



Atelier Génie Logiciel Prise en main de *StarUML sur l'étude de cas Bookinons (1^{ère} partie)*



Le **Génie Logiciel** est un ensemble de techniques tendant à faire passer la production d'un logiciel au stade de l'industrie. Le génie logiciel intègre aussi le suivi et la maintenance. Le génie logiciel est également caractérisé comme "une aide apportée par les méthodes et les outils à différents niveaux du cycle de développement" ou encore comme "une collection d'outils et de méthodes automatisés qui assistent les concepteurs de logiciels".

Un Atelier de **Génie Logiciel** (noté **AGL** ou en anglais *CASE* pour *Computer Aided Software Environment*) est un ensemble d'outils logiciels à l'usage des informaticiens pour automatiser la production et la maintenance de logiciels spécifiques en allant de la phase de conception à la phase d'implémentation. Les **AGL** proposent ainsi des outils, des assistants visuels et disposent la plupart du temps d'un débogueur.

Il existe plusieurs types d'Atelier Génie Logiciel.

StarUML est un outil exclusivement destiné à la modélisation graphique objet du langage **UML**.

Rappelons qu'UML (**U**nified **M**odeling **L**anguage ou langage de modélisation unifié) est un langage de modélisation graphique avec un formalisme **orienté objet**.

StarUML supporte donc la plupart des diagrammes spécifiés par UML 2.0.

StarUML permet de générer des "squelettes" de code dans un langage objet (Java, C++, C#)

StarUML est un projet *open source* visant le développement d'une plateforme UML/MDA libre, rapide, souple, extensible, riche en fonctionnalités...

L'objectif du projet **StarUML** est d'offrir une alternative aux célèbres outils UML commerciaux tel que Rational Rose... Mais **StarUML** ne fonctionne que dans l'environnement Windows

Ce tutoriel va vous guider dans l'utilisation de **StarUML** selon une démarche **pseudo-RUP** basé sur une **approche "vue 4+1"**. L'application à modéliser sera celle de l'étude de case **bookinons** vue en cours ...

Rappelons que l'objectif final de la modélisation est :

- ↳ **la production de programmes** dans un langage informatique constituant le futur logiciel de l'utilisateur. Ces traitements, *éléments dynamiques*, pourront, par exemple, être modélisés à l'aide des diagrammes de Use Case, des descriptions détaillées, des diagrammes de séquences, des diagrammes d'activité,...
- ↳ **la création du SGBDR** où seront stockées les données de l'entreprise (client, commande, article, ...). Ces données, *éléments statiques*, pourront, par exemple, être modélisées à l'aide des diagrammes de classes, ...

Si vous travaillez sur les ordinateurs de l'IUT, l'installation de **Star UML** est déjà réalisée sous WINSV2, vous pouvez passer directement à la partie suivante.

Si vous travaillez avec votre propre ordinateur, vous pouvez télécharger **star UML** à l'adresse suivante <http://staruml.sourceforge.net/en/> à partir du menu **Downloads** → **StarUML Download**. Il vous suffit ensuite de l'installer sur votre PC.

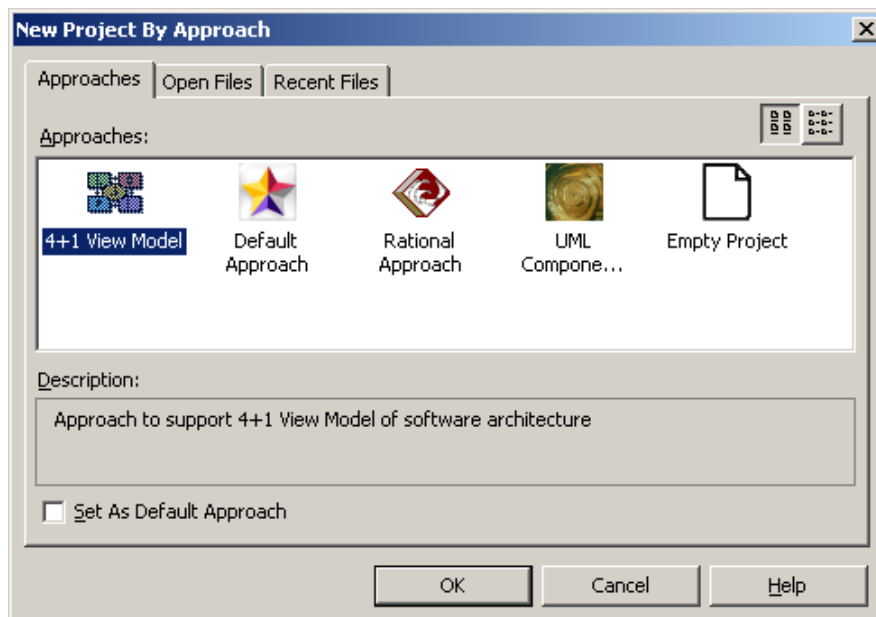
Préliminaire : lancement de Star UML (uniquement sous Windows...)

Lancer **StarUML** en double cliquant sur l'icône suivante :
Ou depuis le menu **Démarrer**

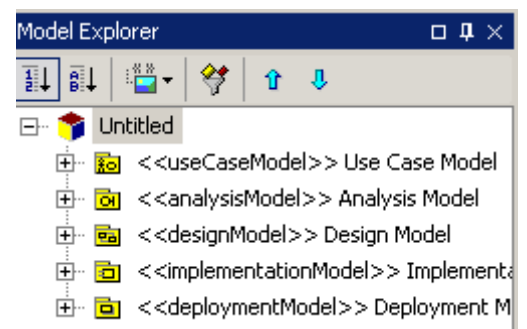


Au lancement du logiciel, **StarUML** propose 4 approches prédéfinies de modélisation :

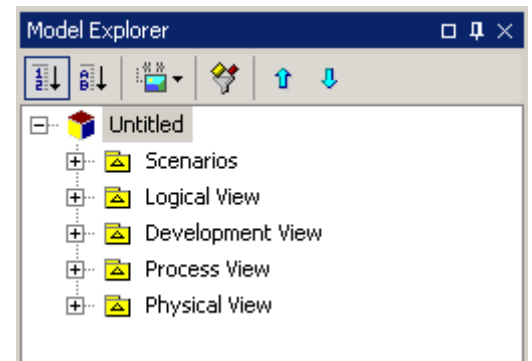
- une approche *par vue 4+1*,
- une approche dite *par défaut*,
- une approche *Rational Rose*
- et une approche *par composants UML*



↳ **L'approche « par défaut »** se décompose en cas d'utilisation (useCaseModel), en diagramme de séquences (analysis Model), en diagramme de classes (designModel), diagramme d'implémentation (ou de composants) (implementationModel), et en diagramme de déploiement (deploymentModel). Cette approche serait suffisante pour initialiser un projet, mais nous préférons utiliser une approche par vue 4+1.



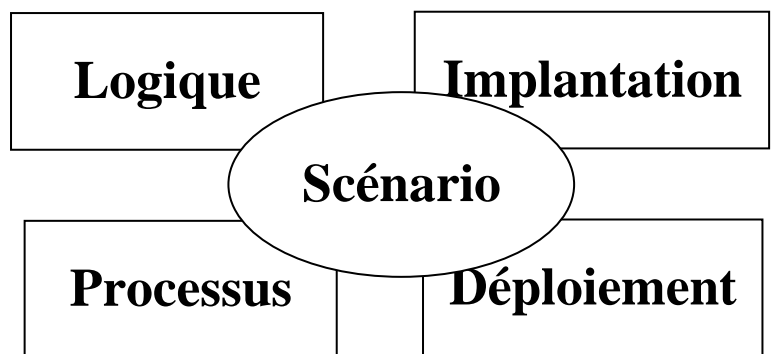
↳ **L'approche « vue 4+1 »** permet de faire le lien entre la conception UML et la méthodologie (démarche) que nous allons adopter pour mener à bien notre projet. Cette démarche permettra de réaliser lors de différentes itérations, un **enrichissement** de l'étude basée sur UML. L'idée étant de partir des besoins utilisateurs, en allant vers le code. Chaque **étape** permettra de mieux comprendre / modéliser le Système d'information. Il est intéressant de remarquer que les approches objets basées sur **UML** ont pour caractéristique et avantage de « proposer » sans « imposer ». Nous choisirons donc une approche vue 4+1 pour organiser notre démarche et "classer nos diagrammes" mais les différents diagrammes UML pourraient tout aussi bien trouver leur place dans une autre approche ...



L'approche vue 4+1 propose **5 vues** :

- la **vue scénario (Scenarios)** (appelée aussi **vue des cas d'utilisation**) guide toutes les autres car elle décrit les besoins attendus par chaque acteur du système
- la **vue logique (Logical View)** décrit le *comportement du système vu de l'intérieur*. Cette *vue de haut niveau* se concentre sur l'abstraction et l'encapsulation : elle modélise les éléments et les mécanismes principaux du système.
- la **vue implantation (Development View)** décrit *l'organisation logicielle* du projet. Cette vue de bas niveau (aussi appelée **vue de réalisation**) identifie les modules qui réalisent (physiquement) les classes de la vue logique, elle montre également l'organisation entre les composants et leurs dépendances.
- la **vue déploiement (Physical View)** décrit *l'organisation matérielle du projet*. Cette vue est très importante dans les *environnements distribués*.
- et la **vue processus (Process View)** qui est intéressante dans les *environnements multitâches* car elle montre la décomposition du système en terme de processus

L'approche vues 4+1 est également souvent représentée par le schéma suivant où la vue scénario (celles des cas d'utilisations) constitue la "colle" qui unifie les 4 autres vues



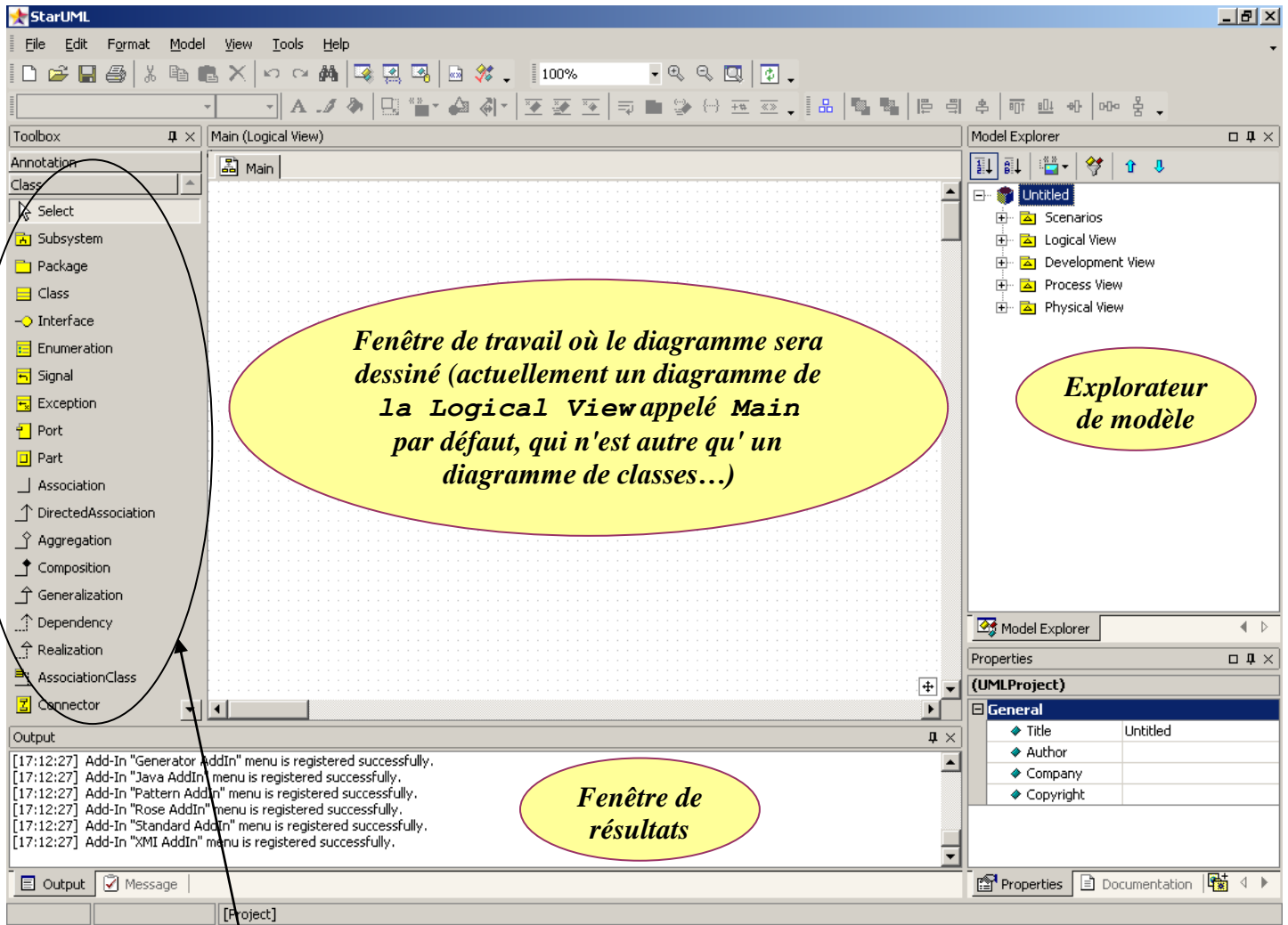
4+1 View Model

Ce schéma est repris par celui de l'icône (ci-contre) proposant une approche par **4+1 View Model** sous Star UML.

*Pour commencer ce tutoriel, choisir donc une approche par vue 4+1 en cliquant sur l'icône **4+1 View Model** proposé par Star UML Puis cliquer sur **OK***

Remarque : Une fois Star UML ouvert, pour créer un nouveau projet respectant une approche vues 4+1, il suffit de cliquer sur : **File → New Project by Approach ...**

Une fois l'approche choisie, **StarUML** ouvre alors l'espace de travail suivant :



Vous pouvez retrouver cette Interface détaillée dans le **chapitre 11 User-Interface Reference** de la documentation détaillée de Star UML (**StarUML 5.0 User Guide**) accessible depuis le site officiel de starUML via : <http://staruml.sourceforge.net/docs/user-guide%28en%29/toc.html>

Une aide est également disponible directement *dans le logiciel starUML* à partir de la barre de menus : **Help → Contents...**

Vous pouvez commencer par enregistrer votre projet en choisissant à partir de la barre de menus **File → Save as**

Appelez par exemple votre projet **bookinons**

N'oubliez pas au cours du tutoriel de sauvegarder fréquemment votre travail à l'aide du bouton :



... Au cours de ce tutoriel, nous allons apprendre à manipuler StarUML en nous appuyant sur les diagrammes de l'étude de cas **bookinons** vue en cours ...

Exercice 1 : Axe fonctionnel : la vue Scénario et les Diagrammes de Use Case ...


La vue scénario est la vue de référence (la vue dite **+I**): elle va permettre de partager les informations collectées entre les **4 autres vues** (Logical, Development, Process et Physical)

Rappelons qu'une des premières étapes de la modélisation consiste à la découverte des use case à partir du cahier des charges. C'est donc la notion de **Use Case** qui est rattachée à la vue scénario : il est également important de souligner que la vue scénario est une vue "plus abstraite" que technique.

La vue Scénario est donc destinée à contenir le(s) diagramme(s) de Use Case.

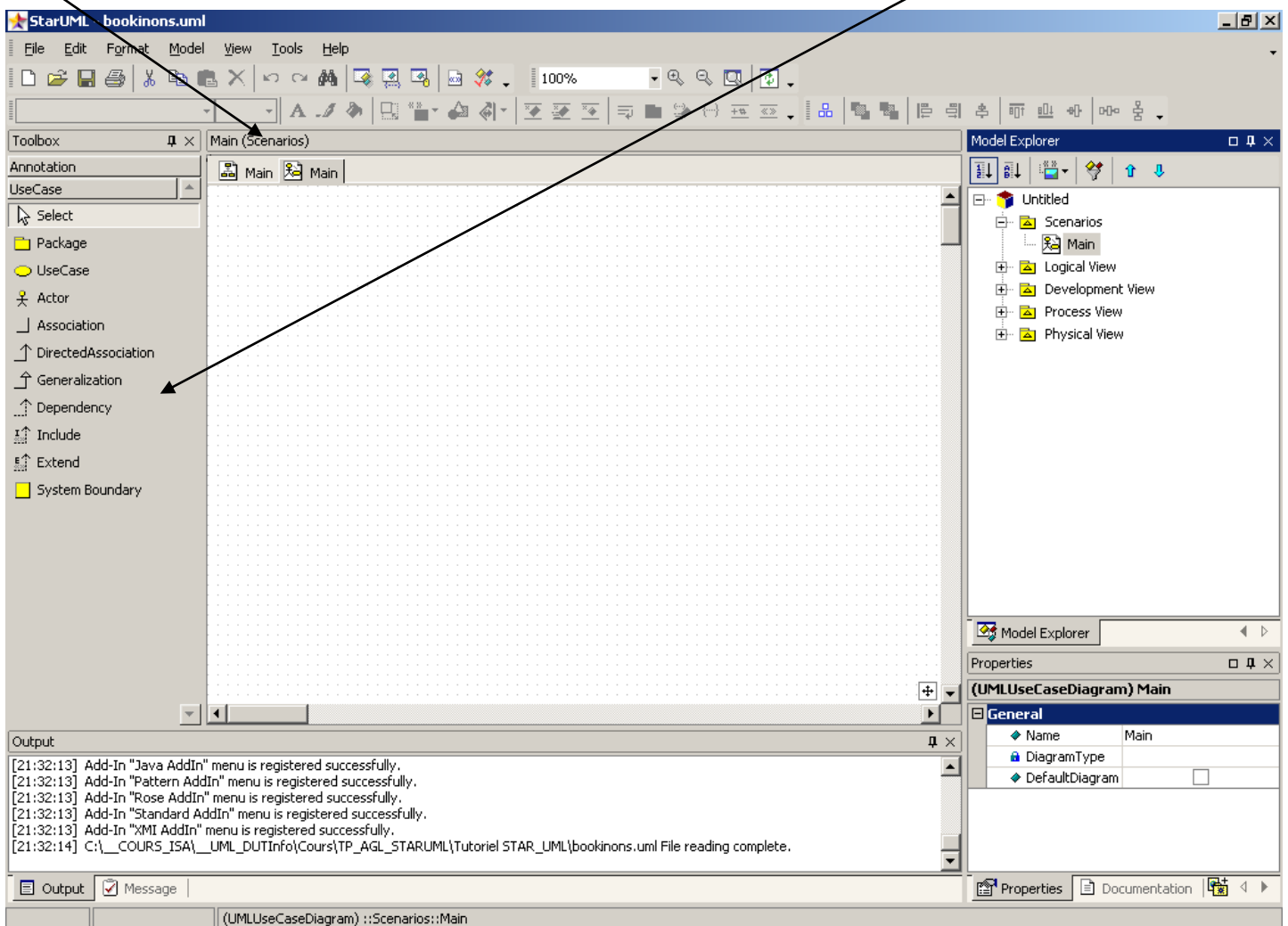
Par défaut, un diagramme de Use Case est déjà disponible dans la vue scénario lorsqu'une approche par vue 4+1 a été sélectionnée.

Pour l'afficher en tant que fenêtre de travail active, il suffit de double-cliquer sur **Scenarios** dans le **Model Explorer** (à droite de votre écran) : la **vue scénario** s'ouvre alors et un objet **Main** précédé

de l'icône  est disponible dans cette vue. Double-cliquer sur ce **Main** (de **Scenarios**) afin d'ouvrir une nouvelle fenêtre de travail qui devient alors la fenêtre active.

Cette fenêtre est la fenêtre active car la barre au-dessus des fenêtres de travail indique maintenant **Main (Scenarios)** et la palette graphique **Toolbox** propose dans un onglet **Use Case** les notations graphiques utilisées par UML pour représenter un diagramme de Use Case.

Vous devez donc vous retrouver maintenant dans la configuration suivante :



Il ne nous reste plus qu'à suivre les instructions suivantes pour dessiner le diagramme de Use case du projet **bookinons**...

Pour créer un objet (élément) sur le *diagramme de Use Case* (acteur, Use Case, relation,...), nous allons utiliser la palette graphique mise à notre disposition par StarUML à gauche de notre écran dans la fenêtre **Toolbox** sous l'onglet **Use Case**.

1.1 Création d'un Acteur

Pour ajouter un acteur dans le diagramme de Use Case, cliquez dans la palette graphique sur



Puis déplacez le pointeur de votre souris dans votre fenêtre active et faites 1 clic gauche.

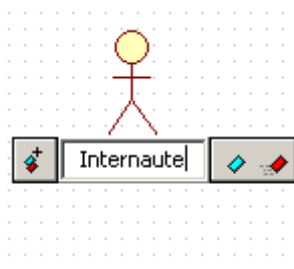
Un acteur apparaît sous sa forme *stick man*.

Lorsque vous créez un nouvel objet dans votre projet, un nom lui est déjà attribué. En fait ce nom est un des noms des "contributeurs" au développement et à la diffusion de StarUML puisque rappelons-le StarUML est un projet open source : ce nom est donc choisi dans la **Contributor List**.

Vous pouvez alors remplacer ce nom par le nom que vous souhaitez réellement donner à votre acteur.

En ce qui nous concerne, nous commencerons par l'**Internaute**.

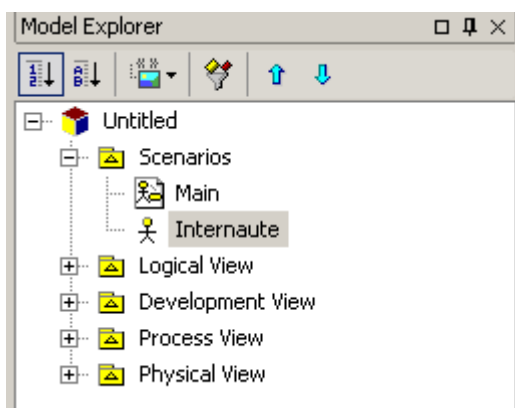
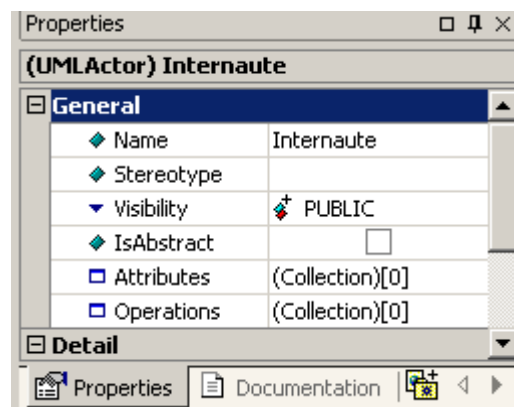
Taper donc **Internaute** puis appuyer sur entrée ou bien cliquer ailleurs dans la fenêtre de travail pour valider.



En dessous de la fenêtre **Model Explorer** se trouve la fenêtre **Properties** (en bas à droite de votre écran).

Comme son nom l'indique cette fenêtre regroupe *les propriétés d'un objet (élément du diagramme)*

Vous pouvez alors remarquer que dans le champ **Name** se trouve le nom que nous venons de donner à notre acteur, à savoir : **Internaute**. Nous nous contenterons pour le moment de cette propriété et explorerons les différentes propriétés possibles au fur et à mesure de notre avancement du tutoriel.



Consulter également la fenêtre **Model Explorer**.

Vous constaterez alors que l'acteur **Internaute** apparaît également dans cette fenêtre comme objet de la vue

Scénarios et qu'il est précédé de l'icône

1.2 Création d'un Acteur et choix du stéréotype

Il existe plusieurs représentations graphiques d'un acteur.


Par défaut sous StarUML, l'acteur est représenté sous sa forme *stick man*, mais on peut également choisir de représenter l'acteur *sous forme rectangulaire*. On réserve d'ailleurs généralement cette dernière notation pour les *systèmes dits connectés*.

Nous allons donc maintenant rajouter l'acteur **Paiement Sécurisé**.

Procédez tout d'abord comme précédemment afin d'obtenir dans votre diagramme de Use Case un acteur avec une apparence de *stick man*.



Pour changer l'apparence de l'acteur, il faut tout d'abord le sélectionner. Pour cela, effectuer 1 clic sur l'acteur afin de le voir apparaître dans un rectangle. Ensuite pour changer son apparence, il y a 3 possibilités :

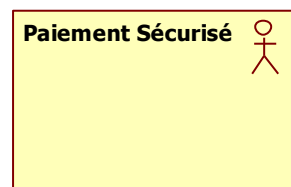
- *soit à partir de la barre de menus* en haut de votre écran : choisir **Format** → **Stereotype Display**, puis **Decoration** pour obtenir la *représentation rectangulaire* ou **Textual** pour revenir dans la représentation *stick man*...
- *soit à partir de la barre des icônes* : une fois l'acteur sélectionné, vous avez accès à l'icône , cliquez dessus et sélectionnez le format **Textual** ou **Decoration**.
- *soit directement à partir du diagramme de Use Case*: une fois l'acteur sélectionné, faites un clic droit, puis choisir **Format** → **Stereotype Display**, puis **Decoration** ou **Textual** selon vos besoins ...

A retenir !!! Sous **Star UML**, un acteur peut avoir 2 apparences (**Stereotype Display**) :

- sa forme *stick man* (option **Textual** de **Stereotype Display**)
- sa forme *rectangulaire* (option **Decoration** de **Stereotype Display**)



Option d'affichage : **Textual**




Option d'affichage : **Decoration**

... Pour continuer ce tutoriel, faites en sorte d'être dans la configuration précédente, c-a-d que :

- l'acteur **Internaute** soit représenté sous sa forme *stick man*
- et l'acteur **Paiement Sécurisé** soit représenté sous sa forme *rectangulaire (Décoration)*

Remarque : Redimensionner automatiquement un objet du diagramme est possible

à l'aide de l'icône **Auto Resize** 

Sélectionner par exemple l'acteur **Paiement Sécurisé** d'un clic de souris, puis cliquer sur l'icône de redimensionnement automatique **Auto Resize**  située juste à gauche de l'icône **Stereotype Display**. L'objet est alors automatiquement redimensionné.



1.3 Supprimer un acteur du diagramme et du modèle :

... Afin de tester la suppression d'un acteur, commencer par rajouter sur votre diagramme un nouvel acteur, vous pouvez laisser son nom par défaut...

☞ Si vous souhaitez ensuite **supprimer cet acteur uniquement du diagramme de Use Case**, vous devez sélectionner l'acteur, puis effectuer un clic droit et choisir **Edit → Delete**. Vous constaterez alors que votre acteur a bien disparu de votre diagramme. Par contre si vous consultez la fenêtre **Model Explorer**, vous retrouverez votre acteur dans la vue **Scenarios**. L'option **Delete** permet donc d'effacer un acteur (ou plus généralement un élément) du diagramme mais pas de le supprimer de la modélisation. Ainsi si maintenant, vous cliquez sur cet acteur dans la fenêtre **Model Explorer** et si vous le faites glisser cet acteur par drag and drop sur votre diagramme de Use Case, il s'affichera à nouveau sur votre diagramme de Use Case.

☞ **Pour supprimer définitivement un élément de votre modélisation**, il est indispensable d'utiliser l'option **Delete from Model** à la place de **Delete**, et c'est ce que nous allons tester maintenant. Sélectionnez par un clic l'acteur à supprimer, puis faites un clic droit, puis **Edit → Delete from Model**. Constatez que votre acteur disparaît non seulement de votre diagramme, mais a également disparu de la vue **Scenarios** de la fenêtre **Model Explorer**.

... A l'avenir faites bien attention au **Delete** que vous utiliserez ...
... Car dans la majorité des cas, c'est le **Delete from Model** que vous allez utilisé ...

1.4 Création d'un élément de type Use Case

Nous souhaitons maintenant modéliser dans notre diagramme, le **use case** :

Rechercher Ouvrage

Pour ajouter un élément de type **Use Case** sur le diagramme, procédez comme pour l'acteur. Cliquez dans la palette graphique sur

 UseCase

Puis déplacez le pointeur de votre souris sur votre fenêtre active et faites 1 clic gauche.

Un **Use Case** apparaît dans le diagramme avec un nom par défaut, qui comme pour les acteurs, est également un nom extrait de la **Contributor List**. Renommez ce Use Case en **Rechercher Ouvrage**.

Rechercher Ouvrage

L'affichage d'un **Use Case** peut avoir 3 apparences (**Stereotype Display**) :



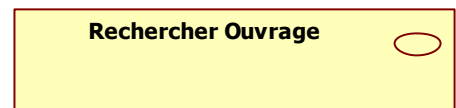
Rechercher Ouvrage

Affichage : Textual



Rechercher Ouvrage

Affichage : Iconic



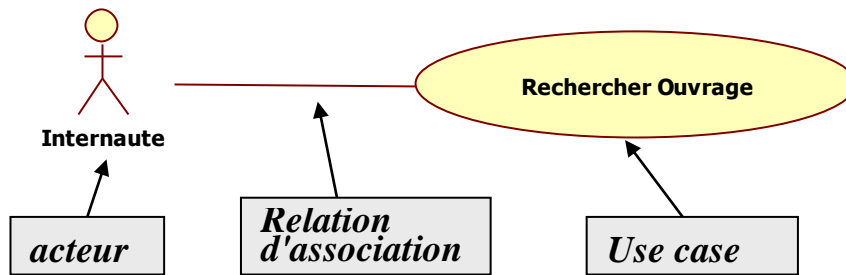
Affichage : Decoration

Vous pouvez essayer ces 3 apparences à partir de l'icône ou du menu **Stereotype Display**.

... Pour la suite du tutoriel, nous travaillerons avec un format **Textual**
pour l'apparence des Use Case ...

1.5 Modélisation complète du Use Case : Rechercher Ouvrage

La modélisation complète du Use Case **Rechercher Ouvrage** est la suivante :

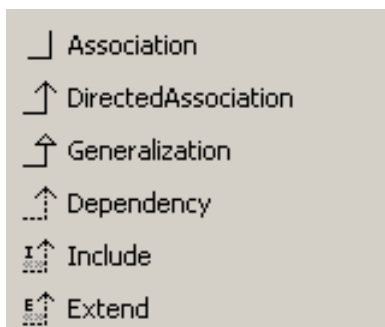


Cette modélisation nécessite la présence de 3 éléments :

- un élément de type *acteur* (l'**Internaute**)
- un élément de type *Use Case* (**Rechercher Ouvrage**)
- une *relation* entre l'élément *acteur* et l'élément *Use case* (une **association** dans le schéma ci-dessus)

Dans notre diagramme, nous avons déjà les éléments de type *acteur* et de type *Use Case*.

Il ne nous reste plus qu'à rajouter une *relation d'association* entre **Internaute** et **Rechercher Ouvrage**.



La palette graphique propose un certain nombre de relations qui sont reprises dans la copie d'écran ci-contre.

Pour modéliser le Use Case **Rechercher Ouvrage**, nous allons utiliser la relation d'**Association**.

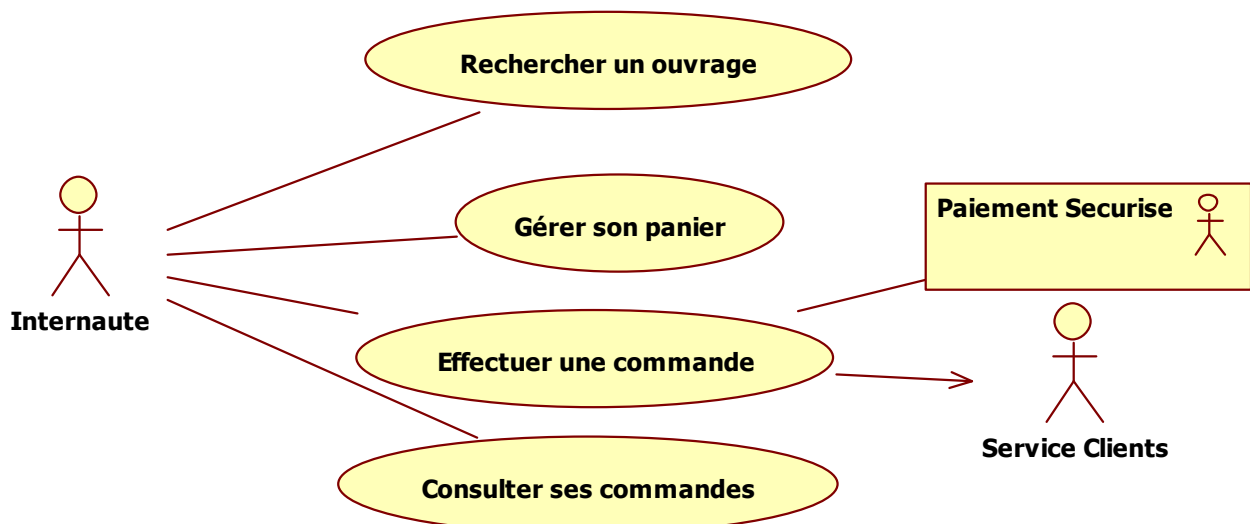
Commencer par cliquer dans la palette graphique sur

Association

Ensuite, sur le diagramme de Use Case, positionnez-vous l'acteur **Internaute**, appuyer sur le bouton gauche de la souris (*drag*) et tout restant appuyé déplacez-vous vers le Use Case **Rechercher**

Ouvrage, une fois sur ce Use Case, relâcher le bouton de la souris (*drop*) : la relation d'association doit alors apparaître entre l'acteur et le Use Case.

Afin de mettre en pratique ce que nous venons de voir dans les pages précédentes, compléter votre diagramme de Use Case afin d'obtenir le diagramme suivant :



1.6 Création d'un Use case à l'aide de raccourcis (*ShortCut Generation Syntax*) :

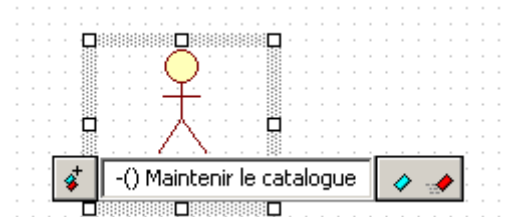
Pour chaque élément acteur et Use Case du diagramme de Use Case, il existe, sous StarUML, des raccourcis qui permettent de générer automatiquement des éléments du diagramme qui lui sont dépendants. Ainsi,

↳ *à partir d'un acteur, il est possible de générer automatiquement un (ou même plusieurs) Use Case* à l'aide du raccourci **- ()**



Commencez par créer sur votre diagramme l'acteur **Libraire**.

Une fois cet acteur créé, double-cliquez dessus afin d'ouvrir la boîte de dialogue dans laquelle vous taperez : **- () Maintenir le catalogue** comme l'indique la copie d'écran ci-contre, puis validez avec entrée.



Vous obtiendrez alors directement le Use Case suivant :



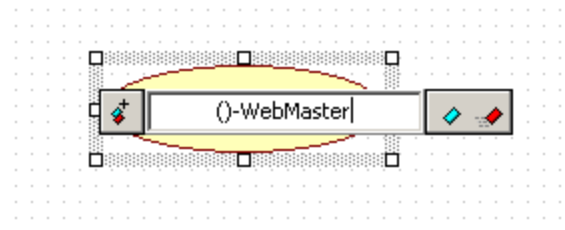
Remarque : Il est même possible de générer plusieurs Use Case depuis un même acteur, pour cela lorsque vous utilisez le raccourci **- ()** vous devez séparer vos noms de Use Case par une virgule.

↳ *à partir d'un Use Case, il est possible de générer automatiquement un (ou même plusieurs) Acteur(s)* à l'aide du raccourci **() -**

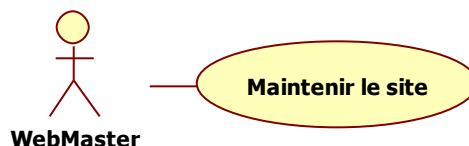
Maintenir le site

Commencer par créer sur votre diagramme l'élément Use Case **Maintenir le site**.

Une fois cet élément créé, double-cliquez dessus afin d'ouvrir la boîte de dialogue dans laquelle vous taperez : **() -WebMaster** comme l'indique la copie d'écran ci-contre, puis vous validerez avec entrée.

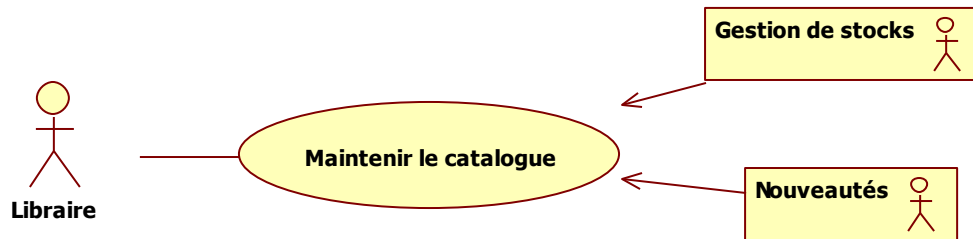


Vous obtiendrez alors directement le Use Case suivant :



Remarque : Comme précédemment, il est même possible de générer plusieurs acteurs depuis un même use Case, pour cela il suffit de séparer le nom des différents acteurs par des virgules...

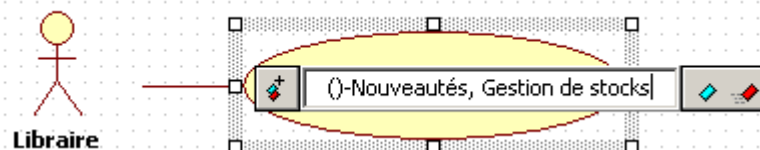
➤ Pour finaliser cette première ébauche du diagramme de Use Case, nous souhaitons maintenant rajouter deux nouveaux acteurs "**connectés**" que nous appellerons **Nouveautés** et **Gestion de stocks** et qui seront reliés à l'élément use case **Maintenir le catalogue** par une **association fléchée**



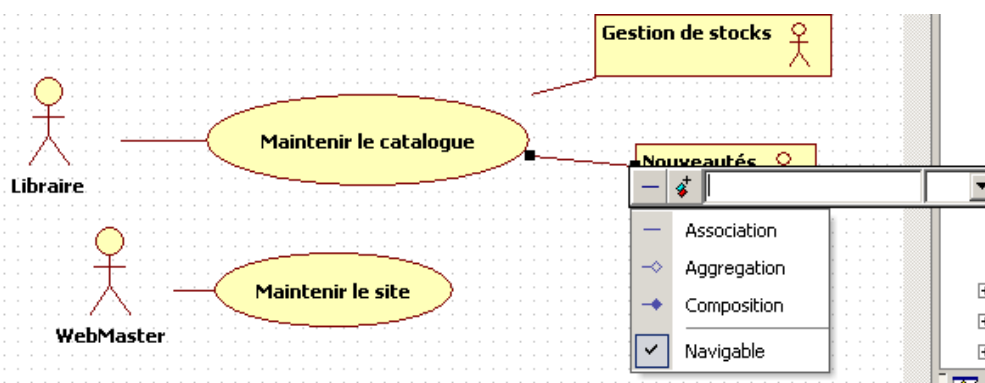
A vous désormais de choisir la méthode qui vous convient le mieux pour modifier votre use case :

- *soit en utilisant la palette graphique* : en cliquant sur le diagramme, vous ferez alors apparaître les acteurs que vous nommerez **Nouveautés** et **Gestion de stocks**. Il faudra ensuite changer leur **Stereotype Display** et à partir de la palette choisir une **association fléchée** de type **DirectedAssociation** qui permettra de relier les acteurs au use case.
- *soit en utilisant les raccourcis clavier* : Dans ce cas-là vous n'avez qu'à double cliquer sur le use case **Maintenir le catalogue**, puis remplir la boîte de dialogue comme suit :
 () -Nouveautés, Gestion de stocks.

↑ DirectedAssociation



Une fois validée, les deux nouveaux acteurs apparaîtront dans votre diagramme de use case sous forme de *stick man* et seront reliés à **Maintenir le catalogue** par une *simple association*. Par drag & drop, vous pouvez bien sûr repositionner les deux acteurs où vous le souhaitez sur le diagramme. Après avoir changé le **Stereotype Display** en **Decoration**, il vous sera également nécessaire de modifier l'association, car par défaut avec un raccourci clavier, l'association est de type *simple* et là nous souhaitons avoir des *associations fléchées*. Pour cela, double cliquez sur l'extrémité de l'association du côté de l'acteur (par exemple du côté de **Nouveautés**), une boîte de dialogue s'ouvre. Cliquez alors sur le - complètement à droite de la boîte de dialogue pour visualiser les options possibles et désélectionner l'option **Navigable**. Vous devriez alors obtenir une *association fléchée*, la flèche pointant vers le use case comme souhaité. Recommencez le même procédé en double cliquant sur l'extrémité de l'association du côté de l'acteur **Gestion de stocks** pour transformer l'association simple en association fléchée indiquant le sens de navigation des données.



Remarque : La liste complète des raccourcis peut être consultée dans la documentation détaillée de StarUML (**StarUML 5.0 User Guide** : <http://staruml.sourceforge.net/docs/user-guide%28en%29/toc.html>). Rappelons que cette aide est également disponible directement dans le logiciel StarUML à partir de la barre de menus : **Help** → **Contents...**

- une liste des raccourcis (dont le tableau suivant est extrait) est ainsi disponible dans le **chapitre 4 : Modeling with StarUML** dans la partie **ShortCut Generation Syntax** de la documentation en ligne :

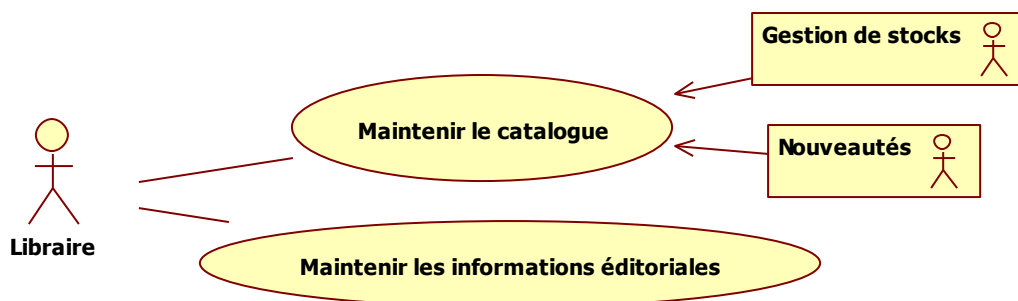
Usecase Diagram	()-	UseCase	The target model(Actor) linking with the current element makes a link of communication.
	-()	Actor	The target model(UseCase) linking with the current element makes a link of communication.
	<i-	UseCase	Makes include relationship from target element to the current element.
	-i>	UseCase	The target element linking with the current element makes a link of include.
	<e-	UseCase	Makes include relationship from target element to the current extend.
	-e>	UseCase	The target element linking with the current element makes a link of extend.

- des exemples sont également donnés dans le **chapitre 5.1 Modeling with UseCase Diagram**

1.7 Création d'une note de commentaires sur un diagramme

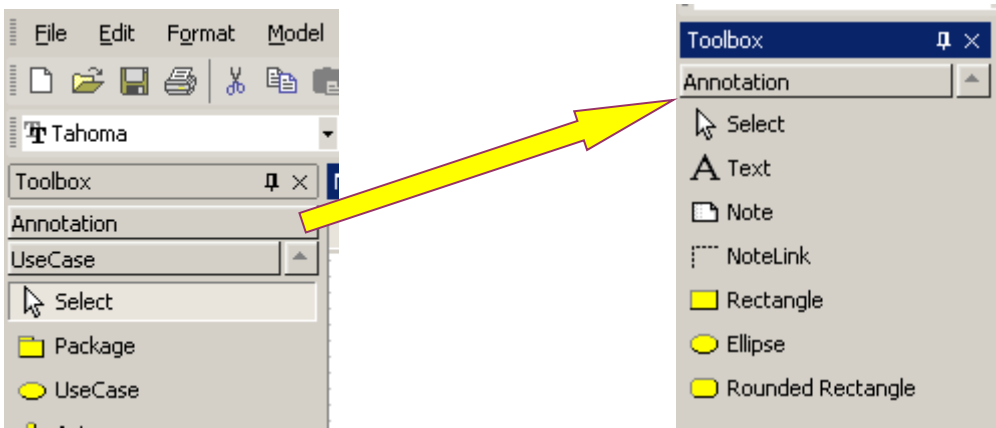
Pour que votre diagramme soit bien explicite, il est conseillé d'y rajouter des notes de commentaires afin de détailler ou de justifier un choix de modélisation.


Commencez par rajouter sur le diagramme un Use Case nommé **Maintenir les informations éditoriales** que vous associerez à l'acteur **Libraire**.



Mais que signifie exactement pour vous **Maintenir les informations éditoriales** ? Vous pouvez le préciser en rajoutant un **commentaire dans une note**.

Pour pouvoir accéder à un élément graphique de type **Note**, il faut commencer par cliquer sur l'onglet **Annotation** dans la **Toolbox** (boîte à outils à gauche de votre écran).



Vous pourrez alors insérer par un clic une note sur le diagramme en ayant au préalable sélectionné l'icône .

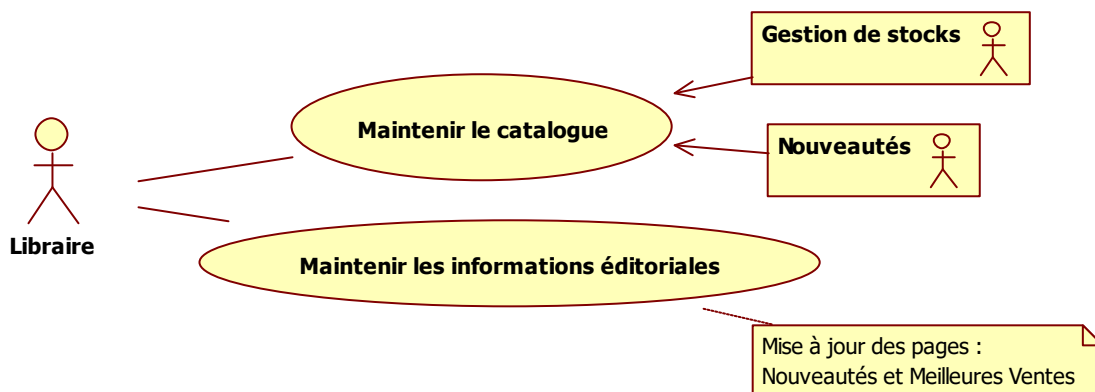
Entrez votre commentaire puis validez.

Mise à jour des pages :
Nouveautés et Meilleures Ventes

Pour relier ensuite la note à l'élément souhaité du diagramme, il faut utiliser un **NoteLink** et procéder de la manière suivante :

Cliquez sur l'élément **NoteLink** dans la **Toolbox** (boîte à outils à gauche de votre écran).

Sur le diagramme de Use Case, cliquez sur l'élément qui doit être associé à la la note (dans notre cas, ce sera le Use Case *Maintenir les informations éditoriales*), puis tout en restant appuyé sur le bouton de la souris déplacez-vous vers la **Note**: une fois sur la note atteinte , relâchez le bouton de la souris : le lien doit maintenant être effectué entre l'élément du diagramme et la note et vous devez obtenir :



... Afin de continuer le tutoriel, n'oubliez pas de re cliquer sur l'onglet UseCase de la Toolbox pour faire réapparaître les éléments graphiques relatifs au diagramme de Use Case...

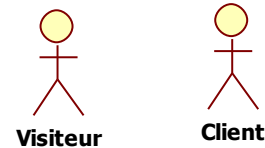
... Arrivé à ce point du tutoriel, le diagramme de Use Case que vous venez de créer correspond au diagramme de use case obtenu lors de la première itération c-a-d celui présenté dans le transparent du cours intitulé : Identification des cas d'utilisations de bookinons: première passe ...

... Nous allons donc maintenant considérer que nous entrons dans la seconde itération et affiner notre diagramme de Use Case ...


1.7 Création d'un Acteur généralisé

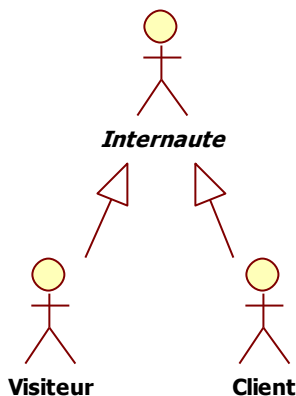
Nous allons modifier notre diagramme pour faire apparaître l'acteur généralisé abstrait **Internaute**.

Commencer par créer deux nouveaux acteurs sur le diagramme que vous appellerez **Visiteur** et **Client**.



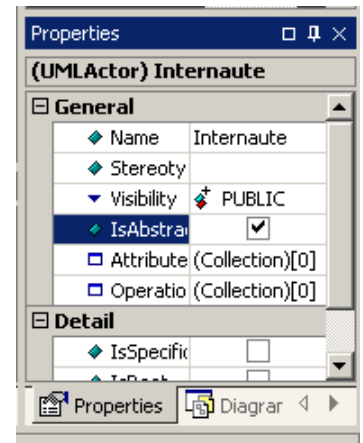
L'acteur **Internaute** doit maintenant devenir la généralisation des rôles **Client** et **Visiteur**.

Pour rajouter une relation de généralisation entre **Client** et **Internaute**, cliquez sur l'icône  **Generalization** de la toolbox, puis cliquez sur l'acteur **Client** et sans relâcher faites glisser jusqu'au super acteur **Internaute**. Faites de même entre **Visiteur** et **Internaute** afin d'obtenir la représentation suivante :



L'acteur **Internaute** peut être vu comme une généralisation abstraite des rôles **Client** et **Visiteur**. Pour faire apparaître ce caractère "abstrait" sur le diagramme, nous devons modifier les propriétés de l'acteur **Internaute**.

Sélectionner d'un clic l'acteur **Internaute** et consulter les propriétés de cet élément dans la fenêtre en bas à droite de votre écran. Il suffit de cocher la propriété **isAbstract** pour que le nom de l'acteur apparaisse désormais en italique dans votre diagramme.



Conformément à l'étude faite en cours pour enrichir le diagramme de Use Case :

⇒ Un **Internaute**, qu'il soit **Client** ou **Visiteur** peut :

- soit Rechercher un ouvrage,
- soit Gérer son panier

⇒ Un **Visiteur** peut en plus :

- Créer un compte Client

⇒ Un **Client** peut en plus :

- Effectuer une commande,
- Consulter ses commandes
- Gérer son compte client

Vous devez donc commencer par supprimer du diagramme de Use Case les associations reliant l'acteur **Internaute** aux uses cases : **effectuer une commande**, et **consulter ses commandes**

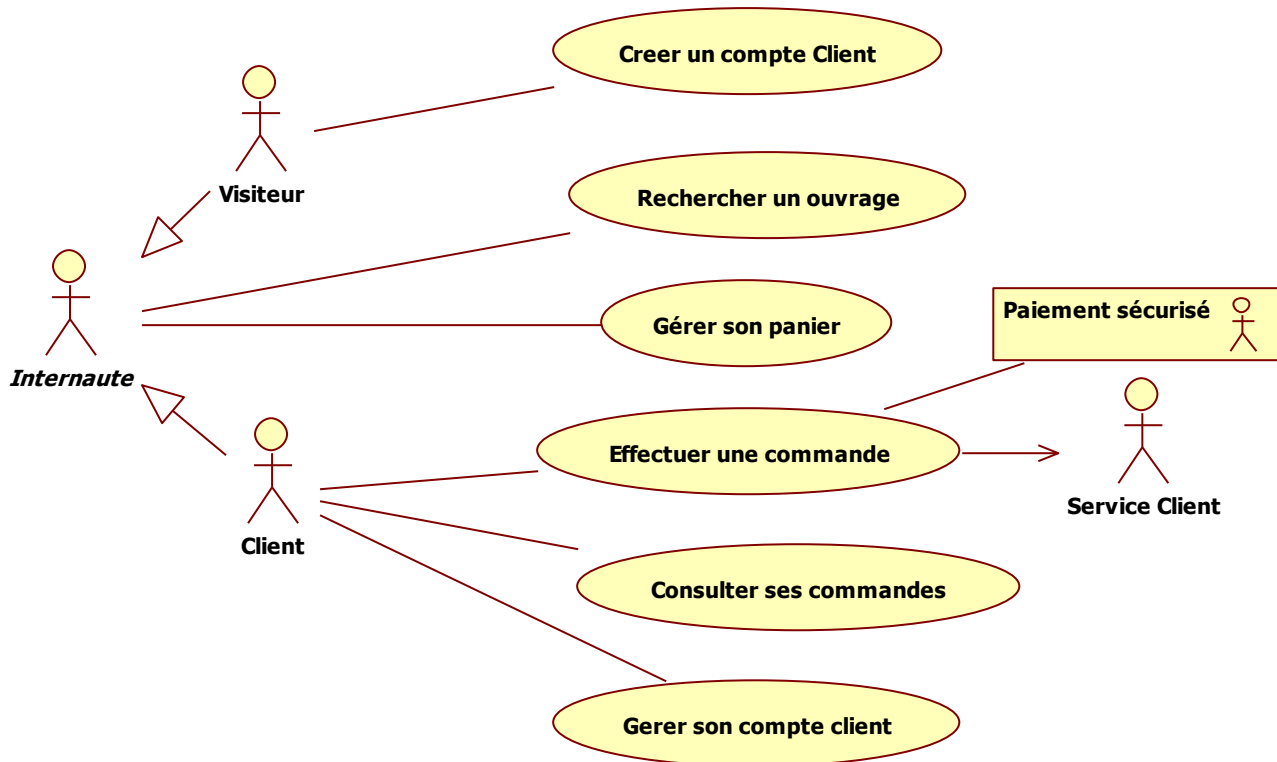


Veillez à bien utiliser une suppression par **Delete from Model** pour supprimer définitivement ces 2 associations de votre modélisation...

Vous rajouterez ensuite un Use Case **Créer un compte Client** qui devra être associé au **Visiteur**.

En ce qui concerne le **Client**, il sera associé aux deux uses cases déjà existants, à savoir **effectuer une commande**, **consulter ses commandes**, ainsi qu'à un troisième use case nouvellement créé et nommé **Gérer son compte client**

Votre bout de diagramme de Use Case concernant les acteurs **Internaute**, **Visiteur** et **Client** doit maintenant être conforme à la copie d'écran suivante afin de continuer le tutoriel :




1.8 Relation d'inclusion entre use case

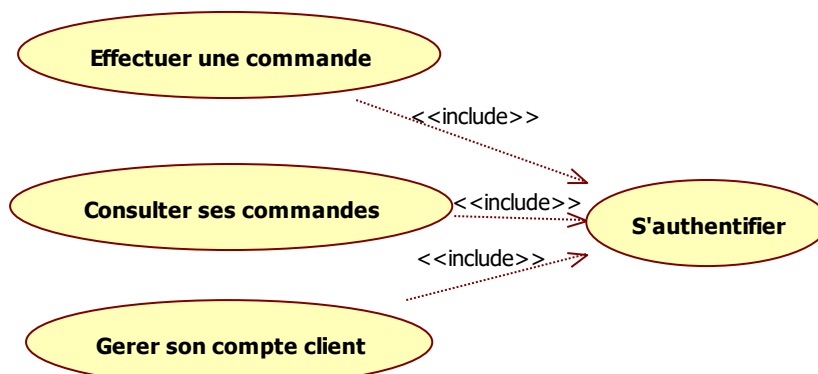
Rappelons que dans le cas d'une inclusion, le cas d'utilisation source comprend le comportement décrit par le cas d'utilisation destination de manière obligatoire .

Dans l'étude de cas **bookinons**, pour pouvoir **Effectuer une commande**, **Consulter ses commandes** ou **Gérer son compte**, le **Client** doit obligatoirement **S'authentifier** au préalable .

Commencez par rajouter sur votre diagramme, un Use Case **S'authentifier** .

Pour rajouter une relation d'inclusion entre **Effectuer une commande** et **S'authentifier**, Cliquez sur l'icône **Include** de la toolbox , puis cliquez dans le diagramme sur le Use Case source **Effectuer une commande** et sans relâcher faites glisser jusqu'au use case destination **S'authentifier**.

Faites de même avec les use case **Consulter ses commandes** et **Gérer son compte** afin d'obtenir la représentation suivante :




1.9 Relation d'extension entre use case

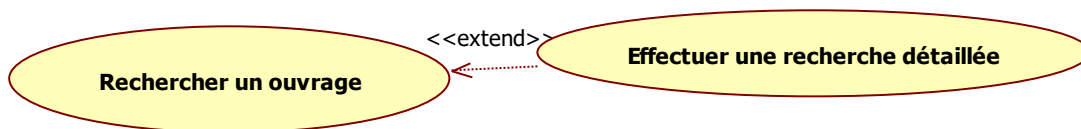
Rappelons que dans le cas d'une extension, le cas d'utilisation source ajoute son comportement au cas d'utilisation destination de manière optionnelle.

Dans l'étude de cas **bookinons**, **Rechercher un ouvrage** peut donner lieu à **Effectuer une recherche détaillée** c-a-d que la recherche détaillée peut être une option dans la recherche d'un ouvrage...

Commencez par rajouter sur votre diagramme, un Use Case **Effectuer une recherche détaillée**.

Pour rajouter une relation d'extension entre **Effectuer une recherche détaillée** et **Rechercher un ouvrage**,

Cliquez sur l'icône **Extend** de la toolbox , puis cliquez dans le diagramme sur le Use Case source **Effectuer une recherche détaillée** et sans relâcher faites glisser jusqu'au use case destination **Rechercher un ouvrage** afin d'obtenir la représentation suivante :

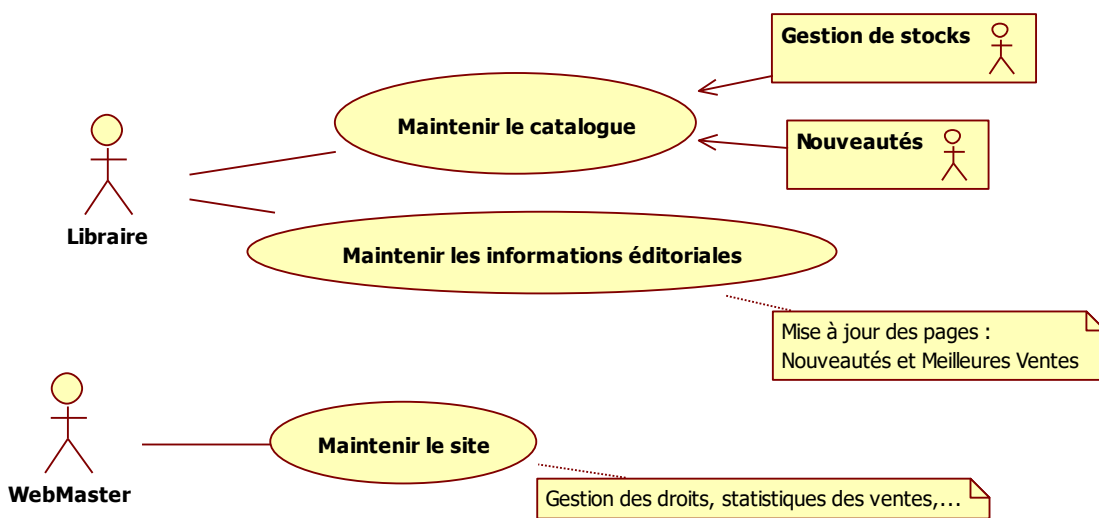
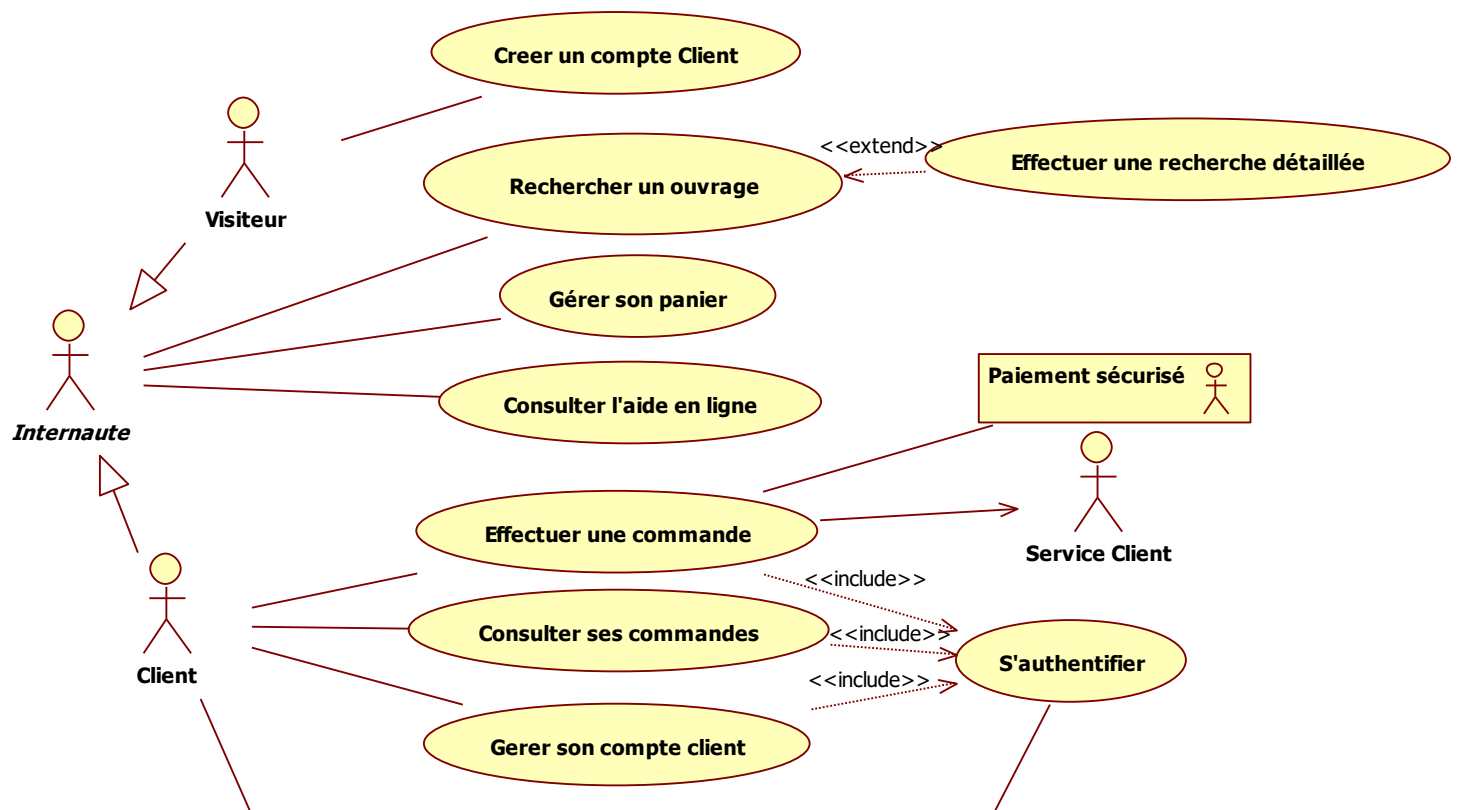


1.10 Version complète de la 2^{ème} passe du diagramme de use case de bookinons :

Lors d'une deuxième itération (2^{ème} passe), nous identifions également un use case oublié lors de la 1^{ère} passe à savoir **Consulter l'aide en ligne** dont l'acteur est l'**Internaute**.

Il ne vous faut pas non plus oublier d'indiquer par une *relation d'association* sur votre diagramme que l'acteur du Use Case **S'Authentifier** est l'acteur **Client**.

Une fois ces modifications effectuées, vous devez obtenir un diagramme de Use Case conforme à la représentation suivante :



1.12 Organiser la vue scénario en package ...

UML propose le concept de **paquetage** (**package** en anglais). Le package est un mécanisme général qui permet de regrouper des éléments UML

Il est ainsi possible de **regrouper les UC en package** en les classant *par ensemble fonctionnel*.

Par exemple, dans *bookinons* :

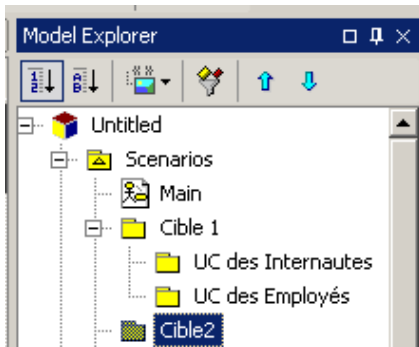
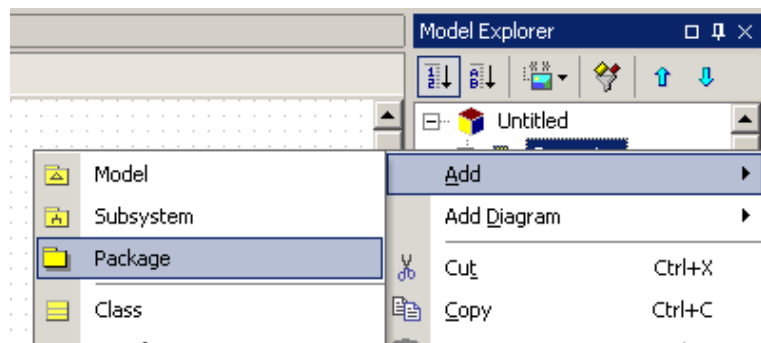
- plusieurs UC concernent tout ce qui est relatif à l'**Internaute** (front-office)
- plusieurs UC concernent et une partie propre aux **Employés** (back-office)

Par ailleurs, il est également possible de créer un package pour les UC que nous qualifierons de « **UC de second rang** » ou de « **UC de seconde cible** » qui ne représentent pas une intention métier à part entière du client (« **UC majeur** » ou « **UC de première cible** »), mais plutôt un niveau intermédiaire comme pour l'UC *S'authentifier* qui permet au client d'exécuter ses propres UC majeurs.

Pour notre étude de case, nous choisissons de regrouper les UC selon leur ordre de priorité par rapport aux intentions métier des acteurs :

- le package **Cible 1** contiendra 2 autres packages :
 - **UC des Internautes** et
 - **UC des employés**.
- le package **Cible 2** contiendra directement les cas d'utilisations : Consulter l'aide en ligne et s'authentifier.

Pour mettre en place cette arborescence de packages, nous allons commencer par créer le package **Cible 1**. Pour cela, placez-vous sur **Scenarios** dans le **Model Explorer**, puis d'un clic droit, sélectionnez le menu **Add**, puis cliquer sur **Package** et entrer le nouveau nom du package à savoir **Cible 1**

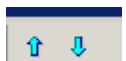


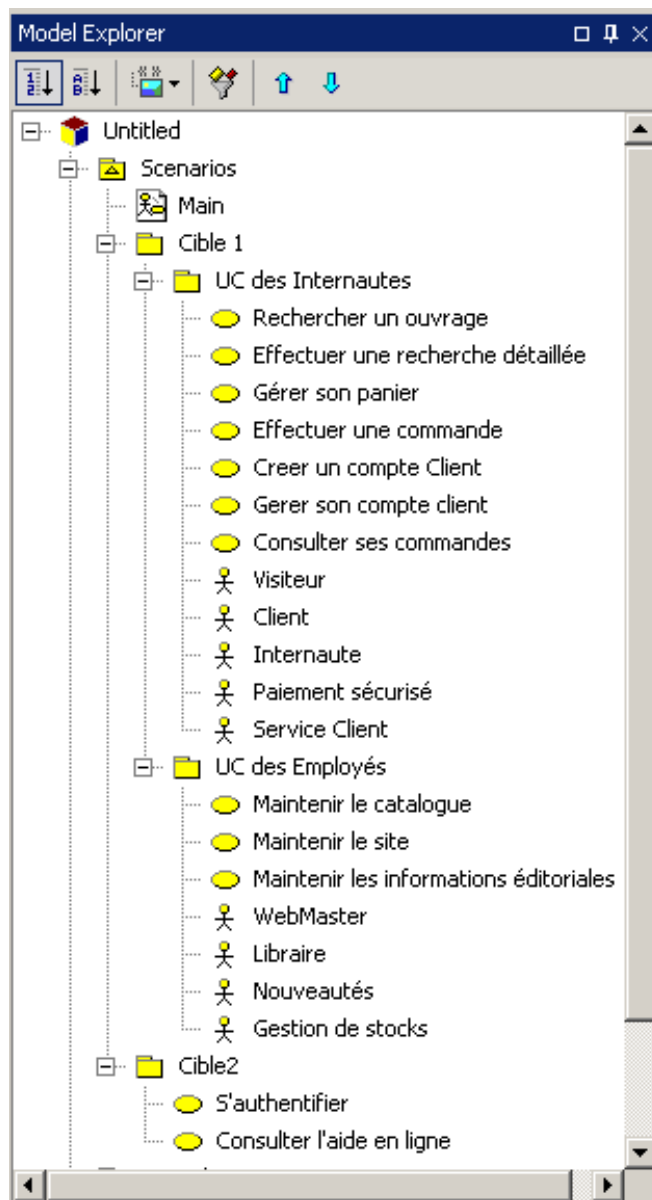
De la même manière à partir du package **Cible 1**, créer deux nouveaux packages **UC des Internautes** et **UC des Employés**.

Enfin, au même niveau que le package **Cible 1**, créer un package **Cible 2**.

Il ne vous reste plus qu'à déplacer les UC dans les packages concernés afin d'obtenir une répartition conforme à la copie d'écran suivante. Peu importe pour le moment l'ordre des UC dans les packages.

Remarque : Si vous souhaitez classer les éléments apparaissant dans le **Model Explorer** dans un certain ordre (par exemple celui dans lequel vous allez les traiter), vous pouvez utiliser les flèches bleues pour faire "remonter" ou "descendre" un élément à la position souhaitée.





1.13 Renommer le diagramme de Use Case

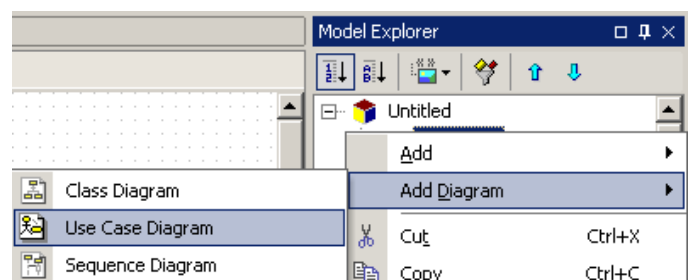
Pour finir, nous allons maintenant renommer notre diagramme de Use Case qui pour l'instant porte le nom de **Main** comme l'indique la copie d'écran ci-dessus du **Model Explorer**.

Pour cela, cliquez dans le **Model Explorer** sur **Main** correspondant au diagramme de Use Case. Vous pouvez alors renommer le diagramme en allant changer la propriété **Name** de la fenêtre **Properties** (en bas à droite de l'écran).

Appelez par exemple ce diagramme : **Diagramme Global** car il contient l'intégralité de votre diagramme de Use Case.

En effet pour des raisons de commodités (meilleure visibilité, détails supplémentaires souhaités,...), vous pourriez être amené dans votre analyse à *créer d'autres diagrammes de Use Case* pour par exemple zoomer sur un point particulier en proposant des use case de granularité plus fine.

Il est tout à fait possible de créer de nouveaux diagrammes de Use Case dans la vue **Scénarios**. Pour cela, il suffit de se placer sur **Scenarios** dans le **Model Explorer**, puis d'un clic droit, sélectionnez le menu **Add Diagramm**, puis cliquer sur **Use Case Diagram**



Remarque : Dans le cadre de ce tutoriel, nous nous contenterons uniquement du **Diagramme Global**.

Rappelons que nous appuyons notre démarche pseudo-RUP sur *l'approche* « **vue 4+1** » qui contient : la **vue scénario (Scenarios)** ou **vue des cas d'utilisation** ainsi que :

- la **vue logique (Logical View)**
- la **vue implantation (Development View)** ou **vue de réalisation**
- la **vue déploiement (Physical View)**
- et la **vue processus (Process View)**



4+1 View Model

Quelques mots de rappel sur la vue Scénario ...

La **vue Scénario** (celle des cas d'utilisations) que nous venons d'enrichir avec le **diagramme de use case** est la vue de référence : toutes les autres vues dépendent de celle-ci.

La vue scénario va permettre de partager les informations collectées (acteur et use case) avec les autres vues. En pratique, on rattache énormément de documentation à la vue scénario comme le cahier des charges ou la description détaillée des use case,... La vue scénario est donc plutôt abstraite que technique.

...Avant de passer à la vue Logique ...

Le passage à la **vue Logique** permet de mettre en route les premières réflexions concrètes sur le futur logiciel. En s'appuyant sur plusieurs diagrammes dynamiques, la vue logique propose une série de vues du système d'information d'un **niveau d'abstraction très élevé**.

La mise en place de la **vue Logique** se mène *en permanence* par rapport aux découvertes faites dans la **vue Scénario**. **Les deux vues s'enrichissent mutuellement** et il ne faut **pas hésiter à passer de l'une à l'autre, à modifier l'une ET l'autre tout en avançant...**

Exercice 2 : Axe dynamique : la vue Logique et les Diagrammes d'activité ...

En ce qui concerne le diagramme d'activité, il devra être réalisé une fois les premiers cas d'utilisation identifiés, et une fois la maquette et la description détaillée d'un UC réalisées.

Sachant que le diagramme d'activité permet de représenter graphiquement les enchaînements des activités, nous avons décidé dans notre démarche de **réaliser un diagramme d'activité par cas d'utilisation** afin d'illustrer graphiquement les enchaînements des actions décrites dans un UC. Ainsi, il sera possible d'identifier d'un seul coup d'œil **toute la famille des scénarios d'un cas d'utilisation** c-a-d tous les flots sur un même diagramme.

Le diagramme d'activité doit donc être construit à partir de la description détaillée textuelle.

Mais bien souvent, en construisant le diagramme d'activité, on va s'apercevoir qu'il manque des étapes dans la description détaillée (des actions dans le diagramme). La description détaillée devra donc bien souvent être modifiée en même temps que le diagramme d'activité sera élaboré.

C'est ainsi que les 2 points de vue (textuel et graphique) sur le UC permettront d'enrichir mutuellement les 2 **vues (Scénario et Logique)** sur le UC.

2.1 Organisation de la vue logique : un package par Use Case :

Comme nous venons de le dire, 1 diagramme d'activité sera associé à chaque UC de la modélisation. De même, plus tard, 1 diagramme de séquence et 1 diagramme de classes participantes seront associés à chaque UC de la modélisation.

Il est donc nécessaire d'organiser, dès maintenant, la **vue logique en packages, afin de pouvoir regrouper dans chaque package tous les diagrammes communs au même UC**.

La création d'un package n'est pas obligatoire, mais elle permet une meilleure lisibilité de la conception.

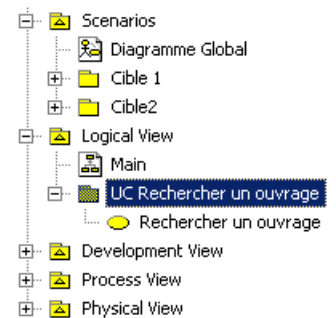
Pour mieux suivre l'évolution de notre modélisation, nous créerons les packages au fur et à mesure des traitements des UC dans la **Logical View**.

Nous allons commencer par traiter le **UC Rechercher un ouvrage**.

Dans **Logical View** du **Model Explorer**, ajoutez un package que vous pouvez nommer: **UC Rechercher un Ouvrage**.

Remarque : Si vous le souhaitez, vous pouvez même déplacer l'élément **Rechercher un ouvrage** de type Use Case du package **Internaute** de la **Cible 1** de la vue **Scénario** dans votre nouveau package.

Ainsi lorsque tous les UC de la **Cible 1** auront été déplacés, cela signifiera qu'ils auront été traités dans la **Logical View** et que vous pouvez passer aux paquetages de la **Cible 2**...

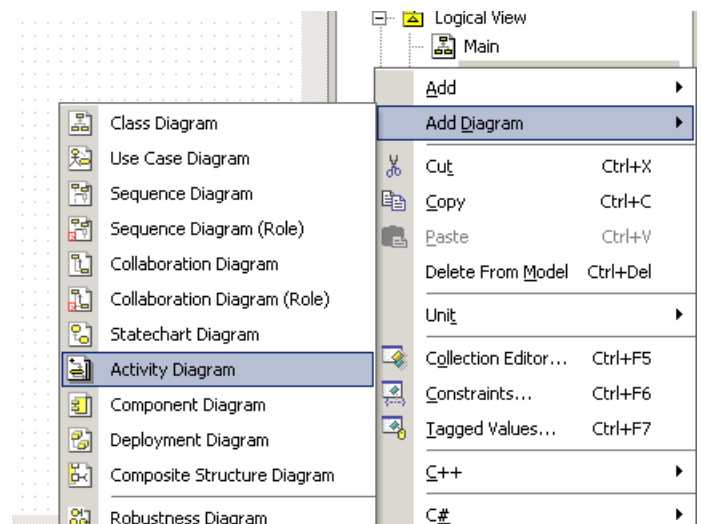


2.2 Création du diagramme d'activité

Tout d'abord, vous devez ajouter un diagramme d'activité dans le package **UC Rechercher un ouvrage** que vous venez de créer. La création d'un diagramme d'activité (comme tout autre diagramme) peut se faire de 2 manières différentes :

- soit directement depuis le **Model Explorer** par un clic droit effectué depuis le package **UC Rechercher un ouvrage** afin de sélectionner le menu **Add Diagramm**, puis de cliquer sur **Activity Diagram**

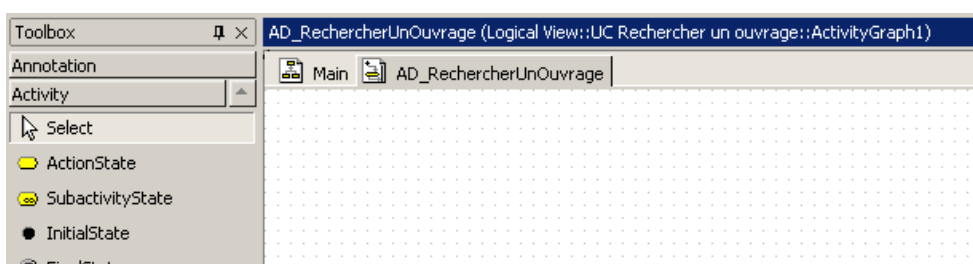
Nommez ce diagramme par exemple : **AD_RechercherUnOuvrage**



- soit en sélectionnant le package **UC Rechercher un ouvrage** dans le **Model Explorer**, puis en choisissant dans la barre d'outils standard en haut de votre écran le menu : **Model** → **Add Diagramm** → **Activity Diagram**

Lorsque vous créez un diagramme, il devient à sa création **le diagramme actif de la fenêtre de travail**.

C'est ce qui s'est produit avec le diagramme d'activité **AD_RechercherUnOuvrage** : vérifiez que le bandeau de votre fenêtre de travail indique **AD_RechercherUnOuvrage** et constatez que la palette graphique de la **Toolbox** propose maintenant un onglet **Activity** où sont disponibles les représentations graphiques représentant les éléments d'un diagramme d'activité UML.



2.3 Construction du flot de base dans le diagramme d'activités

Grâce à la palette graphique **Activity** de la **Toolbox**, nous allons pouvoir commencer la représentation du diagramme d'activité.

Nous allons commencer par représenter le flot de base de la description détaillée:

2.1 - Flot de base (ou flot nominal) :

<Recherche d'un ouvrage à partir de mots clés>

- 2.1.1 Le système recherche les informations relatives à la recherche (Nouveautés, Meilleures ventes,...) et affiche l'écran de recherche ...
- 2.1.2 L'Internaute saisit un ou plusieurs mots-clés (un thème, un titre, un auteur, un nom d'auteur) et valide.
- 2.1.3 Le système recherche dans le catalogue les ouvrages pouvant correspondre à la demande de l'utilisateur
- 2.1.4 Le système trie les ouvrages dans l'ordre souhaité.
- 2.1.5 Le système affiche dans une page de résultat un résumé des ouvrages trouvés.
- 2.1.6 L'Internaute sélectionne un ouvrage
- 2.1.7. Le système recherche le détail de l'ouvrage
- 2.1.8 Le système affiche une fiche détaillée de l'ouvrage qui contient :
 - une image de l'ouvrage, le titre, l'auteur
 - l'éditeur, l'isbn, la langue, la date de parution
 - le prix et la disponibilité

2.3.a Etat initial

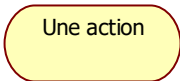


Un scénario nominal commence toujours *par un état initial*.

Pour positionner un état initial sur le diagramme, il suffit de sélectionner un **Initial State** dans le toolbox **Activity** et de venir cliquer sur le diagramme à l'endroit souhaité.

Vous pouvez renommer si vous le souhaitez votre élément à partir de la propriété **Name** proposée par le fenêtre **Properties** : ce nom est seulement pris en compte dans le **Model Explorer** et n'apparaît pas sur le diagramme.

2.3.b Action :



Une action

Un scénario est une séquence d'**actions**. Une **action** est une unité de traitement très petite qui a une incidence sur le système (modification de l'état, récupération d'une information...)

Les actions sont des étapes autour desquelles on détermine des comportements, ainsi on compte différents types d'action : affecter des valeurs à des attributs, accéder à des valeurs (attributs par exemple), créer des liens, effectuer des calculs, émettre ou envoyer un signal...

La première action du flot de base correspond à la première ligne c-a-d :

2.1.1 Le système recherche les informatives relatives à la recherche (Nouveautés, Meilleures ventes,...) et affiche l'écran de recherche ...

Pour représenter cette action sur le diagramme UML, nous allons tout d'abord sélectionner un **Action State** dans le toolbox **Activity** et de venir cliquer sur le diagramme en dessous de l'état initial. Transcrire la phrase précédente, vous pouvez appeler cette activité :

Charger Nouveautés, Meilleures Ventes et afficher écran de recherche

2.3.c Transition

Il faut maintenant relier par une transition l'état initial à l'activité.

Sélectionner **Transition** dans le toolbox **Activity**, cliquer sur le diagramme sur l'état initial, faites glisser jusqu'à l'action précédente et relâcher.

Une transition sous forme de flèche est alors dessinée.

Cette transition est automatique dès que l'état initial est fini : elle n'a donc pas besoin de déclencheur.

2.3.d Etat final



Un scénario nominal finit toujours *par un état final*

Pour positionner un état initial sur le diagramme, il suffit de sélectionner un **Final State** dans le toolbox **Activity** et de venir cliquer sur le diagramme à l'endroit souhaité.

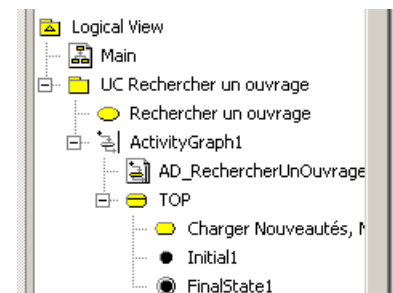
Remarques :

↳ Un diagramme d'activité détaillant un Use Case en représentant la chronologie des opérations réalisés par un acteur :

- aura en général *un seul état initial* qui sera le point de départ du diagramme
- pourra avoir *un ou plusieurs état(s) final (finaux)*

↳ Les éléments ajoutés au diagramme d'activité sont stockés dans le **Model Explorer** dans un état **TOP** positionné au même niveau que le diagramme d'activité. Si vous souhaitez modifier une propriété d'un des éléments du diagramme, vous pouvez pour accéder à la fenêtre **Properties** de l'élément de deux manières :

- soit en double-cliquant directement sur l'élément sur le diagramme
- soit en double-cliquant sur le nom de l'élément sur le **Model Explorer**



2.3.d Représentation complète du flot de base :

Pour vous entraîner à manipuler les *actions* et les *transitions*, commencez par construire dans le diagramme d'activités correspondant au flot de base.

Reprendre un par un les points du flot de base et transformer les en action.

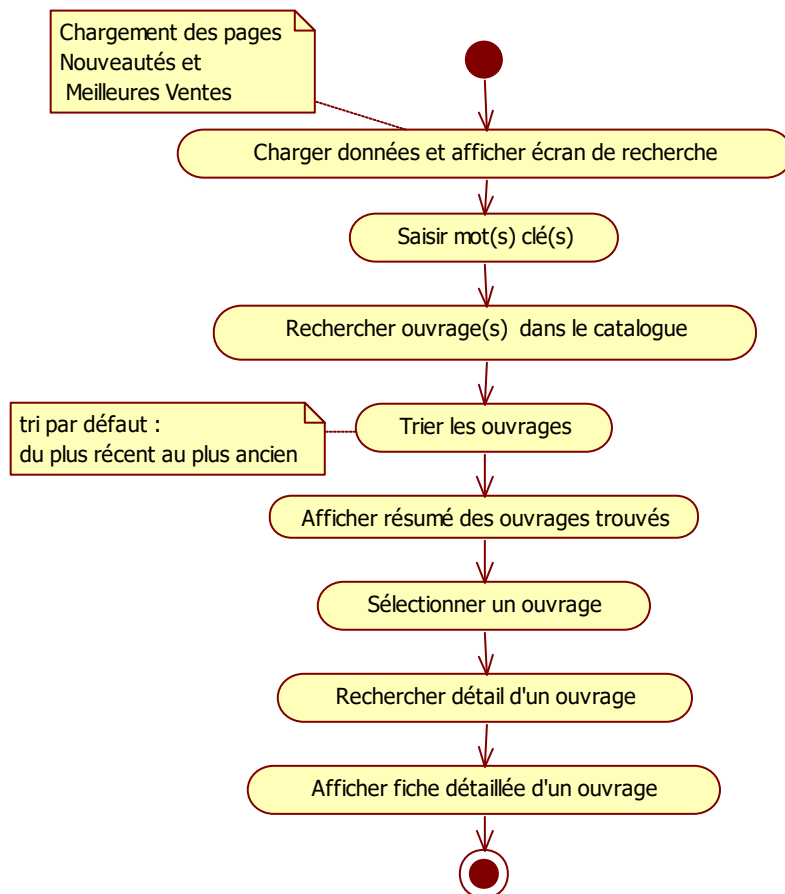
Vous devriez obtenir un diagramme conforme au diagramme ci-après :

Peu importe si vos noms d'actions ne sont pas les mêmes que celles du diagramme proposé ici, ce qui est important c'est que vos actions soient assez explicites pour qu'un client non informaticien puisse lire votre diagramme...et vos noms d'actions ne doivent pas être trop longs pour permettre une meilleure lisibilité (vous avez bien sûr remarqué que l'on a rajouté une note sur la première ...)

Votre diagramme sera d'autant plus facilement lisible, qu'il sera bien présenté : n'oubliez pas que votre diagramme d'activité sera sûrement montré au client.

Ainsi, pour obtenir un "bel alignement verticalement" du flot de base comme sur la copie d'écran ci-dessous, il vous suffit de sélectionner toutes les activités et les transitions du flot de basé et de procédez à l'alignement vertical en utilisant l'icône suivante:





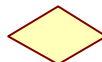
2.4 Ajout d'un flot alternatif dans le diagramme d'activités

Nous allons maintenant représenter sur le diagramme le premier flot alternatif de la description détaillée, à savoir :

2.2.1 - Flot Alternatif 1 : < Recherche avancée d'un ouvrage >

*Le flot alternatif démarre après le point 2.1.1 du flot de base
 2.2.1.1 L 'Internaute choisit d'effectuer une recherche avancée
 2.2.1.2 Appel au UC Effectuer une recherche détaillée
 Le flot de base reprend au point 2.1.3*

2.4.a Condition (Decision)



Pour **modéliser une alternative** sur le diagramme d'activité, on va utiliser une **décision** qui va nous permettre de mettre en place **un branchement conditionnel** sur le flot de base.

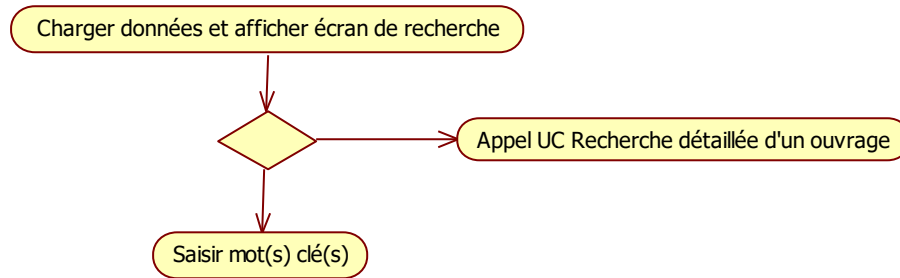
La recherche avancée peut être effectuée une fois la page d'écran affichée (**activité Charger données et afficher écran de recherche**) et à la place de la recherche par mot clé (**activité Saisir mot(s) clé(s)**)

Dans un premier temps, la transition entre ces deux premières actions du diagramme doit être supprimée, puisque **cette transition n'est plus automatique**, mais **conditionnelle** puisque soumise à la **décision** que prendra l'utilisateur pour réaliser sa recherche. Pour supprimer la transition, sélectionner la puis par un clic droit choisir **Edit → Delete from Model**

Une **condition** sera modélisée sous starUML par un élément **Decision**.

Pour positionner une condition sur le diagramme, il suffit de sélectionner **Decision** dans la toolbox **Activity** et de venir cliquer sur le diagramme à l'endroit souhaité entre les deux actions.

Vous devez ensuite ajouter une transition de la *première action* vers la **Decision**, puis une autre transition de la **Decision** vers la *seconde action*, puis créer une nouvelle action **Appel UC Rechercher Détaillée** (correspondant à l'action 2.2.1.2 du flot alternatif) et rajouter une transition entre la **Decision** et cette *nouvelle action* diagramme, afin d'obtenir un bout de diagramme similaire à celui ci-après :



Il est maintenant nécessaire de rajouter un peu de texte afin d'explicitier **quelles sont les conditions** qui vont permettre de la *bonne décision* vers la *bonne transition*.

Cliquer sur l'élément **Decision** du diagramme afin d'ouvrir sa fenêtre **Properties**.

Vous pouvez commencer par donner un nom à cette condition en utilisant la propriété **Name**, par exemple : **Recherche Classique ?**

On pourrait s'attendre à ce que ce nom apparaisse sur le diagramme... et bien non, il apparaît juste dans le **Model Explorer**.

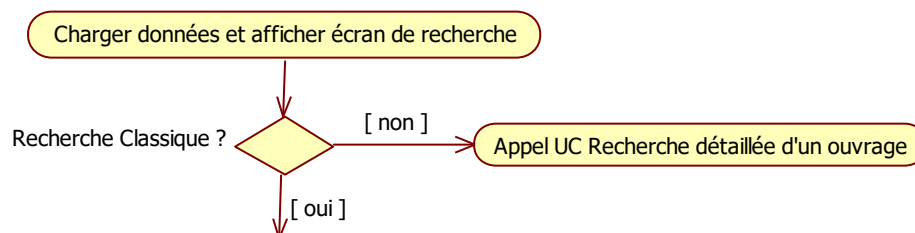
Pour faire apparaître **Recherche Classique ?** sur le diagramme, deux solutions s'offrent à nous :

- soit rajouter un simple texte sous forme de commentaire en le positionnant à côté de la condition. Pour écrire un simple texte dans le diagramme, il suffit de choisir à partir de la toolbox **Annotation** un élément de type **Text**.
- soit nommer la première transition en double cliquant dessus et en remplissant la zone de texte qui s'ouvre ou ce qui revient au même en la sélectionnant d'un clic et en remplissant sa propriété **Name** avec par exemple : **Recherche Classique ?**

Il faut ensuite faire apparaître les valeurs des **conditions de garde** pour savoir quelle transition est choisie lorsque la **Decision** est rencontrée.

Cliquer par exemple sur la transition entre la **Decision** et l'action **Saisir mot(s) clé(s)** et entrer dans la propriété **GuardCondition** de la fenêtre **Properties** : **oui** et valider avec entrée ou cliquer quelque part ailleurs. La condition de garde **[oui]** apparaît sur le diagramme. Cela signifie que cette transition ne sera franchie que si la réponse à la condition **Recherche Classique ?** est positive. On parle alors de "transition gardée" (notion que l'on reverra dans les diagrammes d'états-transitions)

Ajouter également une garde à **[non]** vers l'action du flot alternatif à savoir la recherche détaillée d'un ouvrage.



Remarques :

↳ La description détaillée du flot alternatif indiquait : "*Le flot alternatif démarre après le point 2.1.1 du flot de base*", sur le diagramme d'activité, on retrouve bien le branchement qui nous amène au flot alternatif *après la première activité*.... Il y a donc bien **cohérence et conformité entre le diagramme d'activité et la description détaillée**.

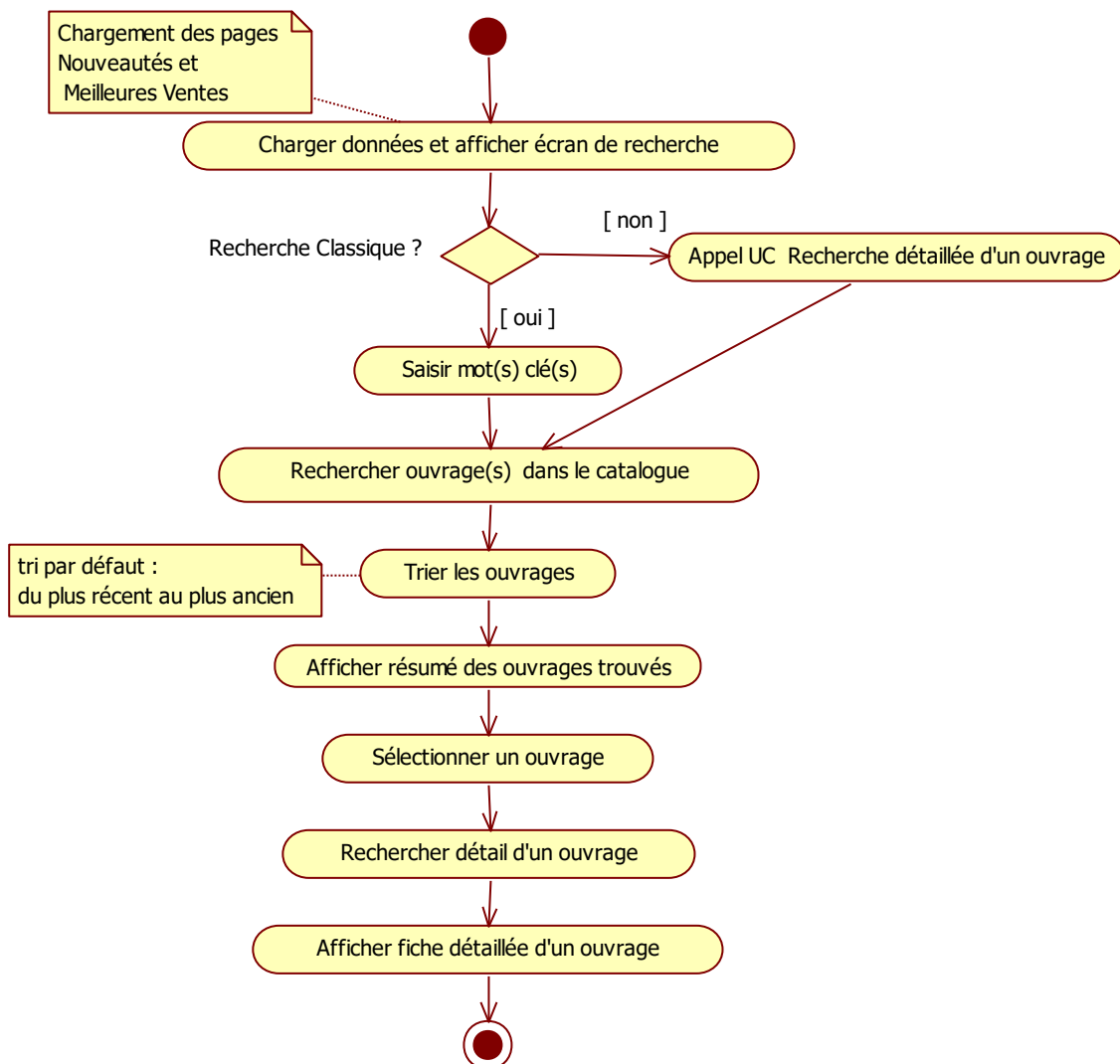
↳ La première étape du flot alternatif "*L'Internaute choisit d'effectuer une recherche avancée*" est bien représentée sur le diagramme par le Decision.

2.4.b Fin du premier flot alternatif : retour dans le flot de base

Ce premier flot alternatif se termine après l'appel à l'UC Effectuer une recherche détaillée par : "*Le flot de base reprend au point 2.1.3*"

Il faut donc retrouver dans le diagramme d'activités une transition vers l'action correspondant au point "*2.1.3 Le système recherche dans le catalogue les ouvrages pouvant correspondre à la demande de l'utilisateur.*"

Rajouter cette transition de manière à obtenir un diagramme d'activité similaire au diagramme ci-après qui montre comment le scénario alternatif *se branche* sur le scénario nominal.



2.5 Représentation complète de la famille de flots dans le diagramme d'activité

Vous devez maintenant reprendre toute la description détaillée (document distribuée en cours d'UML) afin de représenter sur votre diagramme d'activité **l'ensemble des flots (de base et alternatifs) pour modéliser graphiquement la description détaillée complète du use case Rechercher un ouvrage.**

Une fois terminé, vérifiez que votre diagramme d'activités et que votre description détaillée soient bien cohérents et conformes par une lecture simultanée des deux vues.

Vous pouvez alors comparer votre diagramme d'activité au diagramme d'activité proposé sur la page suivante.

Et pour finir rappelons que pour *documenter des cas d'utilisation*, la *description textuelle est indispensable*, car elle permet de *communiquer facilement* avec les utilisateurs et de s'entendre sur la *terminologie métier* employé. En revanche, le texte présente des désavantages car il est difficile de montrer comment les enchaînements se succèdent, ou à quel moment les acteurs secondaires sont sollicités. Il est donc recommandé de compléter la description textuelle par un ou plusieurs *diagrammes dynamiques* UML.

En effet, comme vous avez pu le constater le diagramme d'activité permet de représenter graphiquement les enchaînements des activités au sein d'un même cas d'utilisation. Il est ainsi possible **d'identifier d'un seul coup d'œil la famille de tous les scénarios du cas d'utilisation** et d'envisager ainsi toutes les possibilités d'exécution offertes par ce Use Case.

