TP

Objectif

L'objectif des 6 séances de TP est d'appliquer la démarche introduite en cours. Ces séances s'articuleront de la manière suivante :

Séance 1 : Modélisation métier : processus et concepts métier

Séance 2 : Capture des besoins fonctionnels : diagramme des cas d'utilisation

Séance 3 : Description des cas d'utilisation : description textuelle et/ou diagrammes de séquence de niveau système

Séance 4 : Réalisation des cas d'utilisation : diagrammes de séquence détaillés

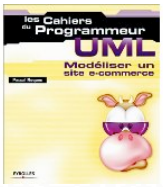
Séance 5 : Diagrammes de classes participantes ; structuration en couches et paquetages

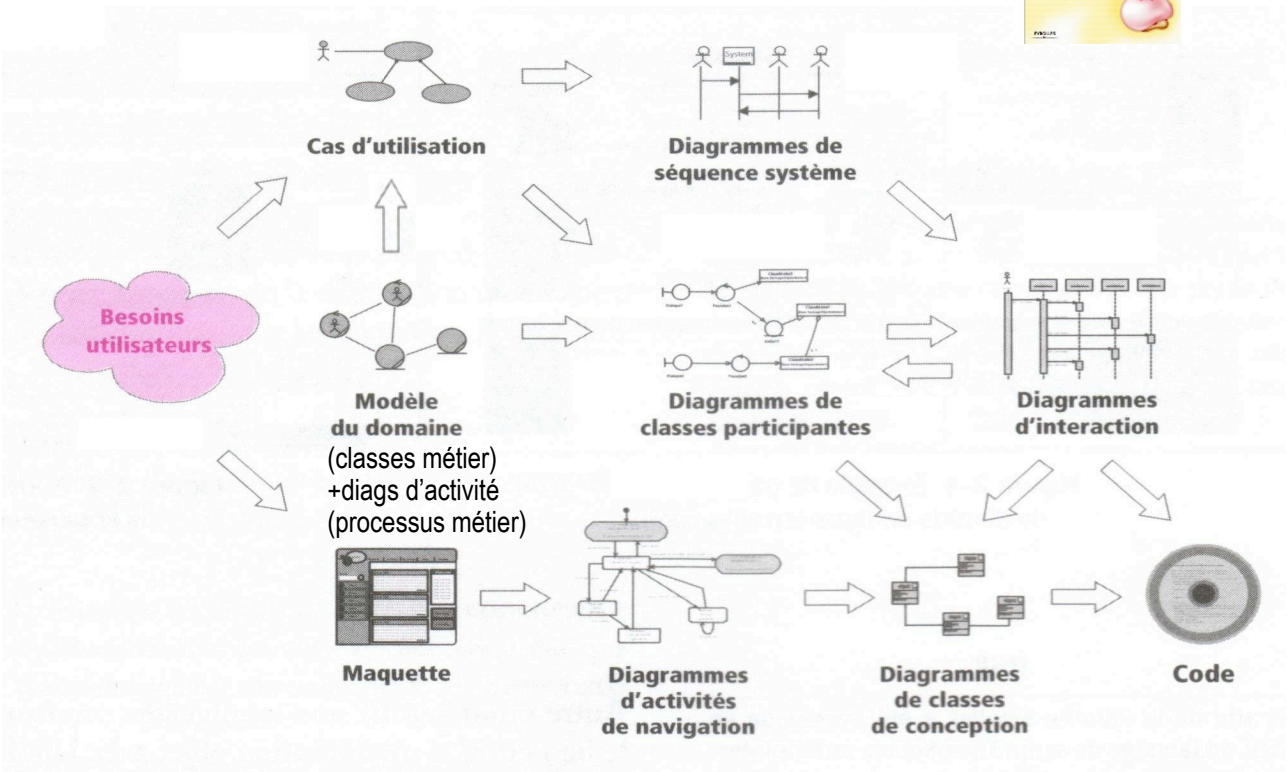
Séance 6 : Diagrammes états/transitions.

Rappel du schéma de la démarche de modélisation

Procéssus de modélisation adapté du livre :

Pascal Roques *UML : Modéliser un site e-commerce* Eyrolles 2002





Sujet - organisation du covoiturage au sein d'une communauté universitaire

Les six séances de TP porteront sur la modélisation d'une application pour gérer le co-voiturage au sein d'une université. Plusieurs applications généralistes proposent des services de ce type (e.g. site www.covoiturage.fr), on propose ici de modéliser une application destinée à organiser le covoiturage en confiance, entre étudiants et personnels d'une même université.

Des études montrent que le manque de confiance entre partenaires constitue un frein majeur pour le covoiturage : peur d'être bloqué au bureau (pas de garantie de retour), peur de l'inconnu (qui incite à le limiter à des communautés fermées), peur des comportements routiers à risque, etc.

Dans notre contexte, la confiance s'établira sur la base

- de l'appartenance à une même communauté
- de la réputation des partenaires (statistiques sur le nombre de trajets effectués et sur les appréciations recueillies, tant pour les conducteurs que pour les passagers).

Pour le premier point, le système s'appuiera sur le mécanisme d'identification en vigueur à l'université, qui ne devra donc pas être modélisé ici (mais avec lequel l'interfaçage devra se faire de manière explicite)

Pour le second point, le système devra proposer un dispositif d'évaluation par les pairs, au sein de la communauté.

Le système devra également proposer des solutions pour garantir le retour (i.e. trouver des solutions alternatives) et s'adapter aux défaillances qui peuvent être anticipées (un partenaire annonce qu'il ne pourra pas effectuer le trajet) ou à celles qui sont découvertes de manière fortuite (rendez-vous manqué, à l'heure et au lieu prévus).

Les fonctions essentielles d'un tel système traiteront notamment :

- des offres de trajet, faites par les conducteurs,
- des recherches de trajet, faites par les passagers,
- des profils des partenaires (conducteurs et passagers),
- des évaluations des partenaires (conducteurs et passagers).

Evolutions de la spécification

S'il y a des points sur lesquels vous sentez avoir besoin de faire des choix, vous êtes autorisés à les faire (éventuellement avec accord de votre enseignant de TP). Tous les évolutions par rapport à la spécification proposée doivent être documentées.

Fournitures

Le travail peut s'effectuer en équipe (fixe) de jusqu'à 3 étudiants (du même groupe de TP).

A la fin du mini-projet (la date limite sera communiquée au cours des dernières séances de TP) vous devez déposer sous moodle une archive (zip) contenant

- La documentation succincte de la modélisation (choix effectués, évolutions de la spécification, diagrammes représentatifs, descriptions textuelles des cas d'utilisation, etc)
- Le modèle UML

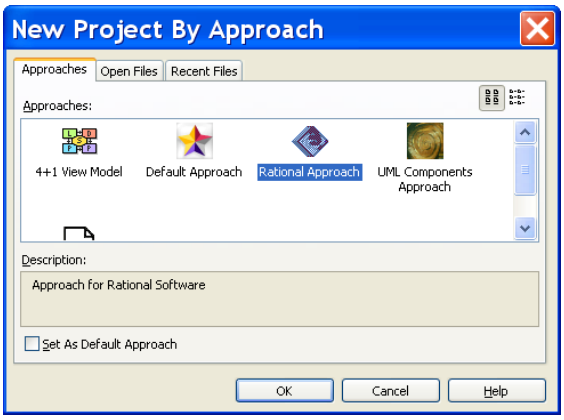
I. TP1 (modélisation métier)

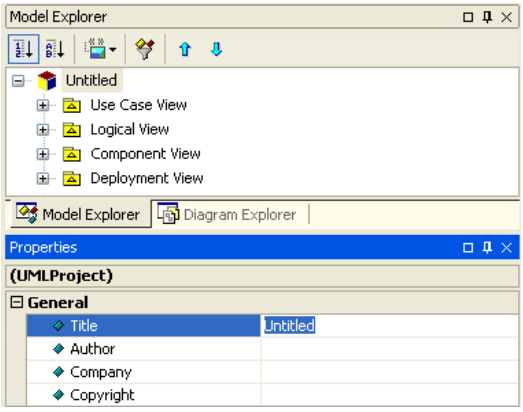
StarUML :

Lancer StarUML, puis choisir l'approche : « Rational Approach ».

*StarUML crée un **projet**, structuré en 4 **modèles** (ou vues), avec les diagrammes principaux correspondants à l'intérieur (vides).*

PENSER A ENREGISTRER REGULIEREMENT VOTRE TRAVAIL !

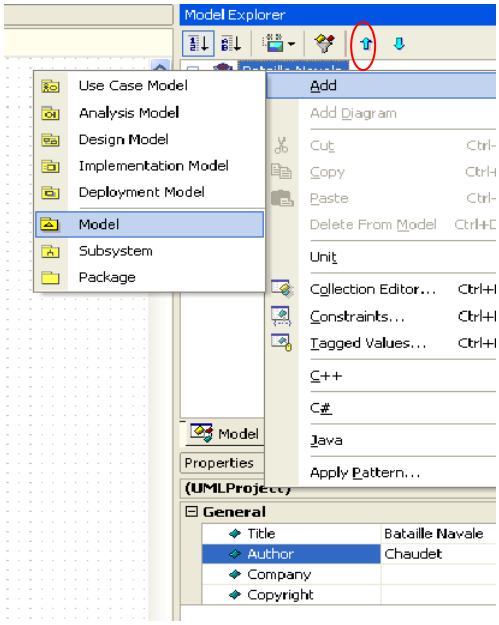


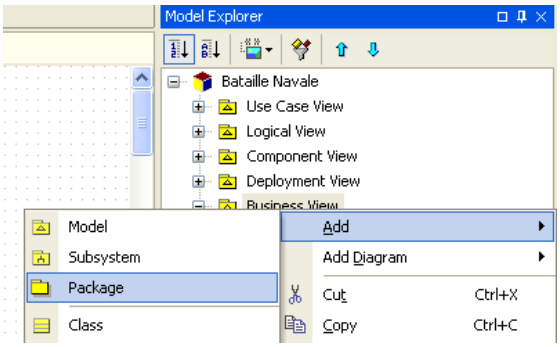


Personnaliser votre projet en modifiant ses propriétés (titre et auteur).

Rajouter dans le projet un modèle « Business View » (clic droit sur le projet, Add, puis Model).

Faire remonter ce modèle jusqu'en première position (bouton bleu flèche vers le haut), de manière à ce que l'ordre des modèles soit conforme à la démarche présentée en cours.





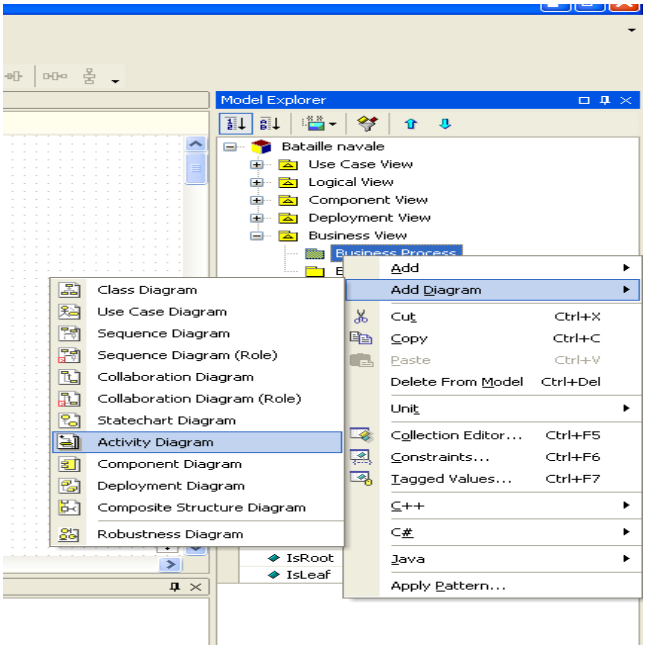
Créer deux **paquetages** dans cette nouvelle vue : « Business Process » et « Business Classes » (clic droit sur « Business View », Add, puis Package).

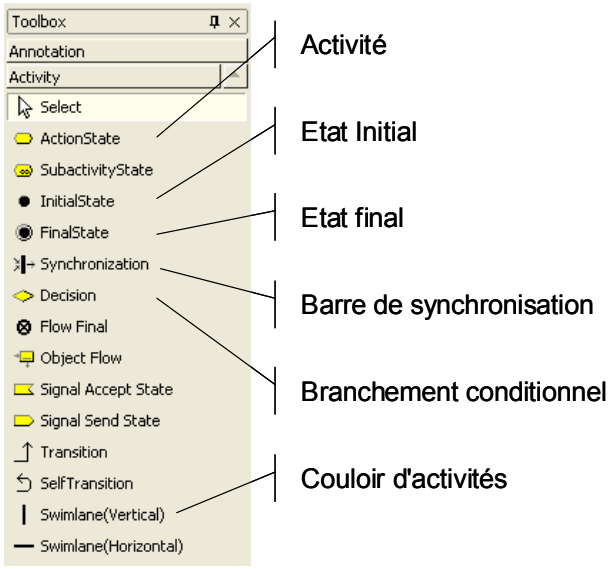
Par la suite, nous travaillerons successivement sur les modèles « Business View », « Use Case View » puis « Logical View ».

Diagramme d'activité du domaine

StarUML :

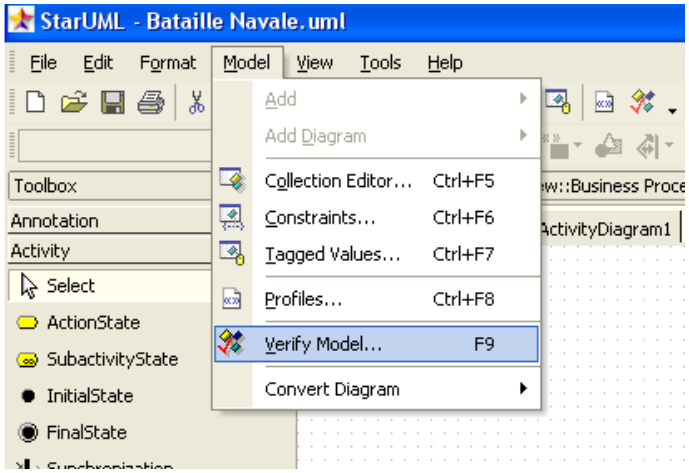
Dans le paquetage « Business Process » du modèle « Business View », ajouter un **diagramme d'activité** (clic droit sur le paquetage, Add Diagram, Activity Diagram). Dans les propriétés, donner un nom significatif au diagramme.





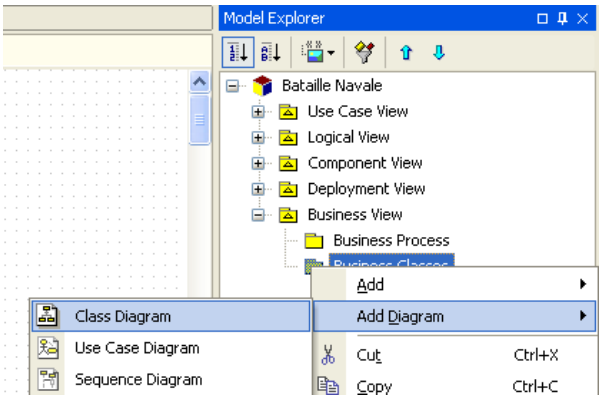
Travail à réaliser :

- Identifier le ou les responsables d'activité que vous souhaitez représenter (couloirs).
- Etablir le(s) diagramme(s) d'activité.
- Vérifier votre modèle et corriger si besoin (l'ensemble du projet peut être vérifié à tout moment : Model > Verify Model …)



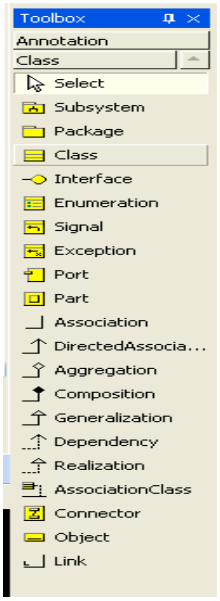
- Enregistrer votre projet sous le nom <nom de fichier>.

Diagramme des classes du domaine (ou classes métier)



StarUML :

Dans le paquetage « Business Classes » du modèle « Business View », ajouter un **diagramme de classes** (clic droit sur le paquetage, Add Diagram, Class Diagram). Dans les propriétés, donner un nom significatif au diagramme.



Travail à réaliser :

- Trouver les différentes classes « métier ».
- Etablir le(s) diagramme(s) de classes métier.
- Vérifier votre modèle et corriger si besoin (Model > Verify Model …).
- Enregistrer votre projet. C'est la version que vous reprendrez lors du prochain TP.
- Enregistrer une copie de votre projet sous le nom <nom de fichier>_VueMétier, afin de figer l'état actuel de celle-ci et ne pas risquer de l'altérer dans la suite de votre travail.

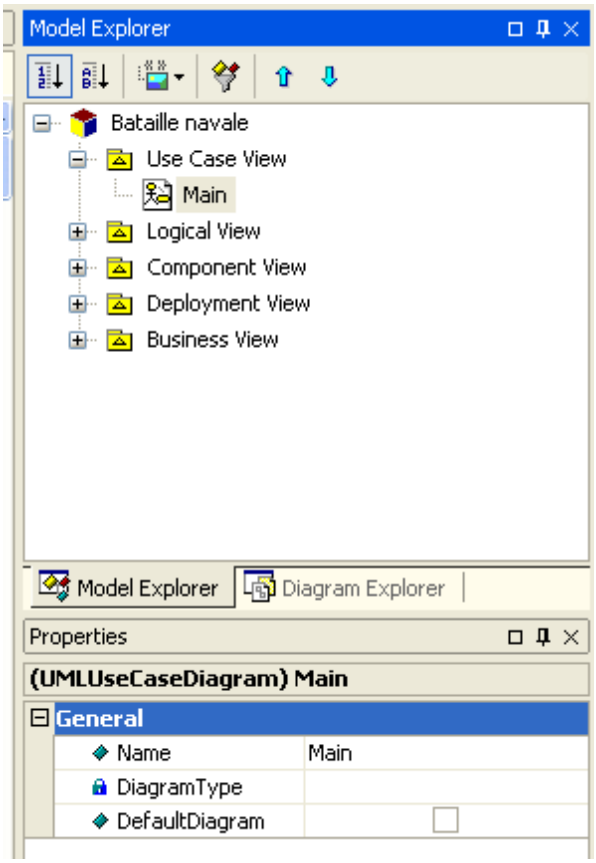
II. TP2 (définition du logiciel)

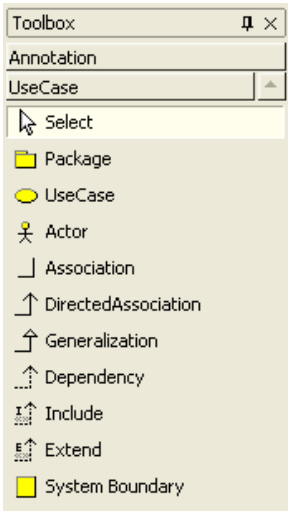
Diagramme de cas d'utilisation

StarUML :

Ouvrir le projet enregistré en fin de TP1 sous le nom <nom de fichier>.

Dans le modèle « Use Case View », renommer le **diagramme de cas d'utilisation** existant en lui donnant un nom significatif.





Travail à réaliser :

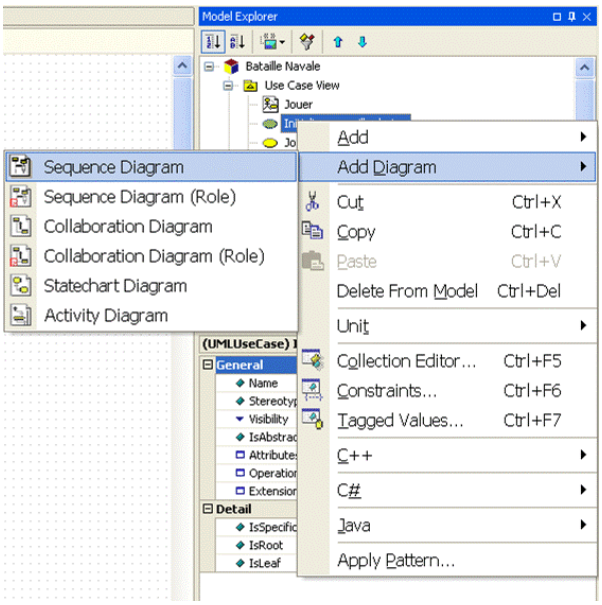
- Trouver les différents acteurs et les différents cas d'utilisation pour chacun d'eux.
- Etablir le diagramme des cas d'utilisation.
  Si nécessaire, vous pouvez organiser les cas d'utilisation en plusieurs paquetages et créer plusieurs diagrammes de cas d'utilisation dans ces paquetages ; dans ce cas, il faudra donner les liens de dépendance entre les paquetages (diagramme de paquetages).
- Vérifier votre modèle et corriger si besoin (Model > Verify Model …).
- Documentez les éléments du modèle (fenêtre Documentation) puis générer le document Use Case Specification (Tools>StarUML Generator) et compléter le .doc produit.
- Enregistrer votre projet.

III. TP3 (définition du logiciel)

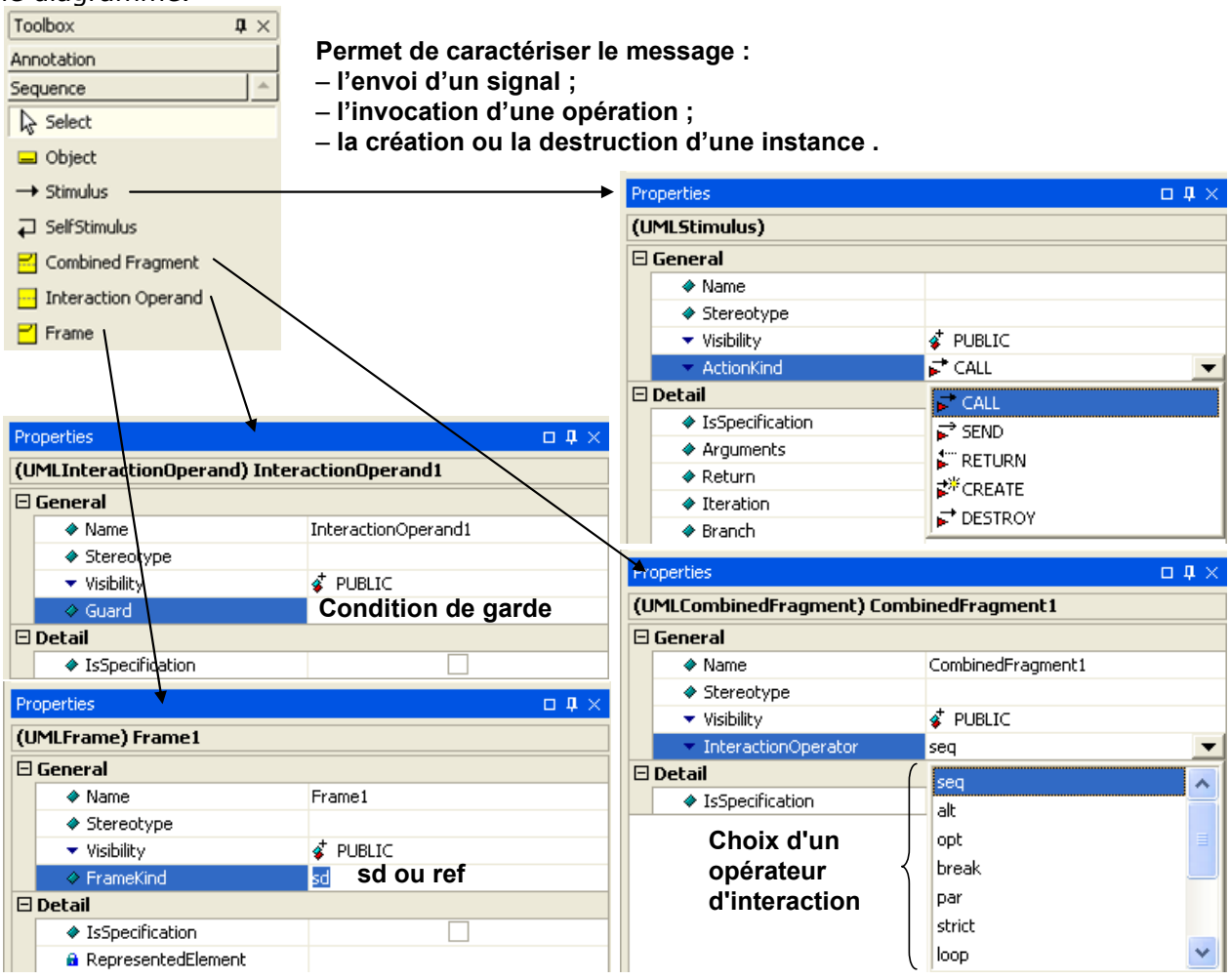
Diagrammes de séquence système

StarUML :

Sous chaque cas d'utilisation identifié au TP précédent (dans le modèle « Use Case View »), ajouter un **diagramme de séquence** (par clic droit). Dans les propriétés, donner un nom significatif à ce diagramme, reflétant à la fois le type de diagramme et le cas d'utilisation qu'il définit.



Rappel : un diagramme de séquence de niveau système servant à définir un cas d'utilisation, il ne doit faire apparaître que l'acteur à l'initiative de ce cas d'utilisation, les acteurs secondaires éventuels et l'objet « Systeme » (à ce niveau, le système est vu comme un tout). Les acteurs ayant été créés à l'étape précédente, il suffit de les faire glisser sur le diagramme.



Travail à réaliser :

- Spécifier les échanges entre l'acteur (ou les acteurs) et le système.
- Vérifier votre modèle et corriger si besoin (Model > Verify Model …).
- Enregistrer votre projet. C'est la version que vous reprendrez lors du prochain TP.
- Enregistrer une copie de votre projet sous le nom <nom de fichier>_VueDéfinitionCas, afin de figer l'état actuel de celle-ci.

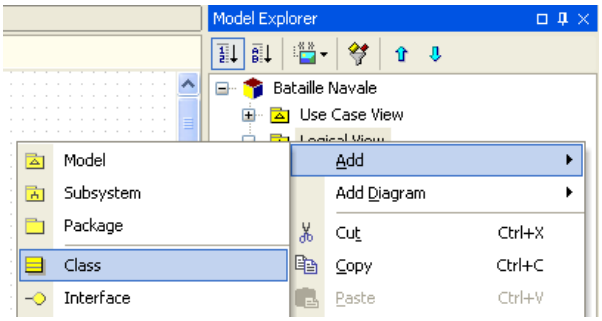
IV. TP4 (Réalisation des cas d'utilisation)

Diagrammes de séquence détaillés

StarUML :

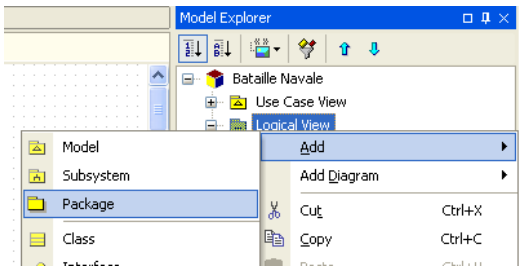
Ouvrir le projet enregistré en fin de TP2.

1. Dans le modèle « Logical View » :
   a. renommer le **diagramme de classes** existant en « Classes d'analyse »
   b. ajouter les **classes** d'analyse, indifféremment dans ce diagramme ou par clic droit sur la « Logical View ». Dans les propriétés, donner un nom à ces classes, ainsi que leur stéréotype (boundary, control, entity).

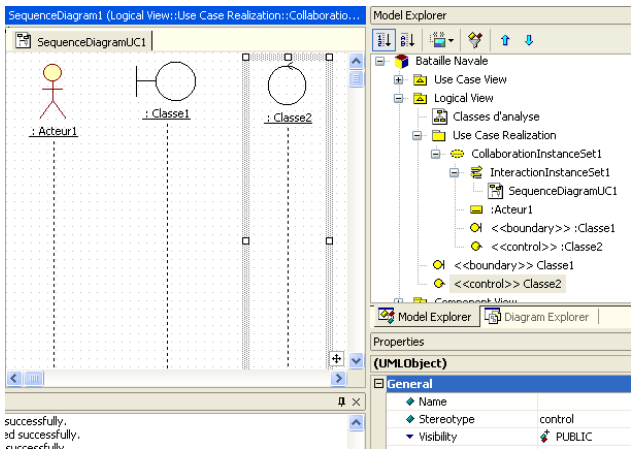


Les classes d'analyse peuvent être identifiées soit globalement dès le début de cette étape, soit au fur et à mesure de la réalisation des cas d'utilisation.

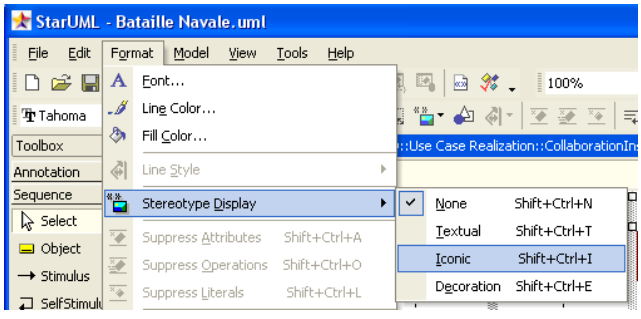
Rappel : les classes « entity » sont issues de la « Business View » (il suffit de les faire glisser dans la « Logical View »), tandis que les classes « boundary » et « control » sont issues de la « Use Case View » (application de règles simples pour commencer).



2. Dans le modèle « Logical View » :
   a. ajouter un **paquetage** (par clic droit) et lui donner pour nom « Use Case Realization ».
   b. Dans ce paquetage, créer un **diagramme de séquence** par cas d'utilisation (comme dans le dernier TP) et lui donner un nom significatif, reflétant à la fois le type de diagramme et le cas d'utilisation dont il exprime la réalisation.

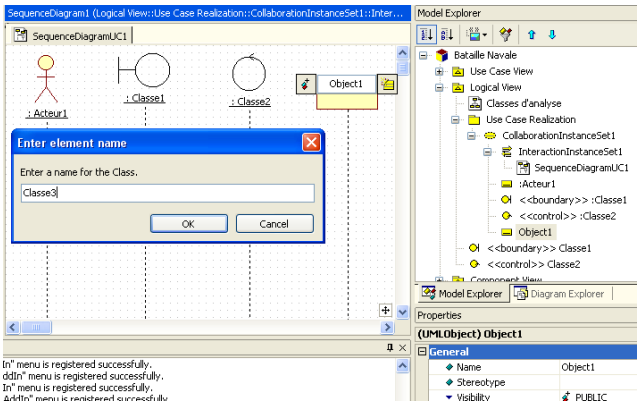


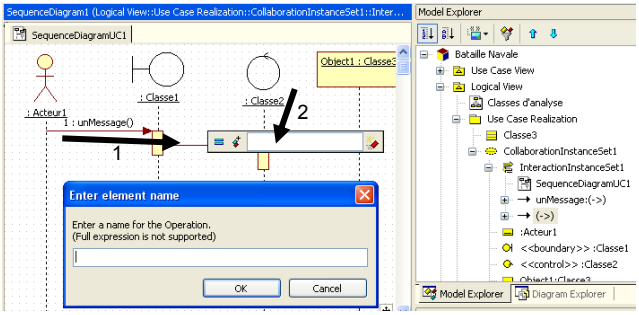
   c. Faire glisser sur le diagramme de séquence l'acteur à l'initiative de ce cas d'utilisation et les acteurs secondaires éventuels. Les objets impliqués étant des instances des classes d'analyse précédemment identifiées, il suffit de faire également glisser les classes correspondantes sur le diagramme de séquence. Il ne reste plus qu'à donner un nom à l'objet (si nécessaire), dans les propriétés.



   d. L'affichage des stéréotypes peut être modifié via le menu Format pour un ou plusieurs éléments sélectionnés au préalable.

Remarque : Il est possible de créer un **objet** dont vous n'avez pas encore identifié la classe ; dans ce cas, créer la **classe** correspondante en même temps en donnant son nom (la syntaxe UML autorise des objets sans spécification de la classe, mais celle-ci est nécessaire pour la génération de code), puis la déplacer dans le modèle « Logical View ». Il ne reste alors plus qu'à donner un nom à l'objet (si nécessaire), dans les propriétés.





e. Pour chaque message synchrone (appel de méthode), créer la méthode correspondante : celle-ci se rajoute automatiquement dans la classe de l'objet destinataire.

Travail à réaliser :

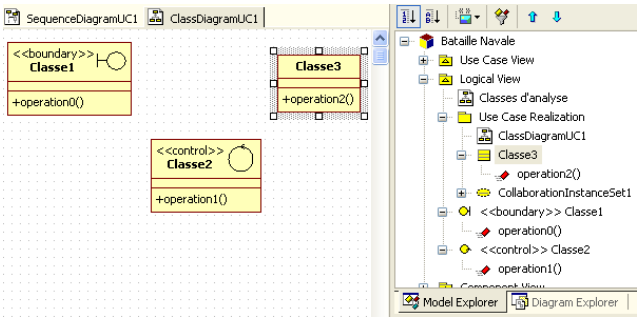
- Détailler chaque diagramme de séquence système par un diagramme de séquence détaillé, en prenant soin de créer les classes et les opérations des classes nécessaires aux communications entre objets.
  Rappel : la structure du diagramme de séquence détaillé doit être cohérente avec celle du diagramme de séquence de niveau système correspondant.
- Vérifier votre modèle et corriger si besoin (Model > Verify Model …).

V. TP5 (Réalisation des cas d'utilisation)

Diagrammes des classes participantes

StarUML :

Dans le paquetage « Use Case Realization » du modèle « Logical View », créer un **diagramme de classes** par cas d'utilisation et lui donner un nom significatif, reflétant à la fois le type de diagramme et le cas d'utilisation correspondant. Ces diagrammes sont destinés à montrer les classes des objets qui participent à la réalisation de ces cas.



Dans le diagramme de classes, faire glisser les classes des objets impliqués dans le diagramme de séquence détaillé correspondant, créées à l'étape précédente.

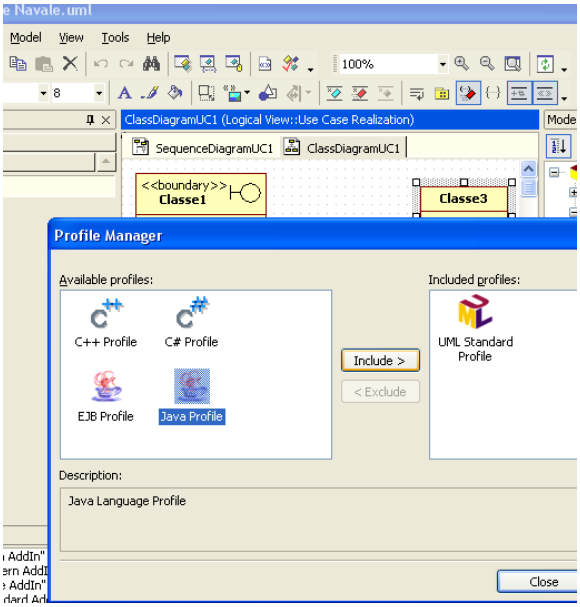
Travail à réaliser :

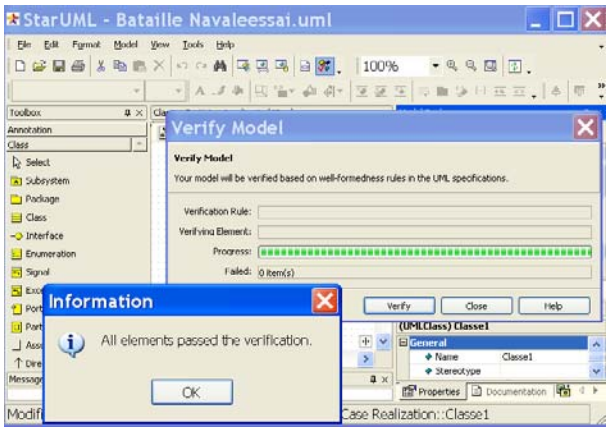
- Compléter les attributs de chacune des classes en fonction du diagramme de séquence détaillé correspondant.
- Définir les relations entre les classes (association, agrégation …) à partir des messages échangés entre les objets sur le diagramme de séquence détaillé correspondant. Compléter par les multiplicités, noms de rôle et indications de navigabilité.
- Compléter si besoin le diagramme en introduisant la généralisation/spécialisation.
- Vérifier votre modèle et corriger si besoin (Model > Verify Model …).
- Structurer le modèle en couches (paquetages stéréotypés « layer » : Interface, Application, Stockage au moins) et éventuellement paquetages à l'intérieur des couches.

Génération en Java

StarUML :

Modifier le profil, pour prendre en compte le langage cible : Model > Profiles …

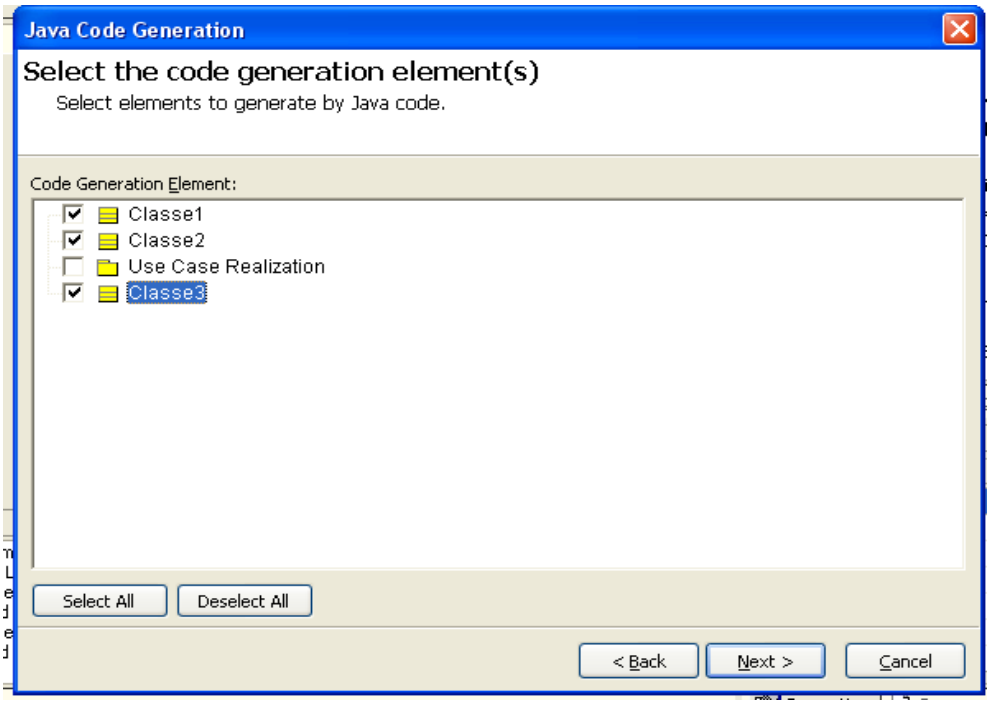


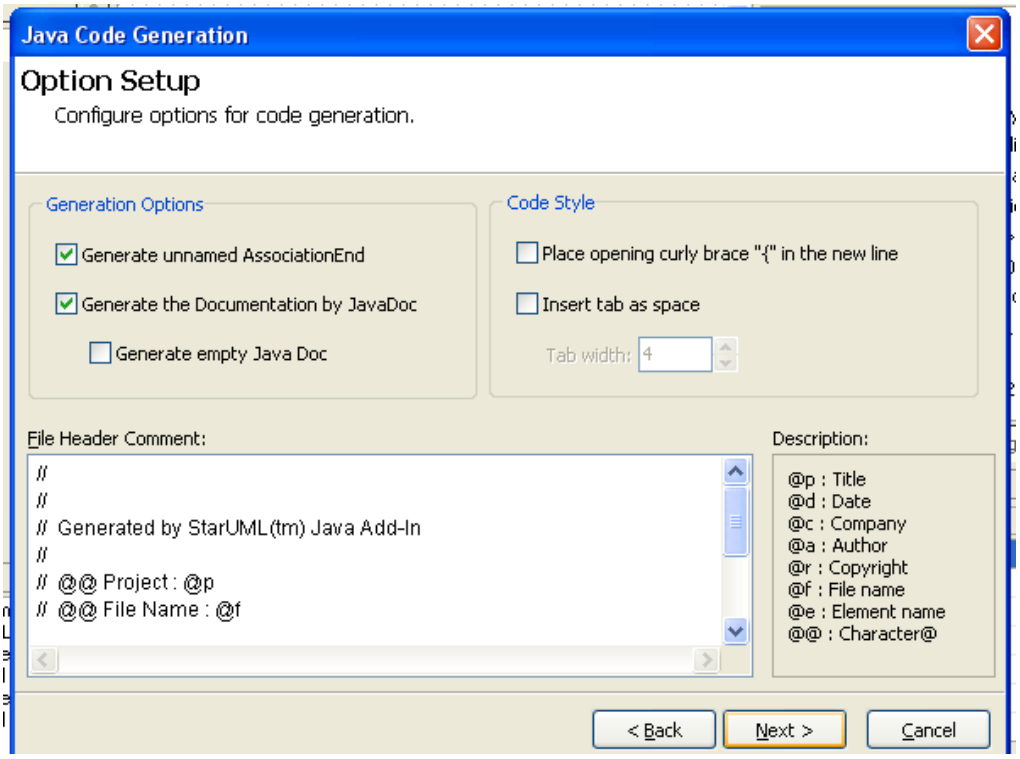


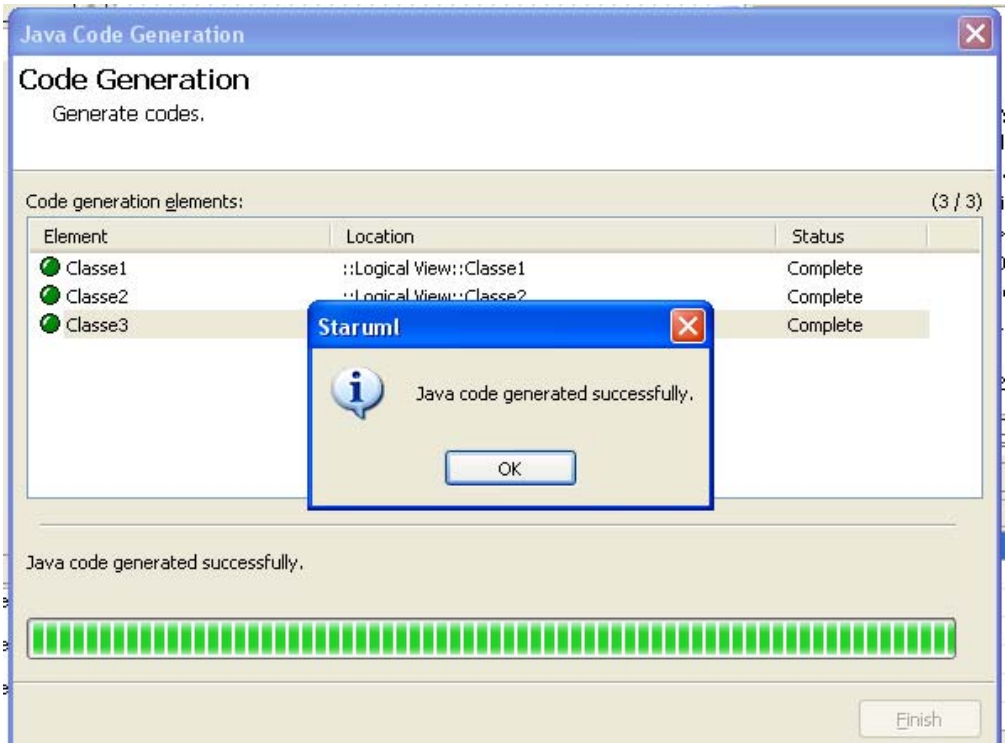
Vérifier votre modèle et corriger si besoin.

Travail à réaliser :

- Générer le code Java : Tools > Java > Generate Code…




Java Code Generation
Option Setup
Configure options for code generation.
Generation Options
Generate unnamed AssociationEnd
Generate the Documentation by JavaDoc
Generate empty Java Doc
Code Style
Place opening curly brace "{" in the new line
Insert tab as space
Tab width: 4
File Header Comment:
//
//
// Generated by StarUML(tm) Java Add-In
//
// @@ Project : @p
// @@ File Name : @f
Description:
@p : Title
@d : Date
@c : Company
@a : Author
@r : Copyright
@f : File name
@e : Element name
@@ : Character@
< Back
Next >
Cancel


Java Code Generation
Code Generation
Generate codes.
Code generation elements:
(3 / 3)
Element
Location
Status
Classe1
::Logical View::Classe1
Complete
Classe2
Complete
Classe3
Complete
Staruml
Java code generated successfully.
OK
Java code generated successfully.
Finish