

**LO19**

**Tutorial Rotional Rose**

**Alexandre Penfornis  
Thomas Derive**

## ***Qu'est ce que Rational Rose ?***

Aujourd'hui, la réalisation d'un logiciel doit se faire dans des délais de plus en plus courts et doit permettre de pouvoir revenir dans le code pour effectuer diverses modifications dans les plus brefs délais pour répondre à la demande des testeurs ou encore à la demande du client utilisateur. Ce gain en temps et en modularité doit passer par une étape de conceptualisation du logiciel, des objets et des relations entre les objets manipulés par le programme. La génération de code n'est plus l'étape essentielle, le code est un outil technique permettant de répondre aux besoins conceptuels. Ainsi UML (Unified Modeling Language) est un outil qui permet aux concepteurs de logiciels de représenter les objets répondant au problème et les méthodes qui permettront la manipulation de ces objets. Rational Rose est un outil de modélisation d'application édité par la société IBM. Il permet entre autre la réalisation de diagrammes UML. Mais il permet également de couvrir toutes les étapes de la phase descendante du cycle en V de développement d'un programme jusqu'à l'étape de génération automatique de code dans le langage souhaité (C/C++, Visual Basic, Java, XML, Application web etc...).

## ***A qui s'adresse cet outil ?***

Rational Rose s'adresse aux concepteurs de logiciels principalement dans les étapes de modélisation du besoin et du comportement de l'application. Pour les étudiants utcéens qui suivent l'unité de valeur de Génie Logiciel LO19 (Walter Schön – Jean-Louis Boulanger), cet outil s'avère être un outil intéressant car il permet de réaliser tous les types de diagrammes vus en cours (Statecharts, Diagramme de classes, Use case, Diagrammes de collaboration, Diagrammes de séquence etc...) et bien sur d'aller encore plus loin dans la compréhension du génie logiciel.

## ***Remarque***

Ce tutorial a été réalisé dans le but premier d'aider les prochains étudiants en LO19 à utiliser les outils fournis par Rational Rose dans le cadre de l'UV. Les exemples présentés dans les différentes parties ont été choisis dans les transparents du cours afin de limiter les efforts de compréhension des diagrammes en favorisant leur réalisation avec Rational Rose.

Bonne découverte !

# Sommaire

---

|             |   |           |
|-------------|---|-----------|
| <b>I.</b>   | <b>DEMARRAGE DE RATIONAL ROSE .....</b>                     | <b>3</b>  |
| A.          | ENVIRONNEMENT DE TRAVAIL .....                              | 3         |
| i.          | <i>Barre d'outils .....</i>                                 | <i>4</i>  |
| ii.         | <i>Navigateur.....</i>                                      | <i>11</i> |
| iii.        | <i>Fenêtre de description/documentation.....</i>            | <i>13</i> |
| iv.         | <i>Espace de dessin.....</i>                                | <i>14</i> |
| v.          | <i>Fenêtre d'erreurs ou fenêtre de Log .....</i>            | <i>15</i> |
| B.          | CONFIGURATION DE L'ESPACE DE TRAVAIL.....                   | 16        |
| <b>II.</b>  | <b>EXEMPLES DE REALISATIONS SIMPLES.....</b>                | <b>19</b> |
| A.          | DIAGRAMME DE CLASSES .....                                  | 19        |
| i.          | <i>Ajout d'une classe .....</i>                             | <i>21</i> |
|             | Paramétrages de la nouvelle classe .....                    | 22        |
|             | Ajout d'un attribut.....                                    | 23        |
|             | Spécifications de l'attribut .....                          | 24        |
|             | Ajout d'une méthode .....                                   | 25        |
|             | Spécifications de l'attribut .....                          | 25        |
| ii.         | <i>Ajout d'une relation .....</i>                           | <i>28</i> |
|             | Mise en place d'une association.....                        | 28        |
|             | Spécifications de l'association .....                       | 28        |
|             | Association réflexive.....                                  | 29        |
|             | Classe-Association .....                                    | 30        |
|             | Qualification d'une association.....                        | 31        |
|             | Agrégation .....  | 31        |
|             | Composition .....   | 32        |
|             | Contraintes sur les associations.....                       | 33        |
| iii.        | <i>Généralisation/Spécialisation .....</i>                  | <i>34</i> |
| B.          | DIAGRAMME DE COLLABORATION.....                             | 34        |
| C.          | DIAGRAMME DE SEQUENCE .....                                 | 36        |
| D.          | USE CASE (DIAGRAMME DE CAS D'UTILISATION).....              | 39        |
| E.          | STATECHARTS .....   | 39        |
| <b>III.</b> | <b>OBSERVATIONS LIEES A NOTRE PROPRE UTILISATION.....</b>   | <b>42</b> |
| A.          | ASTUCES ET REMARQUES IMPORTANTES .....                      | 42        |
| B.          | DEFAUTS OBSERVES .....                                      | 43        |
| <b>IV.</b>  | <b>LIEN ET DOCUMENT INTERESSANTS SUR RATIONAL ROSE.....</b> | <b>44</b> |

---

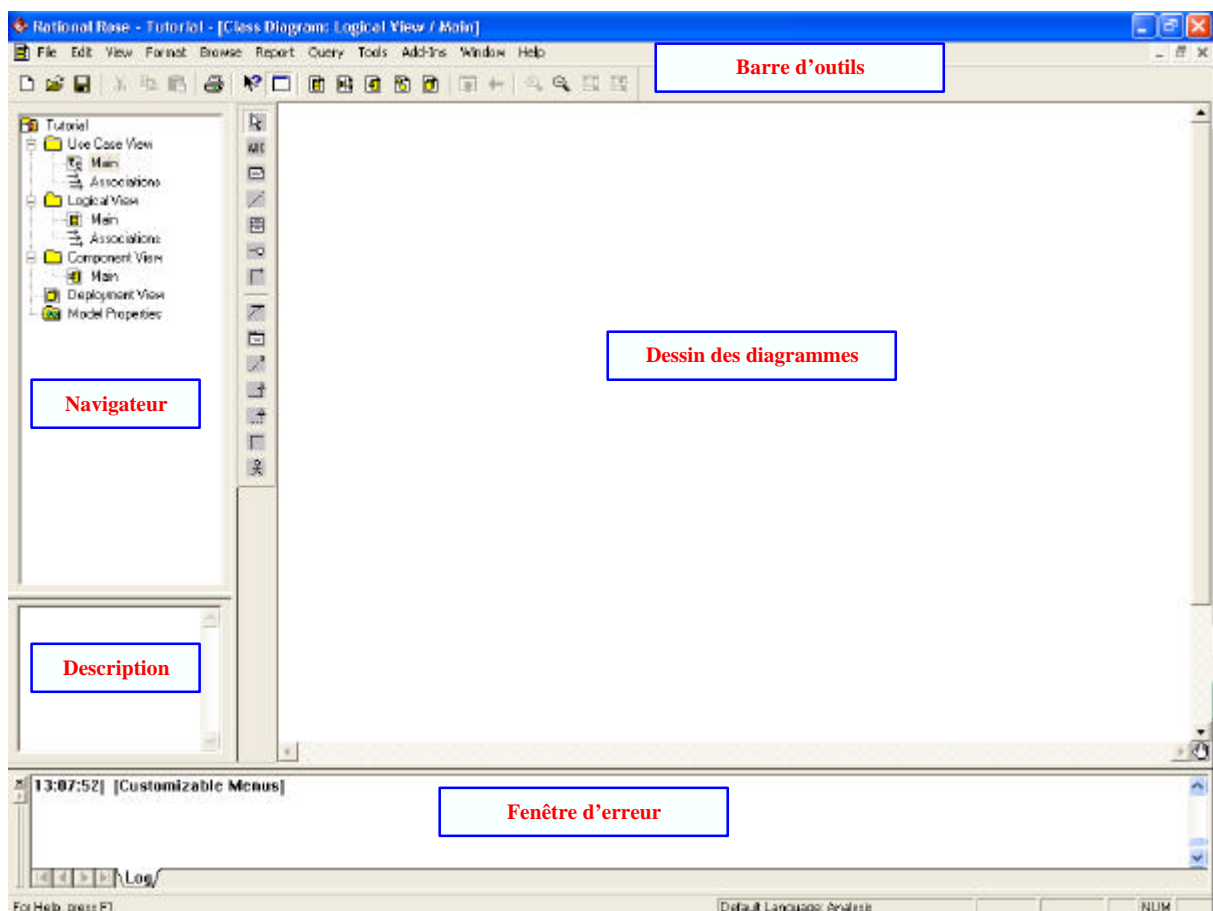
# I. Démarrage de Rational Rose

Rational Rose Entreprise Edition propose une gamme importante d'outils, celui utilisé en LO19 est l'outil de modélisation (Modeler Edition).

Lors de son lancement, une fenêtre s'ouvre pour demander à l'utilisateur quel ancien projet il désire ouvrir ou quel template il souhaite utiliser s'il démarre un nouveau projet. Nous conseillons de cliquer sur Annuler afin de démarrer un nouveau projet plus personnalisé.

## a. Environnement de travail

L'interface utilisateur principale de Rational Rose se compose comme suit :



On peut décomposer la fenêtre principale en quatre parties. Une **barre d'outils** qui fournit les fonctions usuelles d'un programme sous forme de boutons et de menus déroulant. Les boutons de la barre d'outils donnent accès aux fonctions les plus couramment utilisées par les utilisateurs en général, nous verrons plus loin que cette barre est totalement paramétrable selon les choix de l'utilisateur. Les menus déroulant donnent accès à toutes les fonctions du programme. La **fenêtre d'erreur** que l'on peut considérer comme une boîte de dialogue entre l'application et l'utilisateur. Une **boîte de description** des objets et un **navigateur** type explorateur Windows permettant d'accéder facilement aux différents fichiers et diagrammes composant le projet et de les organiser. Nous allons voir par la suite une description plus détaillée de ces quatre parties.

## i. Barre d'outils

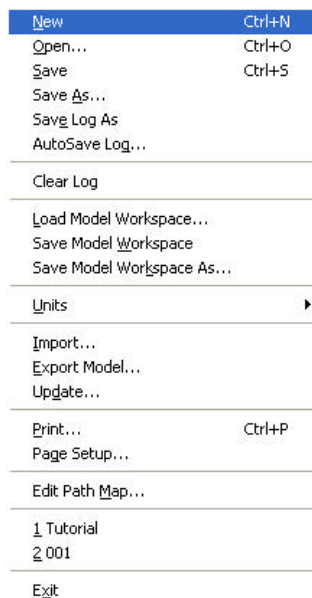
La barre d'outils est composée des menus déroulant, une barre d'outils générale horizontale et une barre d'outils spécifique au type de diagramme que l'utilisateur est en train de dessiner.

### ➤ Barre de menus



La barre de menu est affichée en permanence, elle donne l'accès à toutes les fonctions du logiciel. Ces fonctions sont grisées ou activées selon les éléments sélectionnés dans les autres fenêtres.

- Le menu **File** propose les fonctionnalités habituelles d'un programme :

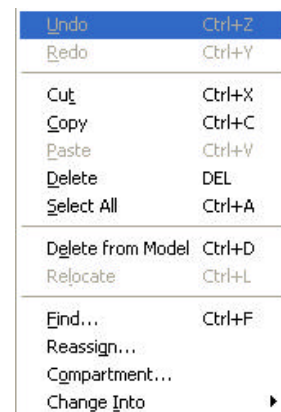


Les sous-menus sont les mêmes que ceux que l'on peut trouver dans d'autres logiciels comme un logiciel de traitement de texte. Ils permettent par exemple d'ouvrir un nouveau fichier ou projet (New) ou un ancien fichier (Open). Les fonctions de sauvegarde habituelles sont également présentes. Un seul fichier .mdl contient toutes les données concernant un projet. La commande **AutoSave Log...** permet la copie des informations contenues dans la fenêtre d'erreur dans un fichier à part. L'extension de ce fichier est .log.

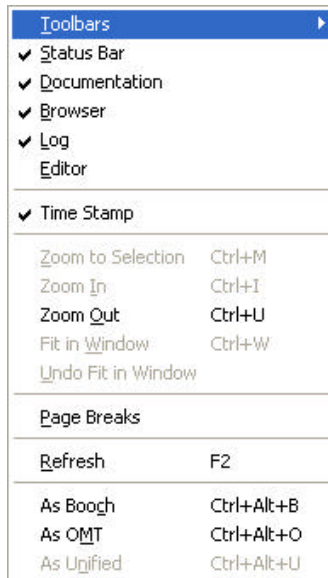
Comme d'habitude, la commande **exit** de sortie du programme est également présente dans le menu File. L'option **Edit Path Map...** permet de paramétrer manuellement les répertoires de travail.

- Le menu **Edit** comporte également les fonctionnalités usuelles (Copier/Coller...) :

Ce menu permet de faire des copier/coller – couper/coller d'un ou plusieurs éléments d'un diagramme par exemple. Il est important de noter ici que l'on réalise en fait un duplicata exact de l'objet d'origine et qu'une modification ultérieure d'un des deux éléments (l'ancien ou le nouveau) sera effective sur les deux objets. Le sous-menu **Change Into** permet de modifier les propriétés principales de l'objet sélectionné. Par exemple lorsqu'il s'agit d'une classe, il est possible grâce à cette action de la modifier en classe instanciée ou en classe paramétrée (l'affichage est alors différent).

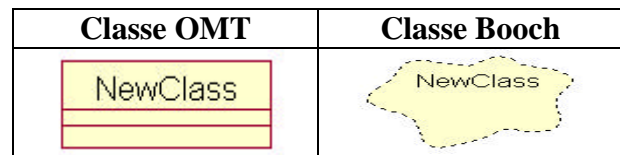


- Le menu **View** permet de spécifier les zones que l'on souhaite afficher



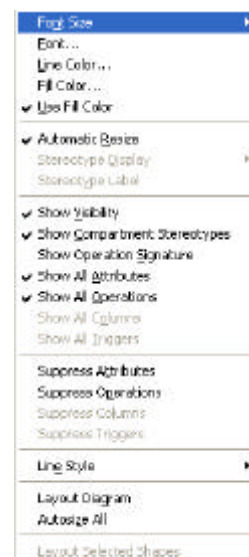
Ce menu permet d'abord de définir quelles parties de l'espace de travail l'utilisateur souhaite afficher. L'affichage des deux barres d'outils se règle dans le sous-menu **Toolbars**. On voit sur la figure ci-contre que lorsqu'une partie est visible un signe de coche est situé devant (par exemple, ici la Fenêtre d'erreur (Log) est marquée comme affichée. On note la présence dans ce sous-menu des options de zoom et dézoom très utiles lors de dessin de diagrammes dépassant la taille de l'écran de l'utilisateur. Le dernier bloc de fonction de ce menu permet de modifier les conventions d'affichage des diagrammes (par défaut les convention sont celles d'UML).

Par exemple l'affichage de classe en **OMT** sera différent de celui en **Booch**.



- Le menu **Format** permet de régler les paramètres d'affichage

Il permet par exemple de changer la police utilisée pour la dénomination des classes. Ou encore de changer la couleur de fond et les lignes de contours d'une classe particulière afin de la mettre en valeur. Pour augmenter la lisibilité de certains diagramme il est intéressant de ne pas visualiser les attributs et/ou les opérations associés à une classe ou l'autre, on peut donc les cacher grâce aux options **Supress Attibutes** et **Suppress Operations**.



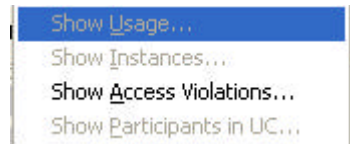
- Le menu **Browse** permet de naviguer dans le projet mais reste moins pratique que le navigateur

Ce menu permet par exemple de sélectionner quel type de diagramme on veut afficher dans la fenêtre de dessin. Sur l'image ci-contre on voit que **Class Diagram...** est coché, ce qui indique que le diagramme actuellement à l'écran dans la fenêtre principale est un diagramme de classes. Si on veut créer le diagramme de séquence ou de collaboration lié à ce diagramme

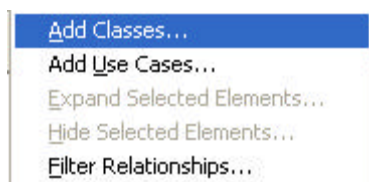
de classes, on pourra sélectionner **Interraction Diagram...**, pour le diagramme d'activité ou le statecharts, on pourra sélectionner **State Machine Diagram...**. Enfin on peut noter la possibilité dans ce menu d'ouvrir la fenêtre de spécification de l'objet en cours (nous verrons plus long que le clic droit ou encore le double clic sur l'objet en question semblent plus appropriés.

- Le menu **Report** permet d'avoir des informations sur les classes et leur utilisation dans tout un projet

**Show Usage** permet de lister tous les endroits où l'objet sélectionné est impliqué dans une relation. **Show Instances** montre toutes les instances d'un objet dans les différents diagrammes. **Show Access Violation** détermine s'il n'a pas été fait appel à une classe d'un autre projet sans importer sa définition et enfin **Show Participants in UC** liste tous les participants d'un Use Case en donnant par exemple les opérations réalisées par cet objet.



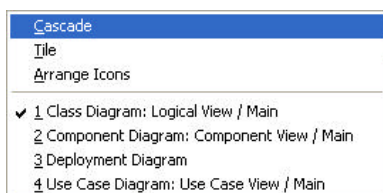
- Le menu **Query** permet de paramétrer quels éléments doivent être visibles ou non dans le diagramme courant



L'option que l'on trouve particulièrement intéressante ici est celle qui permet de filtrer les relations entre classes. Si l'on veut faire disparaître toutes les relations de type généralisation spécialisation pour ne garder que les associations, on peut facilement les faire disparaître en sélectionnant **Filter Relationships...**

- Le menu **Tools** offre de nombreux outils notamment pour la génération de code

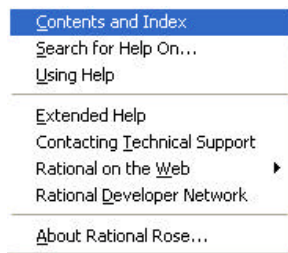
Nous n'avons présenté ici que le haut du menu **Tools** pour des raisons de commodité. Les fonctions que nous ne montrons pas sont réservées à la génération de code ou aux fonctions de reverse engineering offertes par Rational Rose. L'outil **Create** permet de placer sur le diagramme le type d'objet que l'on souhaite créer. On sélectionne par exemple **Text** dans le sous menu **Create** et une flèche verticale apparaît pour permettre à l'utilisateur de spécifier l'endroit où il souhaite placer une zone de texte dans son diagramme. **Check Model** permet d'utiliser l'outil de vérification de Rational Rose. Les incohérences du modèle sont alors spécifiées dans le **Log** (fenêtre d'erreur). On peut également noter la présence de l'outil **Options** qui permet de gérer tous les principaux paramètres de la modélisation (police de caractères, options de sauvegarde, visualisation des diagrammes, paramètre du navigateur etc...).



- Le menu **Window** permet de gérer la disposition des fenêtres

Chaque fenêtre ouverte reste ouverte tant qu'elle n'a pas été fermée. L'utilisateur peut utiliser ce menu pour passer d'une fenêtre à l'autre ou encore sélectionner la fonction **Tile** pour les voir toute simultanément à l'écran.





- Enfin le menu **Help** permet d'accéder à l'aide de Rational Rose mais aussi à l'aide en ligne et aux communautés d'utilisateurs de Rational Rose. Enfin le menu Help permet de visualiser les caractéristique de la version de Rational Rose de l'utilisateur.

#### ➤ Barre d'outils standard horizontale

Pour afficher ou non cette barre d'outils, on peut utiliser le menu **View>Toolbars>Standard**













Un clic droit sur cette barre affiche le menu suivant :













Activer ou désactiver **Allow Docking** permet d'autoriser ou non la fusion de la barre d'outils standard avec les bords de la fenêtre.

Use Large Buttons permet d'afficher les boutons avec des grandes icônes. Enfin **Customize...** permet de choisir quelle fonction

l'utilisateur souhaite pouvoir appeler directement avec cette barre d'outils. Le tableau suivant décrit toutes les fonctions que l'on peut éventuellement appeler à l'aide de cette barre d'outil.

| Bouton  | Nom                         | Fonction  |
|---|-----------------------------|---|
|  | Create New Model or File    | Créer un nouveau modèle / Reviens à l'écran de démarrage de Rational Rose   |
|  | Open Existing Model or File | Ouvre une fenêtre qui permet de naviguer dans l'ordinateur à la recherche d'un fichier existant .mdl ou .ptl  |
|  | Save Model File or Script   | Sauvegarde les modifications / si premier appel demande à l'utilisateur de spécifier un nom pour le fichier contenant le modèle et un emplacement dans l'ordinateur |
|  | Cut                         | Coupe l'élément sélectionné et le met dans le presse-papier.  |
|  | Copy                        | Copie l'éléments sélectionné et le met dans le presse-papier.   |
|  | Paste                       | Colle l'éléments contenu dans le presse papier.   |
|  | Print                       | Lance l'impression de la fenêtre de diagramme en cours.   |
|  | Context sensitive help      | Permet de pointer un élément dont on désire obtenir l'aide  |
|  | View Documentation          | Permet d'afficher la documentation sur l 'élément sélectionné.  |
|  | Browse Class Diagram        | Ouvre une fenêtre répertoriant tous les diagrammes de classe du modèle afin de choisir  |






















|   |                              |  |
|---|------------------------------|--|
|   |                              | celui que l'utilisateur souhaite afficher.   |
|    | Browse Interaction Diagram   | Ouvre une fenêtre répertoriant tous les diagrammes d'interaction du modèle afin de choisir celui que l'utilisateur souhaite afficher.                |
|    | Browse Component Diagram     | Ouvre une fenêtre répertoriant tous les diagrammes de composition du modèle afin de choisir celui que l'utilisateur souhaite afficher.               |
|    | Browse State machine Diagram | Ouvre une fenêtre répertoriant tous les statecharts du modèle afin de choisir celui que l'utilisateur souhaite afficher.                             |
|    | Browse Deployment Diagram    | Ouvre une fenêtre répertoriant tous les diagrammes de déploiement d'application du modèle afin de choisir celui que l'utilisateur souhaite afficher. |
|    | Browse Previous Diagram      | Recherche le diagramme précédemment affiché  |
|    | Zoom In / Zoom out           | Fonctions de zoom classique  |
|    | Fit in window                | Adapte affichage à la fenêtre  |
|   | Undo Fit in window           | Inverse action précédente  |
|  | Help Topics                  | Affiche l'index d'aide de Rational Rose  |
|  | Browse Use Case Diagram      | Ouvre une fenêtre répertoriant tous les Use Case du modèle afin de choisir celui que l'utilisateur souhaite afficher.                                |

### ➤ Barre d'outils verticale





Cette barre d'outils est spécifique au diagramme que l'utilisateur est en train de réaliser.




Dans le cas d'un Use Case, les boutons qui nous ont parus les plus intéressants sont les suivants.

| Bouton  | Nom                 | Fonction   |
|---|---------------------|--|
|  | Selection tool      | Lorsque ce bouton est appuyé, l'objet sur lequel on clique est alors sélectionné |
|  | Text Box            | Permet d'ajouter une zone de texte dans un diagramme                             |
|  | Note                | Permet d'ajouter un commentaire ou une explication sous forme d'un post-it       |
|  | Anchor note to item | Relie un commentaire à l'objet qu'il décrit                                      |
|  | Package             | Pour ajouter un projet complet dans un modèle.                                   |








|   |                            |   |
|---|----------------------------|---|
|    | Use Case                   | Pour ajouter un cas d'utilisation   |
|    | Actor                      | Pour ajouter une class de stéréotype Acteur   |
|    | Unidirectional association | Association unidirectionnelle entre deux entités d'un diagramme   |
|    | Dependency or instantiates | Pour marquer une dépendance fonctionnelle entre deux objets   |
|    | Generalization             | Pour marque un héritage (généralisation)  |
|    | Class                      | Pour ajouter une classe.  |
|    | Parametrized Class         | Pour ajouter une classe paramétrée.   |
|    | Class Utility              | Pour ajouter une classe fournissant des opérations aux autres classes.  |
|    | Parametrized Class Utility | Pour ajouter une classe fournissant des opérations aux autres classes paramétrées   |
|    | Association                | Pour créer une association entre deux classes.  |
|    | Aggregation                | Pour donner la prédominance d'une classe par rapport à l'autre on réalise une agrégation.   |
|    | Unidirectional aggregation | Agrégation unidirectionnelle.   |
|  | Association Class          | Pour créer une classe association. On utilise cet outil entre une association préexistante et la classe que l'on veut définir comme classe association.   |
|  | Lock Selection             | Lorsque ce bouton est enfoncé, on peut répéter la mise en place de l'objet sélectionné sans avoir à le resélectionner dans la barre d'outil. Cet outil est utile par exemple lorsqu'on a mis en place nos classes et que l'on souhaite placer les associations les unes après les autres. |

Les boutons proposés par Rational Rose pour la réalisation d'un diagramme de classes sont sensiblement les mêmes. Dans le cas d'un diagramme de collaboration des boutons spécifiques sont à la disposition de l'utilisateur.









| Bouton  | Nom             | Fonction   |
|---|-----------------|--|
|  | Class Instance  | Instance de classe   |
|  | Object Link     | Lien entre deux objet différents   |
|  | Link to himself | Lien d'un objet sur lui-même   |
|  | Link Message    | Envoi message entre deux objets le long d'un lien (droite vers gauche – bas vers haut) |

|   |                      |  |
|---|----------------------|--|
|  | Reverse Link Message | Envoi message entre deux objets le long d'un lien (gauche vers droite – haut vers bas) |
|  | Data Token           | Echange de données dans un message obtenu par link message                             |
|  | Reverse Data Token   | Echange de données dans un message obtenu par reverse link message                     |

Pour un diagramme de séquence, les boutons ont encore des fonctionnalités différentes :

| Bouton  | Nom                  | Fonction                                       |
|---|----------------------|--|
|    | Object               | Ajout d'un objet dans le diagramme de séquence |
|    | Object Message       | Echange de message entre deux objet            |
|    | Message to self      | Envoie d'un message d'un objet à lui-même      |
|    | Return Message       | Retour de message                              |
|    | Destruction Marker   | Destruction d'un objet                         |
|  | Processus Call       | Appel de procédure                             |
|  | Asynchronous Message | Envoi d'un message asynchrone                  |

Le tableau suivant présente les boutons de création de statecharts :

| Bouton  | Nom                        | Fonction  |
|---|----------------------------|---|
|  | State                      | Représentation d'un état  |
|  | Start State                | Entrée dans une état  |
|  | End State                  | Sortie d'un état  |
|  | State transition           | Transition entre deux états   |
|  | Transition to self         | Transition d'un état sur lui-même   |
|  | Decision                   | Prise de décision (OUI/NON) = IF THEN IF NOT THEN   |
|  | Horizontal Synchronization | Synchronisation horizontale (pour délimiter des états qui doivent être terminé avant de passer la transition) |
|  | Vertical Synchronization   | " " "   |

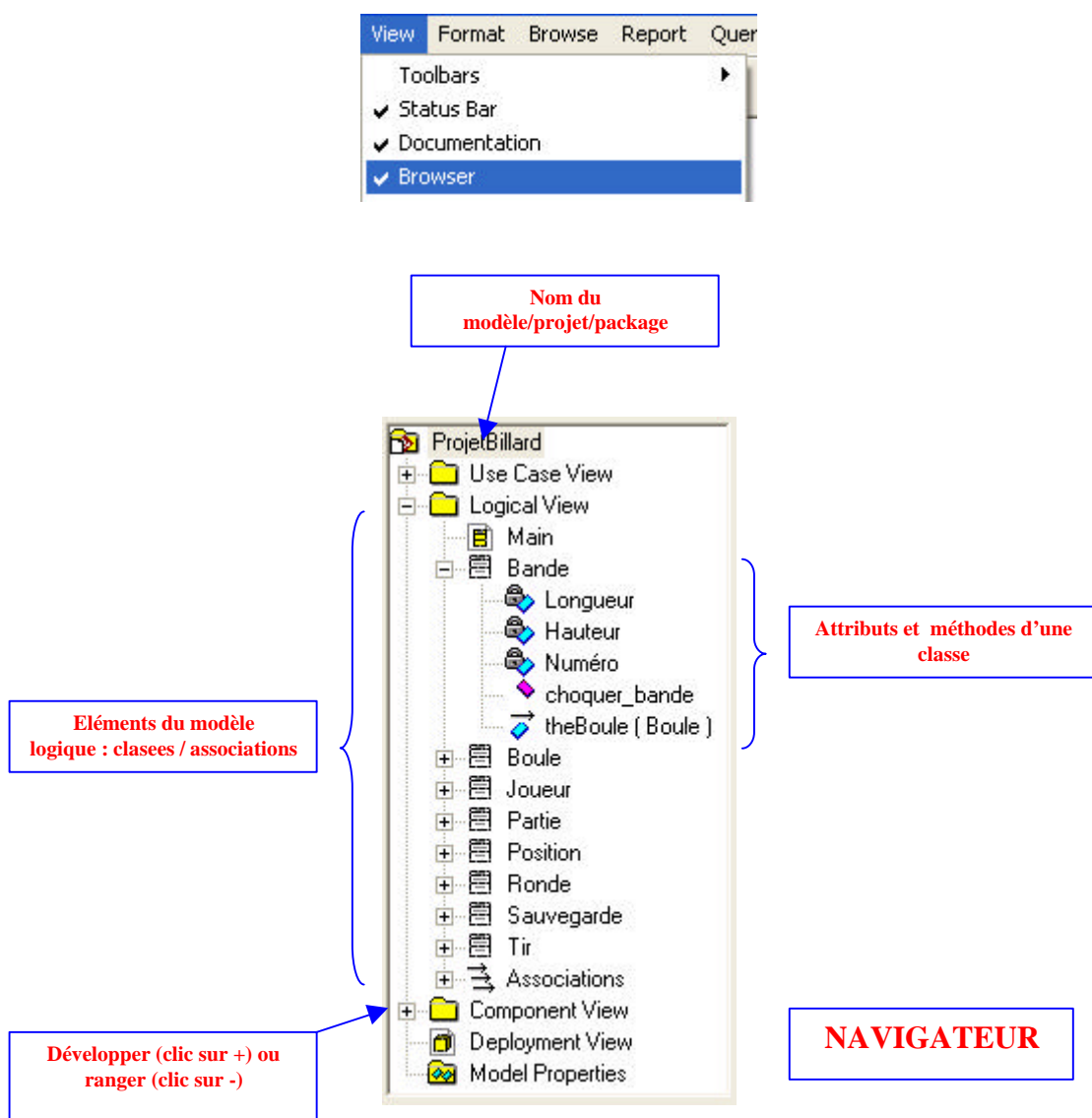
Nous avons désormais à notre disposition tous les principaux outils de réalisation des différents diagramme possible sous Rational Rose.

Une autre partie de l'écran très pratique est le navigateur (Browser) qui permet de se déplacer rapidement dans un projet complet.

## ii. Navigateur

L'utilisateur peut déplacer le navigateur (drag'n drop) et le placer à sa convenance dans son écran sous forme d'une petite fenêtre séparée de la fenêtre principale ou encore le faire fusionner avec le bord droit ou gauche de la fenêtre principale. Par défaut, le navigateur est fusionné et placé à gauche de l'espace de travail.

On peut le faire apparaître ou disparaître avec la commande : **View>Browser**.



Le navigateur est une représentation hiérarchique du modèle. Chaque nouvel élément du modèle (classe, acteur, diagramme, use case) est ajouté dans le navigateur.

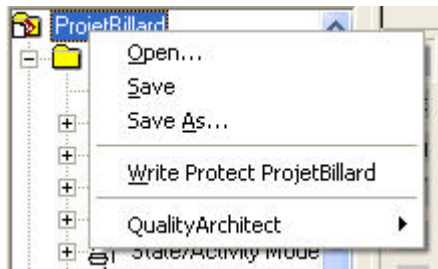
Le navigateur permet :

- d'ajouter un éléments au modèle
- d'ouvrir et de visualiser un élément existant du modèle
- de visualiser les relations entre les objets
- de déplacer facilement un éléments
- de renommer les éléments
- etc...

Les éléments sont regroupés dans des répertoires selon leur place dans le modèle.

| Use Case View                  | Logical View           | Component View     | Deployment View    |
|--------------------------------|------------------------|--------------------|--------------------|
| Business actors                | Classes                | Components         | Processes          |
| Business workers               | Class diagrams         | Interfaces         | Processors         |
| Business Use cases             | Associations           | Component diagrams | Connectors         |
| Business Use Case Diagrams     | Interfaces             | Packages           | Devices            |
| Business Use Case realizations | Sequence diagrams      |                    | Deployment diagram |
| Actors                         | Collaboration diagrams |                    |                    |
| Use cases                      | Statecharts            |                    |                    |
| Associations                   | Packages               |                    |                    |
| Use case documentation         |                        |                    |                    |
| Use case Diagrams              |                        |                    |                    |
| Acitivity diagrams             |                        |                    |                    |
| Sequence diagrams              |                        |                    |                    |
| Collaboration diagrams         |                        |                    |                    |
| Packages                       |                        |                    |                    |

Clic droit sur le nom du projet :



**Open** permet d'ouvrir un nouveau projet.

**Save et Save As...** pour sauvegarder le projet en cours.

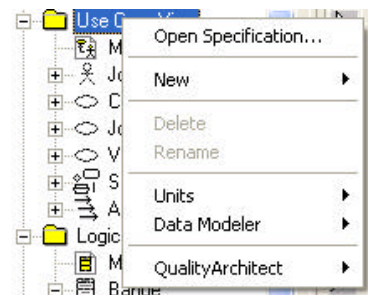
**Write Protect NomDuProjet** permet de verrouiller le projet pour éviter des mauvaises manipulations pendant une présentation par exemple.

**QualityArchitect** pour ouvrir des templates.

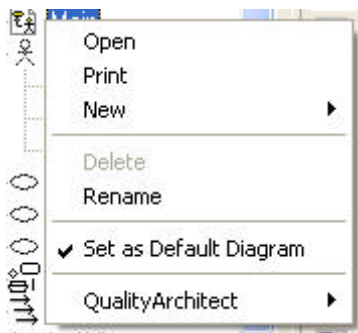
Clic droit sur Use Case View :

**Open Spécification...** permet de régler les propriétés générales des Use Case.

**New** permet d'ajouter un des éléments listés dans le tableau précédent.



Clic droit sur un diagramme (Main) :



**Open** affiche le diagramme dans l'espace de travail.

**Print** permet d'imprimer le diagramme sur lequel on a cliqué.

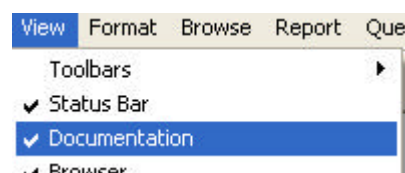
**New** permet de lier un fichier ou une URL.

**Set as Default Diagram** est coché quand le diagramme est défini comme diagramme principal.

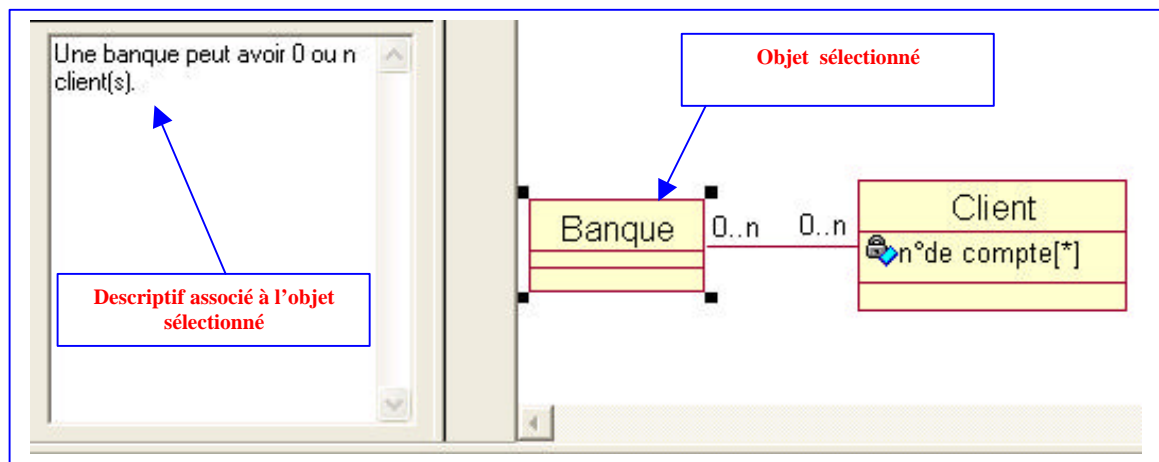
Le clic droit sur les autres éléments du navigateur (classes, associations, attributs, méthodes) permet d'ouvrir les spécifications (Open Specifications...), de renommer, de déterminer l'accès (Private, protected, public etc...) et les propriétés spécifiques aux éléments ainsi sélectionnés.

### iii. Fenêtre de description/documentation

Pour afficher, faire disparaître cette fenêtre, on peut utiliser le menu **View>Documentation**.



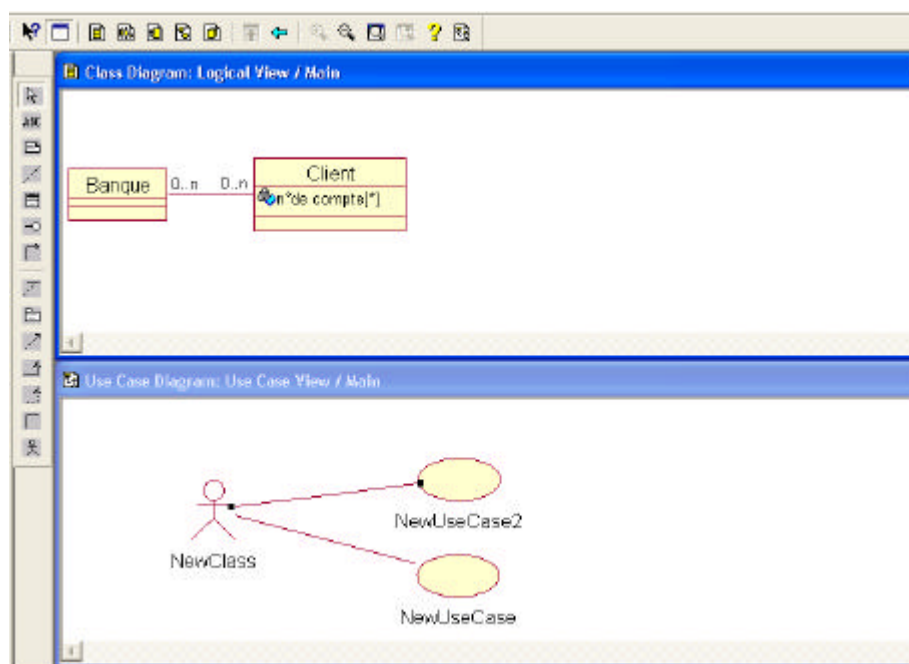
Le fenêtre de description documentation est intéressante pour donner des précisions sur les différents objets qui composent le modèle.



Pour chaque objet on peut rédiger quelques lignes de documentation qui aident à préciser une idée passée par le concepteur afin d'agréments la compréhension de son modèle par un lecteur tiers. Avec la même technique (Drag n' Drop = faire glisser) que pour les autres parties de l'espace de travail, la fenêtre de documentation peut être déplacée à la guise de l'utilisateur, soit sous forme de fenêtre séparée soit fusionnée avec le bord de la fenêtre principale et placée en haut, en bas, à droite ou encore à gauche de l'écran. Par défaut, cette fenêtre est visible sous le navigateur (Browser).

#### iv. Espace de dessin

L'espace de dessin appelée **Diagram Window** est la fenêtre de base de l'espace de travail. Elle est toujours visible et représente le plan de travail du concepteur du modèle.

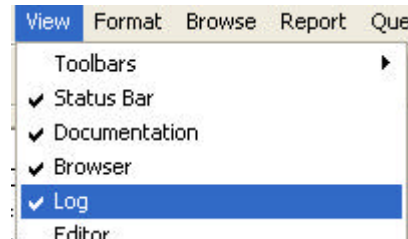




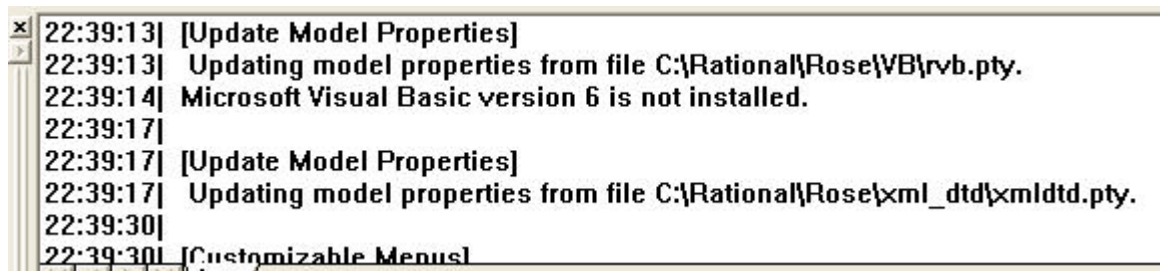
Toutes les fenêtres en cours (une fenêtre correspond à un diagramme), s'ouvrent dans cette zone de l'écran. Nous avons déjà vu que l'on pouvait définir l'affichage de ces fenêtres en mosaïque ou comme ici les unes à côté des autres (option **Window>Tile**).

## v. Fenêtre d'erreurs ou fenêtre de Log

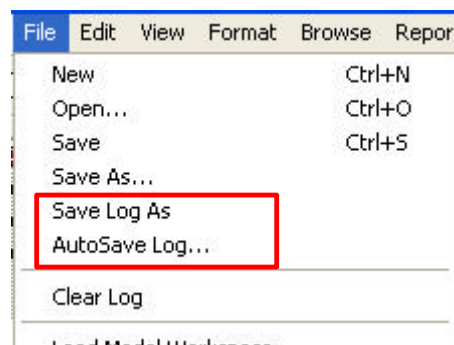
Pour afficher ou non la fenêtre de Log, nous pouvons utiliser la commande **Log** du menu **View**.



Certaines informations sont envoyées dans la fenêtre de Log lors de l'utilisation de Rational Rose, notamment lors de son lancement et de l'initialisation des paramètres mais aussi lors de la génération de code ou lors de tests de vérification du modèle. Toutes les erreurs seront par exemple affichées dans cette partie de l'écran.



Il est également possible d'enregistrer ces informations dans un fichier spécifique .log.



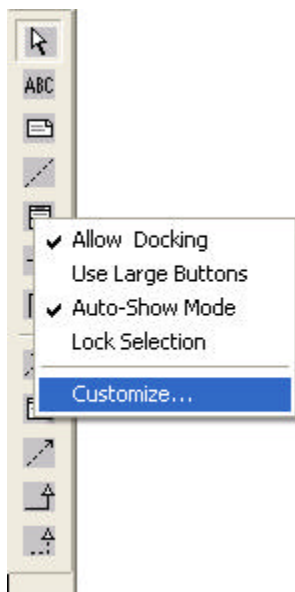
Les commandes **File>Save Log As** et **File>AutoSave Log...** sont les deux principales commandes permettant d'enregistrer le fichier .log. Le nom par défaut de ce fichier est error.log car il est principalement utilisé pour éplucher les erreurs générées dans le code. Enfin une fonction **File>Clear Log** permet d'effacer les informations contenues dans le Log jusqu'à alors.

## b. Configuration de l'espace de travail

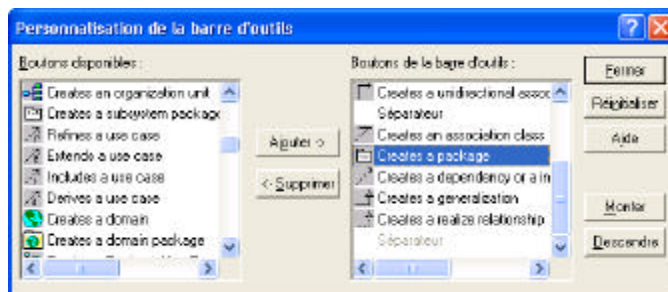
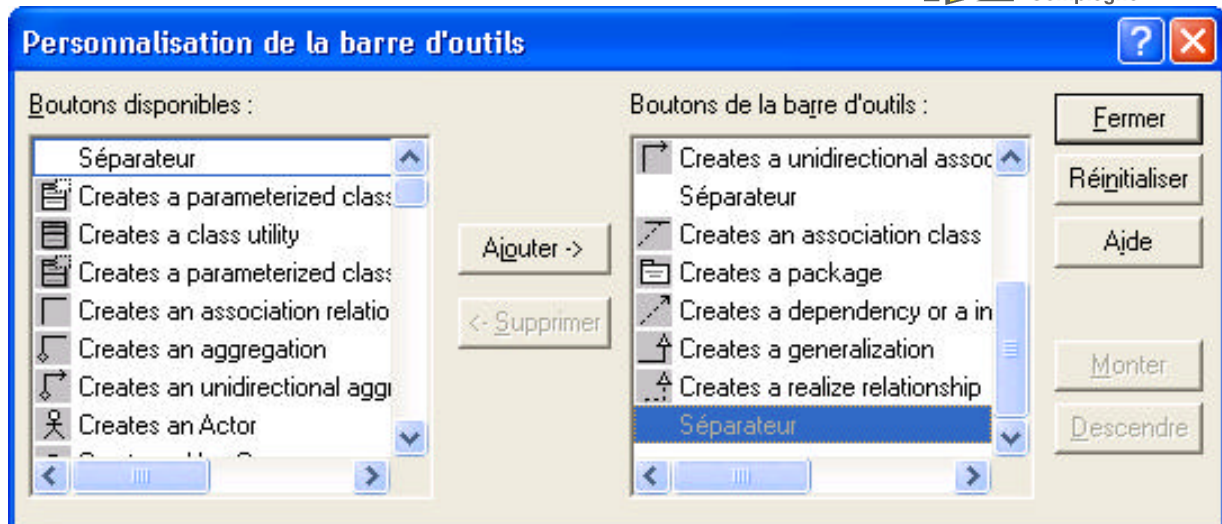
L'avantage de la réalisation d'un modèle à l'aide d'un logiciel tel que Rational Rose est le gain de temps que cela induit dans le cycle de vie du logiciel. Ainsi on comprend bien qu'optimiser le temps passer derrière Rational Rose sera également un critère important. C'est pourquoi il convient de configurer au mieux son environnement de travail et de l'adapter à sa réalisation. Chaque modèle demandera une configuration particulière. La configuration par défaut de Rational Rose est telle qu'elle est adaptée à la plupart des réalisations. Nous conseillons tout de même vivement à tout nouvel utilisateur de commencer par se familiariser avec cet environnement et de le configurer à sa convenance. Rational Rose étant extrêmement complet, il est intéressant de se donner un accès rapide aux seules fonctions que 'on est amené à utiliser tout en masquant du mieux possible les options de moindre utilité.

Il est intéressant d'abord selon la taille de l'écran de l'ordinateur sur lequel Rational Rose est utilisé de définir quelles fenêtres parmi celles décrites dans la partie précédente seront toujours visibles. Selon notre utilisation personnelle dans le cadre de l'UV LO19, nous conseillons de garder la configuration par défaut que Rational Rose propose ou éventuellement de supprimer l'affichage de la fenêtre de Log (**View>Log**).

Ensuite il s'avère important avant de créer un diagramme de configurer la barre d'outils avec les boutons les plus nécessaires à notre besoin. Ici nous prendrons l'exemple de la réalisation d'un diagramme de classes mais la démarche est la même pour un autre type de diagramme.

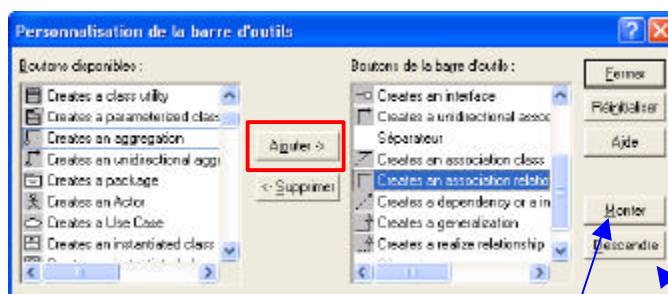
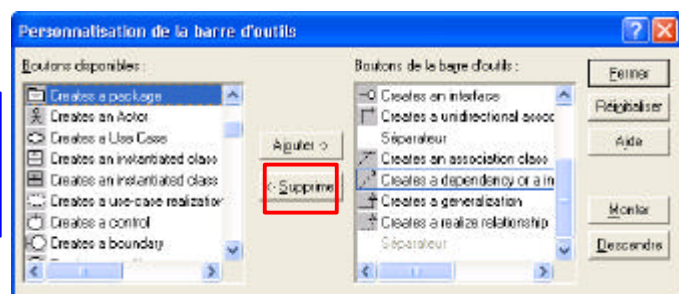


Pour configurer la barre d'outils soit même, faire un clic droit sur la barre d'outils en question et sélectionner **Customize...** pour ouvrir la fenêtre de configuration..



Pour enlever le bouton **Creates a package** qui ne nous semble pas utile pour notre modèle, on sélectionne alors **Creates a package** dans la partie de droite. Le bouton **← Supprimer** devient alors actif.

On peut alors cliquer sur **Supprimer** et le bouton passe dans la patrie de gauche (Boutons Disponibles).



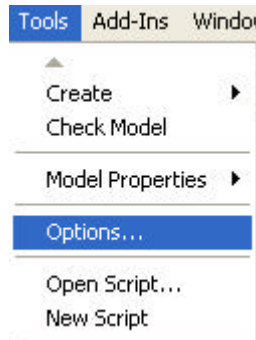
Pour ajouter le bouton **Create an association relationship**, on le sélectionne dans la liste des boutons disponibles et on clique sur ajoute, il passe alors dans la liste des boutons de la barre d'outils.

On peut aussi utiliser les boutons **Monter** et **Descendre** pour positionner les différents boutons de notre barre d'outils dans l'ordre que l'on souhaite. L'utilisation de séparateur pour aussi être un facteur de clarté.

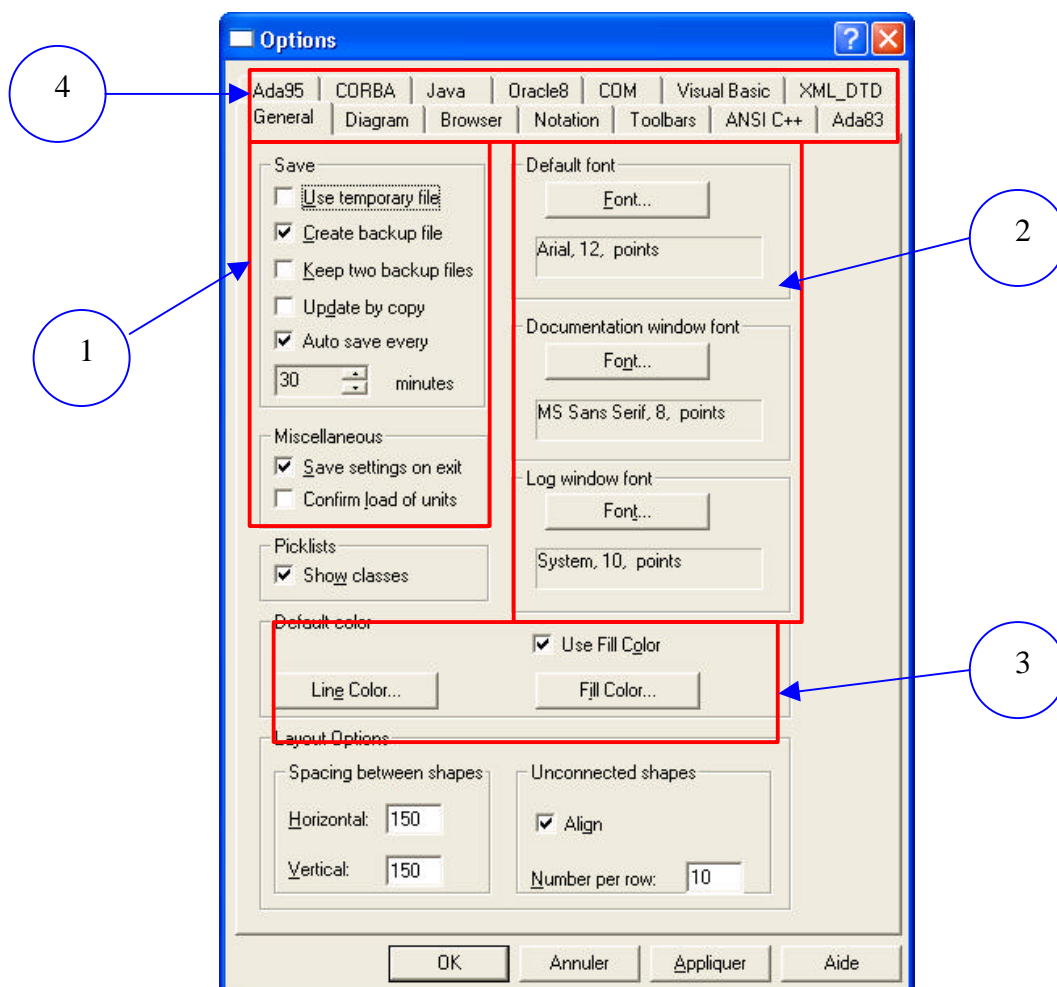
On peut ensuite jouer sur la position de la barre d'outils comme nous l'avons vu dans la partie précédente. Il suffit simplement de faire glisser la barre d'outils vers la zone de l'écran où l'on souhaite la voir s'attacher.

Une fois les outils mis à portée de main, on peut configurer les options générales du programme afin d'optimiser le rendu du modèle mais aussi l'efficacité de la conception.

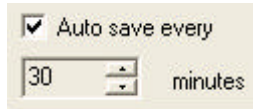
### Tools>Options



La fenêtre de configuration générale des options s'ouvre alors. NB : Le nombre de paramètres étant très important, nous ne nous contenterons ici de ne présenter que ceux qui nous ont paru les plus intéressants. Pour les autres, les configurations sont souvent assez intuitives et l'utilisateur n'aura guère de mal à les réaliser tout seul.



### 1/ Paramètres de Sauvegardes :



Il est recommandé de mettre en place une sauvegarde automatique dans un intervalle de temps assez restreint pour ne pas avoir perdu un temps précieux lors d'une erreur de manipulation ou d'une défaillance du matériel.



Quand cette case est cochée, Rational Rose crée une copie de tous les fichiers qui sont ouverts au cours d'une séance de travail.



Enfin cette option permet de sauvegarder automatiquement à chaque fermeture du programme. Cette option est un peu dangereuse lorsqu'on est habitué à d'autres logiciels qui demandent si l'on souhaite sauvegarder les changements au moment de quitter.

### 2/ Paramétrage des polices :

Ces options permettent de régler les formats de police que l'on souhaite utiliser, par défaut, pour la fenêtre de documentation et pour la fenêtre de Log.

### 3/ Paramètres de couleurs par défaut :

Permet de configurer le rendu visuel des objets que l'on place sur les diagrammes notamment couleur de fond et couleur des traits de délimitation.

### 4/ Autres Onglets :

Les autres onglets permettent d'accéder à des réglages plus spécifiques notamment en rapport avec la génération de code automatique. On peut remarquer que l'onglet **Toolbars** permet d'accéder à une rubrique de configuration de toutes les barres d'outils (**Customize**) (voir le début de cette partie).

Une fois l'espace de travail bien préparé, bien configuré et adapté au besoin du concepteur du modèle, nous allons voir comment réaliser les principaux types de diagrammes à l'aide d'exemples simples.

.

## II. Exemples de réalisations simples

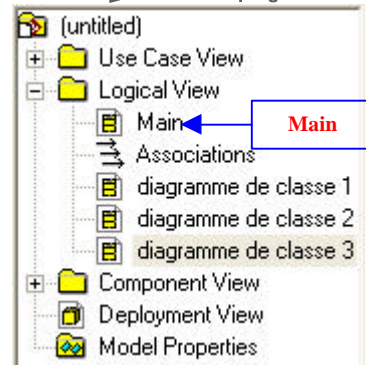
Nous n'allons pas voir dans cette partie comment réaliser un projet complet, mais nous allons donner les outils nécessaires à la réalisation d'un modèle complet avec les principaux types de diagrammes réalisables avec Rational Rose.

### a. Diagramme de classes

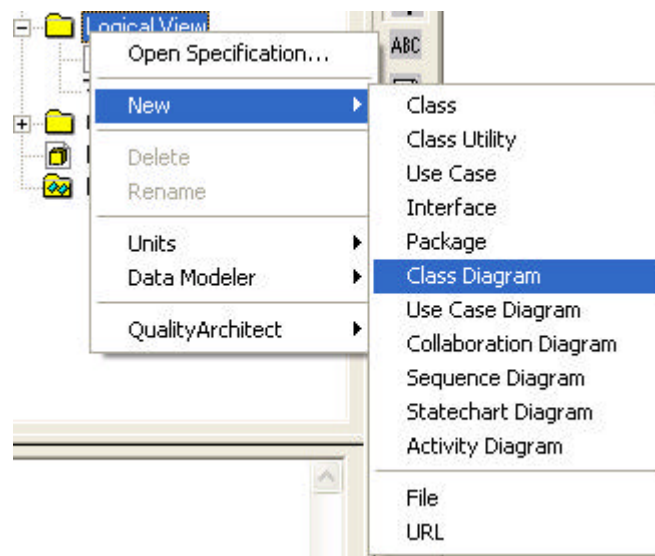
Un diagramme de classes met en jeu les objets utilisés par la future application avec leurs caractéristiques (attributs) et leurs comportements (méthodes). Le diagramme de classes donne une idée sur les relations entre les objets : peuvent-ils communiquer entre eux ? Certains sont-ils dépendants des autres ? Tous sont-ils directement utilisés par l'application ? etc ...



Dans l'outil Rational Rose, les diagrammes de classe sont créés dans le répertoire **Logical View** du navigateur. On peut bien sûr créer plusieurs diagrammes de classe dans un même modèle. Pourtant dès la création d'un nouveau modèle, Rational Rose crée automatiquement un diagramme de classes principal (Main) qui est amené à contenir tous les packages de diagramme de classes définis en-dessous. Si vous pensez ne réaliser qu'un seul diagramme, pour un modèle modeste, il faudra spécifier ce diagramme dans le Main.



Pour ajouter un nouveau diagramme de classes, faites un clic droit sur **Logical View** puis choisissez **New** et **Class Diagram**.



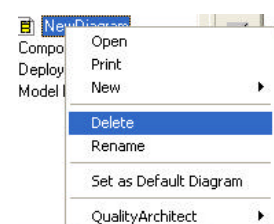
Un nouvel élément s'affiche alors dans la liste **Logical View**, son nom est en surbrillance, l'utilisateur peut alors donner le nom de son choix à son diagramme.



Une fois le nouveau nom entré, le diagramme ainsi créé peut être ouvert par un double-clic.

#### Remarque :

Pour supprimer un diagramme il suffit de faire un clic droit dessus dans le navigateur puis de sélectionner **Delete** dans le menu déroulant.

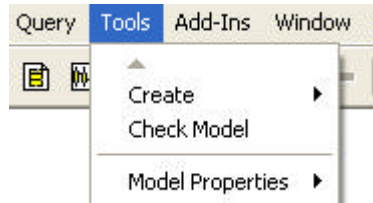


## i. Ajout d'une classe

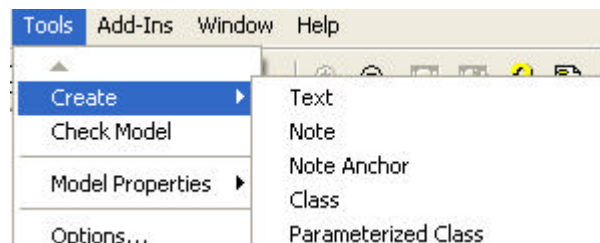
Pour ajouter une classe dans un diagramme de classes, il y a deux méthodes :

### 1- Utilisation du menu Tools :

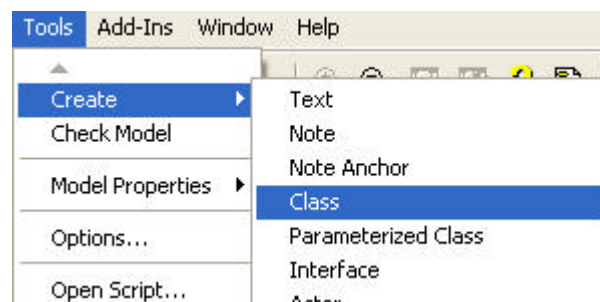
#### a. Cliquer sur **Tools**



#### b. Sélectionner **Create**



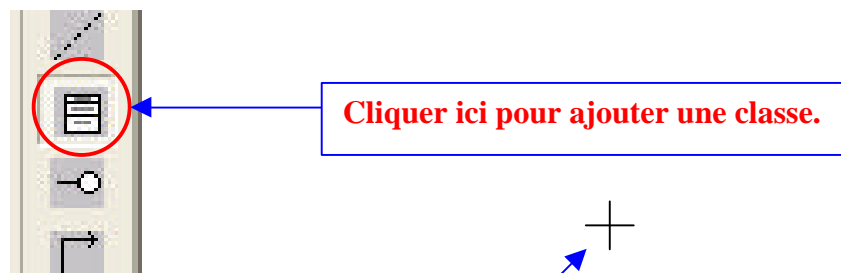
#### c. Choisir l'objet **Class**



#### d. Cliquer sur **Class**

### 2- Utilisation de la barre d'outil :

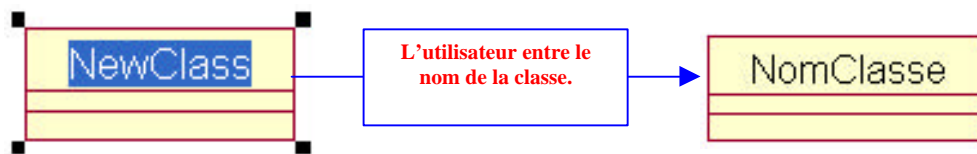
Si le bouton correspondant à l'ajout d'une classe n'est pas présent dans la barre d'outil, il peut être ajouté à l'aide de la méthode vue précédemment. Cliquez alors sur ce bouton.



Une fois l'objet sélectionné, le curseur de la souris devient un signe +.



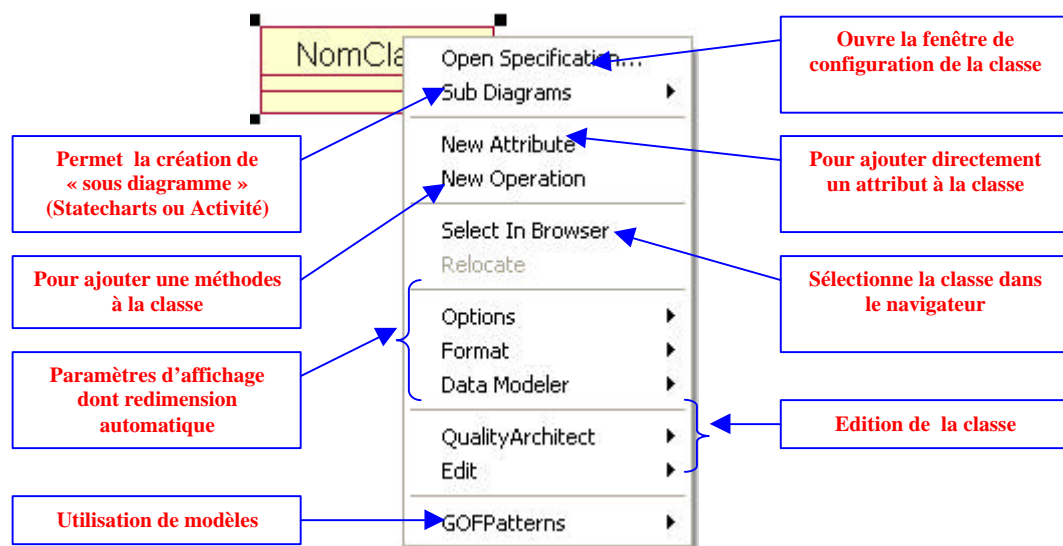
L'utilisateur peut alors placer la nouvelle classe dans le diagramme par un simple clic sur l'emplacement souhaiter. Comme pour tout nouvel objet généré, son nom apparaît en surbrillance, l'utilisateur peut alors le choisir à sa convenance.



Par défaut une classe ne contient aucune méthode et aucun attribut.

### *Paramétrages de la nouvelle classe*

Un clic droit sur la nouvelle classe créée permet d'afficher le menu suivant :



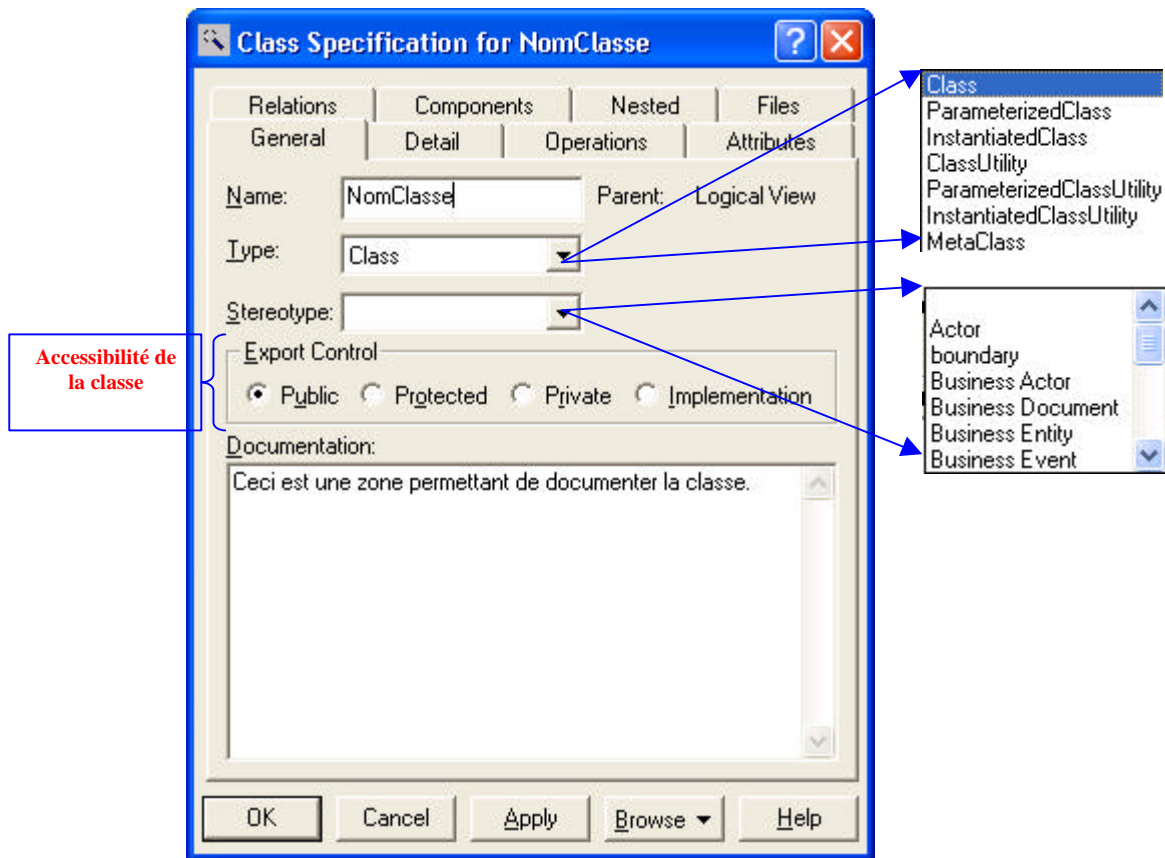
Cliquer sur **Open Spécification...** (un double clic sur la classe permet aussi d'ouvrir directement les spécifications de la classe).

La fenêtre de spécification permet de paramétrer complètement la classe, comme par exemple de lui ajouter un attribut mais aussi d'accéder aux paramètres de ses attributs. Les différents onglets permettent de parcourir ces spécifications.

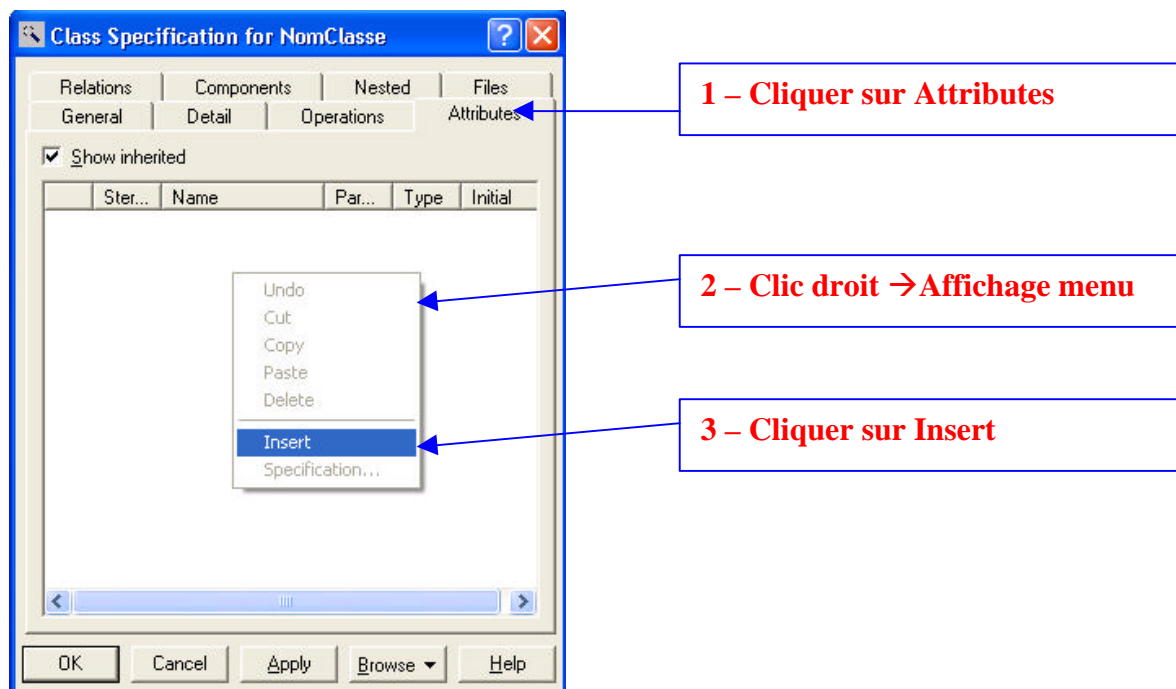
L'onglet **General** permet d'accéder aux caractéristiques générales de la classe :

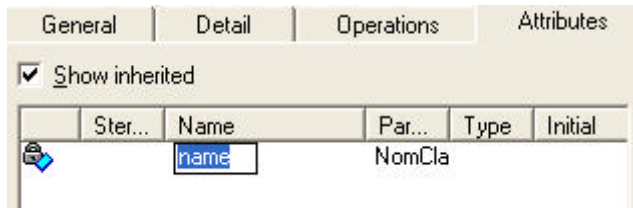
- **Name** : nom de la classe
- **Parent** : Répertoire parent dans le navigateur
- **Type** : Type d'objet, on peut modifier le type si l'on préfère spécifier cette classe comme une Métaclasse par exemple
- **Stéréotype** : Certaines classes ayant des comportements spécifiques peuvent être stéréotypées dans Rational Rose, par exemple une classe représentant une personne agissant dans le modèle peut être stéréotypée comme Actor.
- **Export Control** : permet de définir les droits d'accès à cette classe (Public / Protected / Private / Implementation)

- **Documentation** : fournit une zone de texte qui permet de donner des explications et des précision sur le comportement et l'utilisation de la classe



### Ajout d'un attribut

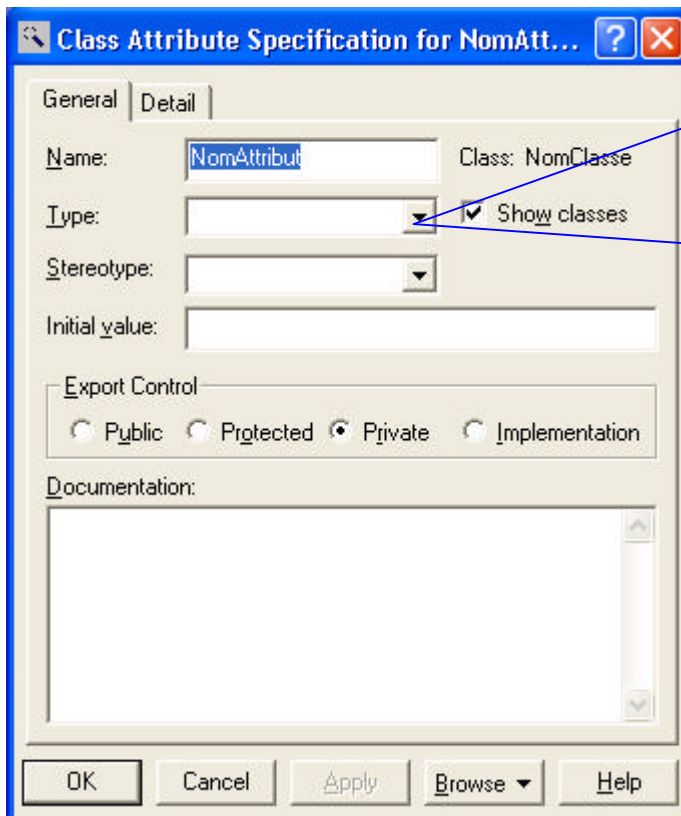




Un opérateur apparaît dans la liste avec son nom en surbrillance, l'utilisateur peut donner un nom à sa convenance.

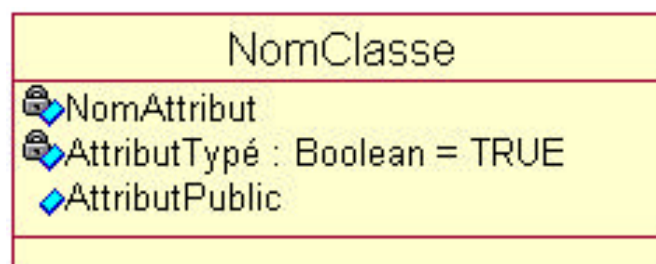
## Spécifications de l'attribut

Un double clic sur l'attribut ainsi créé permet d'ouvrir ses spécifications.



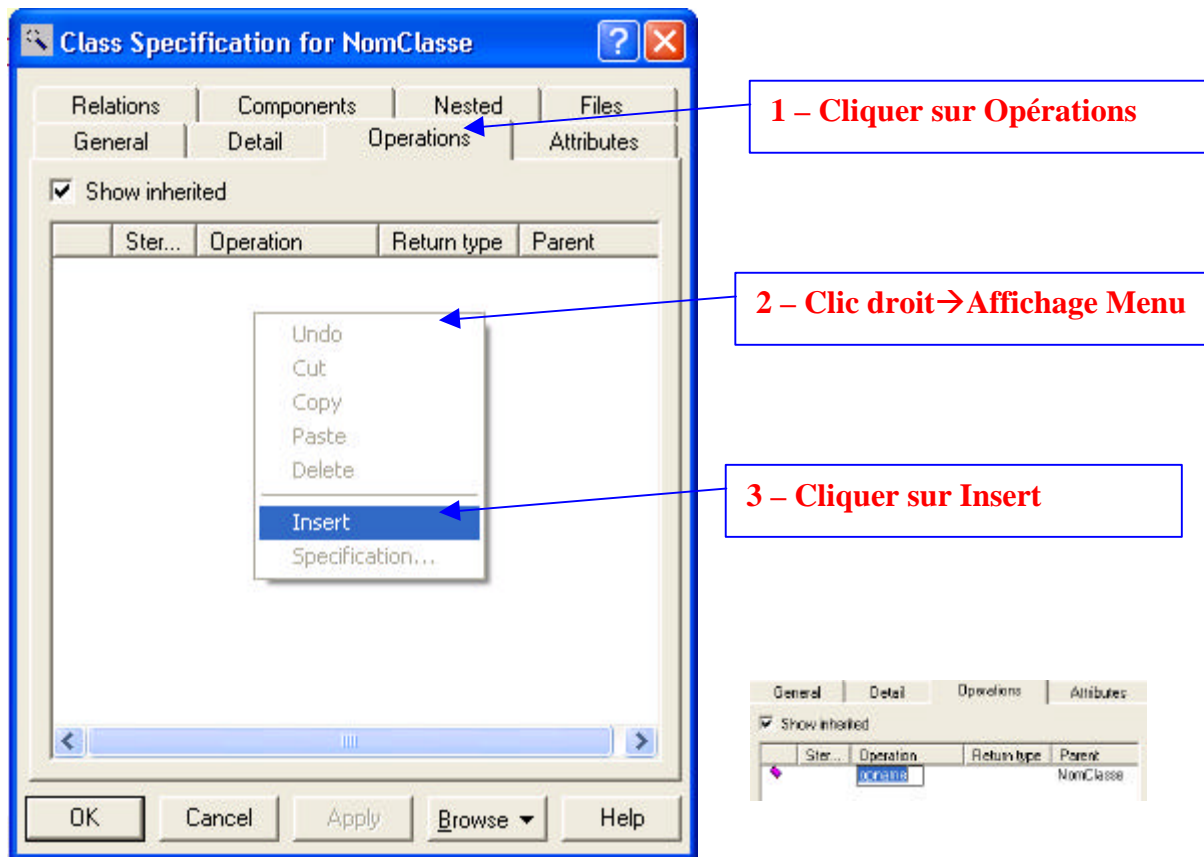
- **Name** : Nom de l'attribut
- **Class** : Classe associée
- **Type** : Type de l'attribut
- **Stereotype** : pas de stéréotype pour les attributs
- **Initial value** : valeur initiale de l'attribut
- **Export Control** : control d'accès à l'attribut (par défaut : private)
- **Documentation** : précisions concernant l'attribut

Un clic sur l'onglet **Detail** permet d'accéder au mode de mise en mémoire de la variable et au passage de paramètre (par valeur, ou par référence). Par défaut rien n'est spécifié. Selon les spécifications, l'affichage dans le diagramme est différent. Nous avons représenté quelques possibilités de définition d'attribut : un attribut privé non typé, un attribut typé de type boolean initialisé à TRUE et un attribut public.



## Ajout d'une méthode

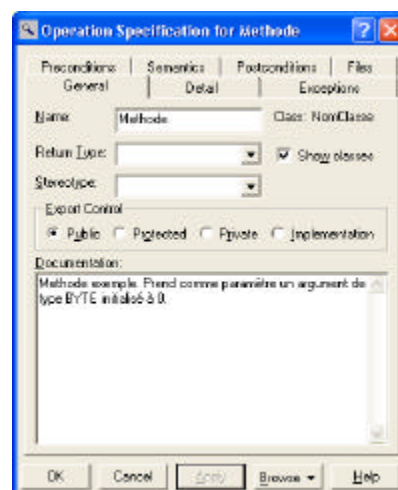
Le principe pour l'ajout d'une méthode est le même que pour l'ajout d'un attribut.



Puis comme pour un attribut on peut double-cliquer sur la méthode pour ouvrir ses spécifications. Les principaux paramètres étant les mêmes que ceux des attributs, nous ne nous intéresserons qu'aux spécificités des méthodes.

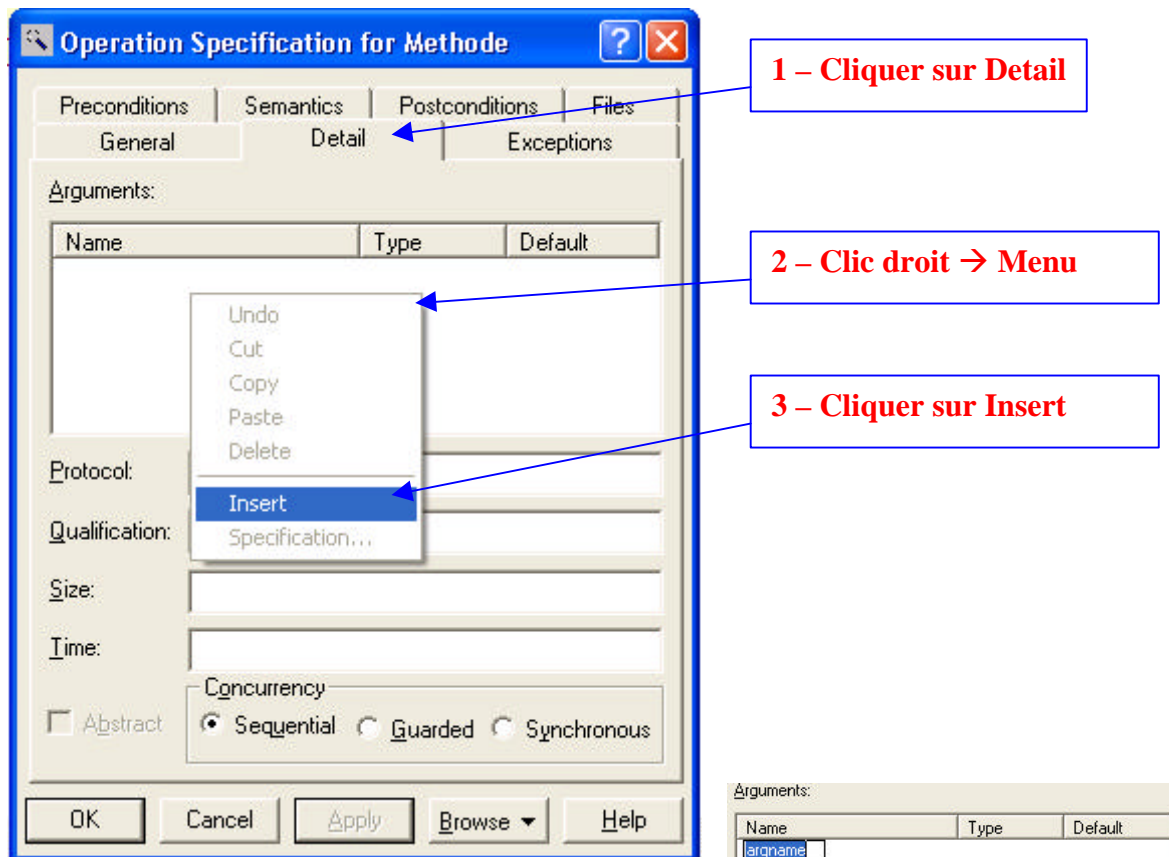
## Spécifications de l'attribut

Par défaut les méthodes sont déclarées comme publiques.

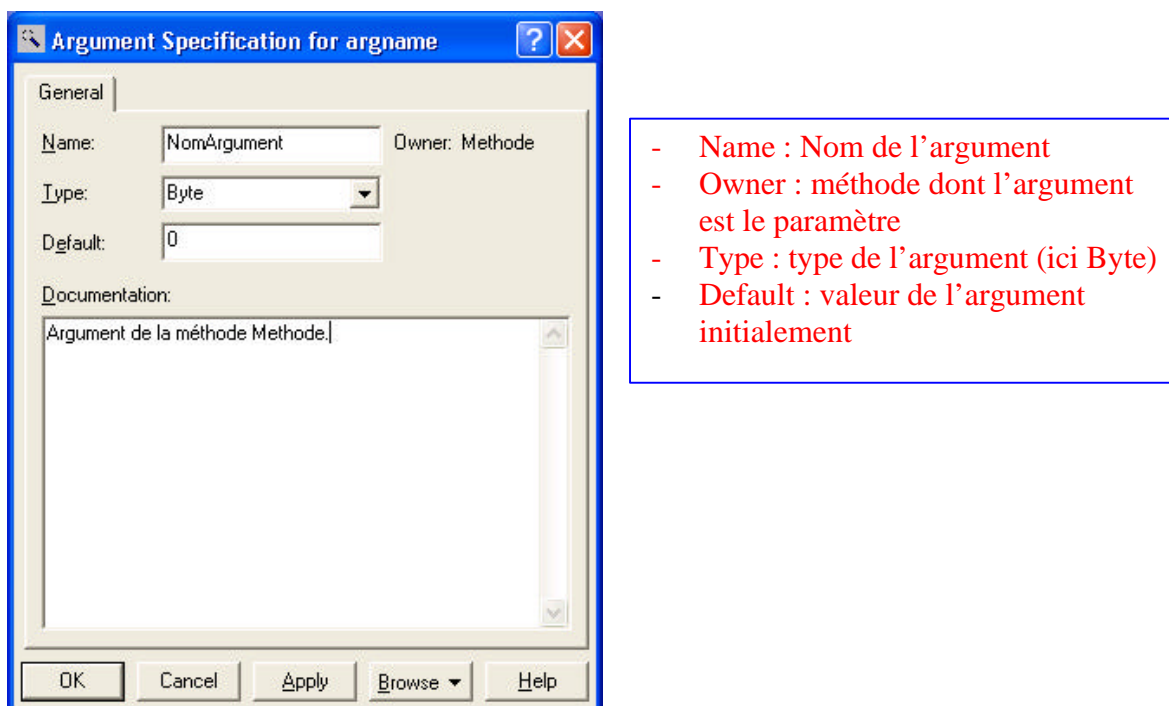


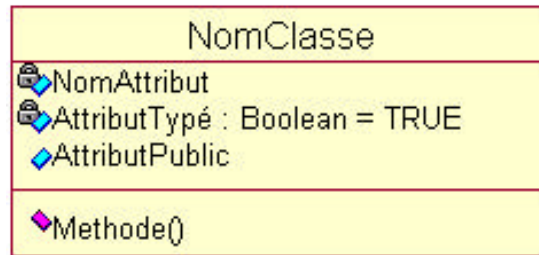
Le champs **Type** des attributs est remplacé par **Returned Type** qui correspond au type de la valeur renvoyée par la méthode.

Pour déclarer une méthode avec un argument :

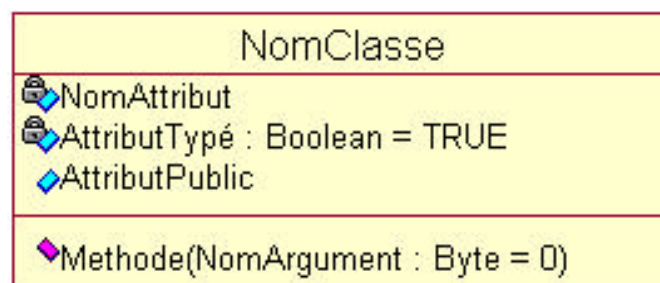
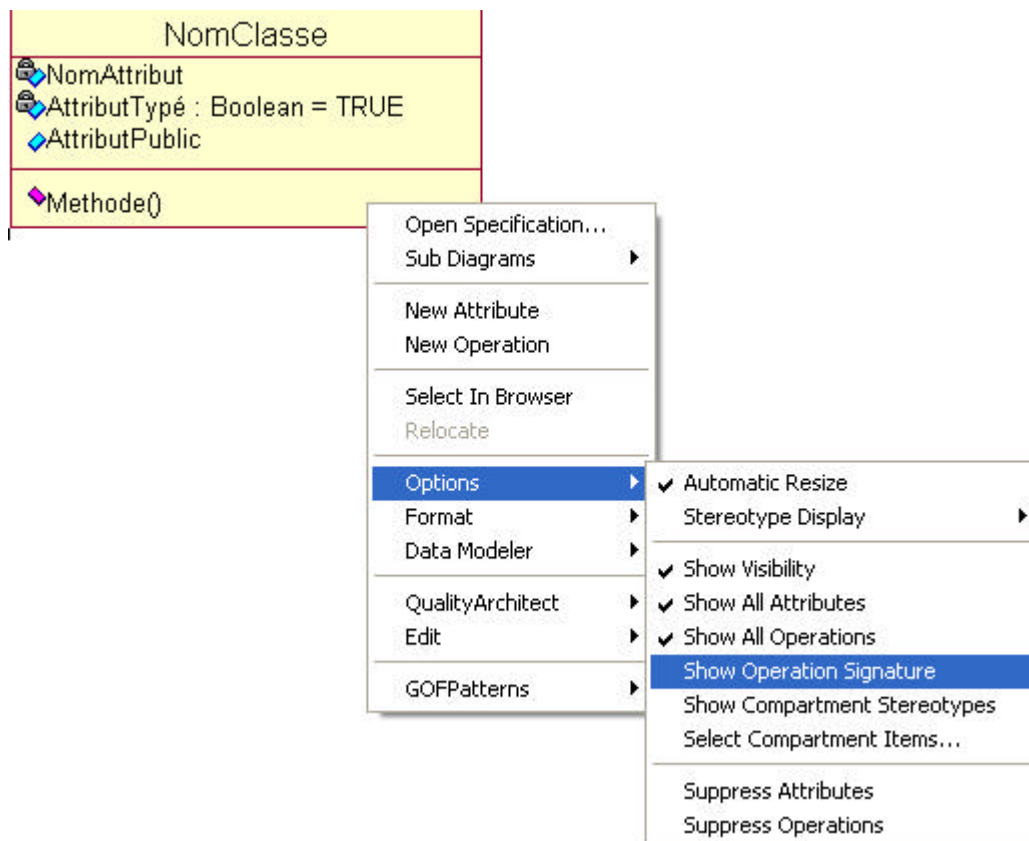


L'utilisateur rentre le nom de son argument et double-clique pour ouvrir ses spécifications.





Par défaut, les arguments ne sont pas visibles. On peut les rendre visible avec l'option **Show Operation Signature**.



Pour effacer une classe d'un diagramme, un simple clic droit puis **Edit>Delete** suffit. Par contre la classe reste présente dans le modèle. Pour la supprimer définitivement du modèle il faut sélectionner **Edit>Delete From Model** (Ctrl + D).

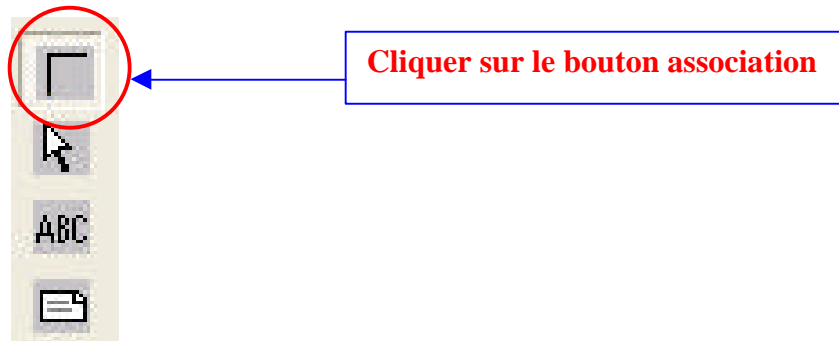
Nous avons maintenant les principaux outils permettant de créer des classes, nous allons voir comment créer des relations entre ces classes.



## ii. Ajout d'une relation

De nombreux types de relations peuvent associer deux éléments d'un diagramme de classes. Nous traitons ici les plus courants.

### *Mise en place d'une association*



Le curseur de la souris devient une flèche vertical, pointé une des classes extrémité de l'association, puis cliquer et laisser le bouton appuyé pour diriger le trait pointillé vers la classe qui représente l'autre extrémité.

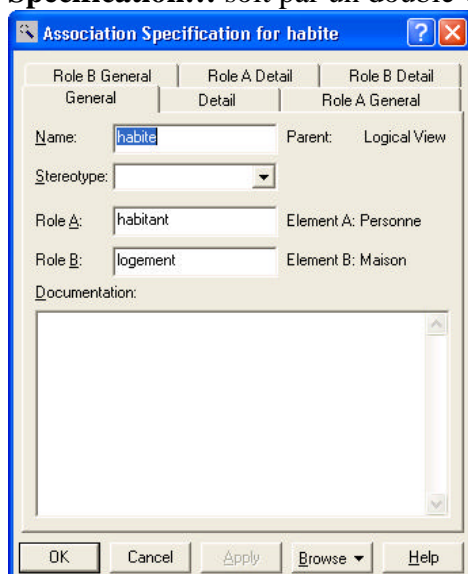


Relâcher le bouton de la souris. L'association est alors matérialisée.



### *Spécifications de l'association*

On peut ouvrir les spécification d'une association soit par un clic droit puis **Open Spécification...** soit par un double-clic sur l'association.

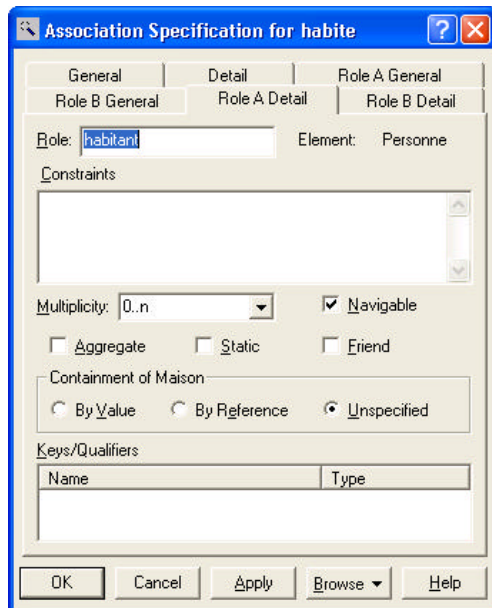


L'onglet **General** permet de :

- donner un **nom** (habite) à l'association (généralement indicatif de la relation entre les deux classes)
- préciser le **rôle** de A (Personne) et le rôle de B (Maison)

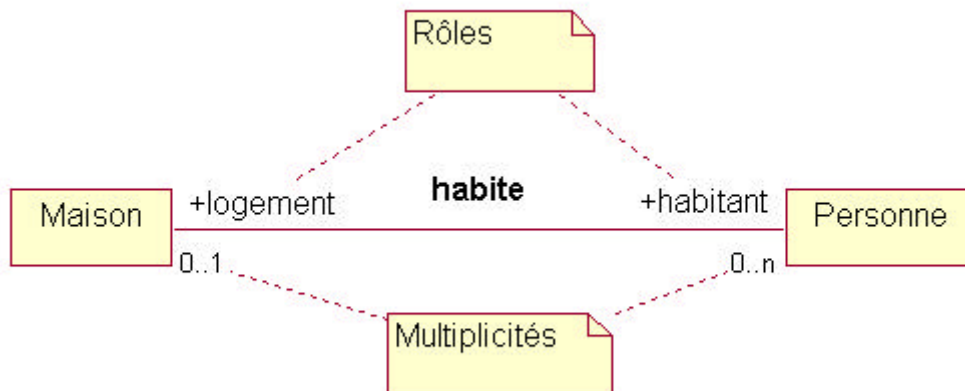


Les onglets Rôle A Général et Rôle B Général permettent de définir l'accès aux extrémités de l'association (Private / Protected / Public / Implementation).



L'onglet **Rôle X Detail** permet notamment de :

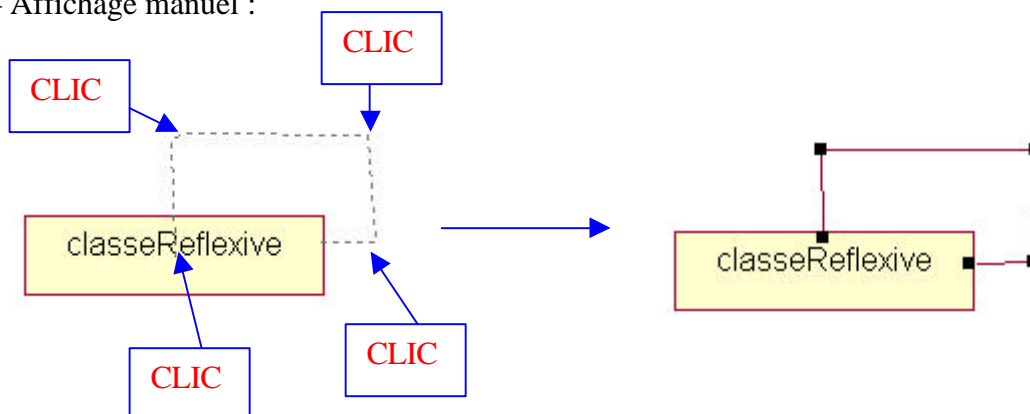
- donner le rôle de X
- définir la multiplicité de X  
(0 / 0..1 / 0..n / 1 / 1..n / n)



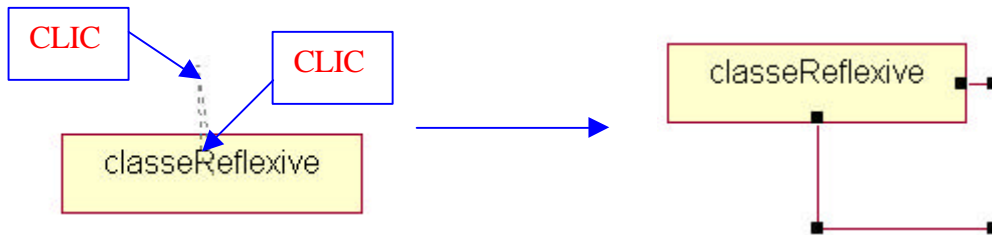
### Association réflexive

Rational Rose permet de réaliser des associations réflexives (d'une classe vers elle-même).

1 – Affichage manuel :



2 – Affichage automatique :



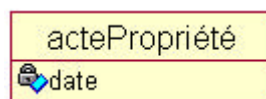
### ***Classe-Association***

Rational Rose permet de créer des classes-associations.

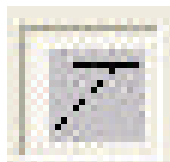
1 – on commence par créer une association simple



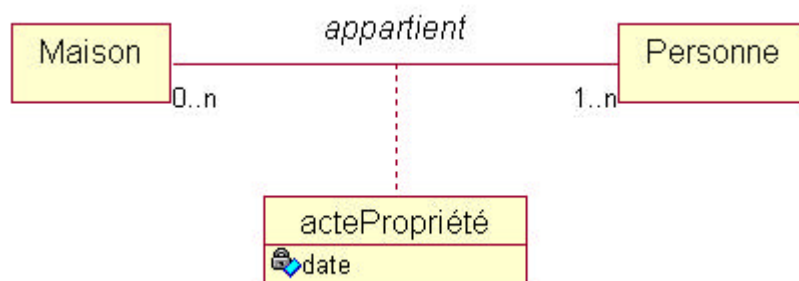
2 – on crée la classe qui formera la classe association



3 – on sélectionne l'opérateur Association-Class



4 – on relie l'association à la classe

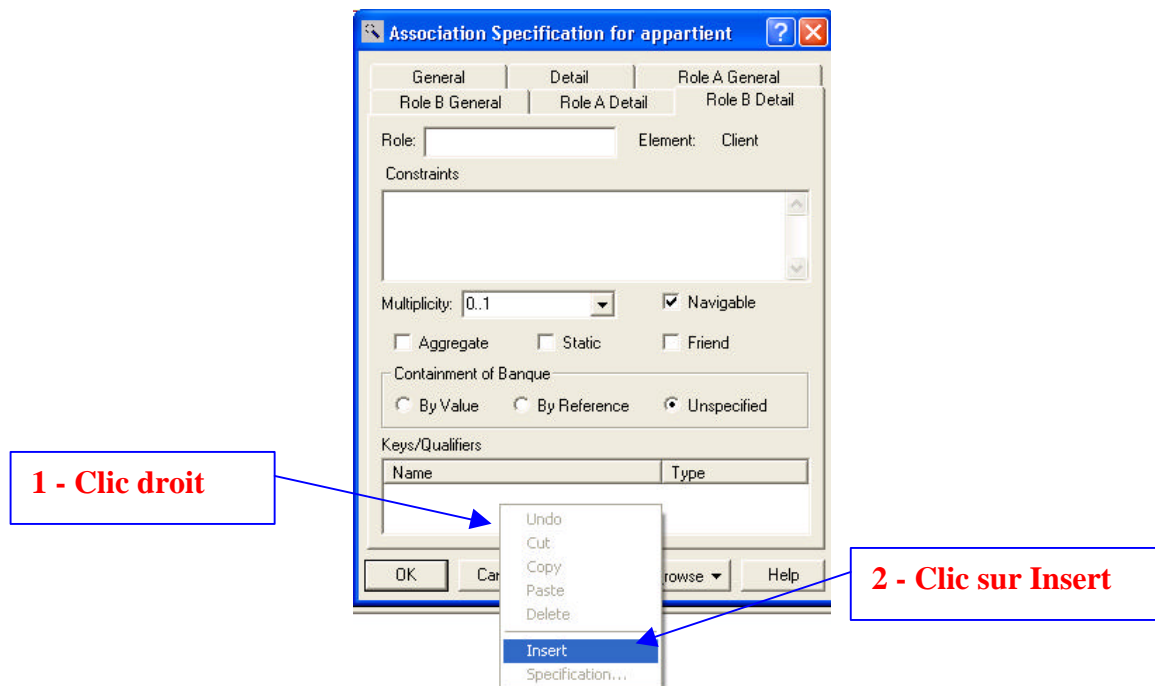


## Qualification d'une association

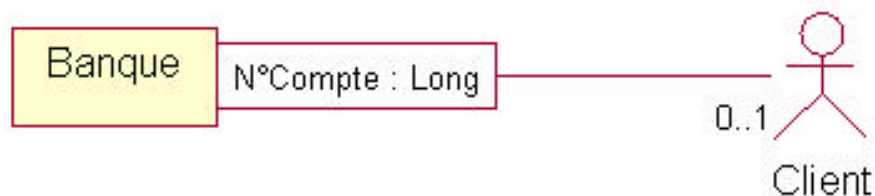
### 1 – Créer association simple



### 2 – Ouvrir les spécifications de l'association et accéder aux paramètres de l'extrémité qualifiée

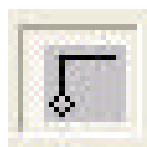


### 3 – Ajouter les qualifications



## Agrégation

### 1 – Sélectionner l'outil d'agrégation dans la barre d'outils



2 – partir de l'agrégué et aller vers l'agregat



3 – relâcher le bouton de la souris pour matérialiser l'agregation



4 – les spécifications sont les même que pour une association classique

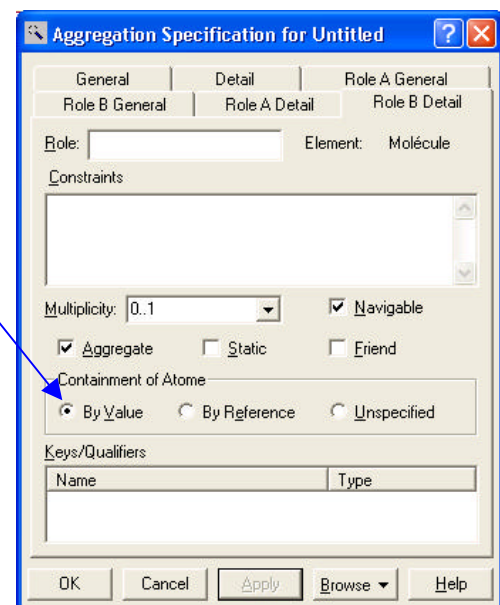
### Composition

1 – Créer une agrégation



2 – ouvrir les spécification de l'agregation et aller voir les paramètres détaillés du composite

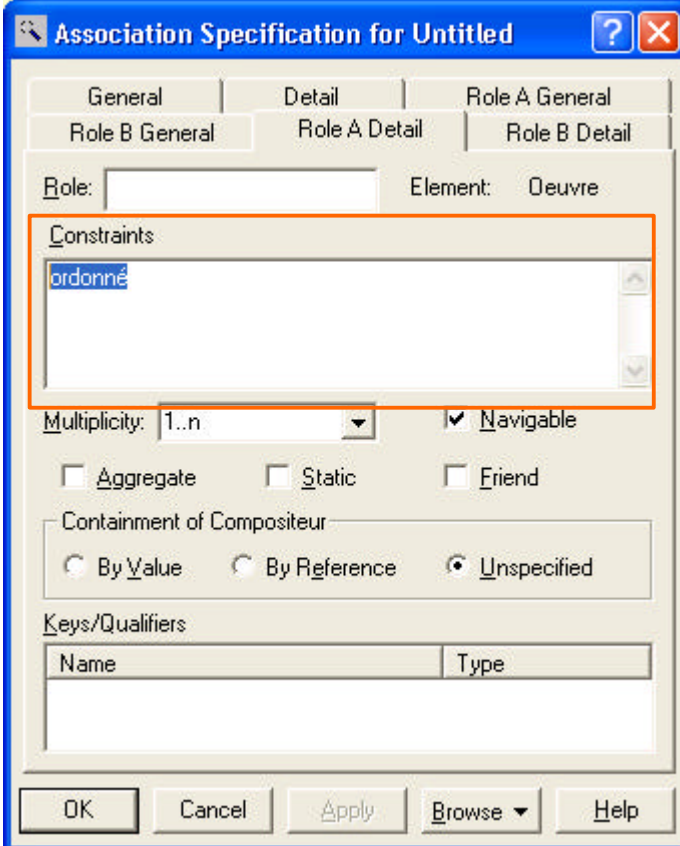
Sélectionner by Value dans  
Containment of « composant »



**ATTENTION :** Rational Rose ne teste pas les cardinalités dans le cas d'une composition !!!  
Si dans l'exemple précédent Molécule avait une cardinalité en 1..n il aurait accepté la composition qui aurait pourtant été erronée.

### *Contraintes sur les associations*

- 1 – Ouvrir les spécifications de l'association
- 2 – Aller voir les paramètres de l'extrémité que l'on souhaite contraindre
- 3 – Décrire les contraintes dans le cadre réservé (**Constraints**)



- 4 – Cliquer sur OK

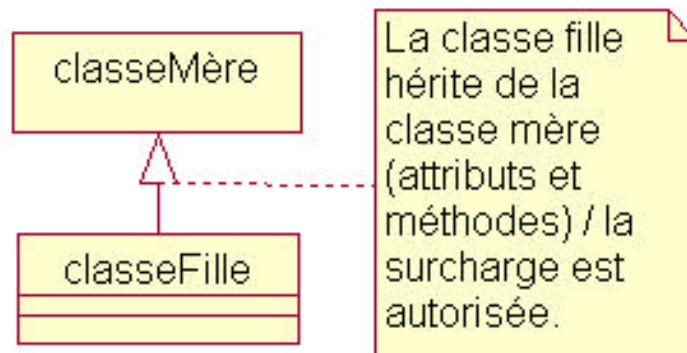


### iii. Généralisation/Spécialisation

Rational Rose permet d'appliquer la généralisation-spécialisation dans un diagramme de classes.



L'outil est l'opérateur **Generalization**.



Les outils nécessaires à la réalisation d'un diagramme de classes sous Rational Rose viennent d'être décrits assez précisément. Pour les autres diagrammes, les grands principes sont les mêmes, nous ne présenterons que les actions spécifiques à la réalisation de chaque type de diagramme.

## b. Diagramme de collaboration

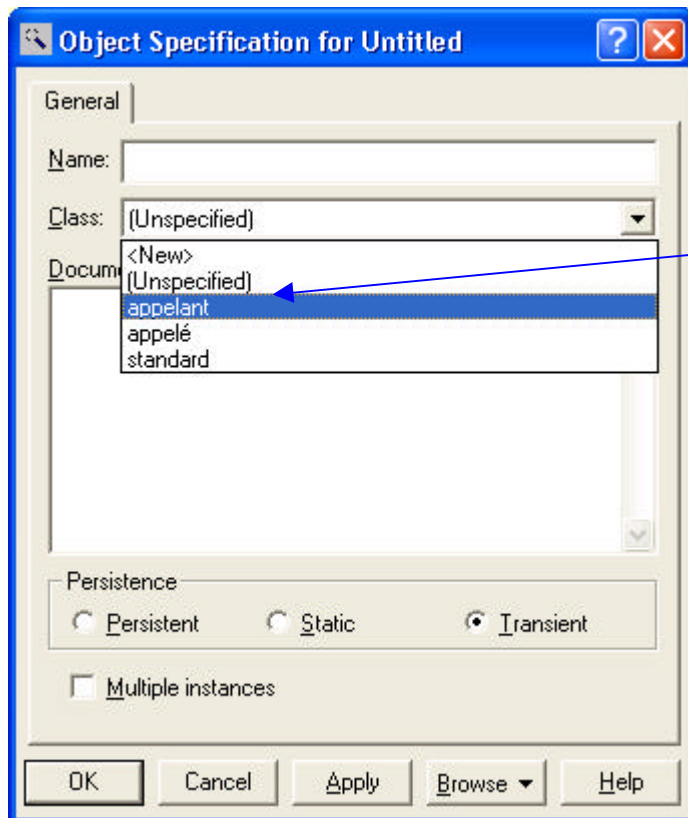
Un diagramme de collaboration met en évidence les échanges de messages entre les objets pendant l'exécution d'un scénario.

Cette fois-ci le diagramme de collaboration sera créé dans le répertoire **Use Case View** du navigateur. Il s'agit en effet de l'observation des événements d'un scénario particulier.

- 1 – Faire un clic droit sur **Use Case View** dans le navigateur et sélectionner **New>Collaboration Diagram**
- 2 – Une nouvelle fenêtre de diagramme s'ouvre présentant la barre d'outils spécifique aux diagrammes de collaboration que nous avons présenté au début de ce tutorial.
- 3 – Cliquer sur le bouton **Object** puis placer les objets sur le diagramme



#### 4 – Ouvrir les spécification des objets **Open Spécification...**



**Choisir la classe à laquelle appartient l'objet**

: appellant

: standard

: appelé

#### 5 – Placer les liens entre les objets pour permettre le transit des messages

opérateur **Object Link**



: appellant

: standard

: appelé

#### 6 – Placer les messages l'un après l'autre dans l'ordre d'exécution sur les liens entre les objets

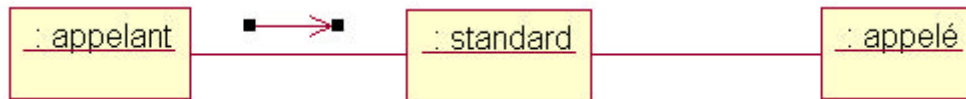
opérateur **Link Message**



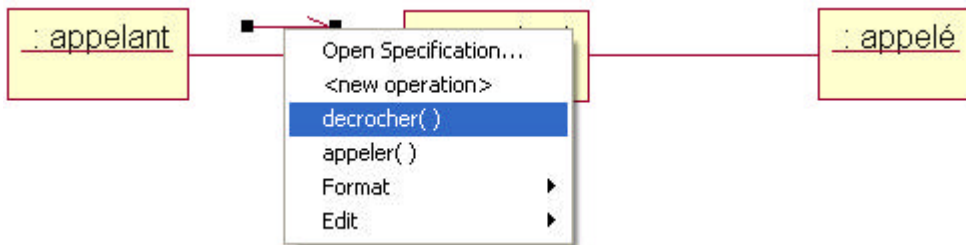
opérateur **Reverse Link Message**



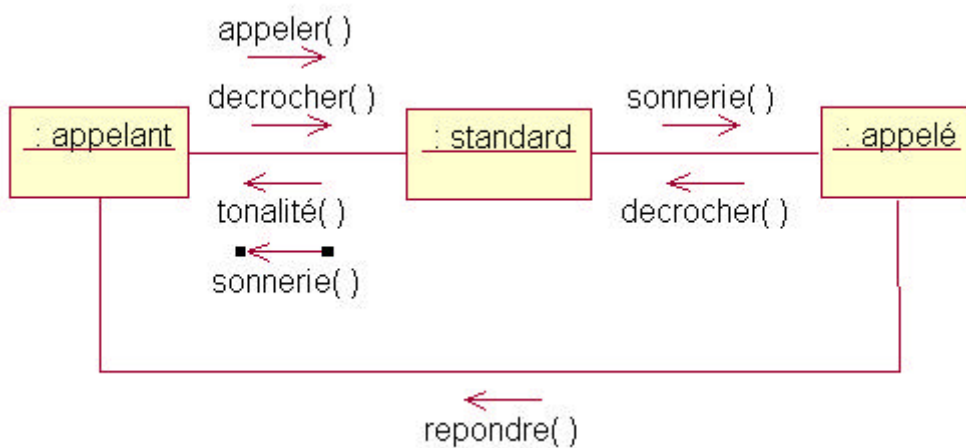




Choix du message (clic droit)



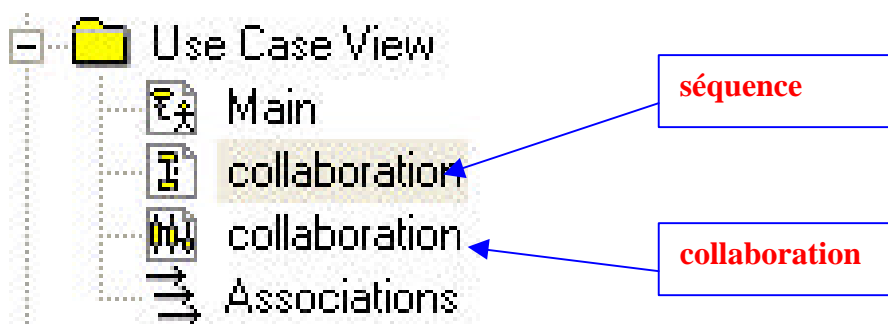
et on continue jusqu'à obtenir le diagramme complet :

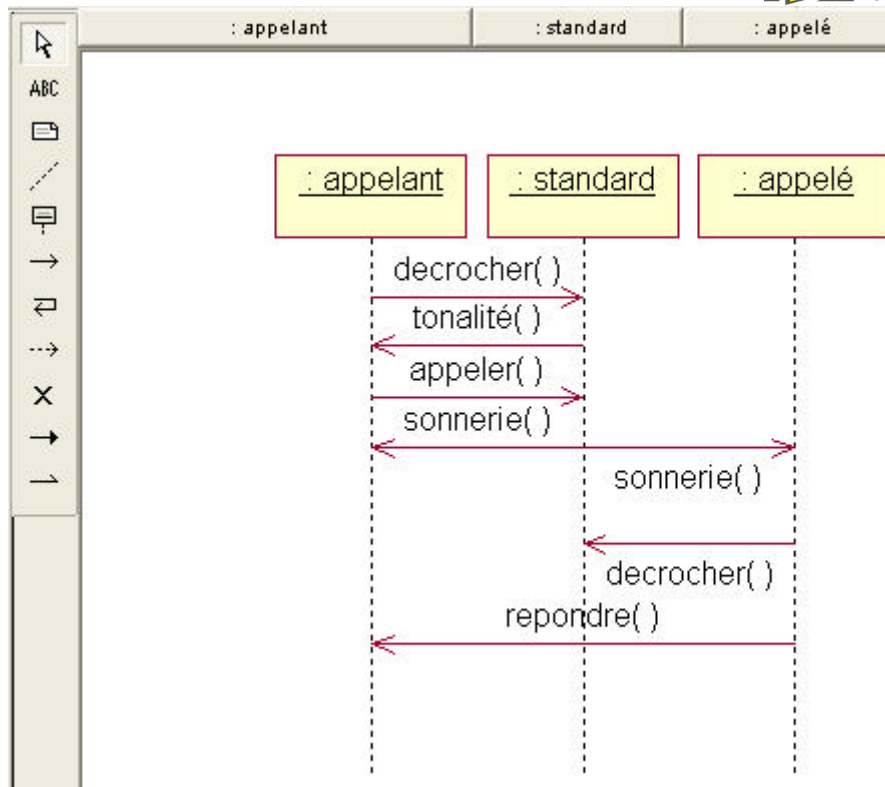


## c. Diagramme de séquence

Une fois le diagramme de collaboration réalisé, le diagramme de séquence peut s'obtenir automatiquement par appui sur la touche **F5** du clavier.

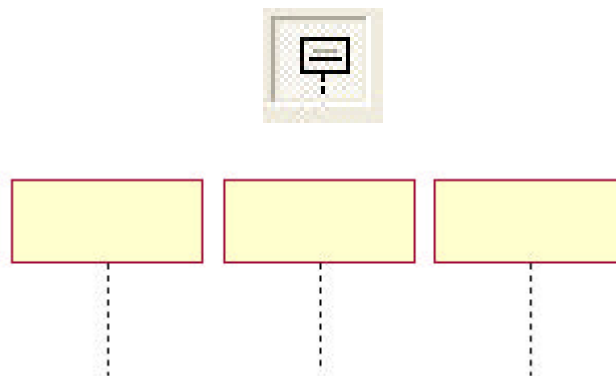
Le diagramme de séquence ainsi obtenu est placé dans le répertoire **Use Case View** dans le navigateur et porte le même nom que le diagramme de collaboration qui a permis de le générer automatiquement.



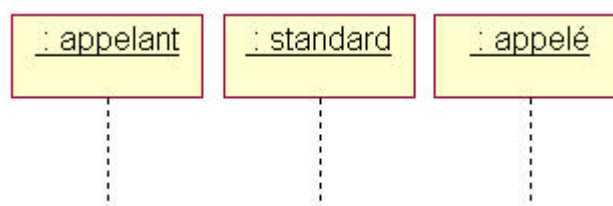


Si l'on n'a pas au préalable réalisé de diagramme de collaboration, on crée un diagramme de séquence dans **Use Case View** (clic droit puis **New>Sequence Diagram**).

1 – On commence par placer les objets avec l'opérateur **Object**



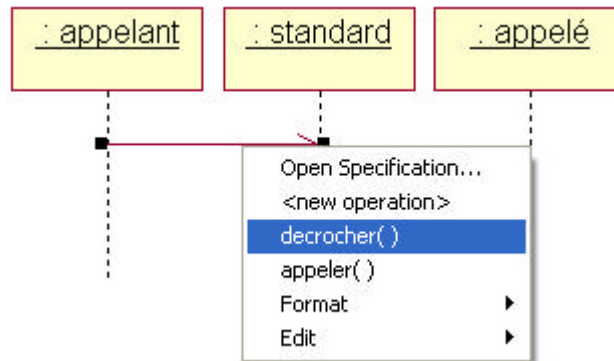
2 – on complète les spécifications de ces objets (double-clic ou clic droit et **Open Specification...**) → les paramètres de spécification sont les mêmes que dans le diagrammes de collaboration



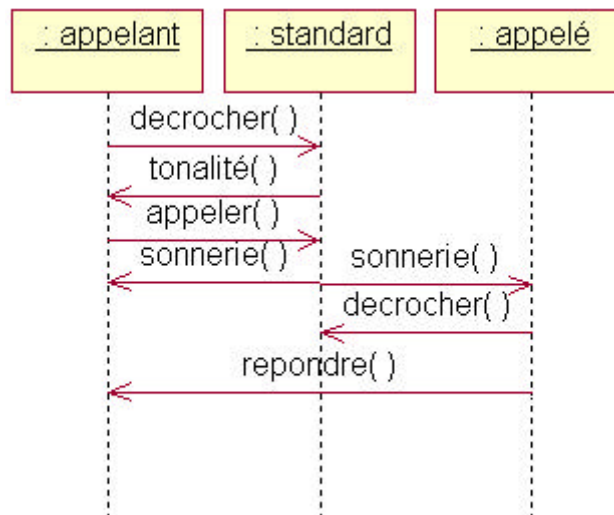
3 – on met en place les messages à l'aide du bouton **Object Message**



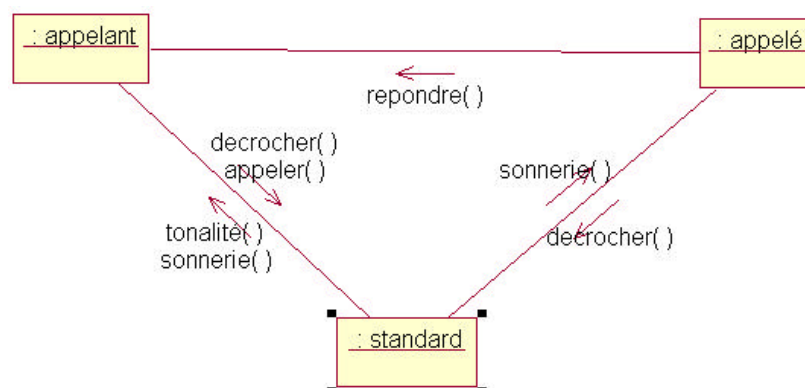
4 – On choisit le message



5 – On continue la mise en place des messages séquentiels, la ligne de vie des objets actifs est allongée automatiquement par Rational Rose.



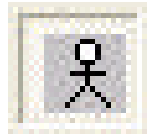
L'appui sur la touche **F5** donne le diagramme de collaboration.



## d. Use Case (Diagramme de cas d'utilisation)

1 – Commencer par créer dans le répertoire **Use Case View** un diagramme de cas d'utilisation (clic droit puis **New>Use Case Diagram**), lui donner un nom approprié.

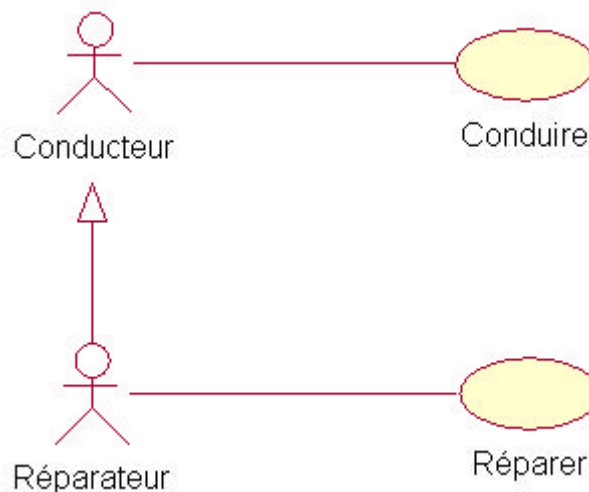
2 – Utiliser le bouton **Actor** pour placer les acteurs



3 – Utiliser le bouton Use Case pour placer les actions réalisées par les acteurs

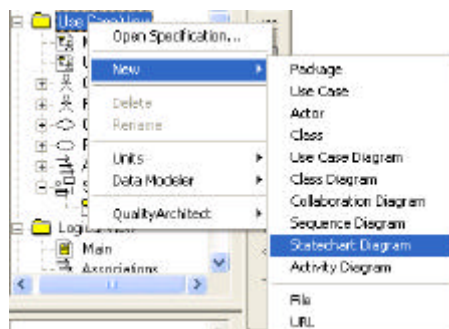


4 – Mettre en place les associations entre les objets comme dans le cas du diagramme de classes.



## e. Statecharts

Rational Rose permet également la création de statecharts. Comme dans le cas des autres diagrammes, il faut commencer par créer un nouveau diagramme dans **Use Case View**.

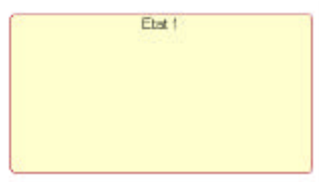


## 1 – Création d'états

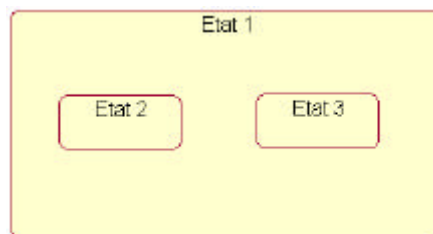
- Cliquer sur le bouton **State**



- Placer l'état dans la fenêtre du diagramme



- Répéter l'opération pour créer les états nécessaires



## 2 – Affecter la transition par défaut à l'aide de l'opérateur **Start State** puis de l'opérateur **State Transition**

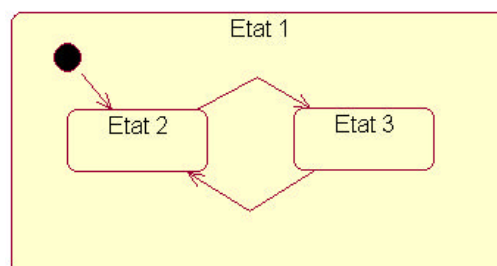
### Start State



### State Transition

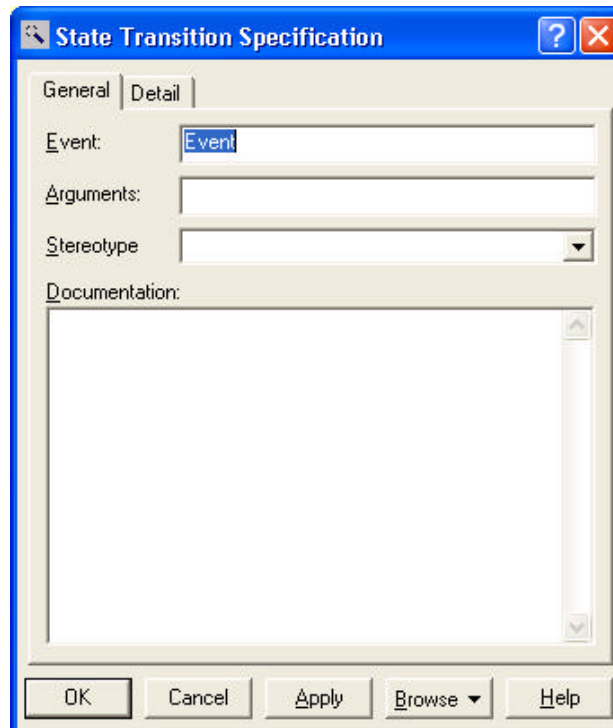


## 3 – Mettre en place les autres transitions avec l'opérateur **State Transition**

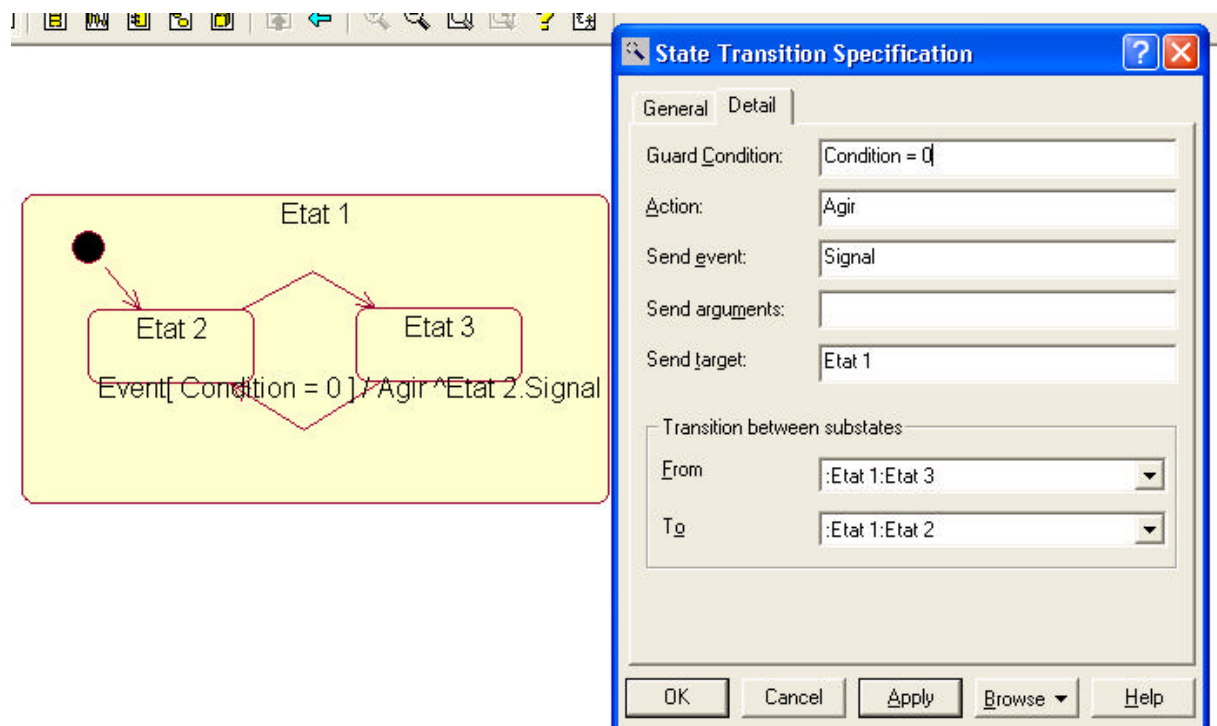


#### 4 – Mise en place d'une condition

- Ouvrir les spécifications de la transition à laquelle on souhaite affecter la condition
- Lui affecter un événement



- Paramétrer **Detail**



### III. Observations liées à notre propre utilisation

Rational Rose est un outil offrant d'énormes possibilités. Nous n'avons ici présenté qu'une petite partie des fonctionnalités de cet outil. La suite de ce tutorial présente nos remarques et conseils directement lié à l'utilisation que nous en avons eu.

#### a. Astuces et remarques importantes

- Quelques raccourcis clavier bien utiles

| Racourcis       | Action  |
|-----------------|---|
| F1              | Affiche l'aide de l'objet actif   |
| F4              | Ouvre les options générale du modèle  |
| Ctrl + S        | Sauvegarder   |
| Ctrl + D        | Delete From Model = efface définitivement l'objet   |
| Ctrl + F        | Rechercher une expression, un nom. Si la case Case Sensitive est cochée, un seul résultat apparaît sinon on obtient toutes les occurrences du mot recherché dans le modèle. |
| Ctrl + U        | Dézoomer  |
| Ctrl + I        | Zoomer  |
| Ctrl + Shift +L | Mets les lignes sélectionnées à angle droit   |
| Ctrl + B        | Ouvre les spécifications de l'objet sélectionné   |

- Manipulations optimisant le temps de travail

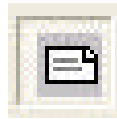
Lorsque l'utilisateur souhaite créer plusieurs fois de suite le même type d'objet il est très intéressant de verrouiller l'opérateur avec le bouton verrou.



Ne pas hésiter à abuser du clic droit qui donne généralement un accès direct aux principales fonctions en relations avec l'objet ainsi sélectionné.



Ne pas oublier d'ajouter des commentaires dans les diagrammes afin de faciliter la relecture et la compréhension du diagramme par une tierce personne.



#### - Remarques

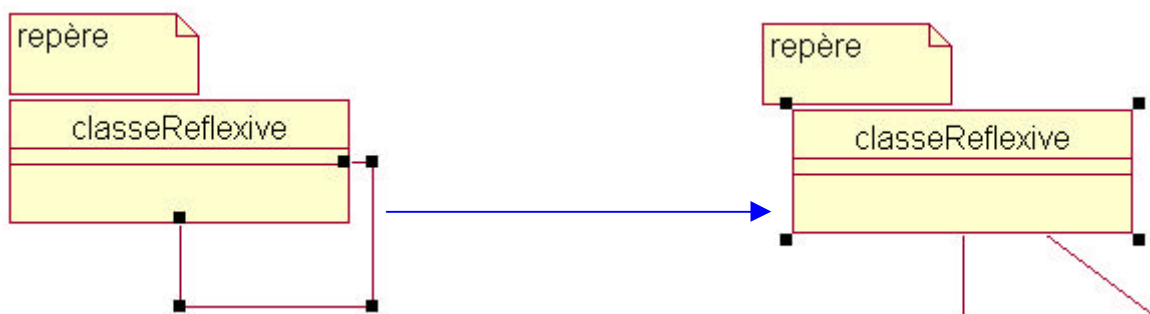
Lors de la création d'une classe, si le nom spécifié existe déjà dans le modèle, une copie exacte de l'ancienne classe est immédiatement effectuée.

Si l'utilisateur désire changer les droits d'accès à un attribut par exemple, il peut cliquer sur le petit logo devant cet attribut (dans la classe), et sélectionner les nouveaux droits.

## b. Défauts observés

Si Rational Rose est un outils extrêmement complet et puissant, il reste de nombreux bugs qui gâchent un peu son utilisation. Les principaux reproches que nous pouvons exprimer quant à notre utilisation de Rational Rose concernant l'interface graphique.

Lors de la création d'une association réflexive, nous avons vu un moyen de faire une mise en page automatique et très pratique, mais lorsqu'on bouge une classe avec une association réflexive, cette mise en page est modifiée et rend moins clair le diagramme.



Rational Rose ne vérifie pas qu'une composition est bien de multiplicité 0..1 et accepte donc une multiplicité 1..n.



Lors de la réalisation de diagramme de collaboration par exemple, les messages le long d'un même liens sont systématiquement superposés les uns sur les autres rendant la lecture impossible. L'utilisateur est alors obligé de repositionner tous les messages à la main...



Les diagrammes de séquence ne permettent pas de voir les paramètres ou les valeurs de retour des méthodes.

Les structures de contrôle dans les diagrammes de séquences n'existent pas, il faut les écrire sous forme de texte.

Dans la composition de statecharts, nous n'avons pas trouvé de fonction permettant de générer une ligne verticale représentative d'un état ET.

Enfin, lors de la composition d'un diagramme de séquence, les messages correspondent bien aux méthodes du destinataire mais il n'y a aucune vérification d'une association entre l'expéditeur et le destinataire dans le diagramme de classes Main.

## IV. Lien et document intéressants sur Rational Rose

[www.rational.com](http://www.rational.com)

Mastering UML with Rational Rose 2002 (Wendy Boggs / Michael Boggs)