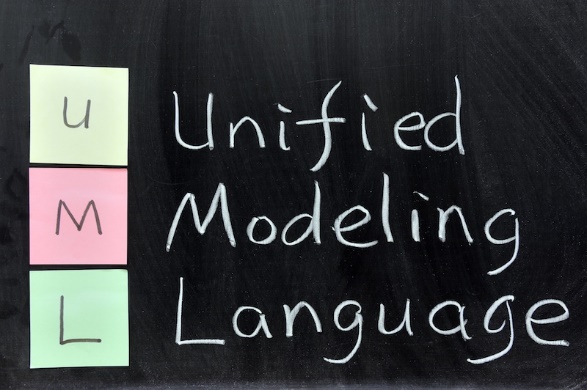
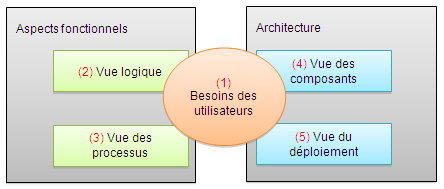
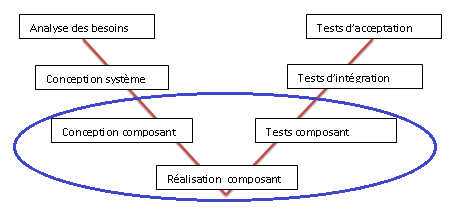
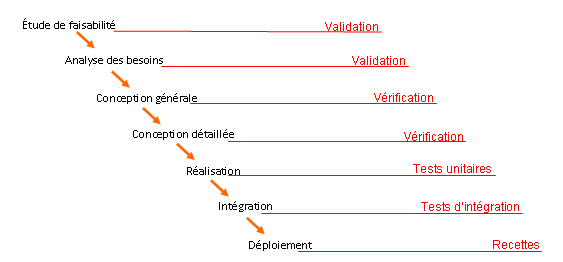
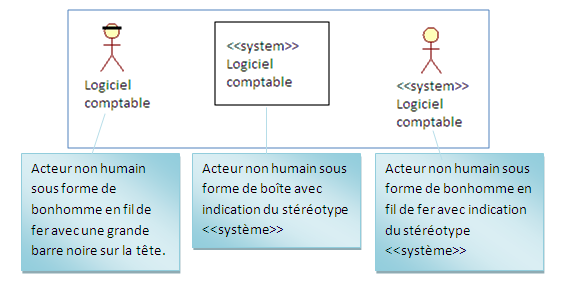
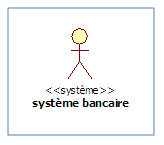
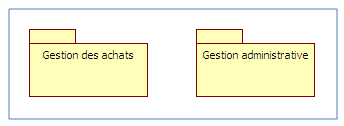
Analyse logicielle UML



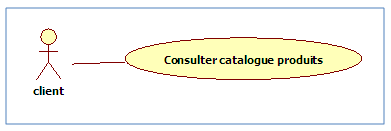
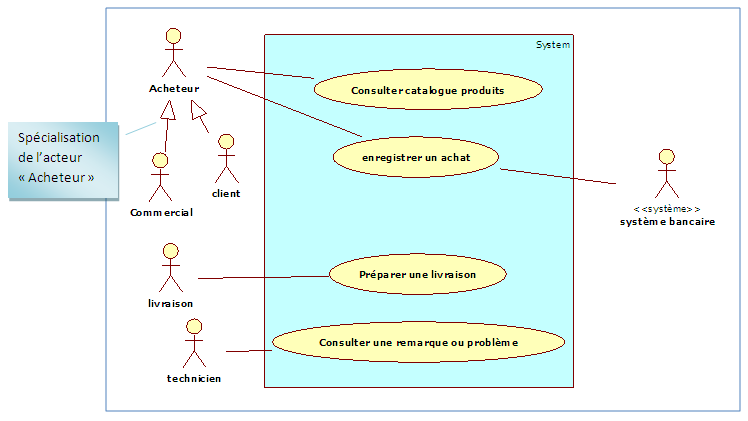
**Langage de modélisation unifié**

* L'Ulm est langage visuel
* Constitué d’un ensemble de schémas, appelés des **diagrammes**, qui donnent chacun une vision différente du projet à traiter
* Réaliser ces diagrammes revient donc à **modéliser les besoins**du logiciel à développer.
* Modéliser, c’est décrire de manière visuelle et graphique les besoins et, les solutions
* Il est important réaliser une étape d'analyse avant-projet au risque de retrouver des anomalies plus tard
* Démarche itérative c’est donc faire des allers-retours entre le plan initial et les modifications apportées par les acteurs du projet.
* Itération (un nouveau passage sur les diagrammes).
* Les deux aspects d'un diagramme :
* **Les aspects fonctionnels** : Qui utilisera le logiciel et pour quoi faire ? Comment les actions devront-elles se dérouler ? Quelles informations seront utilisées pour cela ?
* **Les aspects liés à l’architecture** : Quels seront les différents composants logiciels à utiliser (base de données, librairies, interfaces, etc.) ? Sur quel matériel chacun des composants sera installé ?
* **4+1 vues** : représentés par le schéma de 4 vues, axées sur les besoins des utilisateurs (parfois intitulé des cas d’utilisation), appelé **4+1 vues**
* 
* Système = logiciel = app
* Schéma 4+1 vues permet de définir qui utilisera le logiciel et pour quoi faire, comment les fonctionnalités vont se dérouler
* UML est constitué de 13 diagrammes qui représentent chacun un concept du système ou logiciel.
* Un logiciel peut être vu en considérant les aspects fonctionnels et les aspects d’architecture du logiciel. Ces deux aspects sont composés de 4 vues du logiciel à développer, organisés autour des besoins des utilisateurs. C’est le 4+1 vues.
* Chacune des 4+1 vues est constituée de diagrammes
* **Les étapes de développement logiciel**
* 
* Les recettes ne font pas référence à des recettes de cuisine, mais aux réunions formelles qui ont pour but de valider et d’accepter le produit réalisé. On parle d’ailleurs de recette provisoire et de recette définitive. La recette provisoire a pour but de montrer le produit et de noter les éventuels problèmes restants qui devraient être corrigés pour que le logiciel soit accepté. La recette définitive doit alors démontrer que le logiciel ait été corrigé. Le client et l’équipe projet signent alors un Procès-Verbal de recette.
* Le cycle de vie en v
* **Le Cycle de vie en cascade**

la définition d’un logiciel peut être grossièrement divisée en deux :

* L’étape de l’analyse (analyse des besoins, du domaine, applicative) ;
* La conception de la solution.
* L’itération signifie que l’on fera plusieurs allers-retours sur les diagrammes avant d’avoir un dossier d’analyse entièrement satisfaisant.
* Comprendre ce que veut un client ou un utilisateur n’est pas toujours évident. Les termes utilisés peuvent avoir des sens cachés que nous ne saisissons pas tout de suite. Un besoin peut évoluer en fonction de l’analyse que nous présentons. Notre vision du système peut évoluer en fonction des détails mis en évidence par certains diagrammes.
* Un diagramme peut être couplé avec un glossaire pour définir les termes plus facilement
* Le **système :** Dans UML, on appelle ce qu’on doit analyser, concevoir et réaliser : le système
* Un **acteur ou stick man :** correspond à une entité (humain ou non) qui aura une interaction avec le système
  + **Acteurs principaux** agissent directement sur le système. Il s’agit d’entités qui ont des besoins d’utilisation du système. On peut donc considérer que les futurs utilisateurs du logiciel sont les acteurs principaux.
* ****
* L’environnement du système est composé d’acteurs qui agissent sur le système ou avec lesquels le système aura une interaction
* Représente un acteur secondaire non humain grâce à l'information "système"
* Décomposer le système en parties, dit « package » : on peut découper le futur logiciel en parties distinctes, en fonction des « familles » de fonctionnalités et de façon à pouvoir les analyser séparément. Chacune de ces parties correspond à un domaine fonctionnel ou **package**.
* 
* Les packages sont représentés sous forme de dossier

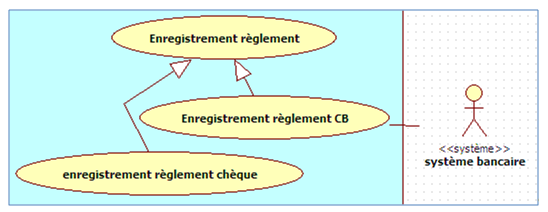
## Les cas d’utilisation

* Chaque package est donc une boîte qu’il faudra ouvrir pour en découvrir le contenu. Le contenu d’un package est illustré par différents diagrammes
* 
* Illustration d'un cas d'utilisation relié à un acteur, on représente un cas d'utilisation sous forme d'élipse
* Ce qu'on doit faire lors **de l’analyse de besoins principaux d’un projet logiciel :** Découvrir les acteurs et les fonctionnalités du futur logiciel.
* Les **étapes de l’analyse des besoins principaux sont :** On définit le contexte, on le décompose éventuellement en packages, puis on définit qui devra pouvoir faire quoi dans pour chaque package.
* cas d’utilisation interne
* La spécialisation des acteurs : Cela se justifie si plusieurs acteurs ont besoin d’un ou de plusieurs cas d’utilisation communs et qu’ils ont également besoin de cas d’utilisation spécifiques.
  + un acteur générique qui est lié aux cas d’utilisations communs ;
  + des acteurs spécialisés qui sont liés à des cas d’utilisation spécifiques.
* 

### La relation de type « include »

* Une relation « include » est utilisée pour indiquer que le cas d’utilisation source (départ de la flèche) contient TOUJOURS le cas d’utilisation inclus

### La relation « extend »

* L’ajout de cette relation se fait en dessinant une flèche en pointillé partant du cas d’utilisation interne vers le cas d’utilisation principal. Puis, on indique le stéréotype « extend » sur la flèche. Dès lors qu’il y a une relation « extend », il faudra toujours définir la condition, c’est-à-dire : à quelle condition cette relation peut avoir lieu ?
* Dès lors qu’il y a une relation « extend », il faudra toujours définir la condition, c’est-à-dire : à quelle condition cette relation peut avoir lieu ? Pour indiquer cela, il faut ajouter une ligne « extension points » et définir la condition « EXT ».
* 

Comme il s’agit ici de **deux variantes** du cas d’utilisation « Enregistrement règlement », je crée donc deux nouveaux cas d’utilisation, qui sont des **spécialisations** de « Enregistrement règlement ».

Dans une relation stéréotypée, un cas d’utilisation a besoin d’un autre cas d’utilisation, soit toujours (« include ») ou sous certaines conditions (« extend »).

Dans une spécialisation, on indique que les cas d’utilisation spécialisés sont des versions différentes du cas d’utilisation générique. Dans notre exemple, les deux nouveaux cas d’utilisation « Enregistrement règlement CB » et « Enregistrement règlement chèque » sont bien 2 versions du cas d’utilisation « Enregistrement règlement ». Ils ont chacun un déroulement spécifique.

Petit rappel des étapes vues :

1. Nous avons commencé par identifier et illustrer notre projet logiciel de façon globale. Nous avons identifié le contexte (c'est-à-dire le système) et ses utilisateurs. Pour rappel, notre projet de site de boutique en ligne a pour objectif de vendre des produits en ligne. Il s’agit donc ici du système. Ses acteurs ou ses utilisateurs sont :

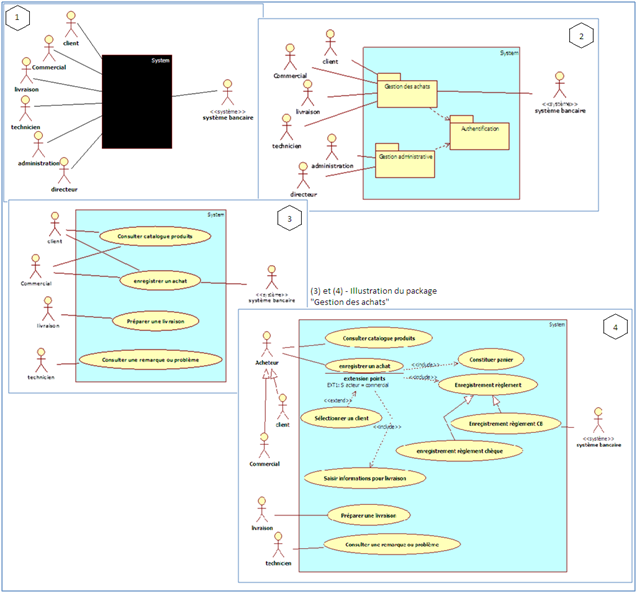
* le client, qui consultera le catalogue et achètera des produits en ligne ;
* le commercial, qui pourra également consulter le catalogue et qui devra récupérer les informations client pour enregistre une commande pour celui-ci ;
* la livraison, qui se chargera de livrer la commande au client ;
* le technicien, qui devra consulter les remarques et problèmes signalés ;
* le service administratif, qui devra gérer le catalogue ;
* le directeur, qui pourra consulter des états concernant les ventes ;
* le système bancaire, qui récupère et valide les informations bancaires afin de confirmer la commande.

Cela nous a permis de réaliser notre premier **diagramme de contexte**.

2. En étudiant les différents besoins de nos acteurs sur le logiciel, nous avons pu repérer 3 grandes familles de fonction : Gestion des achats, Gestion administrative et Authentification. Elles sont illustrées à l’aide d’un **diagramme de package**.

3. Puis, en étudiant dans le détail l’un des packages « Gestion des achats », nous avons défini les grandes fonctionnalités et les acteurs principaux. Nous avons ainsi réalisé le **diagramme de cas d’utilisation** (contenant uniquement des cas d’utilisation principaux).

4. Enfin, à travers l’un des cas d’utilisation principaux, nous avons tenté de définir les différents lots d’actions nécessaires au déroulement de ce cas. Il s’agit des cas d’utilisation internes. Nous avons d’ailleurs vu les différentes formes de relation qui existent entre les cas d’utilisation principales et les cas d’utilisation internes, comme les relations « include », « extend » ou de spécialisation. Nous avons donc réalisé un **diagramme de cas d’utilisation détaillé**(avec include/ extend/ spécialisation)

* Synthèse des diagrammes
* détaillé (avec include/ extend/ spécialisation)
*  Légende :
* (1) Le diagramme de contexte
* (2) Diagramme de package
* (3) Diagramme de cas d’utilisation initial
* (4) Diagramme de cas d’utilisation

## La description textuelle d’un cas d’utilisation

Nous allons désormais parler de l’interaction entre les acteurs et le système : il s’agit de décrire la chronologie des actions qui devront être réalisées par les acteurs et par le système lui-même. On parle d’ailleurs de scénarios.

La description d’un cas d’utilisation permet de :

* clarifier le déroulement de la fonctionnalité ;
* décrire la chronologie des actions qui devront être réalisées ;
* d’identifier les parties redondantes pour en déduire des cas d’utilisation plus précises qui seront utilisées par inclusion, extension ou généralisation/spécialisation. Et oui, dans ce cas nous réaliserons des itérations sur les diagrammes de cas d’utilisation ;
* D’indiquer d’éventuelles contraintes déjà connues et dont les développeurs vont devoir tenir compte lors de la réalisation du logiciel. Ces contraintes peuvent être de nature diverse

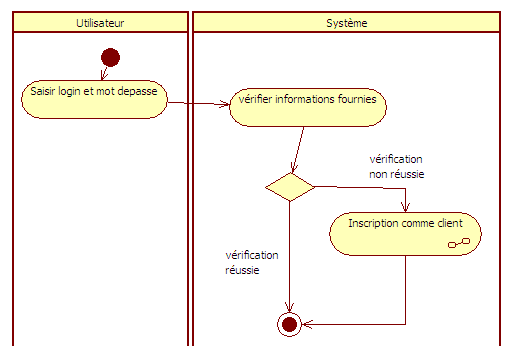
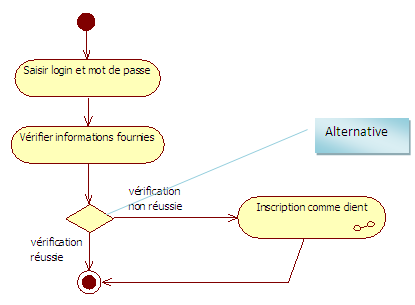
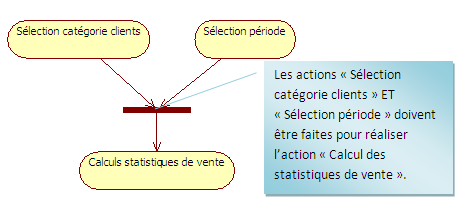
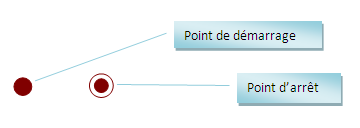
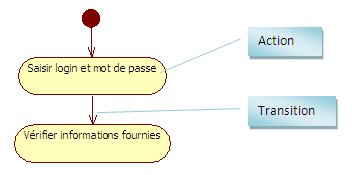
fiche descriptive doit comporter 4 volets :

1. L’identification
2. La description des scénarios
3. La fin et les post-conditions
4. Les compléments

Il existe 3 parties :

* **Le scénario nominal**  
  Il s’agit ici de décrire le déroulement idéal des actions, où tout va pour le mieux.
* **Les scénarios alternatifs**  
  Ici, il s’agit de décrire les éventuelles étapes différentes liées aux choix de l’utilisateur, par exemple. C’est le cas des étapes liées à des conditions.
* **Les scénarios d’exception**  
  On parlera de scénario d’exception lorsque une étape du déroulement pourrait être perturbée à cause d’un événement anormal. Par exemple, lorsqu’une recherche de client ne trouve aucun client correspondant aux critères fournis.
* **La fin** permet de récapituler toutes les situations d’arrêt du cas d’utilisation. Cela permet parfois de s’apercevoir que l’on n’a pas vu tous les cas de figure qui peuvent se présenter.
* Par exemple, notre cas d’utilisation peut s’arrêter aux étapes 2, 5 et 7 car l’utilisateur peut décider de quitter la consultation du catalogue pour revenir à l’accueil par exemple.
* **Les post-conditions** nous indiquent un résultat tangible qui est vérifiable après l’arrêt du cas d’utilisation et qui pourra témoigner du bon fonctionnement. Cela pourrait être une information qui a été enregistrée dans une base de données ou dans un fichier, ou encore un message envoyé par mail, etc.

|  |
| --- |
| **Cas n°1**  **Nom :** Consulter catalogue produit (package « Gestion des achats ») **Acteur(s) :** Acheteur (client ou commercial) **Description :** La consultation du catalogue doit être possible pour un client ainsi que pour les commerciaux de l’entreprise. **Auteur :** Carina Roels **Date(s) :** 10/11/2013 (première rédaction)  **Pré-conditions :** L’utilisateur doit être authentifié en tant que client ou commercial (Cas d’utilisation « S’authentifier » – package « Authentification ») **Démarrage :** L’utilisateur a demandé la page « Consultation catalogue » |
| **DESCRIPTION**  **Le scénario nominal :**  1. Le système affiche une page contenant la liste les catégories de produits.  2. *L’utilisateur* sélectionne une des catégories. 3. **Le système** recherche les produits qui appartiennent à cette catégorie. 4. **Le système** affiche une description et une photo pour chaque produit trouvé. 5. *L’utilisateur* peut sélectionner un produit parmi ceux affichés. 6. **Le système** affiche les informations détaillées du produit choisi.  7. *L’utilisateur* peut ensuite quitter cette description détaillée. 8. **Le système** retourne à l’affichage des produits de la catégorie (retour à l’étape 4) |
| **Les scénarios alternatifs**  2.a *L’utilisateur* décide de quitter la consultation de la catégorie de produits choisie.  2.b *L’utilisateur* décide de quitter la consultation du catalogue.  5.a L’utilisateur décider de quitter la consultation de la catégorie de produits choisie. 5.b *L’utilisateur* décide de quitter la consultation du catalogue. 7.a *L’utilisateur* décide de quitter la consultation de la catégorie de produits choisie. 7.b L’utilisateur décide de quitter la consultation du catalogue. |
| **Fin :**Scénario nominal : aux étapes 2, 5 ou 7, sur décision de l’utilisateur |
| **Post-conditions :**Aucun |
| **COMPLEMENTS**  **Ergonomie**  L’affichage des produits d’une catégorie devra se faire par groupe de 15 produits. Toutefois, afin d’éviter à l’utilisateur d’avoir à demander trop de pages, il devra être possible de choisir des pages avec 30, 45 ou 60 produits. |
| **Performance attendue**  La recherche des produits, après sélection de la catégorie, doit se faire de façon à afficher la page des produits en moins de 10 secondes. |
| **Problèmes non résolus**  Nous avons fait la description basée sur l’information que les produits appartiennent à une catégorie. Est-ce qu’il existe des sous-catégories ?  Si tel est le cas, la description devra être revue.  Est-ce que la consultation du catalogue doit être possible uniquement par catégorie ou est-ce qu’on doit prévoir d’autres critères de recherche de produits ?  Doit-on prévoir un affichage trié sur des critères choisis par l’utilisateur (par exemple : par tranche de prix, par disponibilité, etc) ? |

* Diagramme d'acvité ou textuelle : Certaines personnes préfèrent le diagramme d’activités à la description textuelle. Pour ma part, je préfère commencer par une description textuelle qui donne des précisions que nous n’aurons pas dans le diagramme d’activité (des informations telles que les pré-conditions, le démarrage, les post-conditions, etc.).
* 

#### Les couloirs (dit « swimlanes » en anglais)

Ils permettent d’indiquer qui (de l’utilisateur ou du système) réalise les actions.

## UML, un outil de communication avec le client

* nous pouvons proposer des solutions, mais le choix final appartient au client et aux utilisateurs.
* Il s’agit avant tout de **modéliser les besoins du client et des futurs utilisateurs**