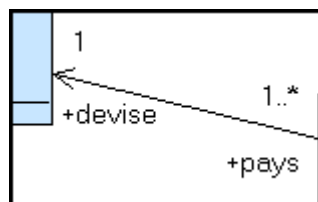


* Choix d'un (ou plusieurs) stéréotype(s) à appliquer:



* Paramétrer les extrémités ("first end", "second end") d'une association:

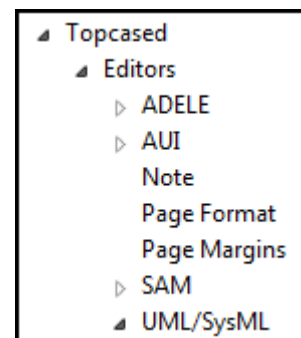
Model	Classifier:	Pays
First End	Role:	pays
Second End	<input type="checkbox"/> isNavigable	
Graphics	Association Type:	none
Advanced	Property Visibility:	public
Requirement	Multiplicity	
Stereotypes	Lower Bound:	1
Stereotype Attributes	Upper Bound:	*
Owned Rules		



* Montrer ou cacher **Graphiquement** les différents éléments d'une association:

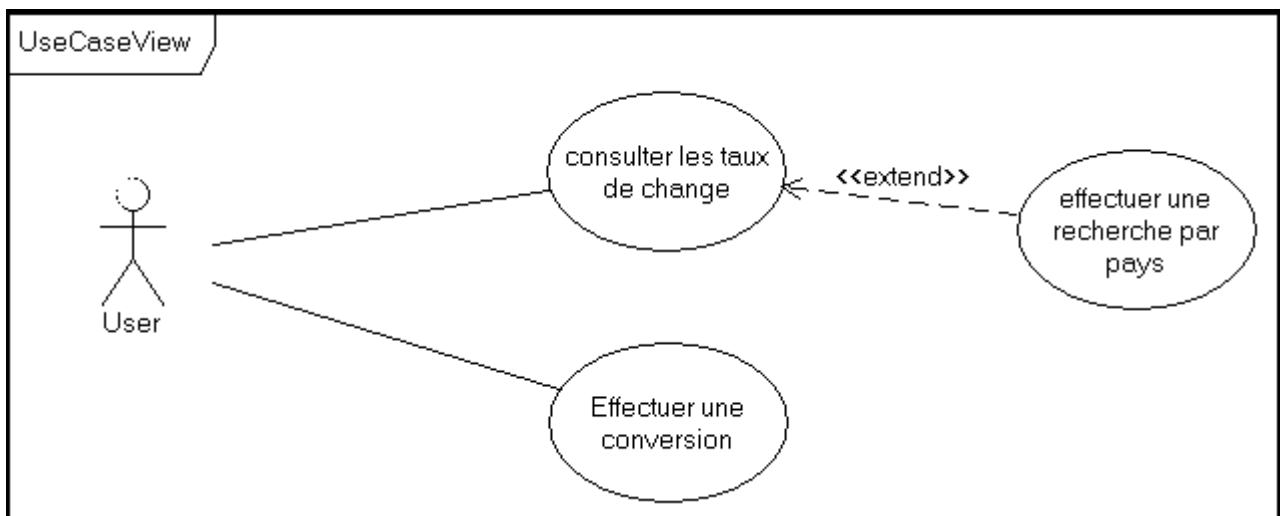
Graphics	Router:	Oblique
Advanced	Visible Elements	Hidden Elements
Requirement	Aa srcNameEdgeObject Aa srcPropertiesEdgeObject Aa srcCountEdgeObject Aa targetNameEdgeObject Aa targetPropertiesEdgeObject Aa targetCountEdgeObject Aa stereotypeEdgeObject	Aa middleNameEdge
Stereotypes	<input type="button" value="Hide >"/> <input type="button" value="< Show"/>	
Stereotype Attributes		
Owned Rules		

1.7. Préférences/options sur l'éditeur topcased UML

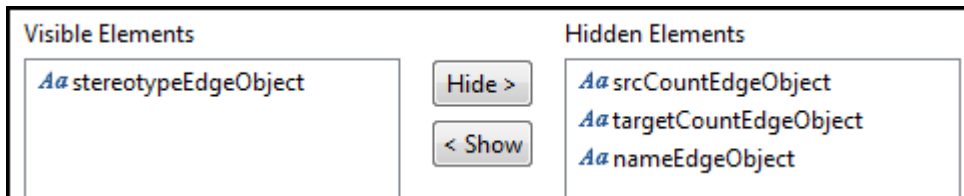


Depuis Navigator , racine du projet , clic droit / "Properties" et

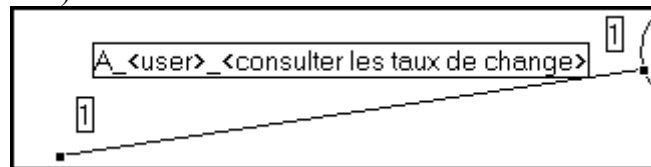
1.8. Edition d'un diagramme de Use Cases (spécificités)



* Montrer ou cacher les différents éléments d'une association sélectionnée:



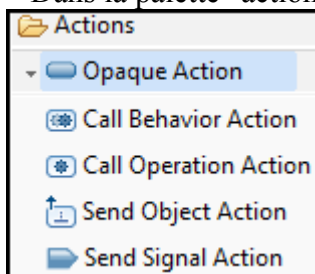
Si tout est visible (et illisible):



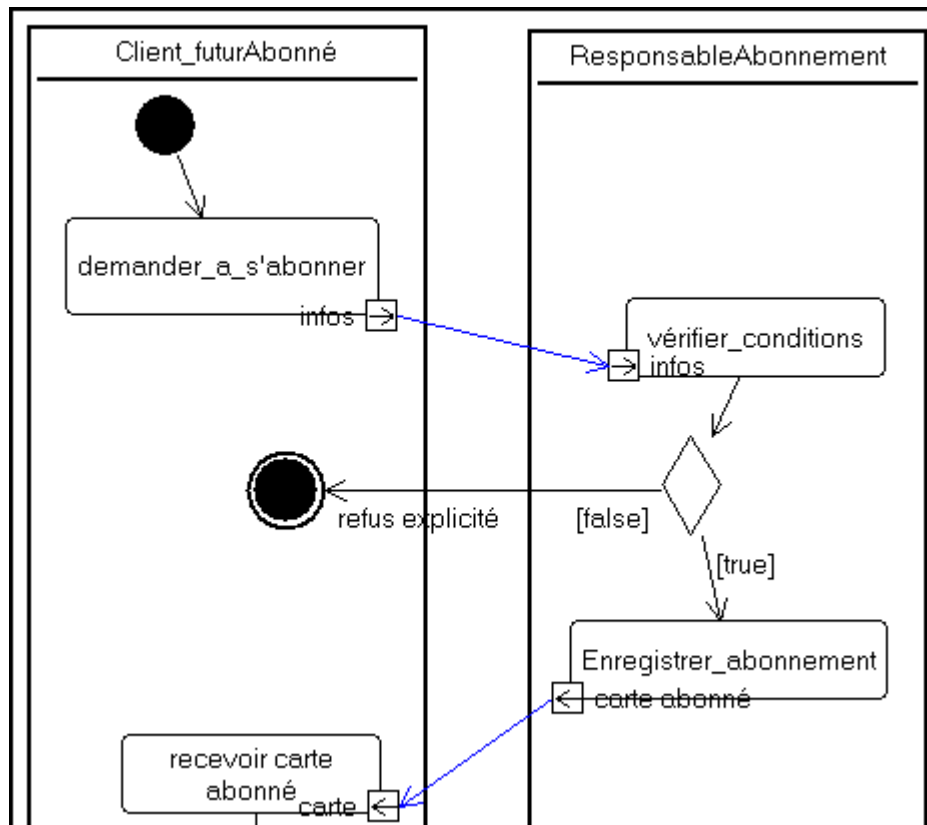
En cachant certaines parties non significatives , c'est mieux (plus lisible) !

1.9. Edition d'un diagramme d'activités (spécificités)

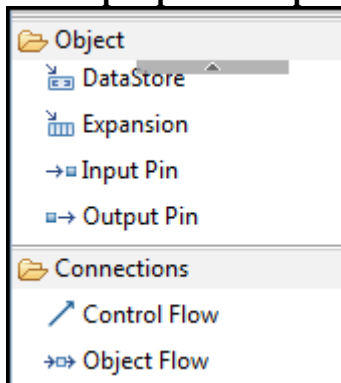
* Dans la palette "actions" , plusieurs variantes d'actions:



NB: Une "opaque action" pourra ultérieurement être transformée en un type d'action plus précis via le menu contextuel "**transform into ...**" de l'arborescence (outline).



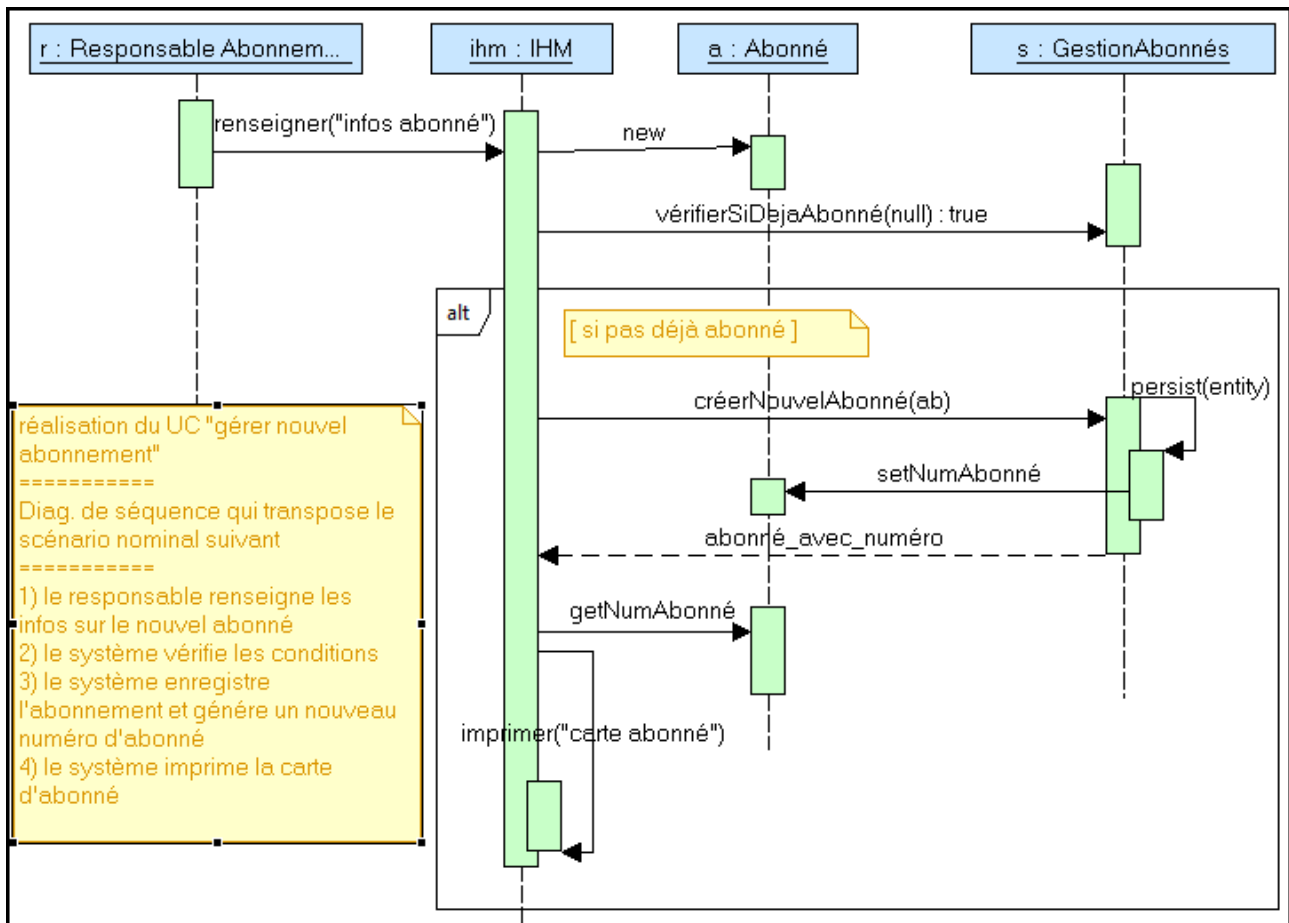
* Dans les versions très récentes d'UML , les "objects flow" nécessitent la mise en place préalable de "output pin" et "input pin" sur les actions:



-----> Une connexion de type "Object flow" pourra être directement placée entre un "output pin" et un "input pin" ou pourra mener vers un éventuel intermédiaire de flux de données (ex: DataStore , CentralBuffer, ...).

NB: Comme toujours, utiliser la sous partie "graphics" de la fenêtre des propriétés permet de cacher les éléments non significatifs et de gagner en lisibilité.

1.10. Edition d'un diagramme de séquences (spécificités)



Mode opératoire:

- créer un diagramme de séquence (en tant que détail d'un "use case")
- placer des **lignes de vie ("LifeLine")** et préciser les **types représentés** (acteurs, classes, ...)

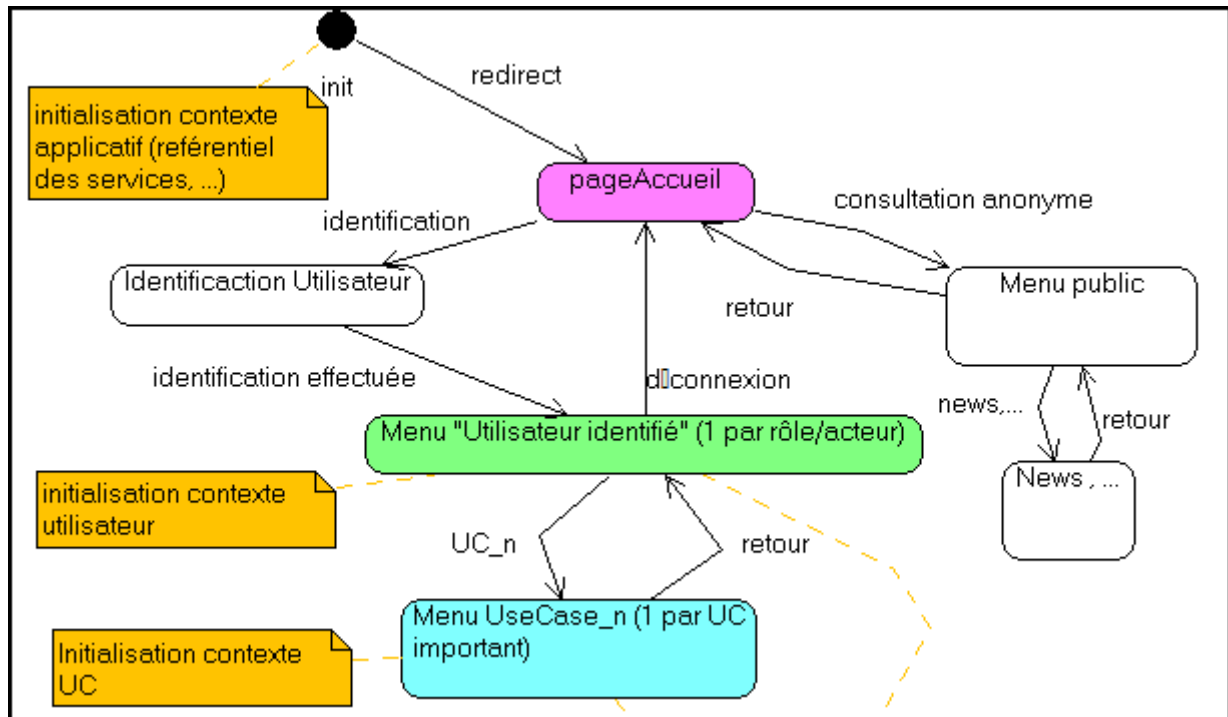
Model	Name:	a
Graphics	Visibility:	public
Advanced	Represents:	<<entity>> <Class> Abonné
Requirement		

- placer des **blocs d'exécution** sur les lignes de vie
- placer des **messages** entre un bloc d'exécution et un autre
- **paramétrer** les messages (*saisir un nom ou bien sélectionner une opération disponible au niveau de type d'objet qui reçoit le message*)
- si une opération sélectionnée en tant que message envoyé/reçu/traité comporte des paramètres de types simples (String, ...), on peut alors spécifier les valeurs de ces paramètres

Operation:	<Operation> renseigner (quelquechose : String)		
	Property Name	Type	Value
	quelquechose	String	infos abonné

NB: On peut également placer des fragments combinés (avec mot clef "alt" , "opt" , "loop" , ...) d'UML2 . Cependant, l'outil Topcased 4.2.1 ne gère pas encore parfaitement tous les paramétrages nécessaires.

1.11. Edition d'un diagramme d'états (spécificités)



* Paramétrage interne d'un état

Model	Name:	NEW_ONE
Internal Transition	Visibility:	public
Graphics	Entry Behaviour:	
Advanced	Exit Behavior:	
Requirement	Do Activity:	<Activity> saisir valeurs initiales
Stereotypes		
Stereotype Attributes		



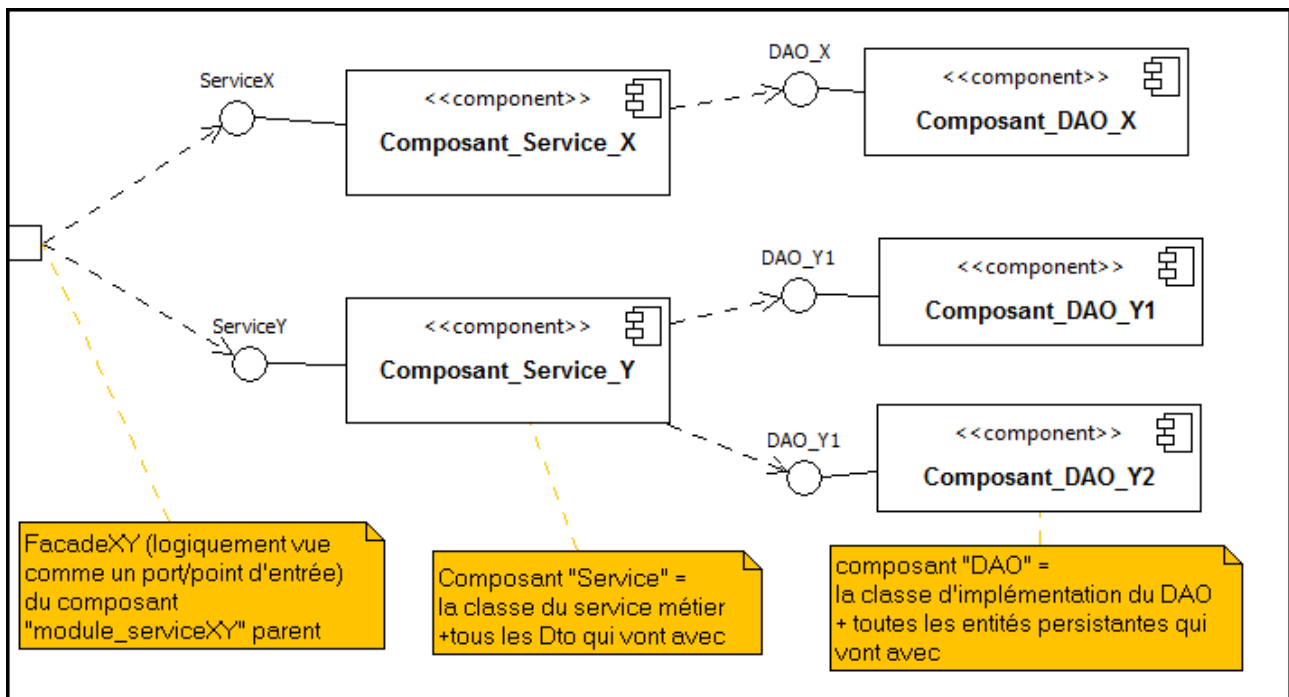
NB: Dans une version récente du produit (4.2.1), le choix d'une action/activité/traitement/comportement à associer à une activité via (entry: , do: , exit:) passe par une sélection d'une chose qui doit avoir été préalablement créée.

Mode opératoire (indirect) et qui fonctionne à peu près:

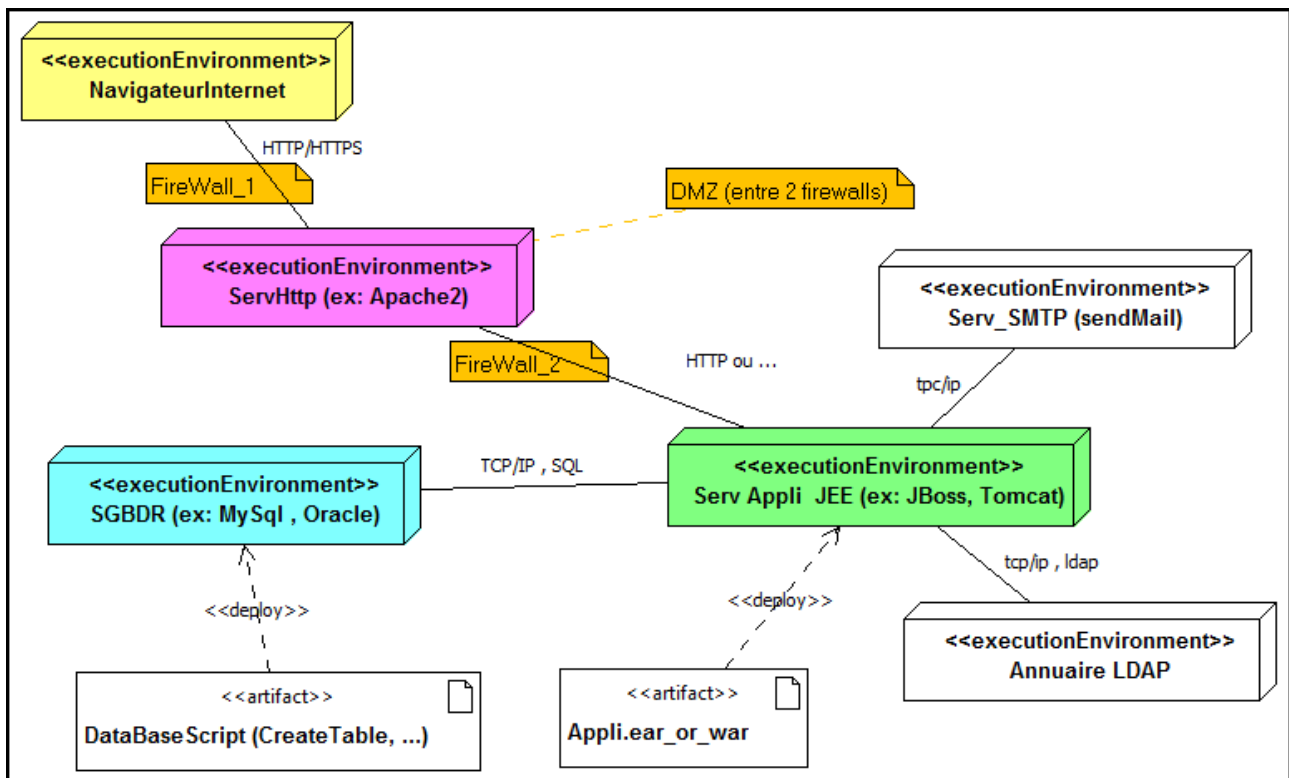
- se placer sur niveau parent "StateMachine" et créer un nouvel élément de traitement via le menu contextuel "Create child / Owned behaviour / Opaque Behaviour"
- renommer cet "opaque behaviour" avec un nom explicite d'action/activité
- retourner sur le paramétrage d'un des états du diagramme d'états.
- Sélectionner l'élément "behaviour" en tant qu'action à associer (entry/do/exit).

NB: En suivant ce procédé on remarque que l'élément "behaviour" à rattacher est devenu une simple chaîne de caractères après une fermeture/réouverture du modèle (.umldi, .uml).

1.12. Edition d'un diagramme de composants (spécificités)



1.13. Edition d'un diagramme de déploiement (spécificités)



Remarque: au sein de la version récente (4.2.1), les multiplicités fixées (1, *, ...) ne s'affichent malheureusement pas (<hide> par défaut).

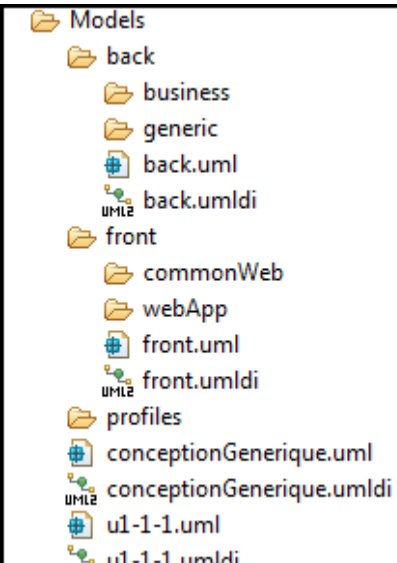
1.14. Glisser/poser d'un élément externe pour y faire référence

En effectuant un **glisser/poser** d'une classe Cx d'un package p1 vers le diagramme de classes du package p2 **tout en maintenant enfoncée la touche "CTRL"** on y introduit **une référence externe avec l'indication (from p1)** qui sera "sans détails/sans contenu".

NB: Si l'on n'enfonce pas la touche "CTRL" durant le glisser/poser, on récupère alors tous les attributs et toutes les opérations comme détails dans une copie "Cx dans p2" malheureusement pas automatiquement synchronisée avec l'original "Cx de p1" lorsque sa structure évolue.

1.15. Décomposition d'un modèle UML en plusieurs sous fichiers

Attention: Cette **fonctionnalité avancée** n'est utile que sur un vrai projet de taille importante. Les chemins relatifs doivent rester stables

	<p><u>Mode opératoire (dans un niveau "xxx"):</u></p> <ol style="list-style-type: none"> 1) créer un sous niveau yyyy (package yyyy) 2) "double-clic" pour créer le diagramme de classes qui va avec 3) sélectionner le sous package yyyy et clic droit "Control" ajouter éventuellement le sous répertoire "yyy" devant yyy.uml (platform:/resource/tests/Models/yyyy.uml --> platform:/resource/tests/Models/yyyy/yyyy.uml) <p>----> <u>résultat</u> : le fichier xxx.uml (et son frère xxx.umldi) ont été modifiés et la sous partie "yyy" sélectionnée a été déplacée dans les nouveaux fichiers yyy/yyyy.uml (et yyy/yyyy.umldi).</p> <p>On peut ainsi directement travailler sur le fichier yyy.umldi si on ne travaille que sur cette partie. --> On peut envisager plusieurs versions de la sous partie "yyy". Si l'on ouvre le fichier de niveau principal "xxx.umldi", les modifications apportées à la partie yyy sont alors automatiquement stockées dans les sous fichiers "yyy" (en suivant les liens).</p>
--	---

Les parties annexes (dans d'autres fichiers) apparaissent en gris mais la navigation (essentiellement en mode lecture) reste possible.

