MATTHIEU REQUENNA - INTRODUCTION UML

Débutez l'analyse logicielle avec UML

Vous êtes développeur ou novice en informatique, et vous souhaitez préparer votre projet logiciel avec UML ? Vous souhaitez proposer une version visuelle de votre projet et compréhensible de tous ?

Je vous présenterai le langage UML et réaliserai, avec vous, les premières analyses de besoin d'un projet logiciel à partir d'un exemple concret. À la fin de ce cours, vous serez capable de réaliser vos premiers diagrammes définissant les éléments de base de votre projet : le contexte, les utilisateurs, les actions et leur déroulement.

Pré-requis :

Ce cours s'adresse aux débutants, qu'ils soient du secteur informatique ou non. En effet, le langage UML est un langage visuel qui permet d'illustrer un projet logiciel et d'échanger / communiquer sur le projet entre le développeur et le client par exemple.

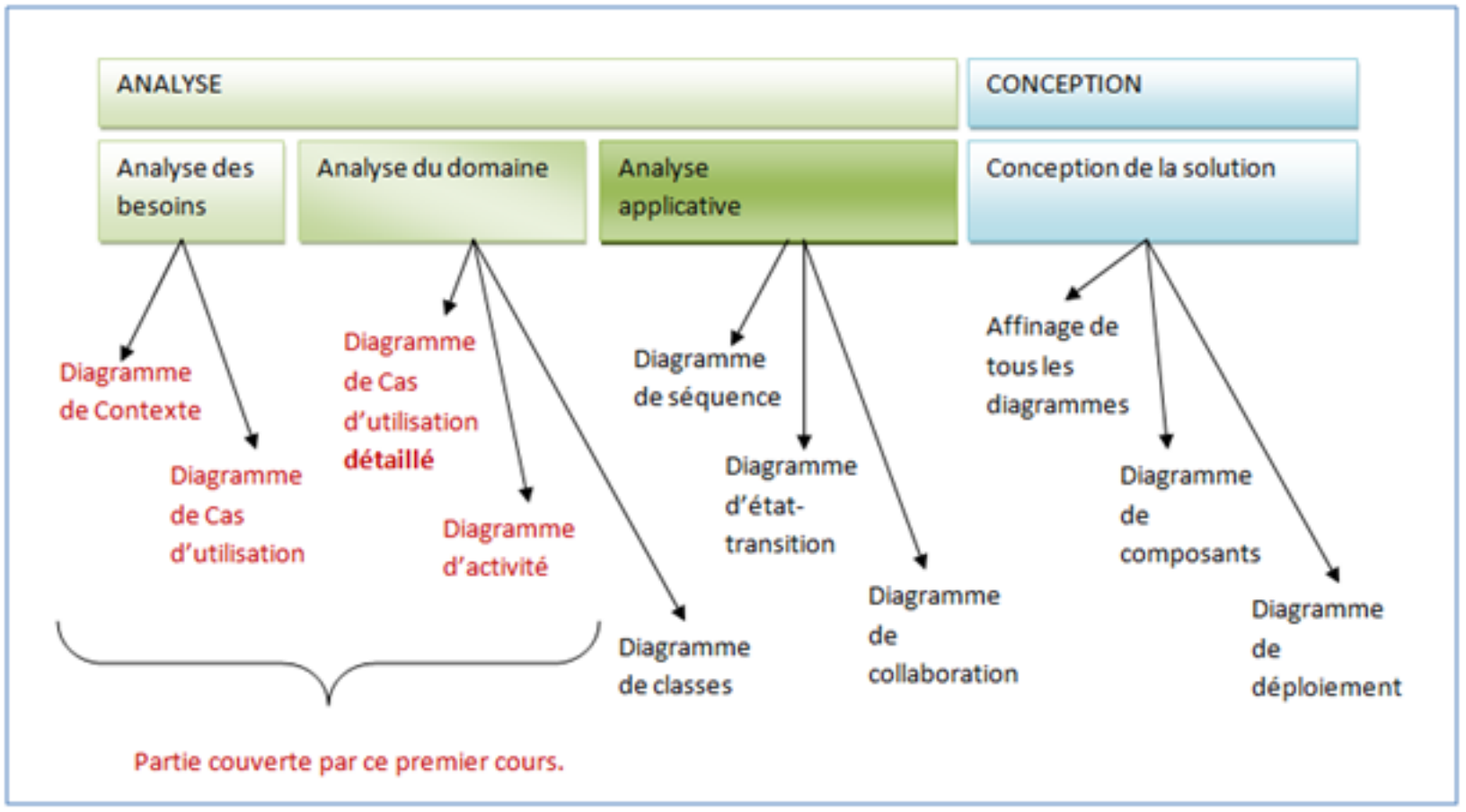
Ce cours sur UML est principalement sous la forme textuelle. Seules les introductions et les conclusions des parties sont en vidéo.

UML, c’est quoi ?

Comme n’importe quel type de projet, un projet informatique nécessite une phase d’analyse, suivi d’une étape de conception.

Dans **la phase d’analyse**, on cherche d’abord à bien comprendre et à décrire de façon précise les besoins des utilisateurs ou des clients. Que souhaitent-ils faire avec le logiciel ? Quelles fonctionnalités veulent-ils ? Pour quel usage ? Comment l’action devrait-elle fonctionner ? C’est ce qu’on appelle « **l’analyse des besoins** ». Après validation de notre compréhension du besoin, nous imaginons la solution. C’est la partie **analyse de la solution**.

Dans **la phase de conception**, on apporte plus de détails à la solution et on cherche à clarifier des aspects techniques, tels que l’installation des différentes parties logicielles à installer sur du matériel.

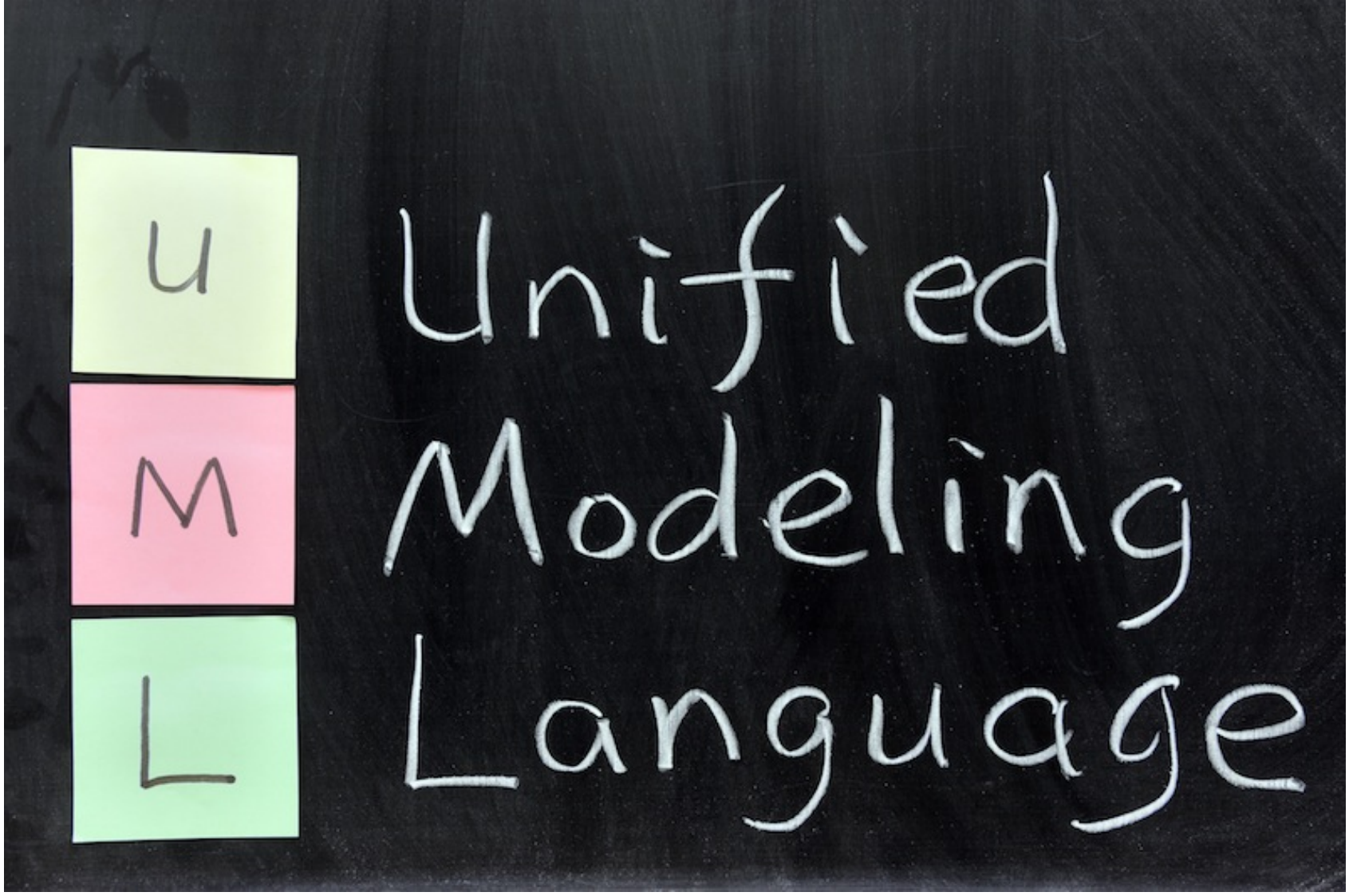


Positionnement du cours dans la démarche de projet

Pour réaliser ces deux phases dans un projet informatique, nous utilisons des méthodes, des conventions et des notations. UML fait partie des notations les plus utilisées aujourd'hui.

Nous allons dans ce chapitre définir le langage UML et ses outils : les diagrammes. Nous verrons comment ce langage peut contribuer à la phase d'analyse des besoins et du domaine d'un projet informatique.

Définition



Fotolia_Crédit Raywoo

UML, c'est l'acronyme anglais pour « Unified Modeling Language ». On le traduit par « Langage de modélisation unifié ». La notation UML est un **langage visuel** constitué d'un ensemble de schémas, appelés des **diagrammes**, qui donnent chacun une vision différente du projet à traiter. UML nous fournit donc des diagrammes pour **représenter** le logiciel à développer : son fonctionnement, sa mise en route, les actions susceptibles d'être effectuées par le logiciel, etc.

Réaliser ces diagrammes revient donc à **modéliser les besoins** du logiciel à développer.

Et ces diagrammes, on les réalise comment ?

Vous avez le choix ! Vous pouvez soit reprendre les normes de ces diagrammes que nous verrons au fur et à mesure de ce cours et les dessiner à la main, soit utiliser des logiciels gratuits ou payants pour les réaliser. Dans ce cours, j'utiliserai par exemple StarUml, mais vous pouvez aussi utiliser ArgoUml, BoUml, PowerDesigner, etc.

UML est né de la fusion des trois méthodes qui ont influencé la modélisation objet au milieu des années 90 : OMT, Booch et OOSE. Il s'agit d'un compromis qui a été trouvé par une équipe d'experts : Grady Booch, James Rumbaugh et Ivar Jacobson. UML est à présent un standard défini par l'Object Management Group (OMG). De très nombreuses entreprises de renom ont adopté UML et participent encore aujourd'hui à son développement.

UML est surtout utilisé lorsqu'on prévoit de développer des applications avec une démarche objet (développement en Java, en C++, etc.).

Cela dit, je suis d'avis que l'on peut tout à fait s'en servir pour décrire de futures applications, sans pour autant déjà être fixé sur le type de développement.

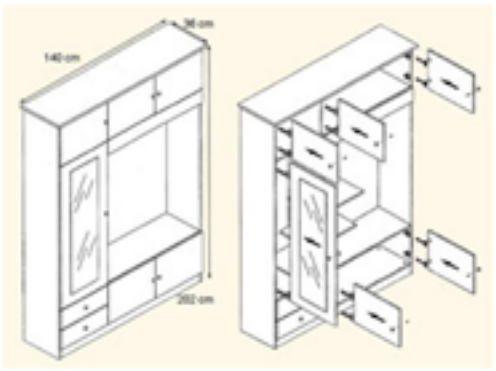
Le langage UML ne préconise aucune démarche, ce n'est donc pas une méthode. Chacun est libre d'utiliser les types de diagramme qu'il souhaite, dans l'ordre qu'il veut. Il suffit que les diagrammes réalisés soient cohérents entre eux, avant de passer à la réalisation du logiciel.

Oui, je vous l'accorde, n'ayant aucune directive quant à la démarche à adopter, on se sent plutôt perdu. C'est pourquoi je vais vous donner une démarche possible (parmi tant d'autres) pour que vous puissiez commencer à utiliser la notation UML dans vos projets de réalisation de logiciels.

Pourquoi modéliser ?

Modéliser, c'est décrire de manière visuelle et graphique les besoins et, les solutions fonctionnelles et techniques de votre projet logiciel. Mais modéliser pour quoi faire ?

Avez-vous déjà eu à constituer un meuble en kit ou à brancher un nouvel équipement électronique? Vous serez d'accord que c'est plus facile à partir de schéma plutôt qu'une page de texte, non ?



Cet exemple nous démontre que l'utilisation de schémas et d'illustrations rend quelque chose de complexe plus compréhensible.

Intéressant. Mais quel est le lien avec UML alors ?

C'est exactement ce à quoi UML sert dans des projets de réalisation de logiciels !

Un document de texte décrivant de façon précise ce qui doit être réalisé contiendrait plusieurs dizaines de pages. En général, peu de personnes ont envie de lire ce genre de document. De plus, un long texte de plusieurs pages est source d'interprétations et d'incompréhension. UML nous aide à faire cette **description de façon graphique** et devient alors un excellent moyen pour « **visualiser** » le(s) futur(s) logiciel(s).

Je suis toujours étonnée de voir le nombre d'informaticiens qui commencent à programmer sans avoir étudié le besoin des utilisateurs, du client, etc. et sans avoir fait une conception du logiciel en bonne et due forme.

Quand j'en fais la remarque, je reçois généralement des réponses du type :

- On n'est pas vraiment obligé de modéliser un logiciel que l'on doit réaliser, non ?
- À quoi bon modéliser le logiciel, puisque je sais exactement ce qu'il faut ?
- C'est une perte de temps. La réalisation de ce logiciel est urgente.
- Etc.

Laissez-moi vous répondre par une analogie.

Est-ce que ça vous viendrait à l'idée de laisser un maître d'œuvre commencer la construction d'une maison sans avoir préalablement fait réaliser des plans par un architecte ? La réponse est bien sûr que non, à moins que l'on veuille bien accepter une maison qui ne réponde pas à nos besoins, qui soit bancale ou qui le deviendrait dans peu de temps, et dans laquelle il serait difficile de faire des travaux faute de plans.

Un logiciel qui a été réalisé sans analyse et sans conception (étapes où l'on modélise le futur logiciel) risque lui aussi de ne pas répondre aux besoins, de comporter des anomalies et d'être très difficile à maintenir.

Tout comme la construction d'une maison nécessite des plans à différents niveaux (vision extérieure, plan des différents étages, plans techniques…), la réalisation d'un logiciel ou d'un ensemble de logiciels a besoin d'un certain nombre de diagrammes.

Le temps que prend la modélisation ne devrait pas être un frein. Cette étape nous permet de ne pas perdre davantage de temps ultérieurement :

- pour « faire coller » un logiciel déjà développé aux besoins (qu'on n'avait pas étudié, ni compris) ;
- pour comprendre comment un logiciel a été créé avant d'y apporter des modifications.

On peut utiliser les termes « application » ou « logiciel ». On pourrait penser que le deuxième terme prête à confusion, puisque cela peut être un logiciel développé pour un besoin spécifique ou un logiciel qu'on achète. Pour être tout à fait exact, un logiciel acheté est appelé un « progiciel » (contraction des mots produit et logiciel), par exemple Word, ou encore Viso, StarUml, etc.

L'approche objet

Dans la gestion de projet, nous pouvons citer deux approches permettant

de définir les besoins :

- La décomposition fonctionnelle (ou l'approche procédurale)
- L'approche objet (sur laquelle est basée UML)

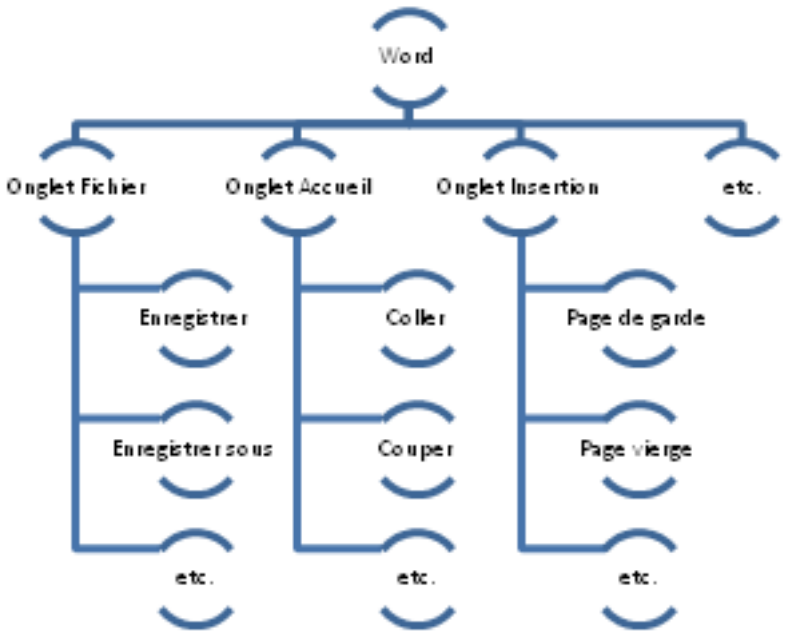
La décomposition fonctionnelle

Avant l'apparition de l'approche objet dans les années 80, une autre démarche était largement utilisée.

Pendant longtemps, de nombreux logiciels étaient conçus par l'approche fonctionnelle descendante, appelée également la décomposition fonctionnelle. Je sais, ce terme parait barbare, mais vous allez vite comprendre.

L'approche par décomposition fonctionnelle considère que le logiciel est composé d'une **hiérarchie de fonctions et de données.** Les fonctions fournissent les services désirés et les données représentent les informations manipulées. La démarche est logique, cohérente et intuitive (voir la figure suivante).

Une fonction ou fonctionnalité est une option mise à disposition des utilisateurs. Lorsque vous utilisez le progiciel Word, vous disposez de plusieurs fonctionnalités qui sont regroupées dans des menus et des onglets. Par exemple, dans l'onglet « Accueil », vous trouverez une fonctionnalité qui permet de modifier la couleur du texte, une autre pour rechercher un mot dans le texte, etc.



Exemple de décomposition fonctionnelle pour le logiciel Word

Pour réaliser une fonction du logiciel, on peut utiliser un ensemble d'autres fonctions, déjà disponibles, à condition qu'on rende ces dernières suffisamment génériques.

Comme vous pouvez le voir dans l'exemple précédent, le progiciel Word contient la fonction « Enregistrer sous », qui utilise très probablement la fonction « Enregistrer ». Cette fonction se sert de l'emplacement du nom du fichier connu à l'ouverture.

Pour la fonction « Enregistrer sous », l'utilisateur doit préciser un endroit où enregistrer tout en donnant éventuellement un autre nom au fichier. L'enregistrement se fait alors exactement de la même manière que dans la fonction « Enregistrer ».

Ce découpage n'a pas que des avantages. **Les fonctions sont alors interdépendantes** : une simple mise à jour du logiciel à un point donné, peut impacter **en cascade** d'autres fonctions. On peut éviter cela en faisant attention à créer des fonctions très génériques. Mais respecter cette contrainte rend l'écriture du logiciel et sa maintenance plus complexe.

Par exemple, si les développeurs de la société Microsoft décident de modifier le déroulement de la fonction « Enregistrer », cela impactera forcément la fonction « Enregistrer sous ». On peut aisément imaginer qu'une modification dans une fonction qui est utilisée par un grand nombre d'autres fonctions sème un peu la pagaille.

Ayant constaté les problèmes inhérents à la décomposition fonctionnelle, de nombreux experts se sont penchés sur une approche différente. De cette réflexion est née **l'approche objet**, dont UML s'inspire largement.

L'approche objet

L'approche objet est une démarche qui s'organise autour de 4 principes fondamentaux. C'est une démarche :

- itérative et incrémentale ;
- guidée par les besoins du client et des utilisateurs ;
- centrée sur l'architecture du logiciel ;
- qui décrit les actions et les informations dans une seule entité.

Allez, voyons cela dans le détail !

=> L'approche objet utilise une **démarche itérative et incrémentale.**

Aie ! Qu'est-ce que ça veut dire ?

Reprenons notre exemple de la construction d'une maison. L'architecte réalise toujours plusieurs versions des plans avant d'avoir la version définitive (c'est-à-dire celui accepté par le client). Après un premier entretien avec son client, il réalise les plans pour la vision extérieure de la maison et les plans de surface. Ces plans sont soumis au client et donnent généralement lieu à des discussions. Cela permet à l'architecte de mieux cerner le besoin de son client. Il apporte alors les modifications qui s'imposent aux plans qu'il avait réalisés. Il s'agit là d'une **première itération** sur les plans de la vision extérieure et les plans de surface.

Par la suite, l'architecte réalisera les plans techniques qui seront nécessaires pour les différents corps de métier (les maçons, les électriciens, les plombiers, etc.). Il arrive parfois que ces plans mettent en évidence certaines contraintes qui obligent l'architecte à revenir sur les plans de surface (par exemple pour ajouter l'emplacement d'une colonne d'aération

qui n'avait pas été prévue au départ). Il y a donc une **deuxième itération** sur les plans de surface.

Une démarche itérative c'est donc faire des allers-retours entre le plan initial et les modifications apportées par les acteurs du projet.

De la même façon, la modélisation d'un logiciel se fera en plusieurs fois. Les diagrammes ne seront pas réalisés de façon complètement satisfaisante dès la première fois. Il nous faudra plusieurs versions d'un même diagramme pour obtenir la version finale. Chaque version est le fruit d'une itération (un nouveau passage sur les diagrammes). Une itération permet donc d'affiner la compréhension du futur logiciel.

☐ La démarche est **guidée par les besoins du client et des utilisateurs**

Les besoins d'un client, dans un projet logiciel sont les exigences exprimées par le client au niveau des fonctionnalités, du rendu et des actions d'un logiciel. Par exemple, lors de la réalisation du progiciel Word, les clients ont probablement souhaité un logiciel qui leur permettrait d'écrire du texte, de l'effacer, d'intégrer des images, enregistrer le contenu, le supprimer, etc.

Les besoins des utilisateurs servent de fil rouge, tout au long du cycle de développement :

- lors de la phase **d'analyse** pour la clarification, l'affinage et la validation des besoins des utilisateurs ;
- lors de la phase **de conception** et **de réalisation** pour la vérification de la prise en compte des besoins des utilisateurs ;
- lors de la phase de **test** afin de garantir la satisfaction des besoins.

Je vous rappelle que dans ce cours, nous traiterons uniquement de la phase d'analyse.

=> La démarche est également **centrée sur l'architecture du logiciel**,

pierre angulaire d'un développement.

L'architecture logicielle décrit les choix stratégiques qui déterminent en grande partie les qualités du logiciel (adaptabilité, performances, fiabilité…). On peut citer l'architecture client serveur, en couche ou en niveaux. Cela n'étant pas l'objet de ce cours, je vous épargnerai les détails. Concentrons-nous sur la modélisation des besoins !

=> L'approche objet **décrit aussi bien les actions que les informations dans une même entité.**

Les différents diagrammes utilisés en UML donnent tous une vision particulière du logiciel à développer. Parmi ces diagrammes, il en est un qui représente particulièrement bien ce qui est à développer si on opte pour un développement objet. Il s'agit du **diagramme de classes**.

Je sais, je n'ai pas encore défini les diagrammes que l'on peut utiliser. Pour l'instant, nous pouvons retenir que le diagramme de classes donnera une vision assez claire des informations qui seront utilisées par le logiciel, mais également des fonctions (ou opérations) qui devront s'appuyer sur ces informations. L'approche objet mélange donc habillement l'analyse des informations et des actions au lieu de les analyser séparément.

Le diagramme de classes ne sera pas étudié dans ce cours, mais je souhaitais simplement l'évoquer dans le cas où vous auriez envie d'aller plus loin !

En résumé

- UML signifie « Unified Modeling Language » ou Langage de modélisation unifié en français. C'est un langage de modélisation qui permet de représenter graphiquement les besoins des utilisateurs. On utilise pour cela des diagrammes.
- UML est une démarche qui se base sur une approche objet. Cette approche s'appuie sur 4 principes fondamentaux.

- L’approche objet nécessite une démarche itérative et incrémentale, c’est-à-dire que le concepteur doit faire des allers-retours entre les diagrammes initiaux et, les besoins du client et des utilisateurs perçus au fur et à mesure de la conception du logiciel afin de le modifier si nécessaire.

- L’approche objet est guidée par les besoins du client.

- L’approche objet est centrée sur le diagramme de classes qui décrit aussi bien des actions que des informations dans une même entité. Les autres diagrammes nous aident à voir clair dans les besoins et dans la solution qui est à développer. Ils permettent de compléter le diagramme de classes.

Les différents types de diagrammes

Tout comme la construction d'une maison nécessite des plans à différents niveaux (vision extérieure, plan des différents étages, plans techniques…), la réalisation d'une application informatique ou d'un ensemble d'applications est basée sur plusieurs diagrammes. Comme je vous le disais dans le premier chapitre, le langage UML est constitué de diagrammes. À ce jour, il existe **13 diagrammes** « officiels ». Ces diagrammes sont tous réalisés **à partir du besoin des utilisateurs** et peuvent être regroupés selon les deux aspects suivants :

- **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 ?

UML modélise donc le système logiciel suivant ces deux modes de représentation.

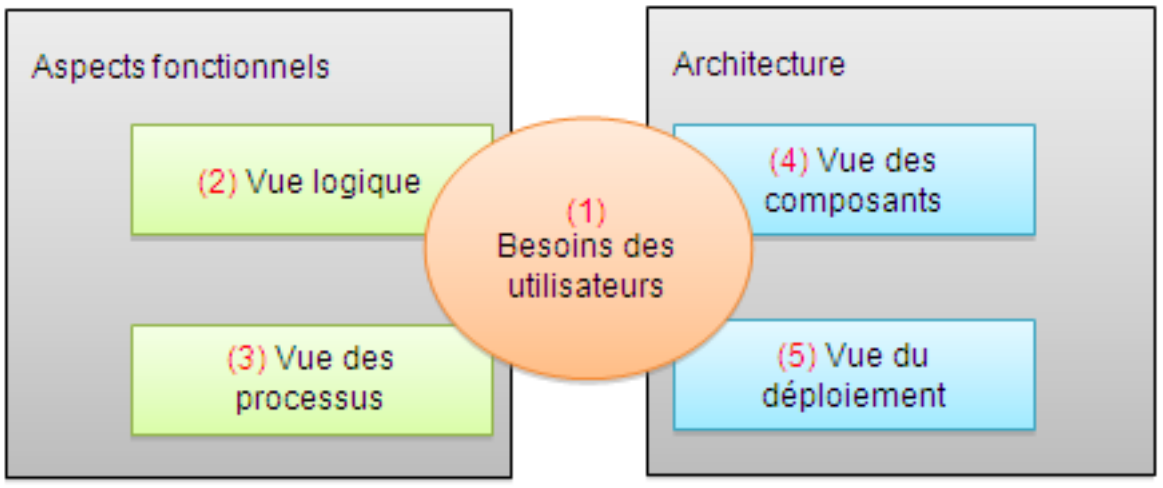
Nous allons donc dans un premier temps décrire ces différents aspects d'un logiciel grâce au schéma 4+1 vues et parcourir brièvement les différents diagrammes qui les composent.

Les 4+1 vues d'un système

Comme je vous le disais en introduction, la conception d'un logiciel est organisée autour des aspects fonctionnels et d'architecture. Ces deux aspects sont représentés par le schéma de 4 vues, axées sur les besoins des utilisateurs (parfois intitulé des cas d'utilisation), appelé **4+1 vues.**

Une première décomposition d'une problématique ou système peut donc être faite à l'aide de 4+1 vues. Le schéma ci-dessous montre les différentes vues permettant de répondre au mieux aux besoins des utilisateurs, organisées selon les deux aspects (fonctionnels et architecture). Chacune des vues est constituée de diagrammes.

Pour rappel, dans ce cours, nous nous concentrerons uniquement sur les besoins des utilisateurs, au centre de la figure.



Les 4+1 vues d'un système

Voici quelques exemples de diagramme qui peuvent être utilisés dans une démarche d'analyse et de conception d'un logiciel. Ne vous inquiétiez pas du nombre de diagrammes ci-dessous. Il s'agit ici simplement de vous familiariser à ces diagrammes.

Je vous rappelle que dans ce cours, mon objectif est d'avancer pas à pas dans l'analyse des besoins initiaux des utilisateurs, afin de travailler plus particulièrement les diagrammes suivants :

- le diagramme de contexte ;

Ce diagramme n'est pas officiellement désigné comme diagramme UML. Il ne fait donc pas partie des 13 diagrammes « officiels », mais il est utile pour la définition des acteurs, avant de commencer à s'intéresser à d'autres aspects, tels que les packages et les cas d'utilisation.

- le diagramme de package ;
- le diagramme de cas d'utilisation ;

- le diagramme d'activité.

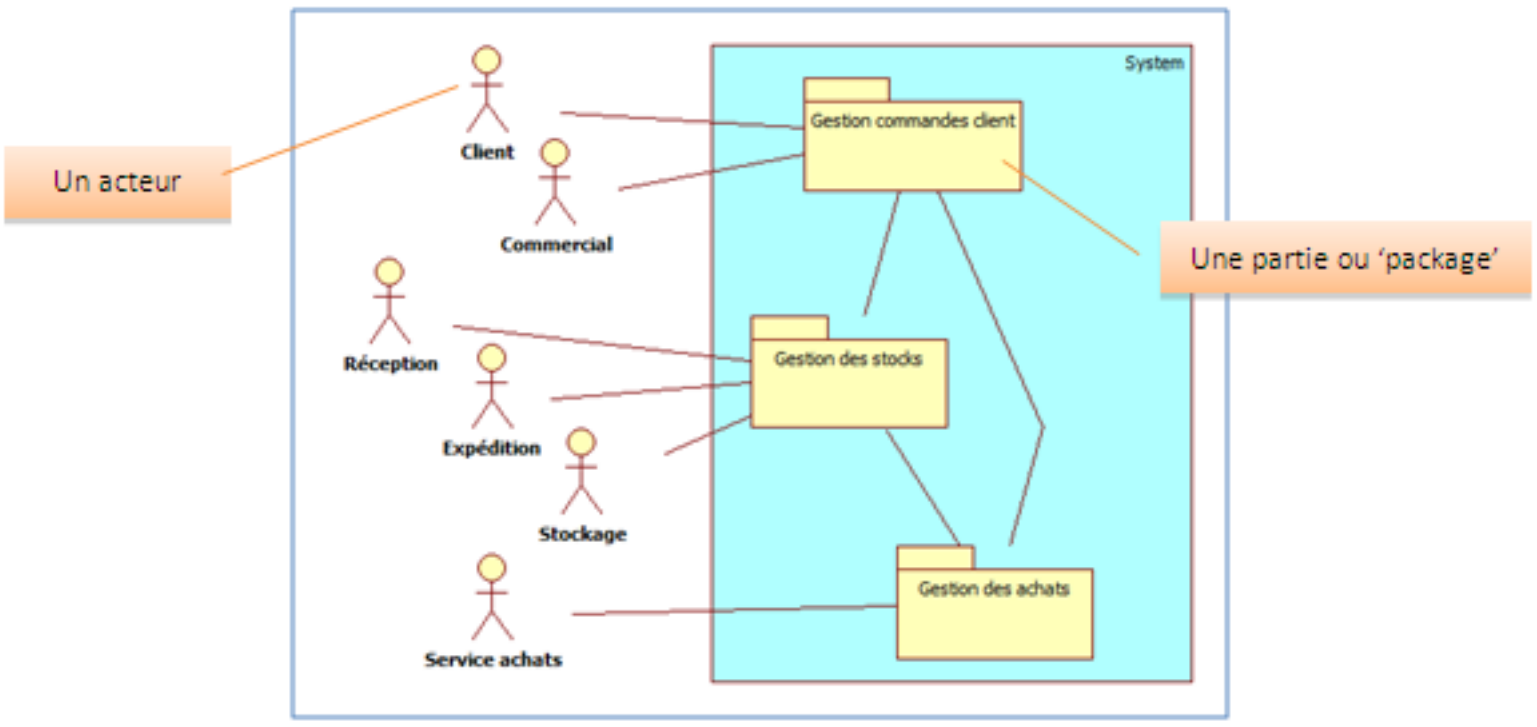
Les autres diagrammes pourront être vus lors d'un autre cours.

Les besoins des utilisateurs (1)

Cette partie représente le cœur de l'analyse. Il est composé de cas d'utilisation (que nous verrons plus tard). On y décrit le contexte, les acteurs ou utilisateurs du projet logiciel, les fonctionnalités du logiciel mais aussi les interactions entre ces acteurs et ces fonctionnalités. C'est d'ailleurs aussi le cœur de notre cours.

Le besoin des utilisateurs peut être décrit à l'aide de deux diagrammes.

- **Le diagramme de packages** permet de décomposer le système en catégories ou parties plus facilement observables, appelés « packages ». Cela permet également d'indiquer les acteurs qui interviennent dans chacun des packages.



Un exemple de diagramme de package

Dans l'exemple précédent, nous voyons que le logiciel que nous concevons peut être divisé en trois parties (ou packages) observables séparément :

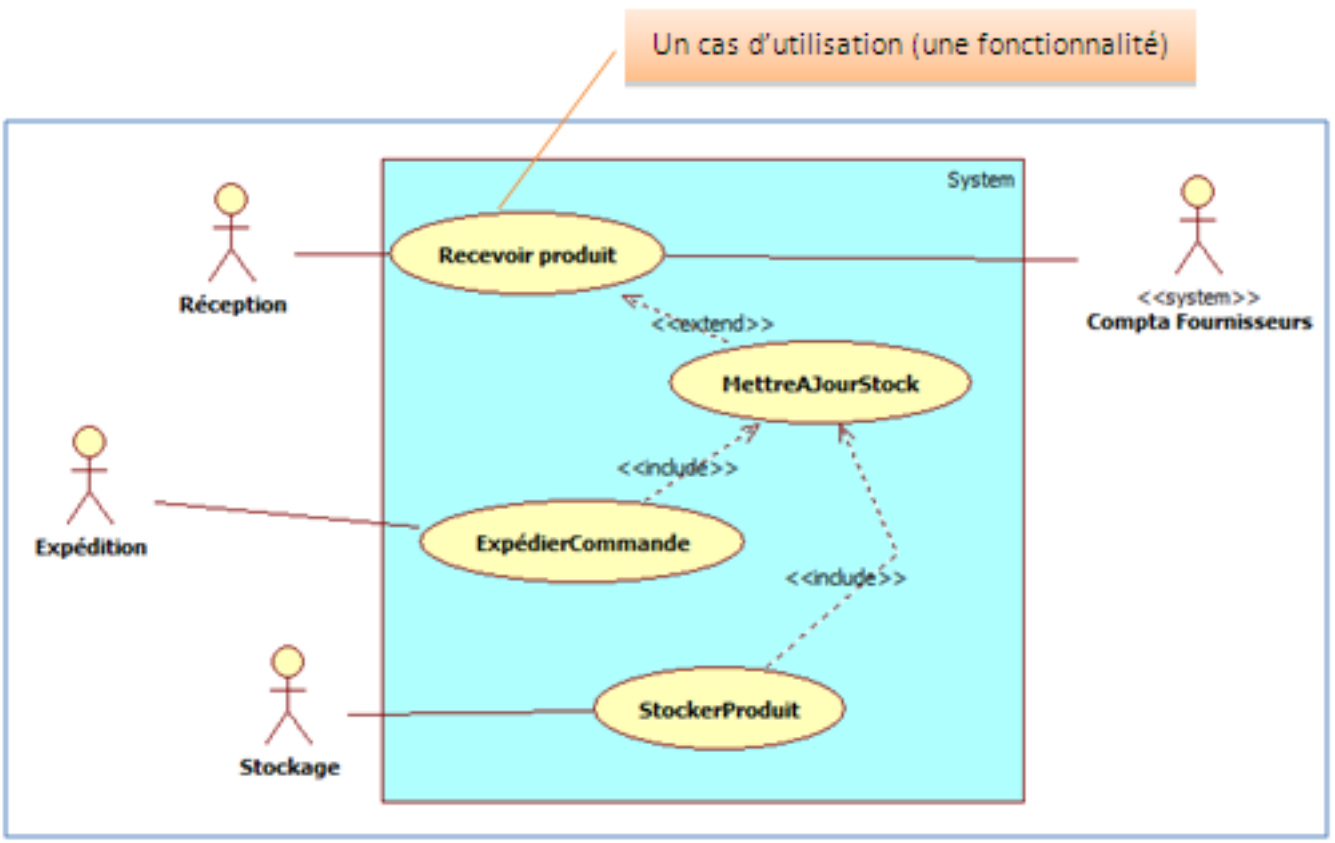
1. La gestion des commandes client

2. La gestion des stocks

3. La gestion des achats

La boîte qui entoure les packages (la boîte bleue) correspond au système (c'est-à-dire le logiciel) qui est analysé.

- **Le diagramme de cas d'utilisation** représente les fonctionnalités (ou dit cas d'utilisation) nécessaires aux utilisateurs. On peut faire un diagramme de cas d'utilisation pour le logiciel entier ou pour chaque package.



Un exemple de diagramme de cas d'utilisation pour un package (Gestion des stocks)

Étant donné que le diagramme de cas d'utilisation détaille le contenu d'un package, ici la boîte bleue correspond au package qui est détaillé.

L'aspect fonctionnel du logiciel

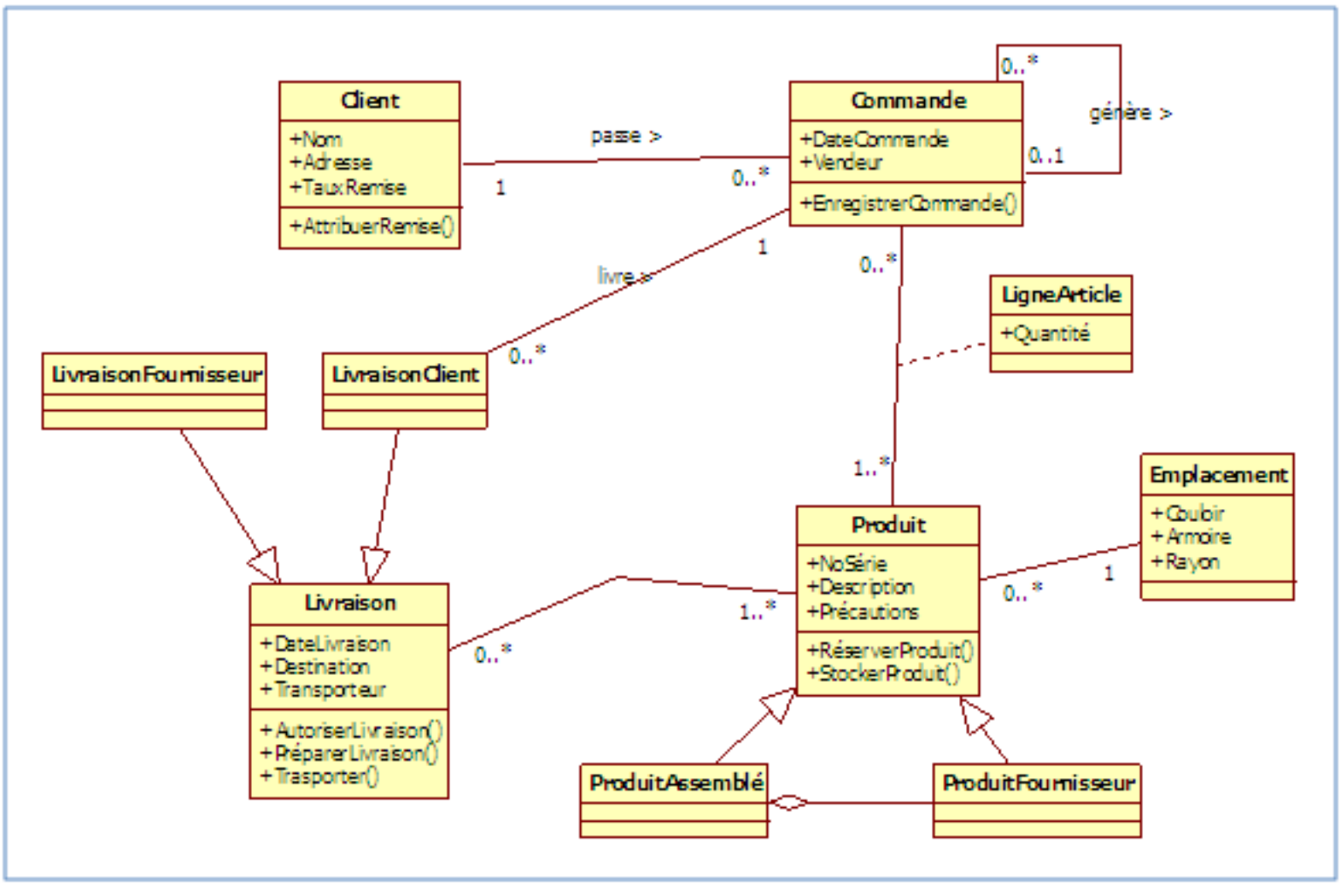
Pour rappel, cette partie du schéma 4+1 vues permet de définir qui utilisera le logiciel et pour quoi faire, comment les fonctionnalités vont se dérouler, etc.

Vue logique (2)

La vue logique a pour but d'identifier les éléments du domaine, les relations et interactions entre ces éléments. Cette vue organise les éléments du domaine en « catégories ». Deux diagrammes peuvent être utilisés pour cette vue.

- **Le diagramme de classes**

Dans la phase d'analyse, ce diagramme représente les entités (des informations) manipulées par les utilisateurs.
Dans la phase de conception, il représente la structure objet d'un développement orienté objet.
Ce diagramme ne sera pas étudié dans ce cours.

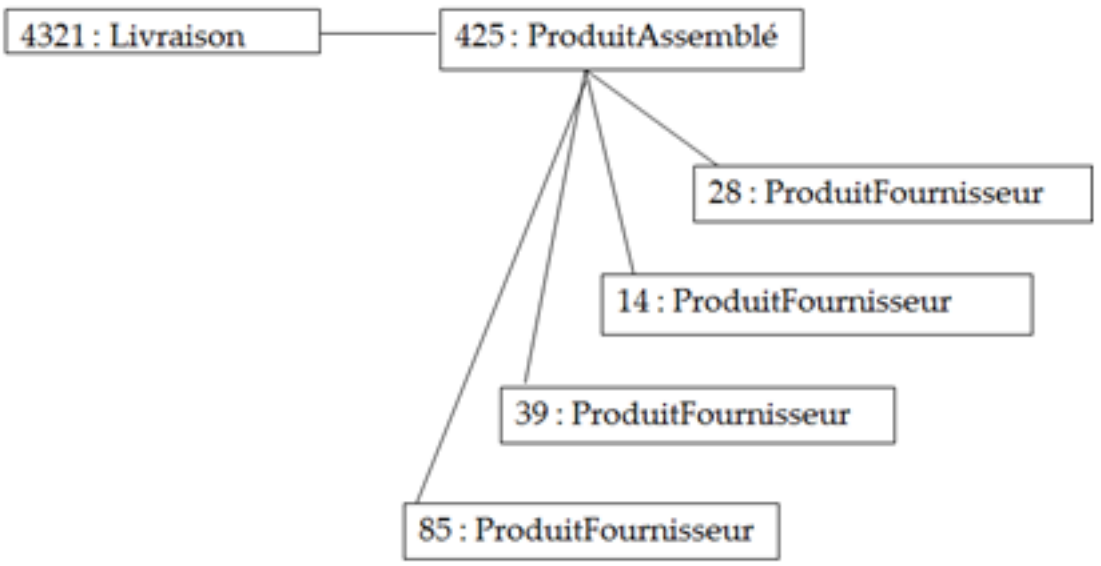


Un exemple de diagramme de classes (utilisé en phase d'analyse)

- **Le diagramme d'objets** sert à illustrer les classes complexes en utilisant des exemples d'instances.

Une instance est un exemple concret de contenu d'une classe. En illustrant une partie des classes avec des exemples (grâce à un diagramme d'objets), on arrive à voir un peu plus clairement les liens nécessaires. Ce diagramme

ne sera pas étudié dans ce cours.



Un exemple de diagramme d'objet

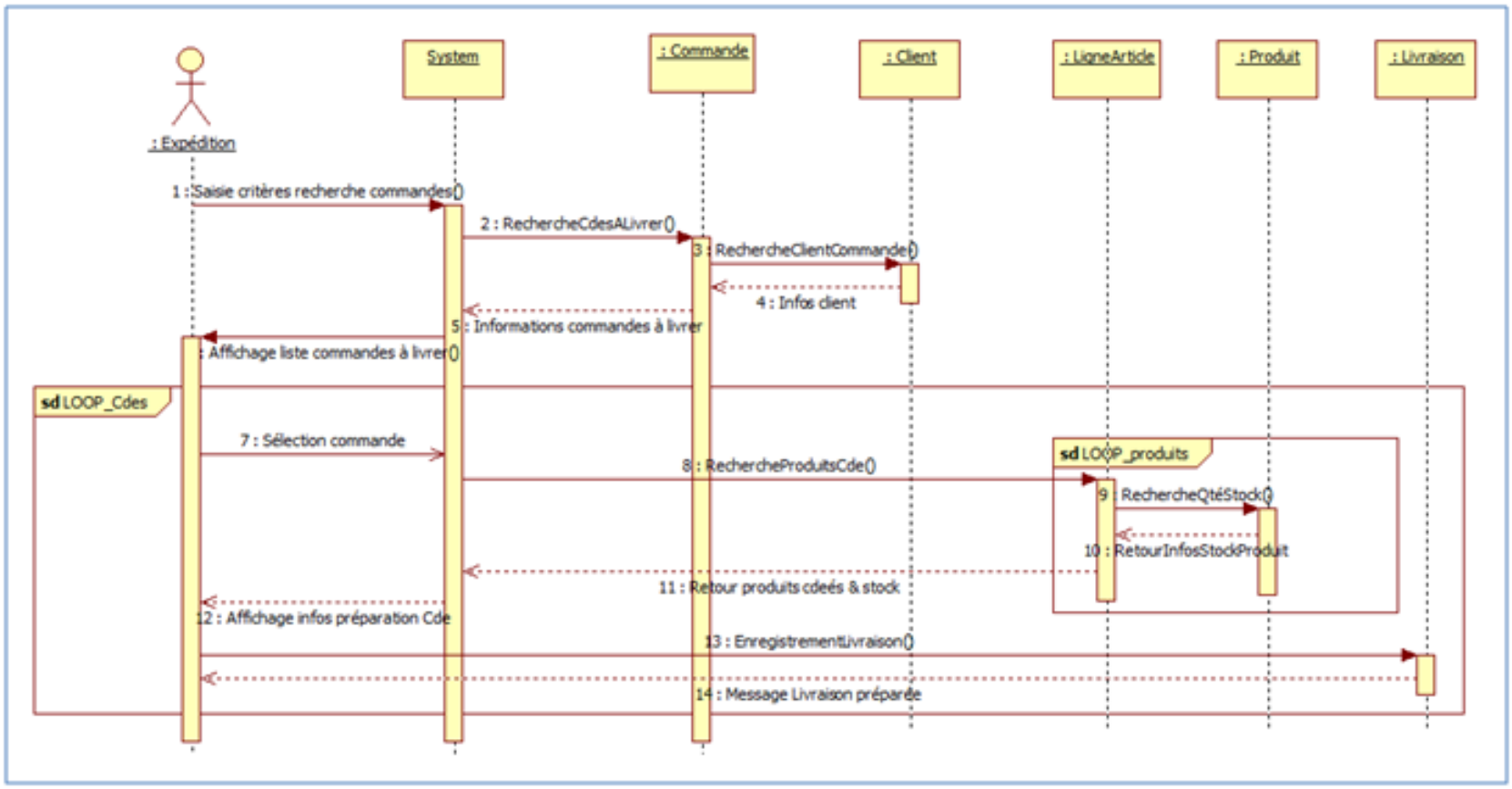
Vue des processus (3)

La vue des processus démontre :

- la décomposition du système en processus et actions ;
- les interactions entre les processus ;
- la synchronisation et la communication des activités parallèles.

La vue des processus s'appuie sur plusieurs diagrammes.

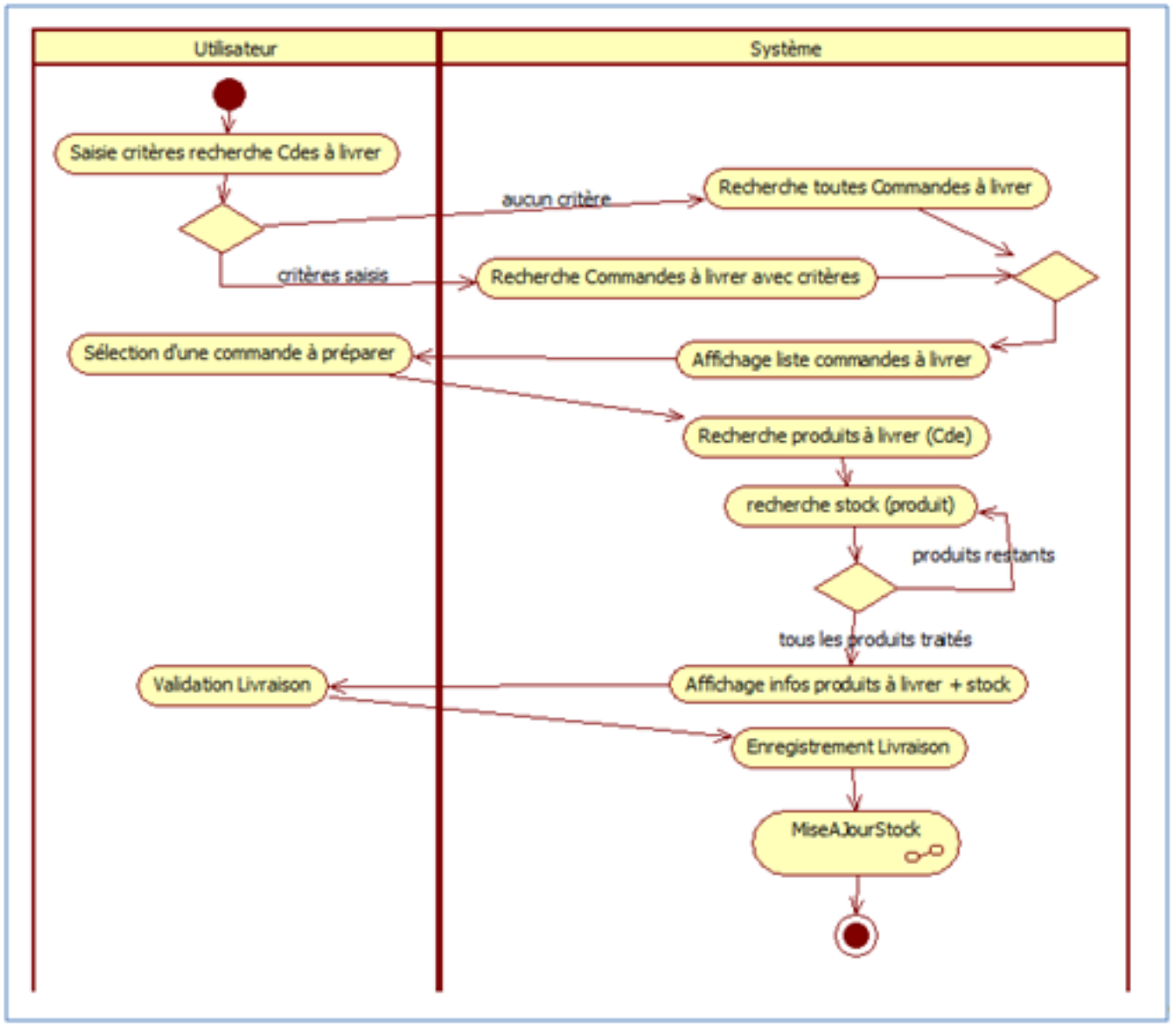
- **Le diagramme de séquence** permet de décrire les différents scénarios d'utilisation du système.



Un exemple de diagramme de séquence

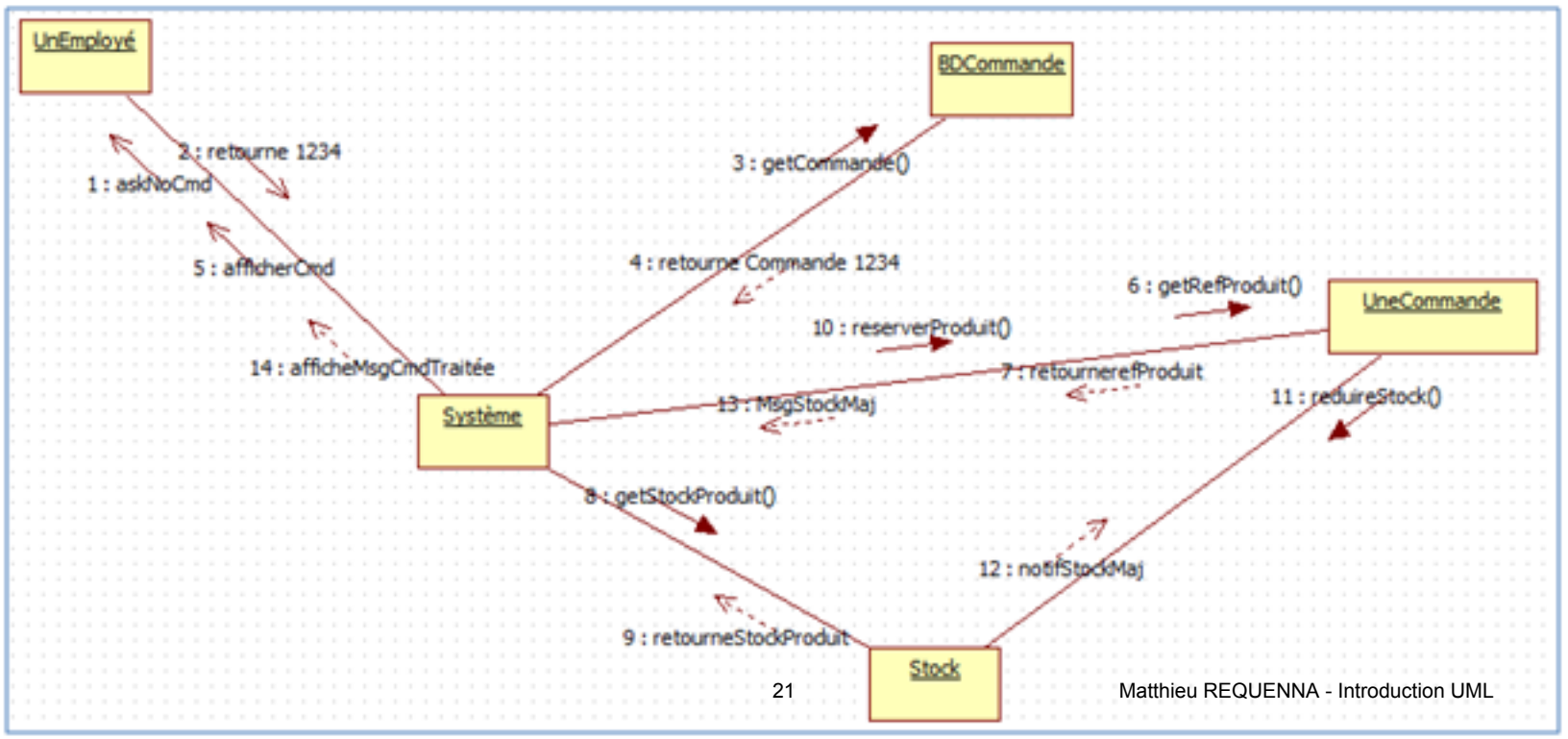
Ce diagramme ne sera pas étudié dans ce cours.

- **Le diagramme d'activité** représente le déroulement des actions, sans utiliser les objets. En phase d'analyse, il est utilisé pour consolider les spécifications d'un cas d'utilisation.



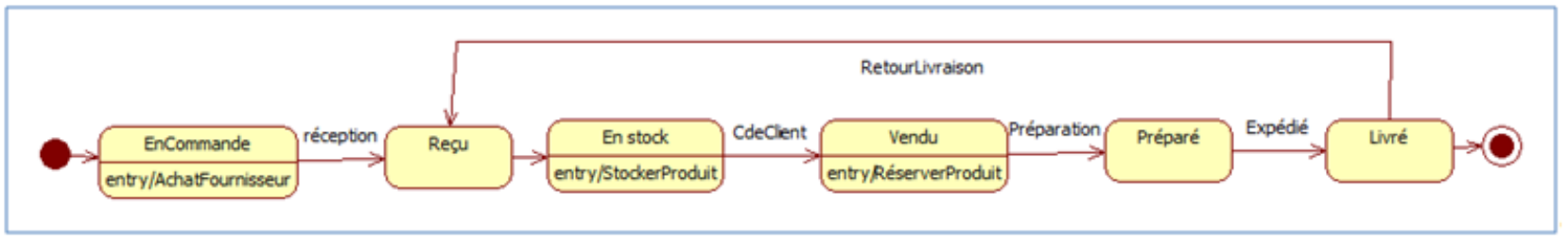
Un exemple de diagramme d'activité

- **Le diagramme de collaboration** (appelé également diagramme de communication) permet de mettre en évidence les échanges de messages entre objets. Cela nous aide à voir clair dans les actions qui sont nécessaires pour produire ces échanges de messages. Et donc de compléter, si besoin, les diagrammes de séquence et de classes.



Un exemple de diagramme de collaboration (de communication)

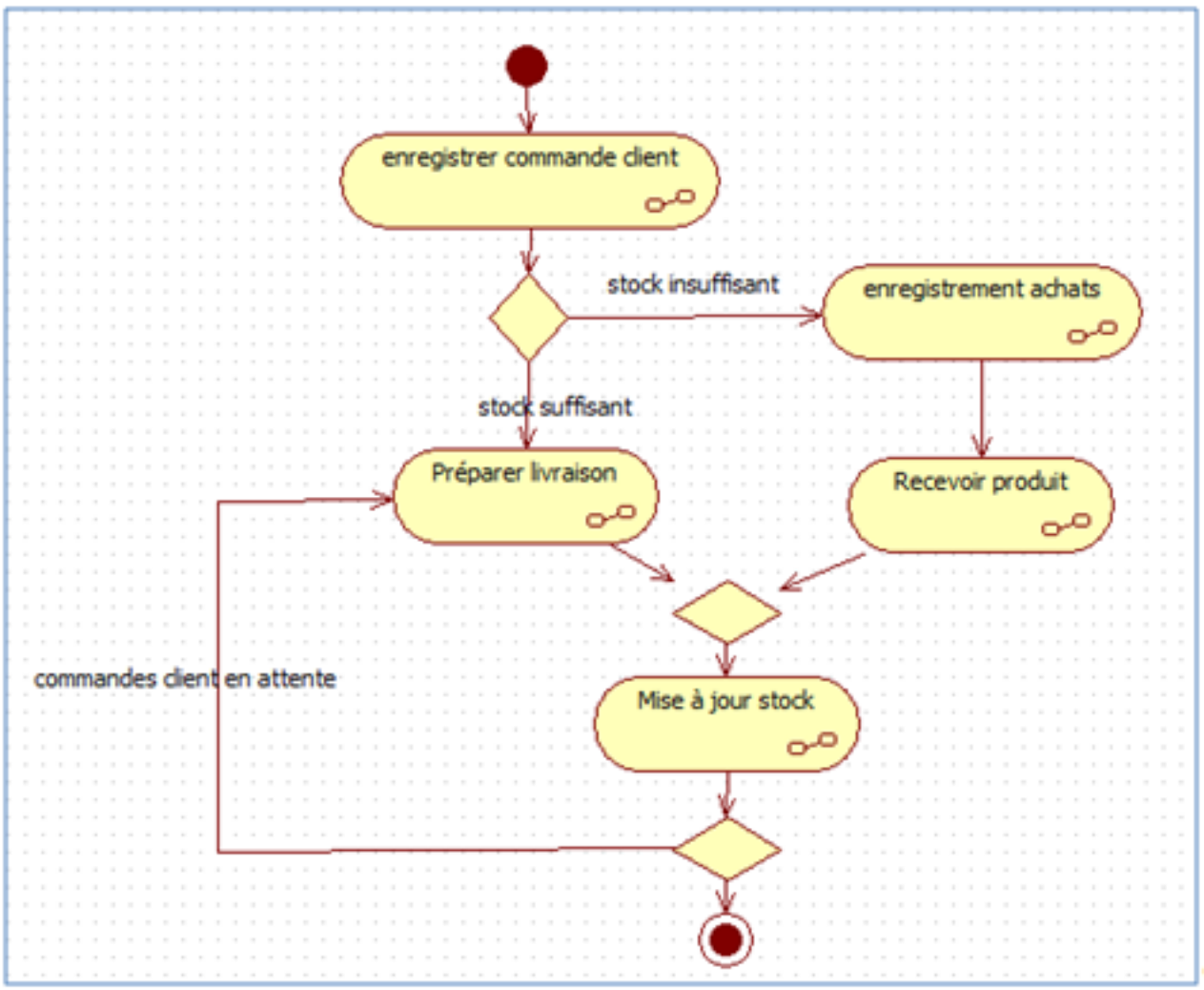
- **Le diagramme d'état-transition** permet de décrire le cycle de vie des objets d'une classe.



Un exemple de diagramme d'état-transition (objets de la classe produit)

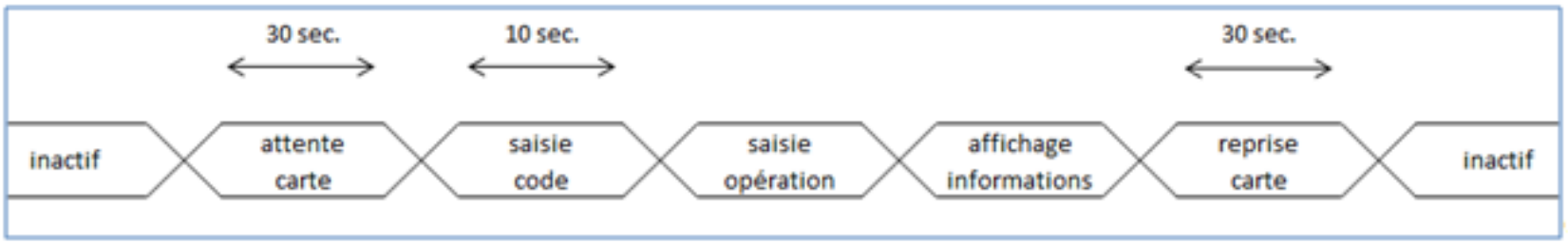
Ce diagramme ne sera pas étudié dans ce cours.

- **Le diagramme global d'interaction** permet de donner une vue d'ensemble des interactions du système. Il est réalisé avec le même graphisme que le diagramme d'activité. Chaque élément du diagramme peut ensuite être détaillé à l'aide d'un diagramme de séquence ou d'un diagramme d'activité. Ce diagramme ne sera pas étudié dans ce cours.

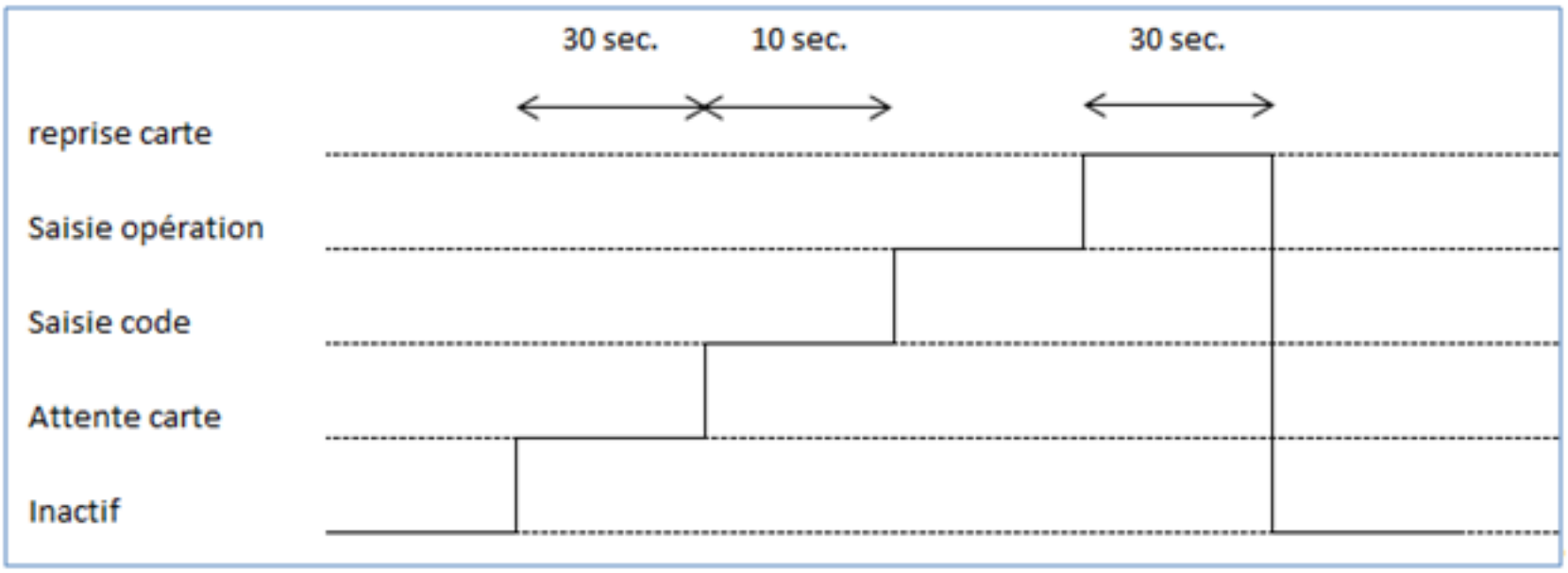


Un exemple de diagramme global d'interaction

- **Le diagramme de temps** est destiné à l'analyse et la conception de systèmes ayant des contraintes temps-réel. Il s'agit là de décrire les interactions entre objets avec des contraintes temporelles fortes. Ce diagramme ne sera pas étudié dans ce cours.



Un exemple de diagramme de temps avec un seul axe temporel



Un exemple de diagramme de temps avec un axe temporel par état

L'aspect lié à l'architecture du logiciel

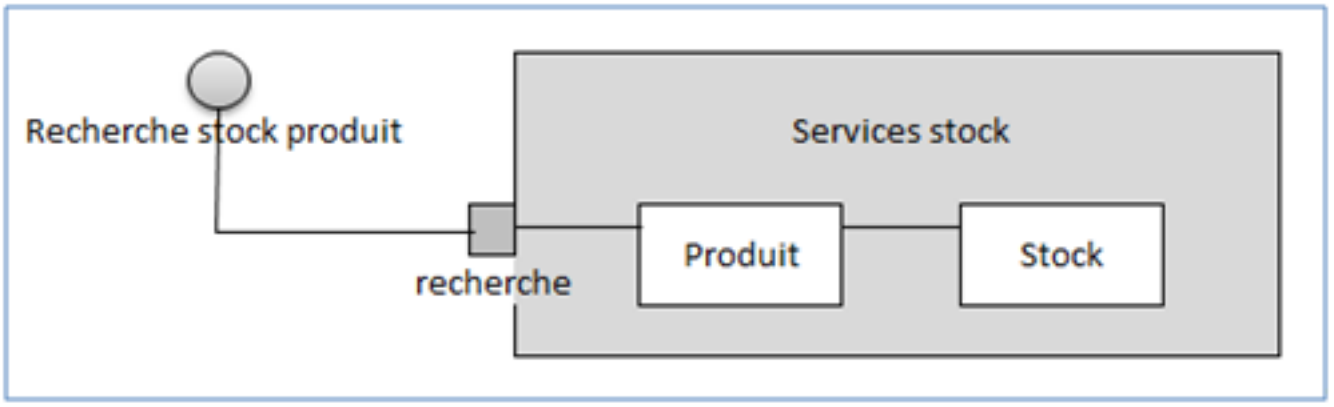
Pour rappel, cette partie du schéma 4+1 vue permet de définir les composantes à utiliser (exécutables, interfaces, base de données, librairies de fonctions, etc.) et les matériels sur lesquels les composants seront déployés.

Vue des composants (4)

La vue des composants (vue de réalisation) met en évidence les différentes parties qui composeront le futur système (fichiers sources, bibliothèques, bases de données, exécutables, etc.). Cette vue comprend deux diagrammes.

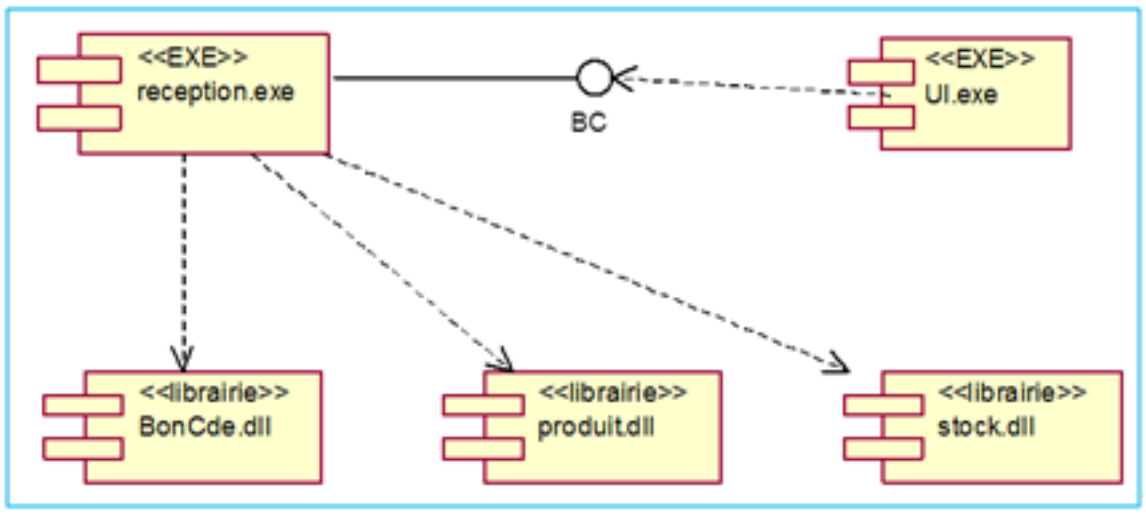
- **Le diagramme de structure composite** décrit un objet complexe

lors de son exécution. Ce diagramme ne sera pas étudié dans ce cours



Un exemple de diagramme de structure composite

- **Le diagramme de composants** décrit tous les composants utiles à l'exécution du système (applications, librairies, instances de base de données, exécutables, etc.). Ce diagramme ne sera pas étudié dans ce cours.

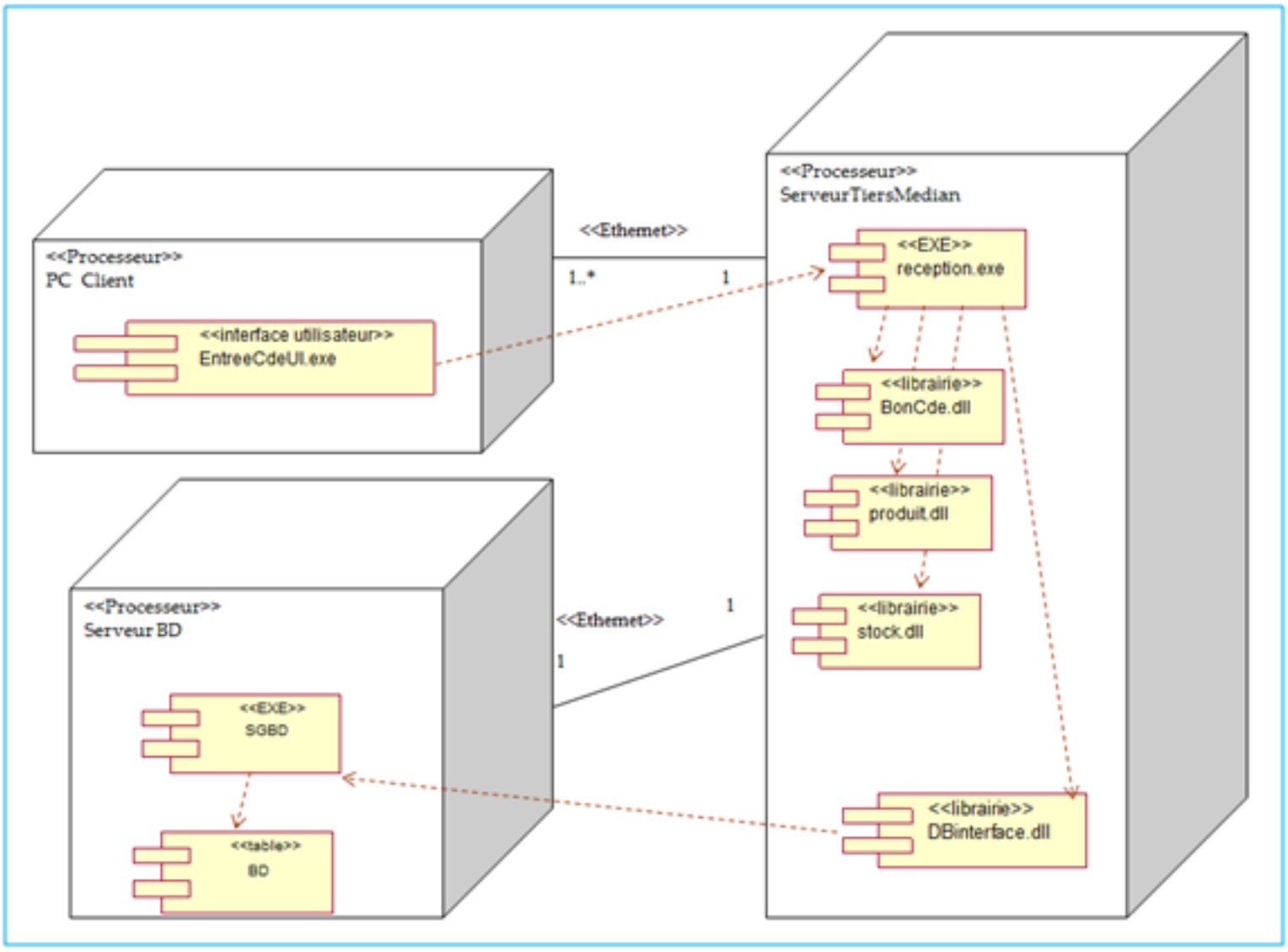


Un exemple de diagramme de composants

Vue de déploiement (5)

La vue de déploiement décrit les ressources matérielles et la répartition des parties du logiciel sur ces éléments. Il contient un diagramme :

- **Le diagramme de déploiement** correspond à la description de l'environnement d'exécution du système (matériel, réseau…) et de la façon dont les composants y sont installés. Ce diagramme ne sera pas étudié dans ce cours.



Un exemple de diagramme de déploiement

Voilà, je viens de vous proposer un aperçu des diagrammes utilisés dans la notation UML. Cela semble complexe, et je vous comprends. Mais rassurez-vous, je vous rappelle que nous n'en verrons que quatre dans ce cours, et je vous accompagnerai pas à pas, à partir d'un cas concret pour donner sens à ces diagrammes. Allez, c'est parti !

En résumé

- 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.

Quelle démarche suivre ?

Je vous rappelle qu'UML ne préconise aucune démarche spécifique. Si vous voulez commencer votre phase d'analyse en réalisant un diagramme de classes, libre à vous.

Toutefois, je suis d'avis qu'il y ait un minimum d'analyse du besoin à faire avant de se ruer sur le diagramme de classes. Rappelez-vous, l'analyse du besoin est la partie centrale des 4+1 vues. Je vais donc ici vous présenter les principales étapes de développement et quelques démarches.

Les étapes de développement logiciel

Le processus de développement logiciel contient un certain nombre d'étapes :

1. Définir les besoins et les exigences du client et des utilisateurs
2. Analyser le système
3. Concevoir le système
4. Programmer le logiciel
5. Tester le logiciel
6. Déployer
7. Maintenir le système

Les étapes d'un projet

Un système peut être un produit ou un ensemble de produits qui seront

livrés au client à l'issu du projet. Lorsqu'il s'agit d'un projet de développement logiciel, le système pourrait donc être un logiciel ou un ensemble de logiciels.

ÉTAPES	DESCRIPTIF
Définition des besoins et des exigences	La définition des besoins et des exigences correspond à l'étape dans laquelle nous discutions avec le client et les futurs utilisateurs afin de comprendre de quoi ils ont besoin : QUI doit pouvoir faire QUOI ? Lors de cette étape, nous définissions également les demandes précises, telles que le respect de certaines normes graphiques, les temps de réponse, le matériel sur lesquels le logiciel devrait fonctionner, etc.
Analyse du système	L'analyse du système permet d'affiner ce qui a été défini dans l'étape précédente. On y détaille davantage le fonctionnement interne du futur logiciel (COMMENT cela doit-il fonctionner ?).
Conception du système	La conception du système correspond à la définition de choix techniques.
La programmation	La programmation est l'étape dans laquelle les informaticiens se donnent à cœur joie ! Ils réalisent le logiciel à l'aide de langages de programmation, de systèmes de gestion de bases de données, etc.
Les tests	Durant les tests, les informaticiens vérifient que le logiciel fonctionne et répond aux besoins définis en début du projet. Cette phase de tests peut intégrer des validations du logiciel avec le client et/ou les utilisateurs. C'est même plus que souhaité.
Le déploiement	Lors du déploiement, les informaticiens installent le logiciel sur le matériel et réalisent des ajustements pour faire fonctionner le logiciel dans l'environnement de travail des utilisateurs.
La maintenance du système	La maintenance correspond à la période qui suit l'installation et pendant laquelle les anomalies et problèmes doivent être corrigés.

Ces étapes ne sont pas forcément utilisées de façon linéaire. On parle souvent de cycles de vie, qui ont pour but d'organiser ces étapes de différentes manières en fonction d'un certain nombre de critères relatifs au projet de développement. Ci-dessous, quelques exemples de cycle de vie.

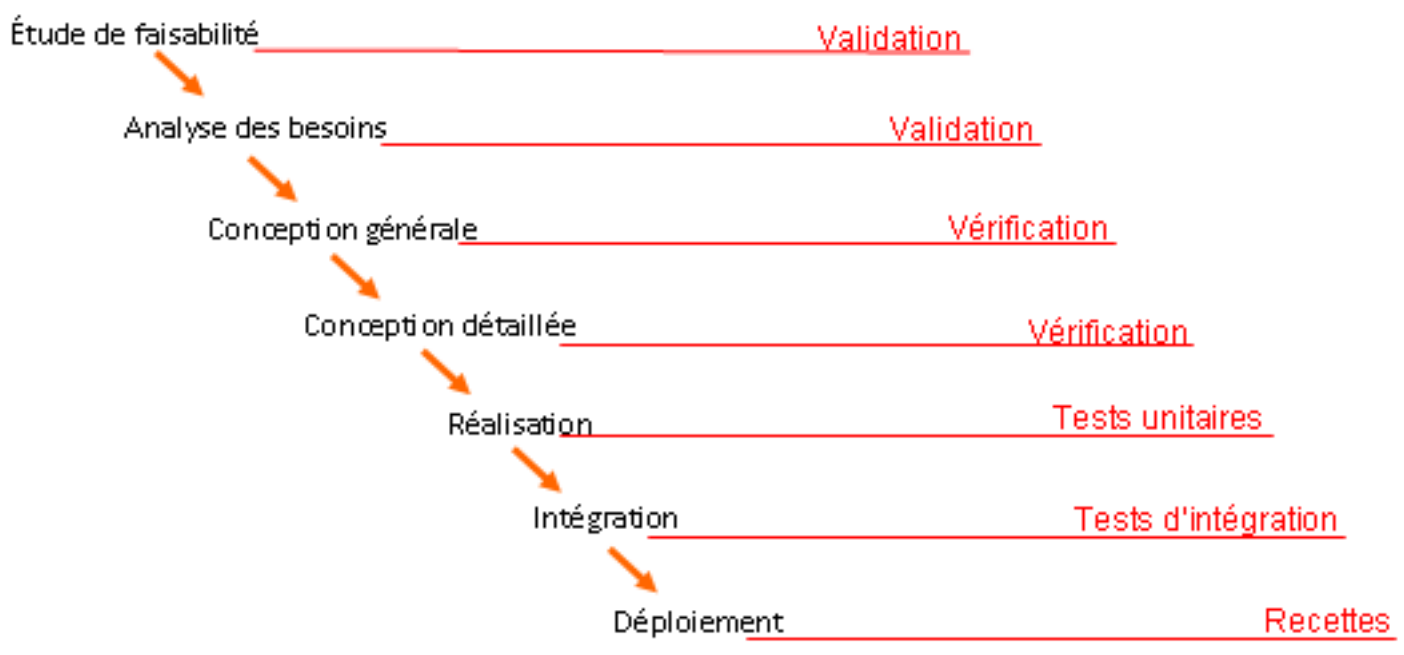
- La cascade
- Le modèle en V
- Le modèle en W
- La spirale
- Le RAD
- Etc.

Voyons ensemble deux exemples afin d'illustrer cela.

Le Cycle de vie en cascade

Exemple d’utilisation :

Le cycle de vie en cascade peut être utilisé lorsqu’un projet est relativement simple, c’est-à-dire pour lequel nous avons la quasi-certitude que les besoins ou exigences n’évolueront pas en cours de projet. En pratique, une étape ne démarre que si la précédente ait été validée par le client et/ou les utilisateurs.



Le cycle de vie en cascade

Les étapes :

Vous constatez probablement que les termes utilisés dans le schéma sont

légèrement différents des 7 étapes indiquées précédemment. Chaque cycle utilise sa propre terminologie, mais nous pouvons voir des correspondances :

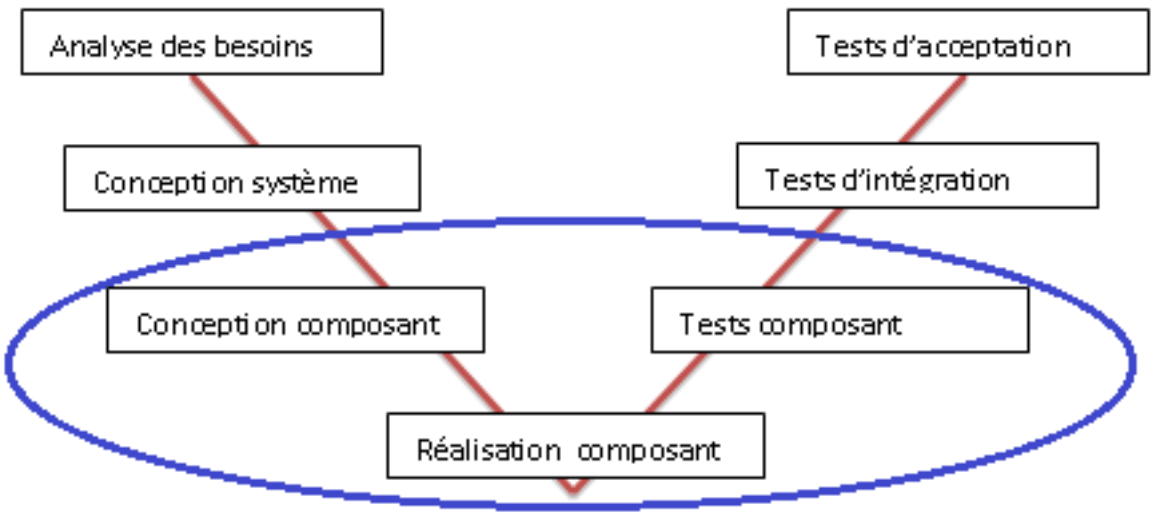
ÉTAPES	ÉTAPES DU CYCLE DE VIE EN CASCADE
Étape n°1 : Étude de faisabilité	Cette étape permet de décider de la nécessité et de l'opportunité de lancer le projet.
Étape n°2 : Analyse des besoins	Cette étape permet de définir les besoins et les exigences des utilisateurs. Cela renvoie à l'étape n°1 énoncé ci-dessus : **La définition des besoins et des exigences.**
Étape n°3 : Conception générale	La conception générale renvoie à l'étape **Analyse du système.**
Étape n°4 : Conception détaillée	Cette étape est équivalente à l'étape **Conception du système**.
Étape n°5 : Réalisation	Cette étape équivaut à l'étape **La programmation.**
Étape n°6 : Intégration	Cette étape équivaut à une partie de l'étape **La programmation.** Notez que l'étape Tests n'est pas vu ici comme une étape, mais comme des validations. Les tests unitaires permettent de tester chaque fonctionnalité du logiciel séparément, alors que les tests d'intégration sont destinés à tester le logiciel dans sa totalité. Ces tests peuvent être faits avec le client et les utilisateurs.
Étape n°7 : Le déploiement	Cette étape est identique.

Notez que ce cycle ne mets pas en avant l'étape de **Maintenance**.

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

Exemple d'utilisation :
Un projet plus important dont les besoins et les exigences risquent d'évoluer pourrait avoir **un cycle en V**. Comme illustré dans la figure suivante, le système à développer serait décomposé en modules. Chaque module serait **conçu, développé et testé séparément**. Les différents modules pourraient alors être intégrés dans le système global au fur et à mesure. Des tests d'intégration permettraient alors de garantir que l'ensemble fonctionne de façon à répondre à la conception générale du système. Les tests d'acceptation permettent ensuite de vérifier la cohérence du système avec les besoins.



Le cycle de vie en V

Les étapes :

ÉTAPES	ÉTAPES DU CYCLE DE VIE EN V
Étape 1 : Analyse des besoins	Cette étape est identique à l'étape n°1 : **Définition des besoins et des exigences.**
Étape 2 : Conception système	Cette étape couvre une partie de l'étape : **l'analyse du système** et de l'étape : **la conception du système**. Il s'agit d'une vision globale du logiciel à développer.
Le système est divisé en composants pour lesquels on réalisera :	
Étape n°3 : Conception	• La conception du composant, c'est-à-dire l'analyse et la conception détaillée du composant

composant	
Étape n°4 : Réalisation composant	• La réalisation ou la programmation du composant
Étape n°5 : Test composant	• Les tests du composant. Cela correspond donc aux étapes **la programmation et les tests**.
Étape n°6 : Tests d'intégration	Les différents composants sont alors intégrés dans un logiciel. Le cycle en V ne mentionne pas d'étape de déploiement mais on pourrait aisément le voir ici.
Étape n°7 : Test d'exploitation	Les tests d'acceptation correspondent à la validation du logiciel par le client et les utilisateurs.

Ce cycle ne met pas en évidence l'étape de **Maintenance**.

Loin de moi l'idée de vous faire un cours sur les cycles de vie. Je voulais juste attirer votre attention sur le fait que **les étapes d'analyse et de conception** ne sont pas forcément à réaliser comme deux gros blocs qui se suivent. Nous allons toutefois les traiter de cette façon dans la suite des chapitres afin de rendre clairement visible la différence entre ces deux étapes.

Quel diagramme pour quelle étape ?

Comme je vous le disais plus haut, 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.

Dans chacune de ces deux parties, nous utilisons un certain nombre de diagrammes UML.

ETAPE : ANALYSE

QUAND ?
Pour définir les **besoins** (contexte et système)

QUEL DIAGRAMME?	POURQUOI?
Diagramme de contexte	Pour identifier les acteurs qui utiliseront le système
Diagramme de cas d'utilisation	Pour indiquer de quelles façons les acteurs utiliseront le système

QUAND ?

Analyse de **domaine**

QUEL DIAGRAMME?	POURQUOI?
Diagramme de cas d'utilisation	Pour **détailler les fonctionnalités** en y ajoutant des liens entre cas d'utilisation
Diagramme d'activité	Afin de **décrire le déroulement** des cas d'utilisation
Diagramme de classes	Pour préciser les **informations** nécessaires pour les actions d'un cas d'utilisation

QUAND ?

Analyse **applicative** (ou analyse de la solution)

QUEL DIAGRAMME?	POURQUOI?
Diagramme de séquences	Afin de détailler le déroulement d'un cas d'utilisation tout en indiquant les informations à utiliser
Diagramme d'état-transition	Afin d'identifier tous les états par lequel un objet peut passer et donc de trouver les actions qui permettent d'obtenir un changement d'état
Diagramme de collaboration (de communication)	Pour identifier les messages échangés entre objets et trouver de nouvelles actions.

ETAPE : CONCEPTION

QUAND ?

Conception de la solution

QUEL DIAGRAMME?	POURQUOI?
Tous les	

diagrammes précédents	Afin de vérifier la cohérence des diagrammes et d'apporter des détails nécessaires à l'implémentation de la solution.
Diagramme de composants	Pour indiquer de quelle façon le logiciel sera construit. Un composant pouvant être un exécutable, un fichier, un module, une base de données, etc.
Diagramme de déploiement	Afin de montrer sur quel matériel chacun des composants devra être installé et pour indiquer les éventuels moyens de communication entre les parties.

Les prochaines parties du cours vous apprendront comment réaliser des diagrammes UML dans la phase d'analyse des besoins et du domaine (hors diagramme de classes).

Le diagramme de classes et les diagrammes utiles dans la phase d'analyse de la solution ne seront pas traités dans ce cours.

Le principe d'itération en UML

Pour rappel, l'itération signifie que l'on fera plusieurs allers-retours sur les diagrammes avant d'avoir un dossier d'analyse entièrement satisfaisant.

Il est utopique de penser que nous pourrons comprendre un besoin exprimé totalement dès le début du projet.

Oui, il nous arrive parfois d'être très optimistes et de penser avoir tout bon du premier coup. Malheureusement, la réalité nous rattrape toujours… souvent un peu trop tard !

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.

Toutes ces raisons font que nous devons revenir sur nos diagrammes un certain nombre de fois.

Je préconise de revenir systématiquement sur l'ensemble des diagrammes lorsque vous modifiez quelque chose dans un diagramme précis. De cette façon, vous allez adoptez la démarche par itérations, de façon assez « naturelle ».

N'est-ce pas ce que l'on a vu précédemment ?

Prenons l'exemple de la réalisation du diagramme de classes.

Lors de la phase d'analyse du besoin, on fait généralement une première version de diagramme de classes après avoir décrit les interactions entre les acteurs et le système (dans les descriptions des cas d'utilisation). Puis, après la description du déroulement des cas d'utilisation, à l'aide de diagrammes de séquence, le diagramme de classes devra être complété avec des éléments complémentaires. Il en va de même après la réalisation d'un diagramme d'état-transition ou d'un diagramme de collaboration. Chacun des diagrammes réalisés pourrait mettre en évidence des éléments à ajouter ou à modifier.

Il faut également garder en tête que la notion de « classe » ou « d'objet » est également utilisé dans d'autres diagrammes, tels que le diagramme de séquence, le diagramme de collaboration, etc.
Une modification du diagramme de classes pourrait donc, à son tour, nécessiter une modification des autres diagrammes. Une modification est complète lorsque tous les diagrammes déjà réalisés sont cohérents.

Dans une démarche de modélisation d'un projet informatique avec UML, les points clés sont :

- une démarche guidée par les besoins utilisateurs ;
- une démarche itérative et incrémentale ;
- une démarche centrée sur l'architecture logicielle ;
- une mise en évidence des liens qui existent entre les actions et les informations.

En résumé

- UML ne préconise aucune démarche.
- La définition d'un logiciel peut être scindée en deux étapes majeures : l'analyse (analyse des besoins, du domaine et de la solution applicative) et la conception.
- L'étape d'analyse comprend 6 diagrammes : diagramme de contexte, diagramme de cas d'utilisation, diagramme de classe, diagramme de séquence, le diagramme d'état-transition et le diagramme de collaboration.
- L'étape de conception apporte des précisions aux diagrammes réalisés lors de l'analyse et comprend 2 diagrammes supplémentaires : le diagramme de composants et le diagramme de déploiement.
- Une démarche itérative permet de garantir que l'analyse soit cohérente et complète.

Cette partie est maintenant terminée. N'oubliez pas de faire vos exercices avant de passer à la partie suivante. À vous de jouer!

La démarche

Lors de la première partie, nous avons vu ce qu'était UML et la démarche que je vous propose de suivre. Dans cette partie, nous allons à partir d'une étude de cas, suivre pas à pas l'analyse des besoins d'un projet en réalisant différents diagrammes. On tentera d'effectuer l'analyse des besoins d'un projet commun : créer une boutique en ligne.

Comme je vous le précisais dans la partie précédente, nous identifierons les premiers besoins grâce aux différents échanges avec le client. On découvrira au fur et à mesure les différents besoins et actions ce que le client souhaite intégrer dans son projet de logiciel, la boutique en ligne. Notre première mission sera donc de décrypter son discours au fur et à mesure afin de préciser ses besoins. C'est ce qu'on appelle la modélisation des besoins fonctionnels. Nous essaierons donc de répondre aux questions suivantes :

- Qui sont les utilisateurs ?
- Que veulent-ils faire avec le logiciel ?

Les étapes

Comme je l'ai indiqué dans l'introduction, il ne sert à rien de commencer à coder sans avoir compris les besoins des utilisateurs au préalable. Sinon, on risque de créer un logiciel qui n'est pas utile, qui est bancale ou en tout cas non satisfaisant pour les utilisateurs.

Mais alors, c'est quoi l'analyse des besoins ?

Analyser les besoins, c'est découvrir des éléments de plus en plus précis.
Voici les étapes :

Étape 1

On commence par décrire le contexte du logiciel à créer.

Qu'est-ce que c'est le **contexte** ? Eh bien, c'est l'environnement direct du logiciel. Il s'agit là de décrire **QUI** devra utiliser le logiciel.
Il faut donc se poser la question : à qui le logiciel devra servir ?

Imaginons par exemple que l'on ait fait partie d'une des équipes d'analyse et de conception de la société MicroSoft. Pour définir le contexte du logiciel « Word » par exemple, on a dû penser aux différents types d'utilisateurs qui s'en serviraient. Je sais, vous n'avez pas réellement fait partie d'une de ces équipes. Moi non plus d'ailleurs. Mais on peut facilement imaginer qu'ils ont pensé aux rédacteurs d'un texte, aux vérificateurs, aux lecteurs, etc.

Étape 2

On décompose ensuite le logiciel en plusieurs parties distinctes, histoire de ne pas avoir à analyser quelque chose de trop énorme d'un coup. On appelle ça **la décomposition en packages**. La décomposition peut se faire en réfléchissant à des « familles » de fonctionnalités qui seraient nécessaires.
Les questions à se poser ? Celles-ci par exemple !

- Quelles sont les grandes parties qui doivent composer le logiciel ?
- Pour une partie précise, qui, parmi les acteurs identifiés (ou utilisateurs) l'utilisera ?

Reprenons notre exemple du logiciel « Word ». On peut le décomposer en plusieurs parties ou packages : la partie « Accueil », la partie « Mise en page », la partie « Publipostage », etc. Oui, on pourrait considérer que

chaque partie correspondra à un menu du logiciel.

La partie « Accueil » sera utilisé aussi bien par un rédacteur que par un vérificateur ou un lecteur.
La partie « Mise en page », en revanche, sera probablement seulement utilisé par un rédacteur.

Étape 3

Chaque partie est alors analysée séparément, en précisant **QUI devra pouvoir faire QUOI** grâce à cette partie du logiciel. C'est ce qu'on appelle **définir les cas d'utilisation.**
Par exemple, demandez-vous, « dans la partie X, qui devra faire quoi avec le logiciel ? »

Toujours sur le logiciel « Word », on distingue des utilisations différentes, en fonction du type d'utilisateur.
Par exemple, dans la partie «Accueil», le rédacteur peut écrire du texte, changer la police et la couleur du texte, aligner le texte, etc.
Dans la partie « Révision », le vérificateur peut insérer des commentaires, indiquer des modifications, etc. Dans cette même partie, le rédacteur peut accepter ou refuser des modifications. Il y a bien d'autres possibilités dans ce logiciel, mais je pense que vous avez saisie ce que je voulais dire.

Pour simplifier la compréhension, on peut considérer que chaque «utilisation» correspond à un sous-menu.

Les outils nécessaires

Pour illustrer chacune des étapes citées ci-dessus, on utilise un diagramme précis, que nous détaillerons au fur et à mesure du cours.

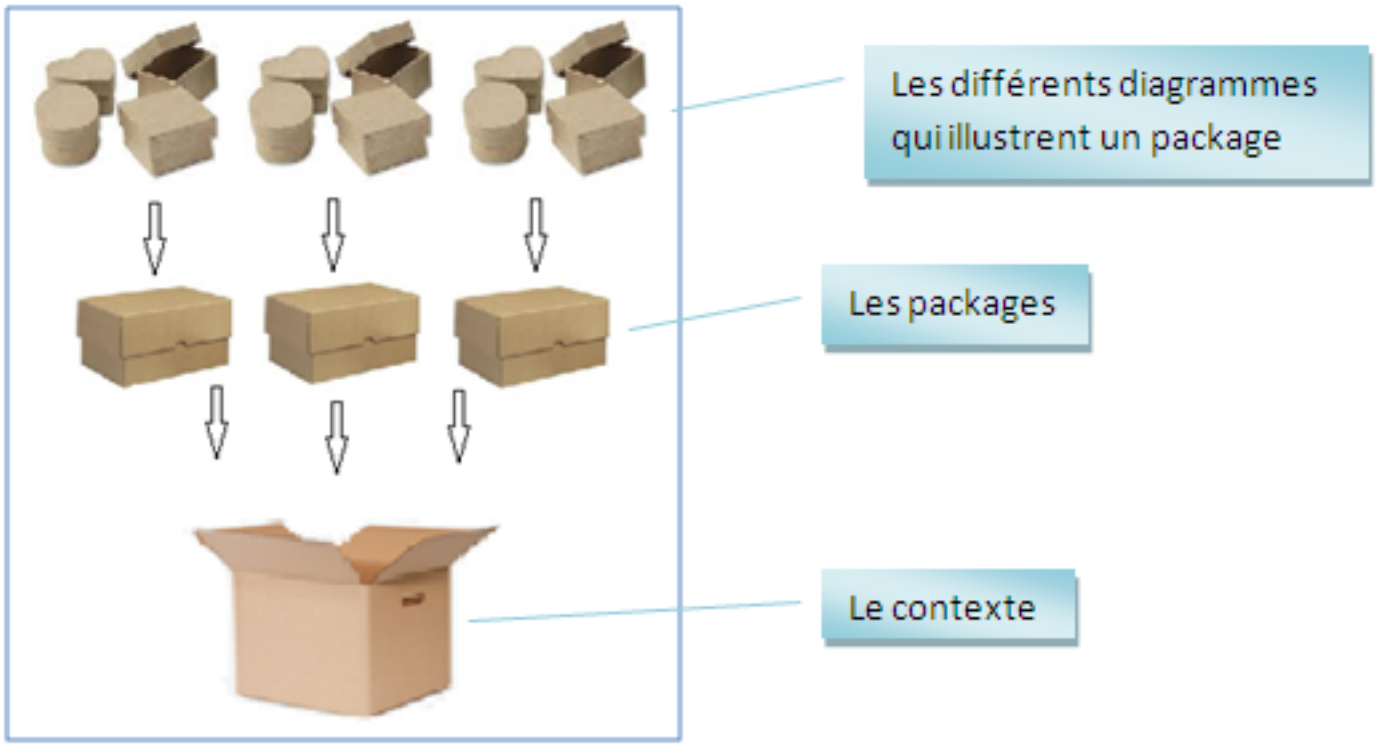
- Pour décrire le contexte, on réalise **un diagramme de contexte** dans lequel on indiquera qui aura une utilité du futur logiciel.

- Pour expliquer la décomposition du logiciel en parties distinctes, on se

sert d'un **diagramme de packages.** Celui-ci nous permettra d'indiquer qui aura besoin de chacune des parties.

- Enfin, pour illustrer ce que le logiciel doit permettre de faire, on utilise **un diagramme des cas d'utilisation.**

En fait, chaque diagramme est une précision du précédent. C'est comme si en ouvrant une grande boîte en carton, on découvrait d'autres boîtes. D'ailleurs, notre découverte ne s'arrêtera pas au diagramme des cas d'utilisation, mais là j'avance un peu sur les parties à venir…



Décomposition de l'analyse

Étude de cas

Comme promis, je vous propose de découvrir l'analyse de besoins d'un projet de logiciel à partir d'un projet commun : créer la boutique en ligne d'une entreprise de papeterie. Nous découvrirons pas à pas la démarche à suivre. Voici le projet.

Qui est notre client ?

Notre client, Monsieur Morin, est le directeur d'une petite entreprise commerciale, MORIN OFFICE, spécialisé dans les produits de bureau. Son entreprise est localisée à Blois.

Que vend-il ?

Voici la liste de produit qu'il vend sur place :

- Matériel de base : crayons, stylos, gommes, papier, cahiers, etc.
- Matériel électronique : ordinateurs, imprimantes, téléphones, etc.
- Mobilier : tables, bureaux, chaises, armoires, etc.

Pourquoi nous contacte-t-il ?

L'entreprise doit faire face à la concurrence de grandes enseignes et a le projet de moderniser toute sa démarche commerciale. Il souhaite passer au e-commerce, c'est-à-dire vendre ses produits sur Internet ! L'entreprise espère ainsi obtenir une meilleure visibilité et obtenir une augmentation des ventes.

Que souhaite-t-il faire sur le site de vente en ligne ?

Lors d'une première discussion avec notre client Monsieur Morin, nous avons pu obtenir les précisions suivantes :

1. Le client (ou client potentiel) doit pouvoir consulter les produits et éventuellement procéder à un achat en ligne.
2. Les commerciaux de la société doivent pouvoir consulter le catalogue des produits en ligne et enregistrer les achats des clients.
3. Le service des livraisons doit pouvoir consulter les commandes pour préparer les colis et les livrer au client.
4. Le technicien doit pouvoir vérifier d'éventuelles remarques ou messages signalant un dysfonctionnement lors de l'achat en ligne.
5. Le service administratif de la société doit pouvoir :
   - ajouter de nouveaux produits au catalogue en ligne ;
   - modifier les descriptions ou les prix des produits ;

- retirer si besoin des produits que l'on ne souhaite plus proposer.

6. Le directeur, quant à lui, souhaite avoir une vision globale des ventes. À travers le site, il souhaite pouvoir :
   - faire un suivi du chiffre d'affaires par mois, sur une certaine durée ;
   - voir quels produits sont les plus vendus sur une durée donnée.

Étape 1 : Définir le contexte du futur logiciel

Notre futur logiciel ? Créer une boutique en ligne. Au départ, le logiciel peut nous sembler un peu « nébuleux ». On a souvent l'impression de ne pas savoir par quel bout commencer. Rassurez-vous, cela est tout à fait normal.

La boîte noire

On commencera d'ailleurs par considérer que le futur logiciel correspond à une boîte noire qui doit fournir des services à son environnement. Par environnement, on entend les **utilisateurs** qui ont besoin de ce logiciel. Dans UML, on appelle ce qu'on doit analyser, concevoir et réaliser : le système. Ici, le **système** est donc le site de vente en ligne.

Le système, style « boîte noire »

Cette « boîte noire » sera donc utile à un ou plusieurs **utilisateurs**. D'ailleurs, on devrait parler d'acteurs et non d'utilisateurs.

Les acteurs

Un **acteur** correspond à une entité (humain ou non) qui aura une interaction avec le système. Parmi les acteurs, nous distinguons :

- les **acteurs principaux** agissent directement sur le système. Il s'agit

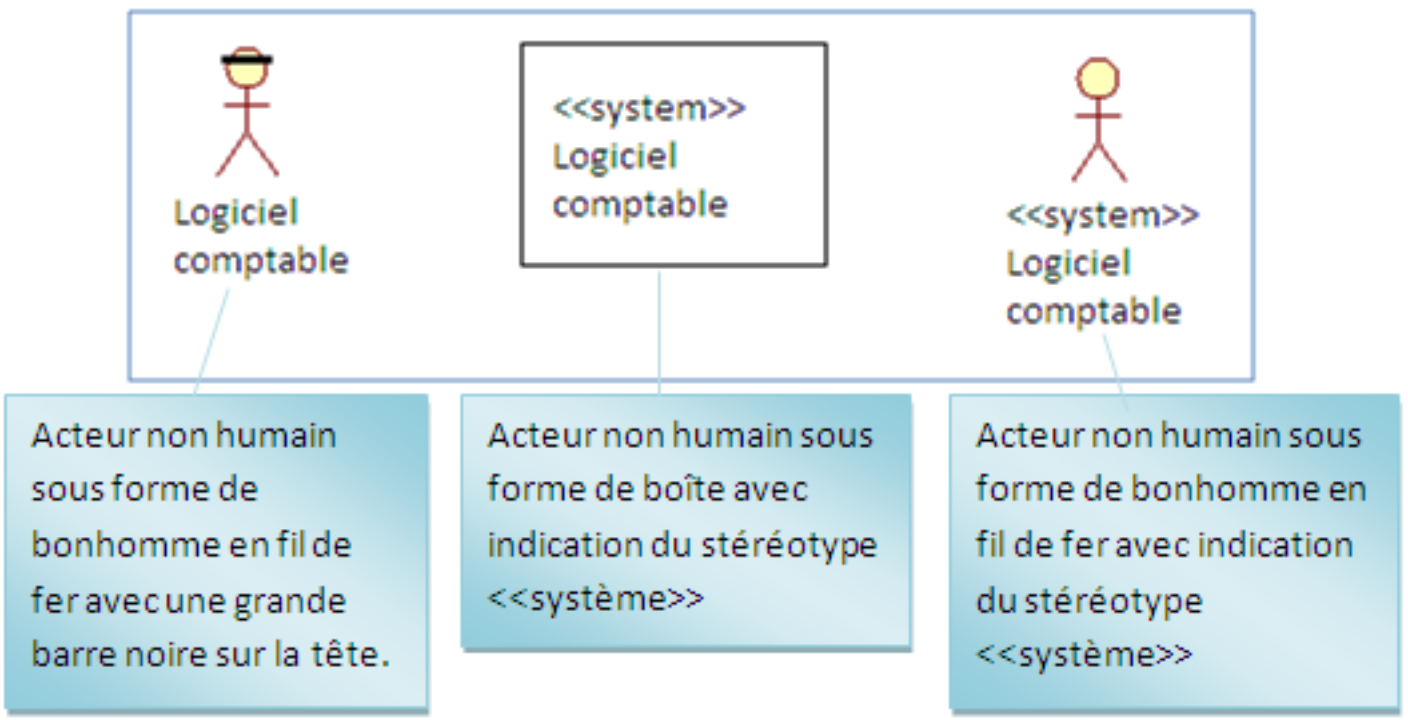
d'entités qui ont des besoins d'utilisation du système. On peut donc considérer que les futurs utilisateurs du logiciel sont les acteurs principaux.

- les **acteurs secondaires** n'ont pas de besoin direct d'utilisation. Ils peuvent être soit consultés par le système à développer, soit récepteur d'informations de la part du système. Cela est généralement un autre système (logiciel) avec lequel le nôtre doit échanger des informations.

Certains acteurs sont de type humain. Ils sont alors représentés par un bonhomme en fil de fer (parfois appelé « stick man » en anglais) et on indique leur rôle en dessous.

La représentation d'un acteur de type humain

Pour représenter un acteur de **type non-humain**, on peut utiliser une représentation graphique différente et/ou ajouter une indication supplémentaire (appelé le **stéréotype**). Vous pouvez voir les différentes représentations du même acteur dans le schéma suivant.



La représentation d'un acteur de type non-humain

La représentation graphique des acteurs non-humains diffère en fonction

de l'outil de modélisation utilisé. Dans ce cours, j'utiliserai la 3e représentation.

Vous l'avez compris, l'environnement du système est composé d'acteurs qui agissent sur le système ou avec lesquels le système aura une interaction.

À cette étape, rien ne garantit que notre vision de l'environnement de notre boutique en ligne soit complète. Les diagrammes qui seront réalisés par la suite peuvent nous obliger à revenir en arrière afin d'ajouter ou de retirer des acteurs. Nous avons ici, une possibilité d'itération.

Le contexte de notre projet

Allez, il est temps de se mettre au travail ! Je vous propose de commencer par l'étape 1 : Décrire le contexte du logiciel, c'est-à-dire la boutique en ligne de la société MORIN OFFICE.

Relisez-bien les demandes de Monsieur MORIN. Pouvez-vous identifier les acteurs de notre site de vente en ligne ? Parmi ces acteurs, lesquels sont des acteurs principaux et des acteurs secondaires ?

Allez, voyons cela ensemble ! Je vous propose de reprendre une partie de la demande du client, MORIN OFFICE.

Que souhaite-t-il faire sur le site de vente en ligne ?

Lors d'une première discussion avec notre client Monsieur Morin, nous avons pu obtenir les précisions suivantes :

1. ***Le client (ou client potentiel)** doit pouvoir consulter les produits et éventuellement procéder à un achat en ligne.*

2. ***Les commerciaux** de la société doivent pouvoir consulter le catalogue des produits en ligne et enregistrer les achats des clients.*

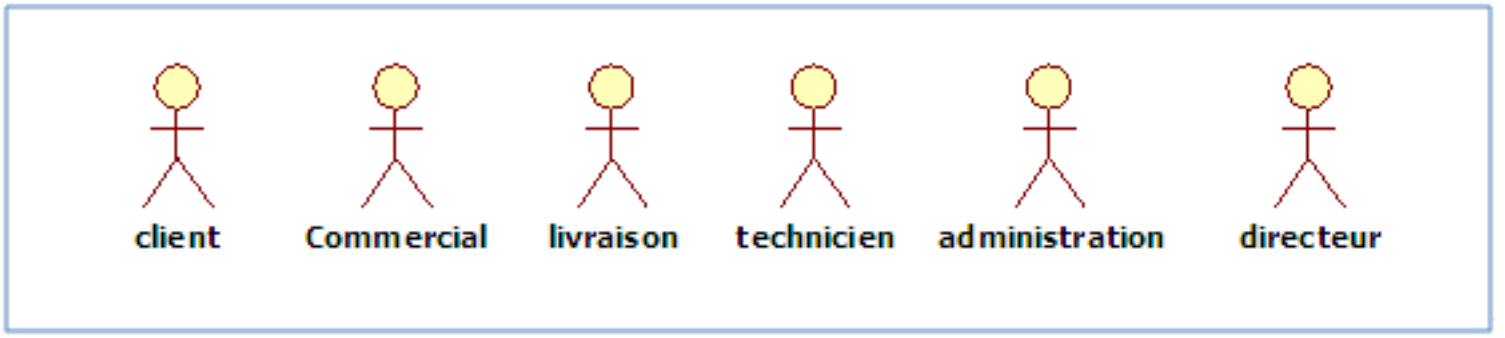
3. ***Le service des livraisons** doit pouvoir consulter les commandes pour préparer les colis et les livrer au client.*

4. ***Le technicien*** *doit pouvoir vérifier d'éventuelles remarques ou messages signalant un dysfonctionnement lors de l'achat en ligne.*

5. ***Le service administratif*** *de la société doit pouvoir :*
   - ajouter de nouveaux produits au catalogue en ligne ;
   - modifier les descriptions ou les prix des produits ;
   - retirer si besoin des produits que l'on ne souhaite plus proposer.

6. ***Le directeur****, quant à lui, souhaite avoir une vision globale des ventes. A travers le site, il souhaite pouvoir :*
   - faire un suivi du chiffre d'affaire par mois, sur une certaine durée ;
   - voir quels produits sont les plus vendus sur une durée donnée.

Vous avez trouvé les mêmes que moi ? Il s'agit là d'acteurs principaux, puisqu'ils devront **tous** utiliser le système pour des actions précises (voir la figure suivante).

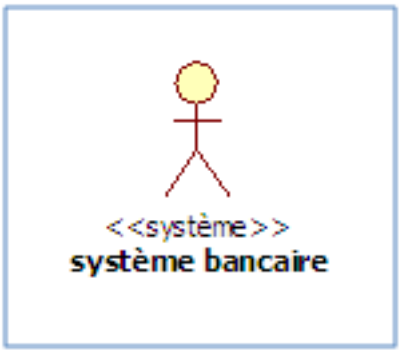


Les acteurs principaux

Vous constaterez qu'on indique un seul acteur pour « client », alors qu'il très souhaitable d'en avoir plusieurs ! Un acteur est bien plus un rôle qu'une personne physique.

Voilà, nous avons nos acteurs. Par contre, il en manque un, qui n'a pas été clairement exprimé par notre client. Réfléchissons un peu… Aujourd'hui, on n'imagine mal un site commercial sans paiement en ligne. Même si le texte ne le mentionne pas, il faut évidemment le proposer au client.

Pour cela, il nous faudra donc un échange avec un système externe : le système bancaire, par exemple, Paypal ou un autre. Voilà, nous venons d'identifier **un acteur secondaire.** Voici comment le représenter : un bonhomme de fil de fer avec un **stéréotype « système »**.



L’acteur secondaire

Voilà, nous avons tous les éléments pour réaliser le diagramme de contexte.

D’accord, j’ai le contexte (c’est-à-dire le système) et les acteurs. Comment je mets en forme tout ça ?

Les acteurs sont placés à côté de la boîte noire, en fonction de leur type. On place généralement les acteurs principaux à gauche et les acteurs secondaires à droite du système.

Le diagramme de contexte

Ça y est, nous venons de créer notre premier diagramme ! Continuons maintenant avec la découverte du contenu du système, c’est-à-dire l’étape 2 : La décomposition en package.

Étape 2 : Décomposer le système en parties, dit « package

Mais pourquoi faut-il décomposer le système en parties ?

Un proverbe chinois connu dit qu'un éléphant ne peut pas être avalé d'un coup. La bête doit être coupée en morceaux si on veut en venir à bout, puis les morceaux sont cuisinés séparément avant d'être mangés.

Ça vous fait sourire ? Vous n'avez jamais eu à manger un éléphant ?
Moi non plus, mais je trouve l'image amusante et je dois avouer qu'un gros logiciel à développer me fait souvent penser à quelque chose d'aussi énorme qu'un éléphant.

Découper en package

Les besoins très différents des acteurs et le nombre de fonctionnalités dont le futur logiciel devra disposer nous semble assez souvent compliqué. Pour y voir clair et pour nous faciliter la tâche, 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**.

Pas de panique si les parties ne sont pas évidentes à voir au départ. On peut toujours procéder avec l'étape suivante : la réalisation du diagramme des cas d'utilisation (cf. chapitre 2.2), pour le logiciel complet. Le diagramme risque de devenir énorme, mais très souvent, on constate alors des groupes de cas d'utilisation qui ont un lien. Il suffit alors de revenir un petit peu en arrière afin de faire la décomposition en packages. Là encore, on fait de l'itération.

Vous l'avez compris, on ne souhaite pas réaliser des diagrammes énormes qui deviendraient très vite illisibles. On décompose le logiciel à développer en plusieurs packages, dès que possible.

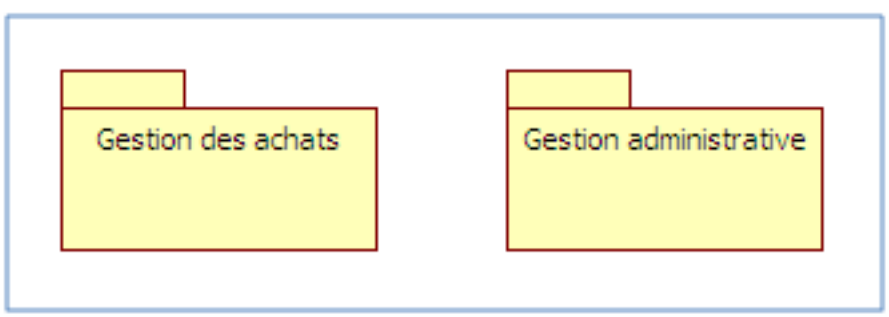
Allez, revenons à notre projet de boutique en ligne ! Essayez d'identifier les parties bien distinctes parmi les fonctionnalités de notre site. Pour cela, posez-vous cette question : « Est-ce que le logiciel comporte des parties différentes qui pourraient être analysées séparément ?

Pour ma part, je vois deux parties distinctes :

- La partie « Gestion de l'achat » qui contiendrait l'achat à proprement parler, la préparation de la commande et la vérification des remarques et problèmes liés à l'achat en ligne.

- La partie « Gestion administrative », incluant :
  - les tâches liées au catalogue : ajouter ou retirer un produits, modifier le produit, etc.
  - les vérifications des ventes : chiffre d'affaires et produits les plus vendus par exemple.

Nous aurons donc deux packages. Un package est représenté sous forme de dossier :



Les packages du système

Comment fait-on pour déterminer les parties ? Est-ce qu'on doit forcément trouver les mêmes parties que vous ?

En réalité, peu importe que vous ayez vu les mêmes parties que moi. Le but étant de trouver des parties qui regroupent des fonctionnalités qui ont un lien ou qui font partie d'une même « famille » et que l'on peut analyser

séparément. On essaye autant que possible de trouver des parties qui n'ont pas trop de liens entre elles.

Nous ne voyons pas encore les liens entre les packages. Pour cela, on devra avancer un peu plus dans l'analyse.

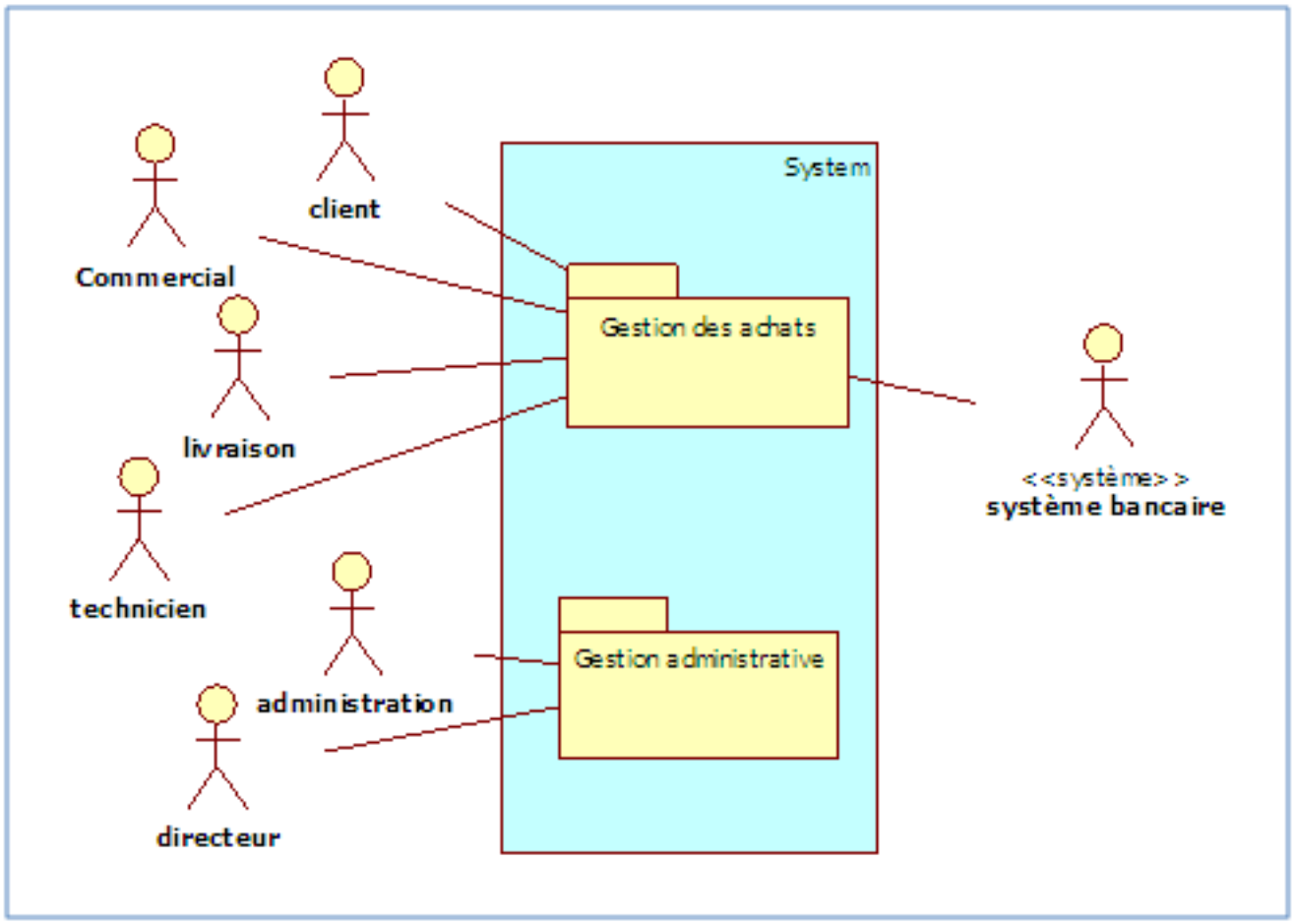
Ne faites pas l'erreur de décomposer le système en fonction des acteurs. Un package (ou une partie) peut être utilisé par plusieurs acteurs !
Il n'est donc pas utile de faire des packages « client », « livraison », « technicien », « administration » et « directeur ».

Réaliser le diagramme

Il ne nous reste plus qu'à réaliser le **diagramme de packages,** en mettant en évidence les acteurs qui interviennent dans chacun de ces packages. Pour cela, on représente au centre le Système, qui comprend les deux packages que nous avons trouvés : la gestion des achats et la Gestion administrative. Autour de cela, nous devons donc relier les acteurs principaux et secondaires au package correspondant !

Ici, le Client, le Commercial, la Livraison et le Technicien sont les acteurs principaux, et le système bancaire est l'acteur secondaire du package Gestion des achats.

L'administration et le directeur sont les acteurs principaux du package Gestion administrative.



Le diagramme de packages

Voilà, c'est notre deuxième diagramme ensemble ! Continuons maintenant avec l'étape 3 de notre analyse de besoins principaux des utilisateurs : les cas d'utilisation.

Étape 3 : Les cas d’utilisation

Nous avons donc identifié les acteurs, et les grandes familles de fonctionnalités (packages). Maintenant, il s’agit de définir plus en détail les besoins de chaque acteur dans ces deux packages, en répondant à ces questions : QUI devra pouvoir faire QUOI grâce au logiciel ? Par exemple, que souhaite faire le technicien sur le site ? Que souhaite faire le client ? Etc.

Les cas d'utilisation

Rappelez-vous, nous avons évoqué des boîtes qui contiennent d’autres boîtes. Notre système (ou logiciel) est une boîte qui contient d’autres boîtes, les packages donc. 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. Un de ces diagrammes représente **les cas d’utilisation,** c’est-à-dire les fonctionnalités ou lots d’actions que devront réaliser nos acteurs. (Voir la figure 2.1.A)

Les acteurs principaux qui sont liés à un package auront besoin de cette partie du logiciel pour réaliser une ou plusieurs lots d’actions. Ces lots d’actions sont le contenu de la boîte « package ».

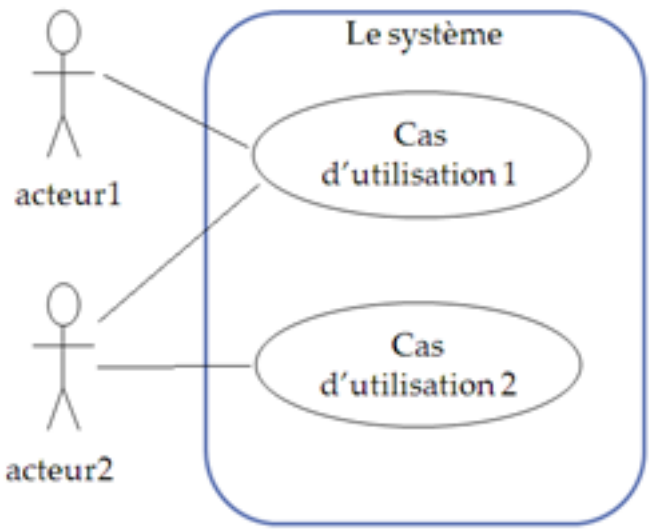
Oui, j’ai bien dit des « lots d’actions » et non des actions. Vous vous demandez quelle est la différence ?

Pensons de nouveau au logiciel Word et imaginons que le rédacteur d’un texte veuille mettre en forme une partie du texte. Le lot d’actions peut être « la mise en forme d’une partie de texte ». Ce lot d’actions comprend plusieurs actions élémentaires, telles que : « sélectionner une partie de texte », « choisir une police différente », éventuellement « aligner la partie du texte d’une manière spécifique », etc.

Un lot d'actions correspond à une fonctionnalité dont certains acteurs principaux ont besoin. On appelle cela **un cas d'utilisation**. Il s'agit, en effet, d'une utilisation particulière du logiciel. **Le diagramme des cas d'utilisations** met donc en évidence de quelle façon les acteurs utiliseront le logiciel : QUI doit pouvoir faire QUOI ?

Réaliser le diagramme

Comment on illustre un cas d'utilisation ?



Exemple diagramme de cas d'utilisation

Pour rappel, les cas d'utilisation principaux sont liés aux acteurs qui en ont besoin. Ils détaillent les diagrammes de packages.

Allez, il est temps d'appliquer ce que nous venons de voir !

Concentrons-nous sur le package « Gestion des achats » pour en réaliser **le diagramme des cas d'utilisation.**

Il faut donc identifier toutes les fonctionnalités dont les différents acteurs concernés par le package auront besoin. Dans notre projet de réalisation d'une boutique en ligne, nous indiquons qu'un client devra pouvoir utiliser le site pour consulter le catalogue des produits. Nous pouvons en déduire que « Consulter le catalogue des produits » est un cas d'utilisation.

Le cas d'utilisation « Consulter le catalogue » de l'acteur principal « Client » est représenté sous forme d'ellipse. On le schématisera ainsi :

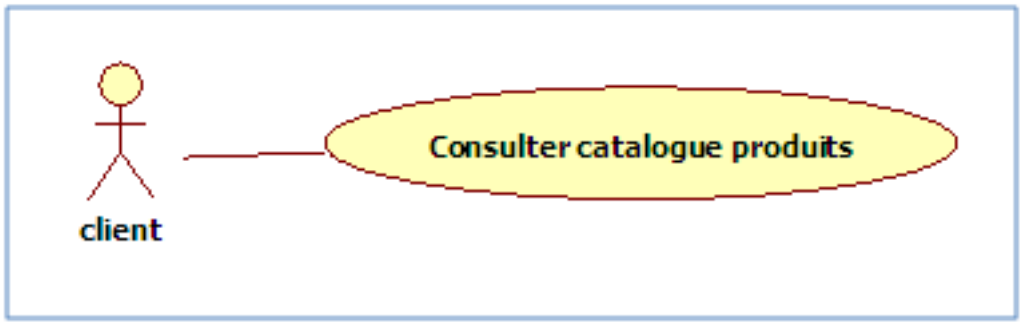


Diagramme de cas d'utilisation, package « Gestion des achats » v1

Un cas d'utilisation n'est pas forcément lié à un seul acteur. Nous pouvons indiquer qu'un commercial devra également pouvoir consulter le catalogue des produits (par exemple, pour montrer des caractéristiques d'un produit lors d'une visite chez son client) en reliant cet acteur au même cas d'utilisation !

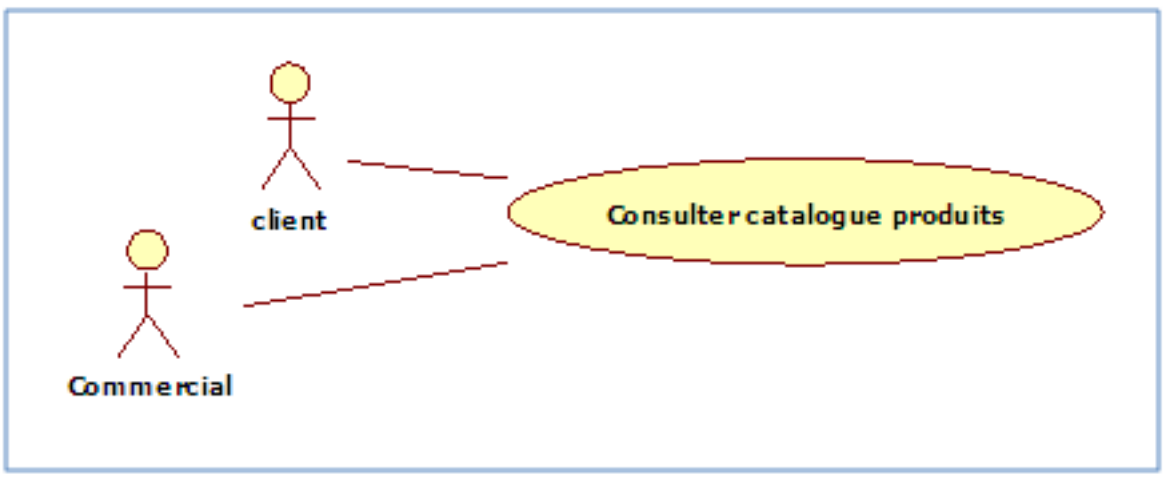


Diagramme de cas d'utilisation, package « Gestion des achats » v2

Pouvez-vous identifier d'autres fonctionnalités liées à chacun de nos acteurs ? Pour rappel, il s'agit de :

- Client
- Commercial
- Livraison
- Technicien

En observant attentivement la demande de notre client, Monsieur MORIN, nous pouvons noter les besoins suivants :

Le client souhaite…	Consulter le catalogue de produits

	Enregistrer un achat
Le commercial doit pouvoir…	Consulter le catalogue de produits Enregistrer un achat
La livraison (ou les membres du service) souhaite(nt)…	Préparer une livraison
Le technicien souhaite…	Consulter une remarque ou problème

Pour le package « Gestion des achats », ceci nous donne une version de diagramme de cas d'utilisation un peu plus complète.

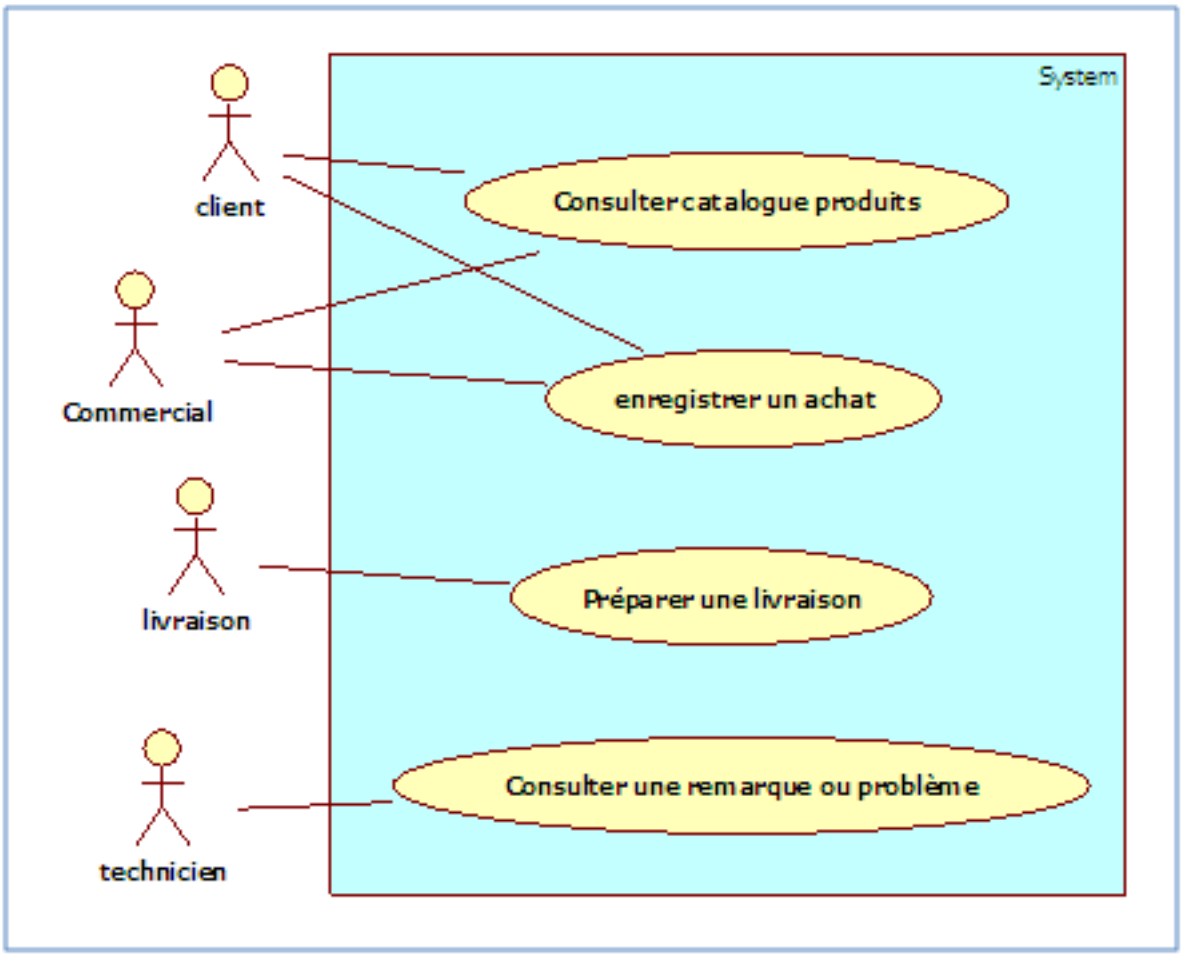


Diagramme de cas d'utilisation, package «Gestion des achats» v3

Il nous manque encore un élément. Dans le diagramme de packages, on avait illustré un lien entre le package « Gestion des achats » et un système externe c'est-à-dire le système bancaire. Il faut maintenant lier ce système externe au cas d'utilisation qui en a besoin. Il s'agit, bien entendu du cas d'utilisation « Enregistrer un achat ».

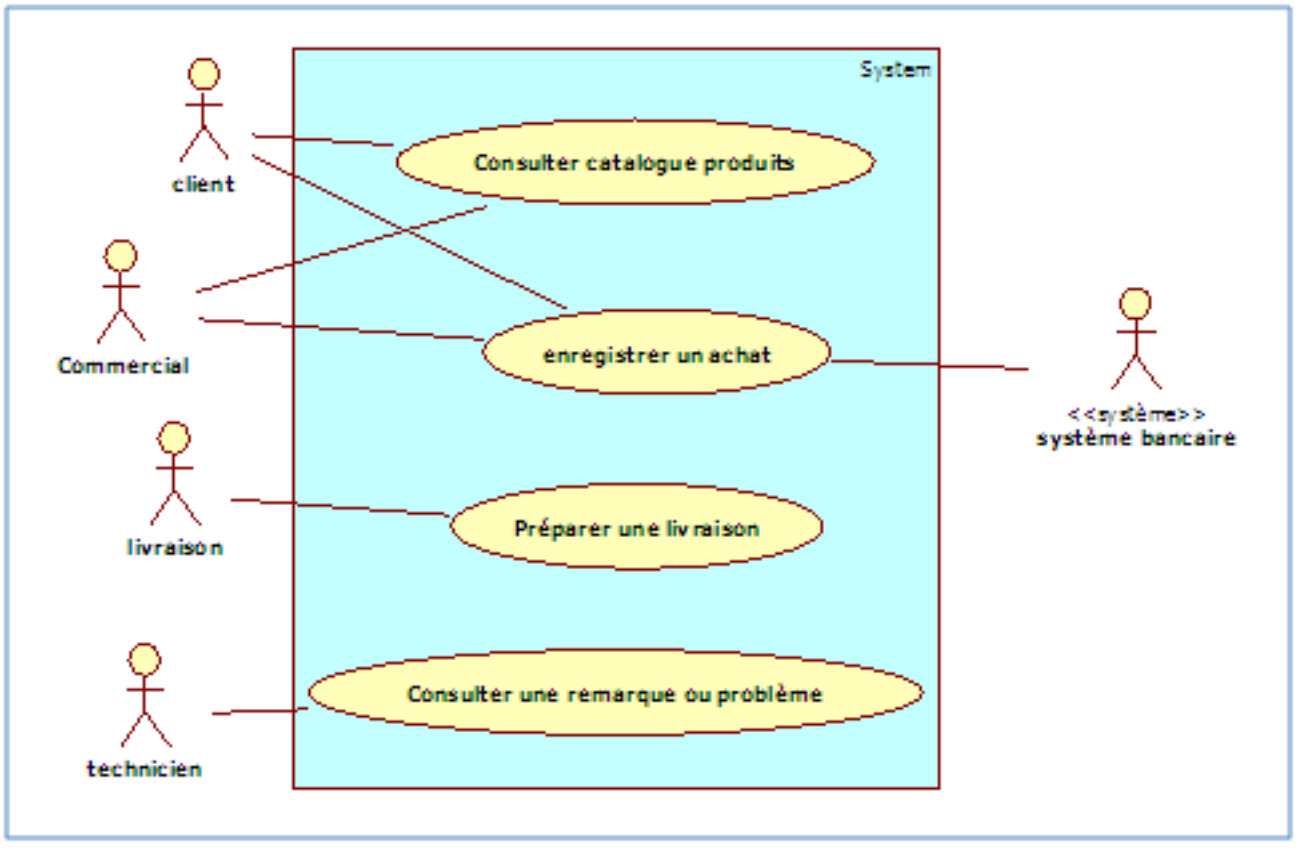


Diagramme de cas d'utilisation, package « Gestion des achats » v4

Bravo ! Nous venons de réaliser une partie importante dans l'analyse des besoins.

Nous avons :

- défini le contexte du futur logiciel, en identifiant les différents acteurs (1) ;
- décomposé le système en packages, c'est-à-dire les familles de fonctionnalités (2) ;
- illustré les fonctionnalités principales pour un des packages, grâce aux cas d'utilisation (3).

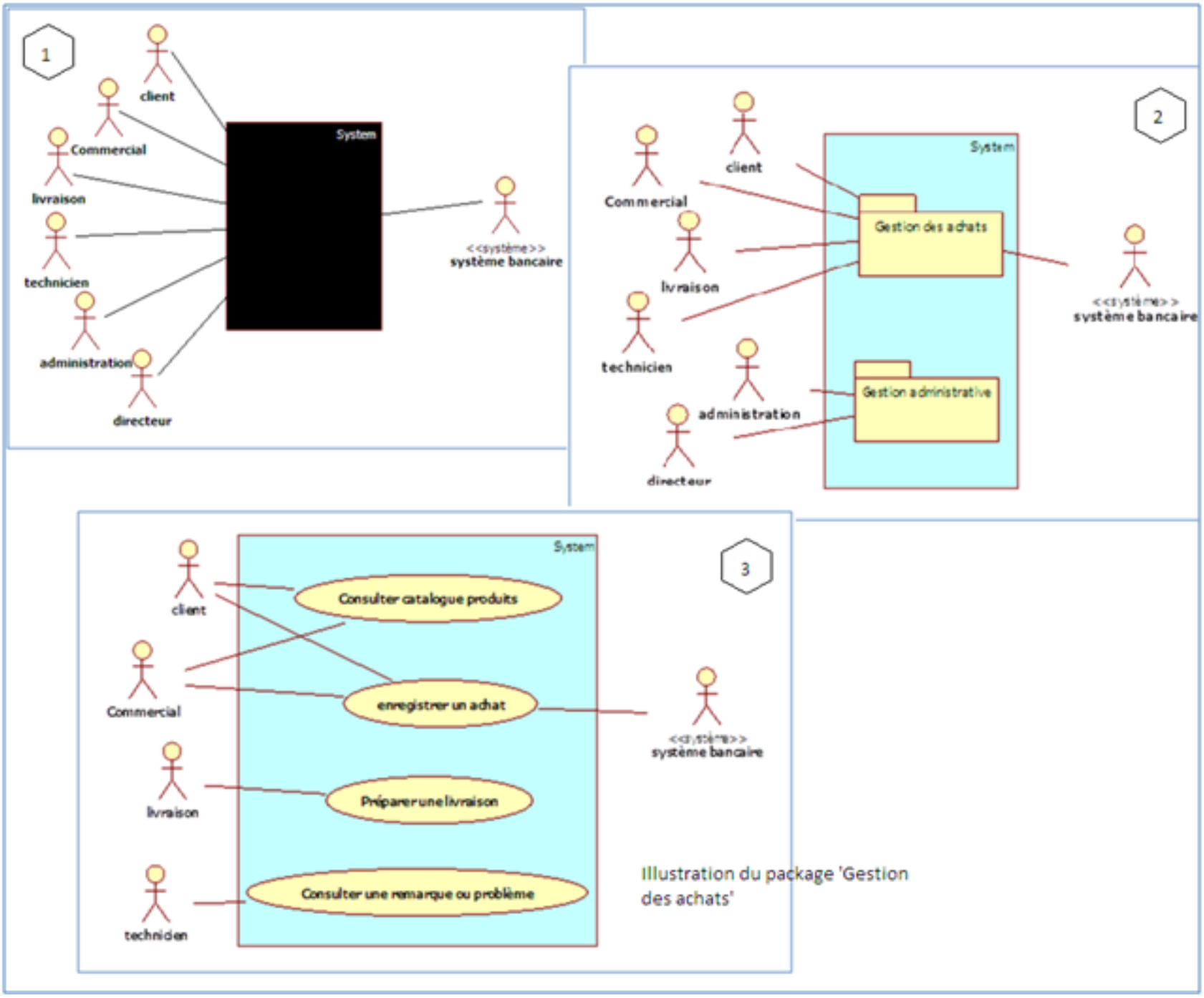


Illustration du package 'Gestion des achats'

Synthèse des 3 diagrammes réalisés

Dans la partie suivante, nous allons détailler davantage les diagrammes de cas d'utilisation en introduisant des **cas d'utilisation internes**. Cela permettra d'indiquer si un cas d'utilisation a besoin d'autre cas d'utilisation.

En résumé

- Un logiciel à développer est donc considéré comme un système, vu au départ comme une boîte noire.
- Le système est utilisé par des acteurs principaux, et parfois, il peut être lié à des acteurs secondaires. Les acteurs secondaires échangent des informations avec le système, mais ne déclenchent aucune des fonctionnalités.

- Un système peut être composé de plusieurs packages. Chacun des packages contient une famille de fonctionnalités nécessaires aux acteurs.

- Les fonctionnalités utiles aux acteurs sont appelées des cas d'utilisation. Un diagramme de cas d'utilisation permet d'illustrer QUI devrait pouvoir faire QUOI, grâce au système. Il en existe un par package.

Cette partie est maintenant terminée. N'oubliez pas de faire vos exercices avant de passer à la partie suivante. Vous trouverez les liens des exercices (quiz et/ou activité) dans le plan principal du cours ICI. À vous de jouer!

Les cas d’utilisation internes

Dans ce chapitre, nous allons découvrir ce qu’on appelle des cas d’utilisation interne. Il s’agit ici de détailler les cas d’utilisation principaux afin de comprendre tous les lots d’actions qu’ils impliquent.

Définition

Les cas d’utilisation que nous avons découvert dans la partie 2 sont directement liés à un acteur et sont appelés des « **cas d’utilisation principales** ». On y décrit ce qu’un utilisateur doit pouvoir faire grâce au logiciel à développer. On parle donc les **fonctionnalités principales** du logiciel à développer.

Chacun de ces cas d’utilisation nécessitera un certain nombre d’actions. Il arrive que l’on doive regrouper certaines actions dans un ou plusieurs cas d’utilisation complémentaires qui ne sont pas directement liés à un acteur. On parle alors d’un **cas d’utilisation interne**.

Il s’agit essentiellement :

- de cas d’utilisation interne qui peuvent être nécessaires à plusieurs autres cas d’utilisation (qu’ils soient des cas d’utilisation principaux, ou même d’autres cas d’utilisation interne) ;

- de cas d’utilisation qui regroupent un certain nombre d’actions qui forment un tout homogène et pouvant être décrit séparément.

Pour trouver les cas d'utilisation internes, nous devrons réfléchir à chaque cas d'utilisation que nous avons déjà indiqué dans notre diagramme.

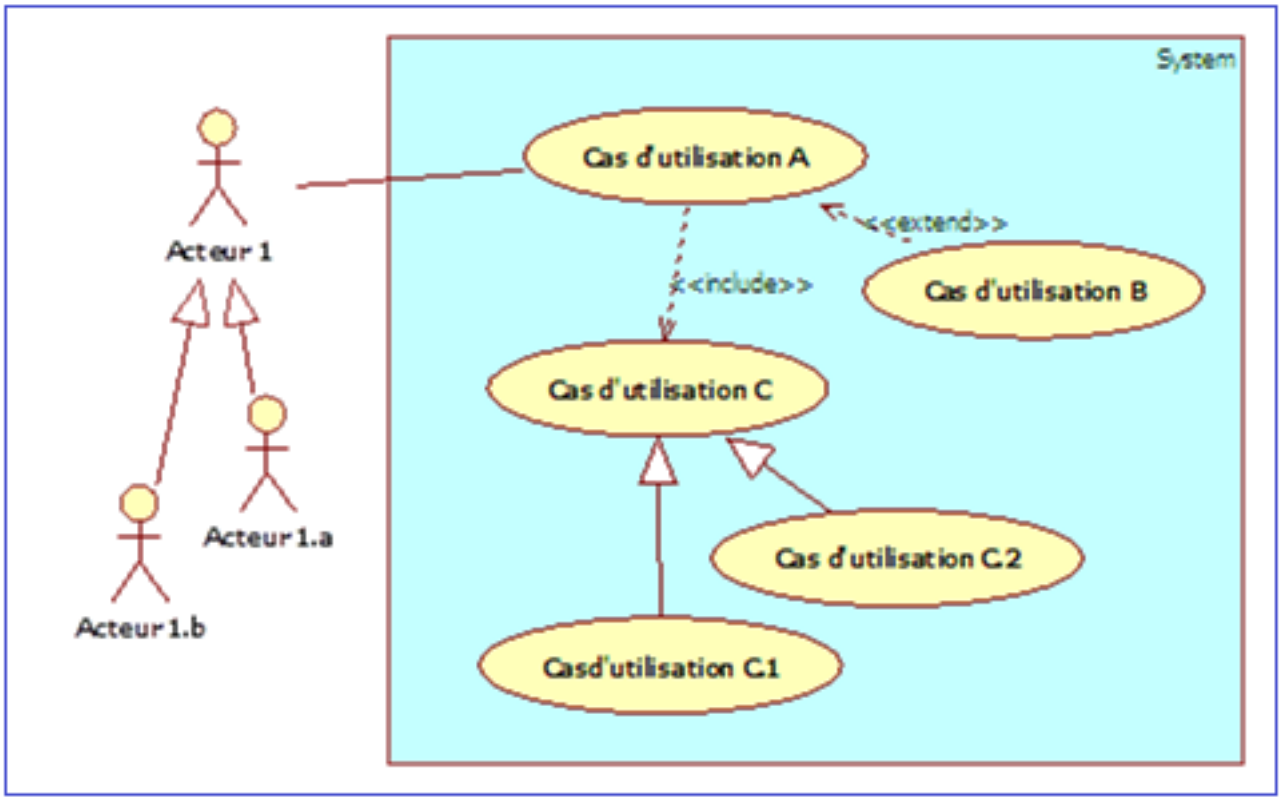


Diagramme illustrant des cas d'utilisation internes

Les cas d'utilisation internes seront indiqués dans le diagramme de cas d'utilisation. Leur particularité réside dans le fait qu'ils ne seront pas directement liés aux acteurs, mais aux cas d'utilisation qui en ont besoin.

Les liens (ou relations) entre cas d'utilisation peuvent être divisés en deux groupes :

- La spécialisation
- Les relations stéréotypées

Cela vous semble un peu nébuleux ?

Revoyons tout cela à partir de notre étude de cas : créer une boutique en ligne. Dans la partie précédente, nous nous étions arrêté au diagramme de cas d'utilisation du package « Gestion des achats » qui mettait en évidence les acteurs et les cas d'utilisation principaux. Cela vous rappelle quelque chose ? (Voir la figure suivante).

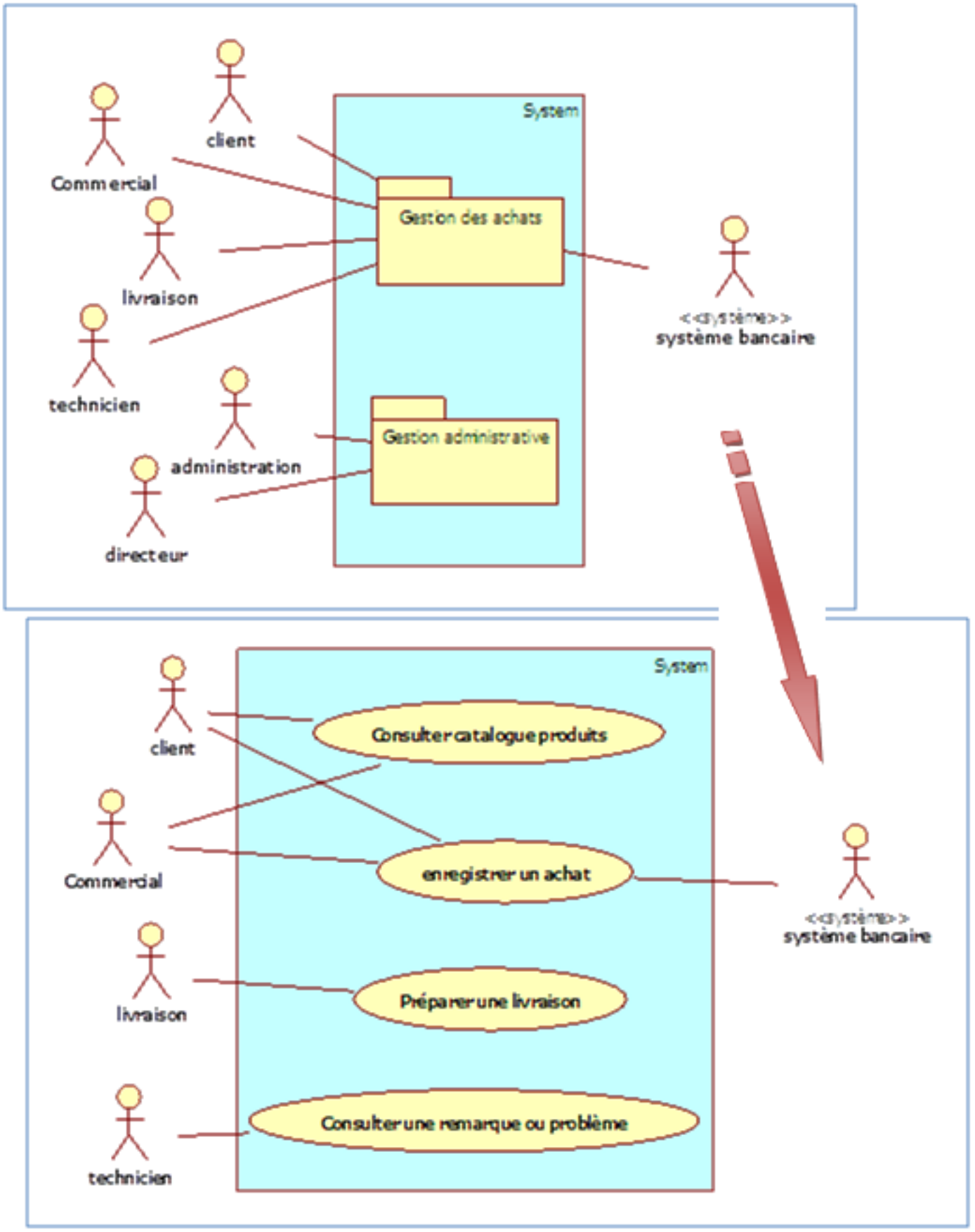


Diagramme de cas d'utilisation du package « Gestion des achats » v4

La spécialisation des acteurs

Un premier complément à notre diagramme de cas d'utilisation pourra être mis en œuvre à l'aide d'une notion appelée « **la spécialisation** ».

La spécialisation peut être indiquée à deux niveaux :

- au niveau des acteurs ;
- au niveau des cas d'utilisation principaux.

Regardons le diagramme de package « Gestion des achats» dans le détail. On constate très vite qu'au niveau des acteurs « Client » et « Commercial », les cas d'utilisation « Consulter catalogue produits » et « Enregistrer un achat » s'entrecroisent.

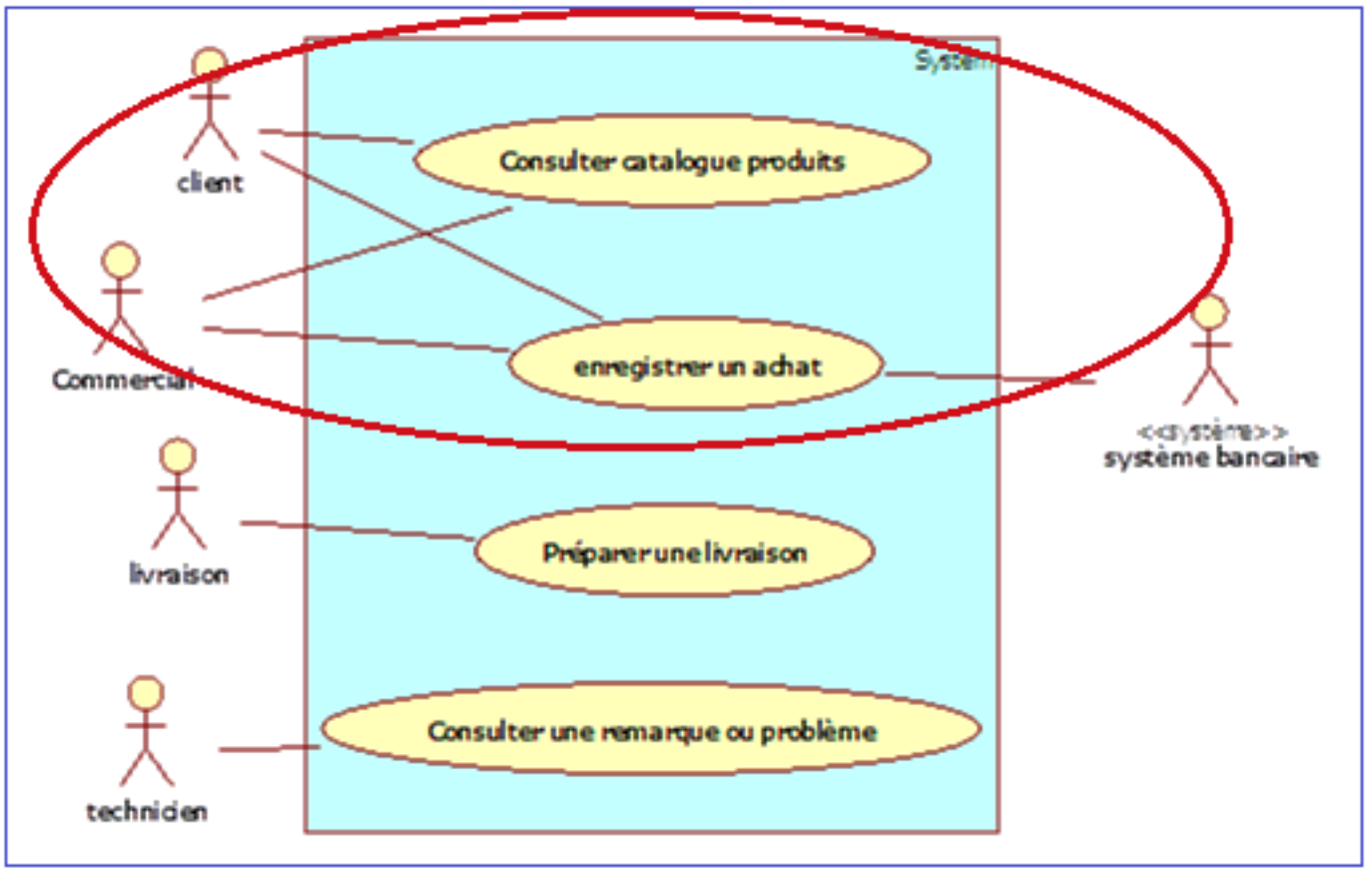


Diagramme de cas d'utilisation du package «Gestion des achats» v4 (détail)

Le diagramme reste lisible puisqu'il n'y a pas trop de liens qui se croisent mais prenons tout de même l'habitude d'éviter les liens qui s'entrecoupent, car cela devient vite affreux lorsqu'il y a beaucoup de cas d'utilisation. Et comme pour plein d'autres choses, une fois que nous avons pris de bonnes habitudes, nous ne les oublierons pas.

Alors, comment remédier à ces liens qui s'entrecoupent?

Pour cela, nous introduirons donc la notion de spécialisation au niveau de certains 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.

On utilise alors :

- un acteur générique qui est lié aux cas d'utilisations communs ;
- des acteurs spécialisés qui sont liés à des cas d'utilisation spécifiques.

On indique qu'un acteur est la spécialisation d'un autre en dessinant une flèche à pointe fermée vers l'acteur principal.

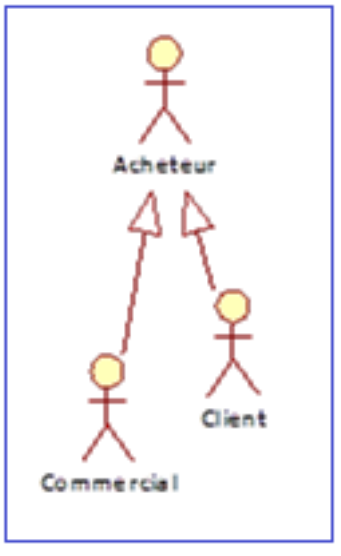


Diagramme illustrant la spécialisation d'un acteur

Dans notre étude de cas, on constate que deux acteurs ont les mêmes cas d'utilisation. Pour indiquer cela, nous créons un nouvel acteur générique appelé « Acheteur », dont « Client » et « Commercial » sont des spécialisations. Nous pouvons alors relier l'acteur générique aux cas d'utilisation « Consulter catalogue produits » et « Enregistrer un achat » (voir la figure suivante).

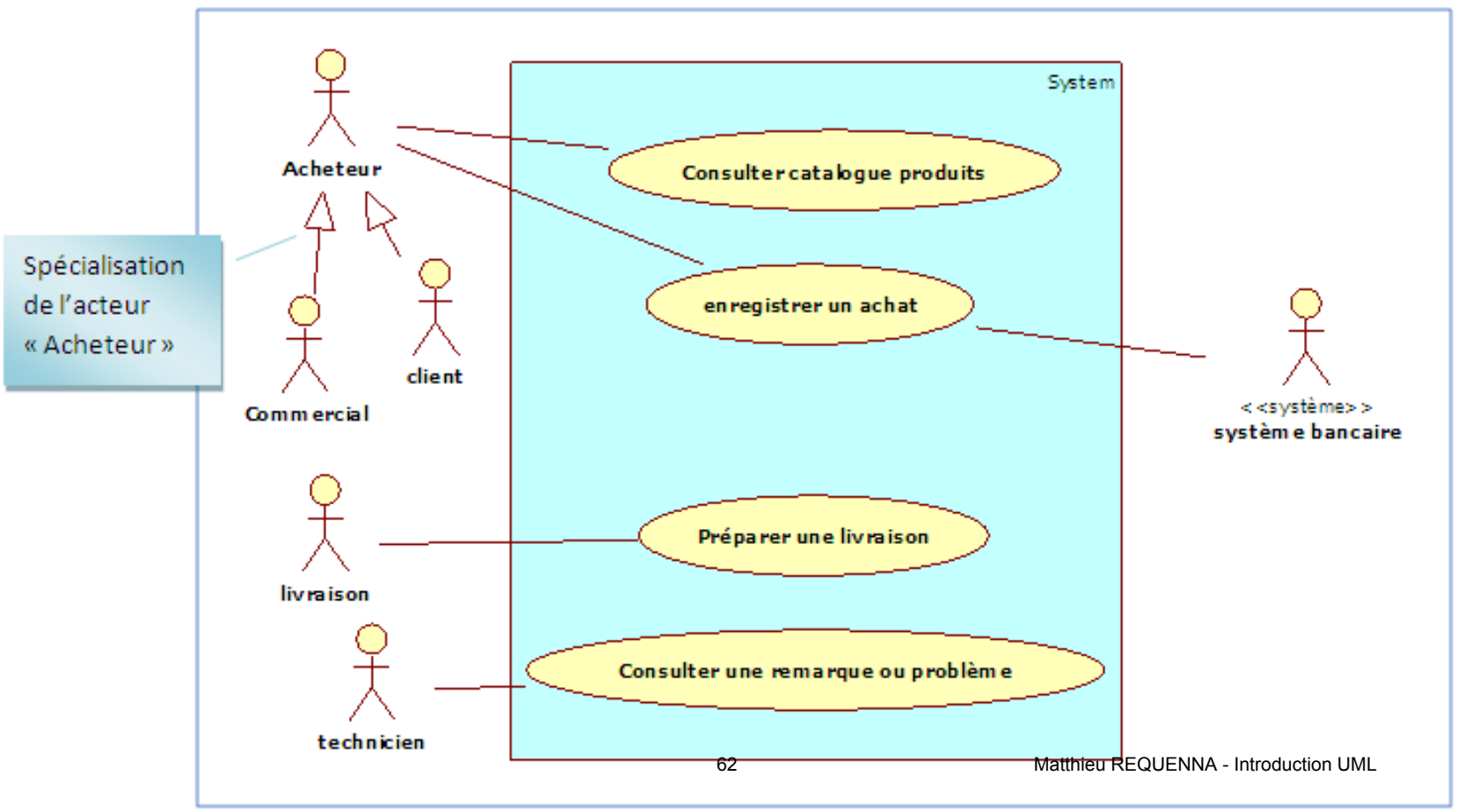


Diagramme de cas d'utilisation, package « Gestion des achats » v5

Même si les acteurs spécialisés ne sont pas liés à des cas d'utilisation spécifiques, nous pouvons les laisser dans le diagramme. Cela explique quels types d'acteurs sont regroupés sous le nom « Acheteur ». De plus, il se pourrait que nous découvrions, plus tard, de nouveaux cas d'utilisation qui seraient utiles à l'un ou à l'autre des utilisateurs spécialisés.

Je vous ai dit que la spécialisation est également possible au niveau des cas d'utilisation, n'est-ce pas ?
Il va falloir patienter un peu, car notre diagramme ne présente pas encore d'éléments nécessitant cela. Je vous promets que vous verrez la notion de spécialisation de cas d'utilisation avant la fin de ce chapitre. Parole de scout !

D'autres précisions

Regardons de nouveau le diagramme de package « Gestion des achats» dans le détail. Intéressons-nous cette fois-ci au cas d'utilisation « Enregistrer un achat ». L'objectif ici est de détailler les actions au sein de cette fonctionnalité.

Quand on y réfléchit plus en détail, on réalise qu'un acheteur qui enregistre un achat doit forcément constituer un panier (comme cela est souvent le cas lors d'un achat en ligne). La constitution du panier nécessitera plusieurs séquences d'actions :

- probablement une recherche de produits selon une catégorie ou un nom ;
- la consultation de la fiche produit ;
- la validation d'un produit à ajouter au panier ;
- etc.

Essayons de voir tout ce que l'on doit faire pour « Enregistrer un achat», que l'on soit client ou commercial.

Actions ou lots d'actions pour le cas d'utilisation « Enregistrer un achat » :

Le client doit…	**Le commercial doit…**
1. S'authentifier	1. S'authentifier
2. Constituer un panier	2. Sélectionner le client pour lequel on réalise l'achat
3. Saisir les informations de livraison	3. Constituer un panier
4. Enregistrer le règlement de l'achat	4. Saisir les informations de livraison
	5. Enregistrer le règlement de l'achat

Vous voyez ? L'enregistrement d'un achat est bien plus complexe que ce qu'on aurait pu penser au début. Ces lots d'actions sont donc des précisions du cas d'utilisation « Enregistrer un achat ». Ces précisions nécessiteront, elles-mêmes, plusieurs actions. On peut en faire des **cas d'utilisation internes**.

D'accord ! Mais comment est-ce qu'on illustre cela dans le diagramme de cas d'utilisation alors ?

C'est très simple. Tout d'abord, nous allons introduire un nouveau cas d'utilisation, appelé « Constituer panier ». Ce cas d'utilisation sera toujours nécessaire pour la fonctionnalité « Enregistrer un achat ». Le nouveau cas d'utilisation interne sera donc lié au cas d'utilisation principal « Enregistrer un achat » par une flèche représentant la relation dite « Include ». Regardez la figure suivante.

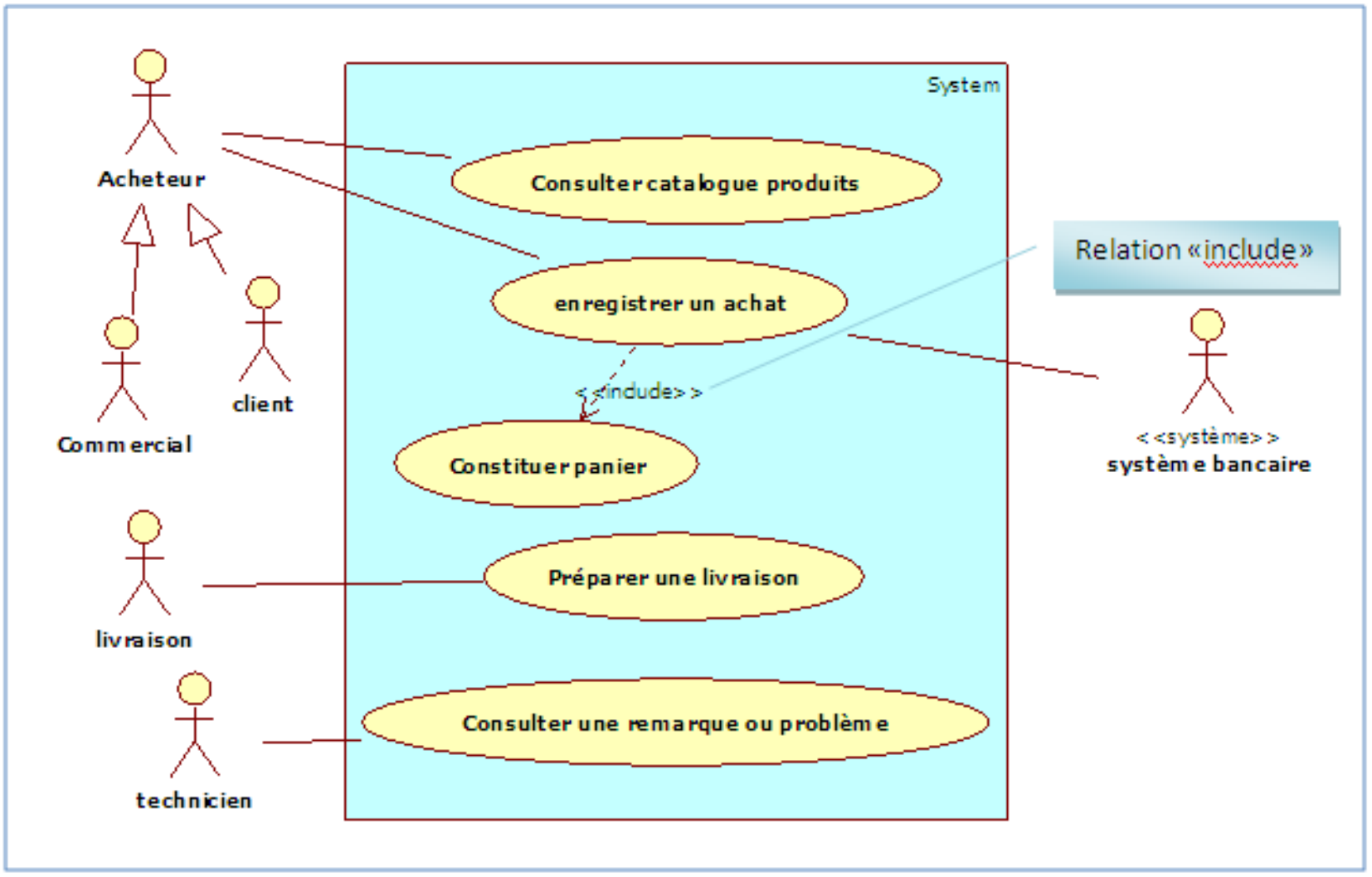


Diagramme de cas d'utilisation, package « Gestion des achats » V6

Une relation dite « include » ? Qu'est-ce que c'est ?

Voyons cela dans le chapitre suivant !

Les relations stéréotypées

Les cas d'utilisation internes sont toujours reliés, par une flèche, au cas d'utilisation initial qui en besoin. Le cas d'utilisation initial peut être un cas d'utilisation principal (donc directement relié à un acteur) ou à un autre cas d'utilisation interne. Ces relations sont appelées des relations stéréotypées car ils définissent le type de lien entre les cas d'utilisation, qui peuvent être de deux sortes :

- les relations « include » ;
- les relations « extend ».

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. Dans la figure précédente, le cas d'utilisation source « Enregistrer un achat » contiendra TOUJOURS le cas d'utilisation « Constituer un panier ». Pour représenter cette relation, on dessine donc une flèche en pointillé partant du cas d'utilisation principal vers le **cas d'utilisation interne**. Puis, on indique le **stéréotype** « include » sur la flèche (voir la figure précédente). Dans une relation « include », le cas d'utilisation source est donc celui qui a besoin d'un autre cas d'utilisation dit interne, indiqué par une flèche.

Dans un diagramme de cas d'utilisation, le **stéréotype** est le mot indiqué écrit entre guillemets sur la flèche, ici « include » par exemple. Il apporte une précision sur la relation qui relie les deux cas d'utilisation.

Cas d'utilisation 1 -> inclut -> cas d'utilisation 2 (interne)

La figure suivante illustre tous les cas d'utilisation internes que nous avons identifiés pour le cas d'utilisation « Enregistrer un achat ». Observez bien le diagramme. Que remarquez-vous ?

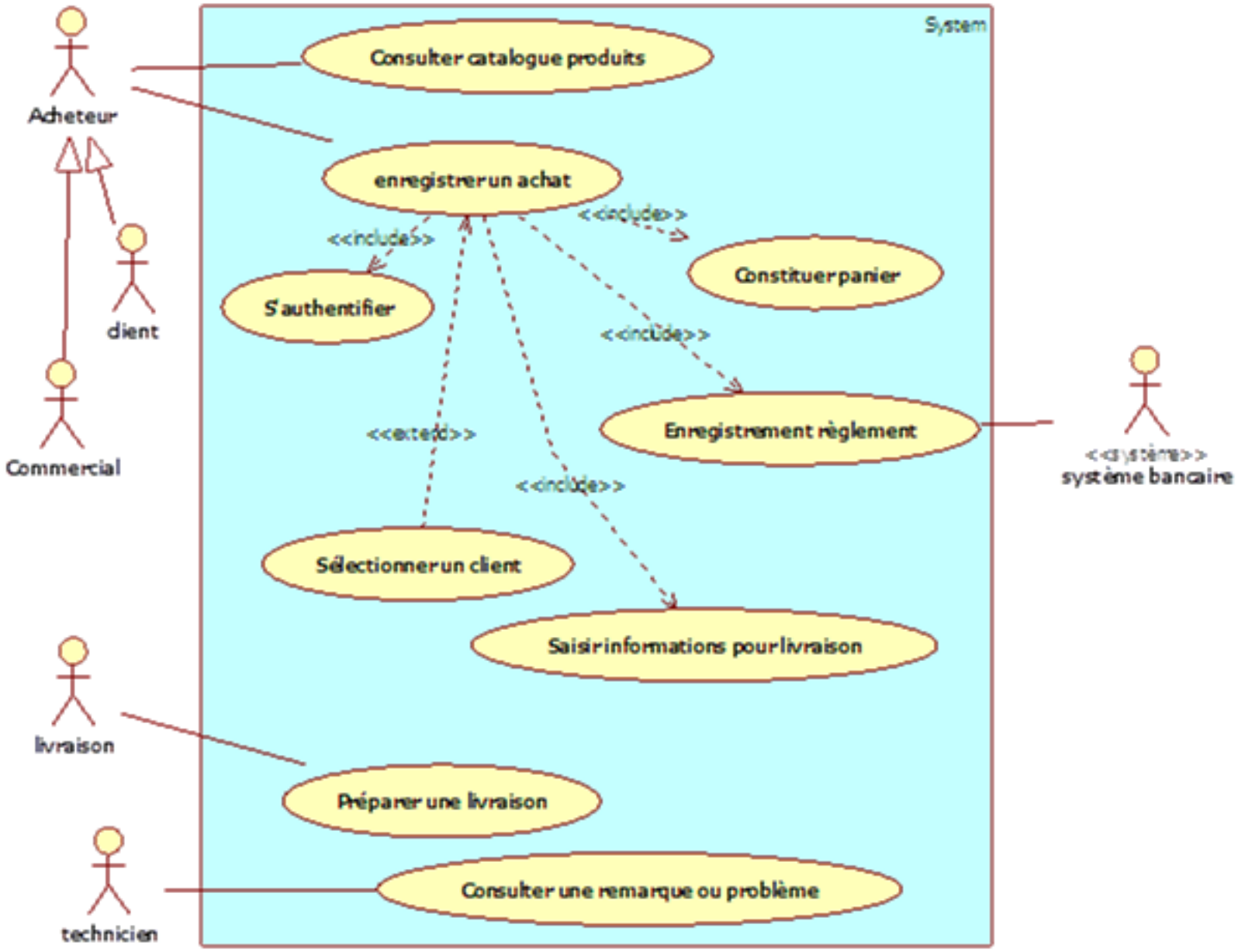


Diagramme de cas d'utilisation, package « Gestion des achats »

Dans la version précédente de notre diagramme, l'acteur secondaire « Système bancaire » était lié au cas d'utilisation « Enregistrer un achat ». Maintenant que ce cas d'utilisation a été complété par des cas d'utilisation internes, nous pouvons être plus précis. Nous avons donc déplacé ce lien vers le cas d'utilisation direct : « Enregistrement règlement ».

Je vois une nouvelle relation, dite « Extend » entre le cas d'utilisation « Enregistrer un achat » et « Sélectionner un client ». Pourquoi ce changement ? C'est quoi une relation « Extend » ?

Voilà, ce sera l'objet de notre section suivante.

La relation « extend »

Une **relation « extend »** est la deuxième forme de relation stéréotypée. Cette relation est utilisée pour indiquer que le cas d'utilisation source (à

l'origine de la flèche) **n'est pas toujours nécessaire** au cas d'utilisation principal, mais qu'il peut l'être dans certaines situations. On doit alors préciser la condition qui fera que le cas d'utilisation en « extend » est nécessaire.

Dans notre cas, le cas d'utilisation interne « Sélectionner le client » n'étant pas une action obligatoire au cas d'utilisation principal « Enregistrer un achat », on peut dès lors parler de relation « extend ». En effet, un client qui réalise un achat sera déjà identifié ; le commercial, par contre, devra sélectionner le client pour lequel il réalise l'achat.

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 ? Pour indiquer cela, il faut ajouter une ligne « extension points » et définir la condition « EXT ».

Regardez la figure suivante : ici, le cas d'utilisation « Sélectionner un client » est une relation « extend » du cas d'utilisation principal « Enregistrer un achat ». Cette action n'est possible qu'à condition que l'acteur soit le « Commercial », et non pas le « Client ».

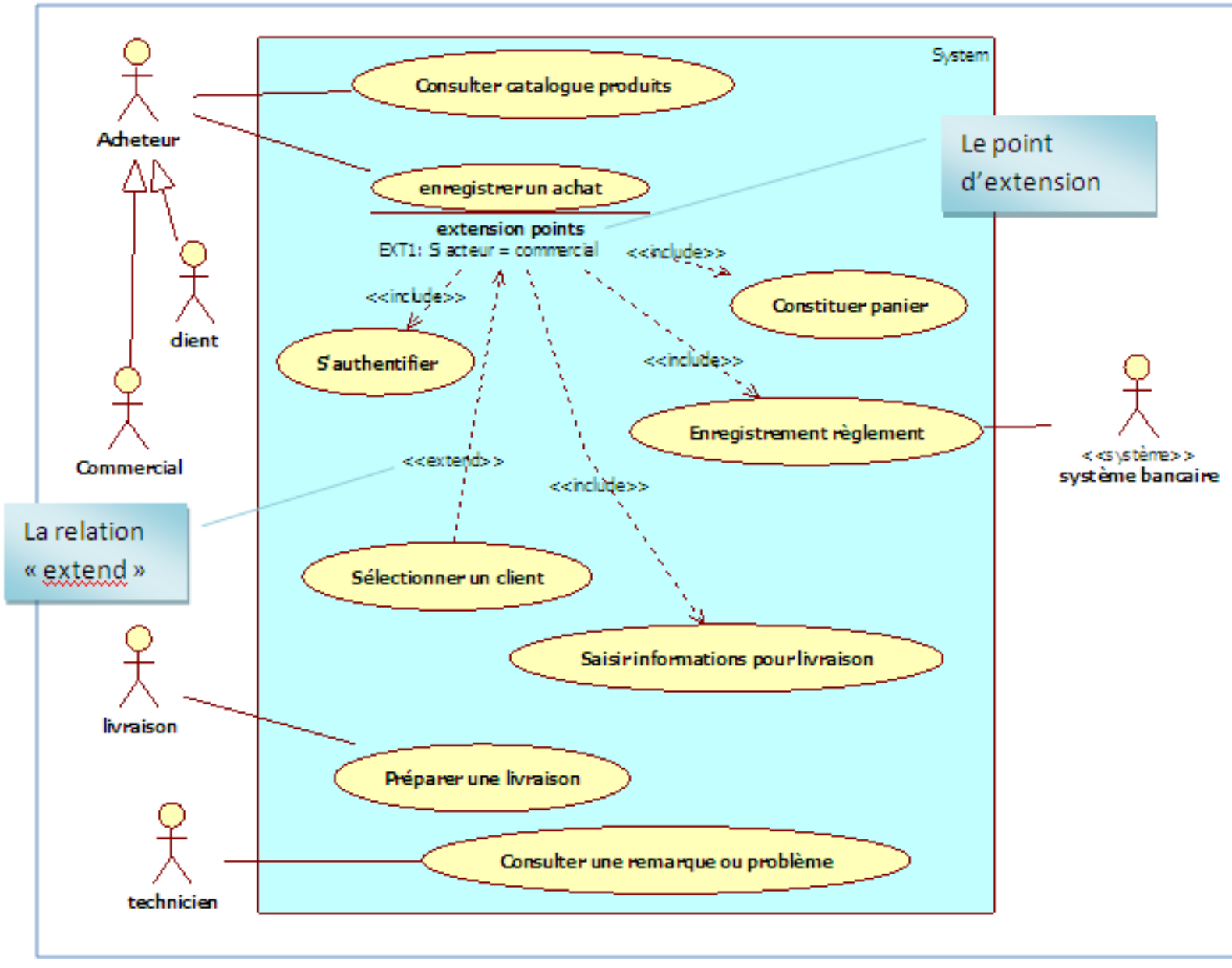


Diagramme de cas d'utilisation, package « Gestion des achats » v8

Voilà, vous savez ce qu'est un cas d'utilisation interne et, ses relations stéréotypées « include » et « extend ».

Le cas d'utilisation « S'authentifier » peut devenir un package à lui seul. Pourquoi ? Et bien parce qu'il sera nécessaire pour tous les cas d'utilisation principaux. Il sera bien nécessaire de vérifier que l'acteur qui souhaite utiliser une fonctionnalité de notre logiciel est bien connu et habilité à utiliser la fonctionnalité en question. Par exemple : seul un utilisateur de type « Livraison » devra avoir accès à la fonctionnalité « Préparer livraison ». Et le package « Gestion administrative » aura également besoin de ce cas d'utilisation. On peut donc dans ce cas créer un package spécifique à « Authentifier ».

Notre diagramme de packages pourra mettre en évidence que les deux packages « Gestion des achats » et « Gestion administrative » utiliseront le package « Authentification » :

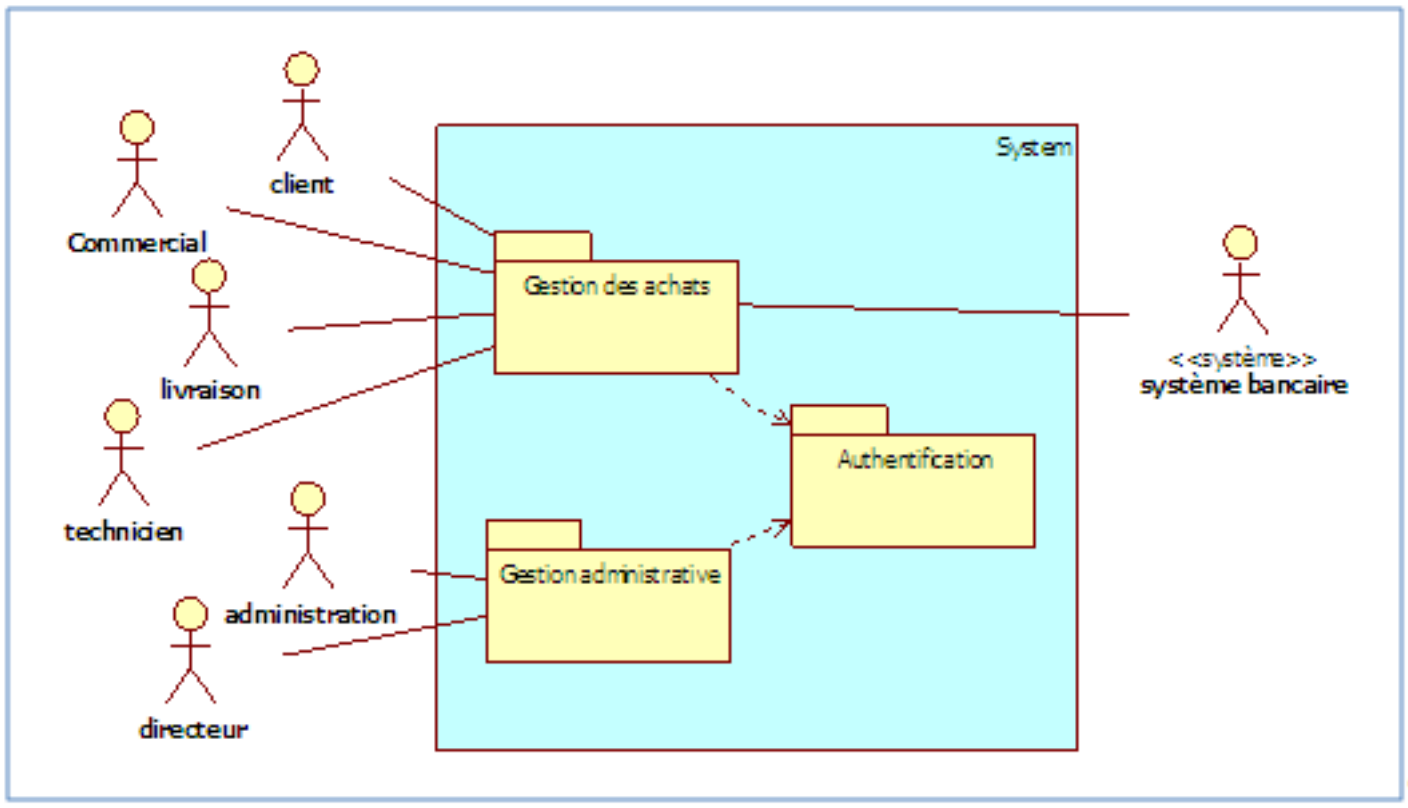


Diagramme de packages modifié

Du coup, le diagramme de cas d'utilisation du package « Gestion des achats » s'en trouve allégé (voir la figure suivante).

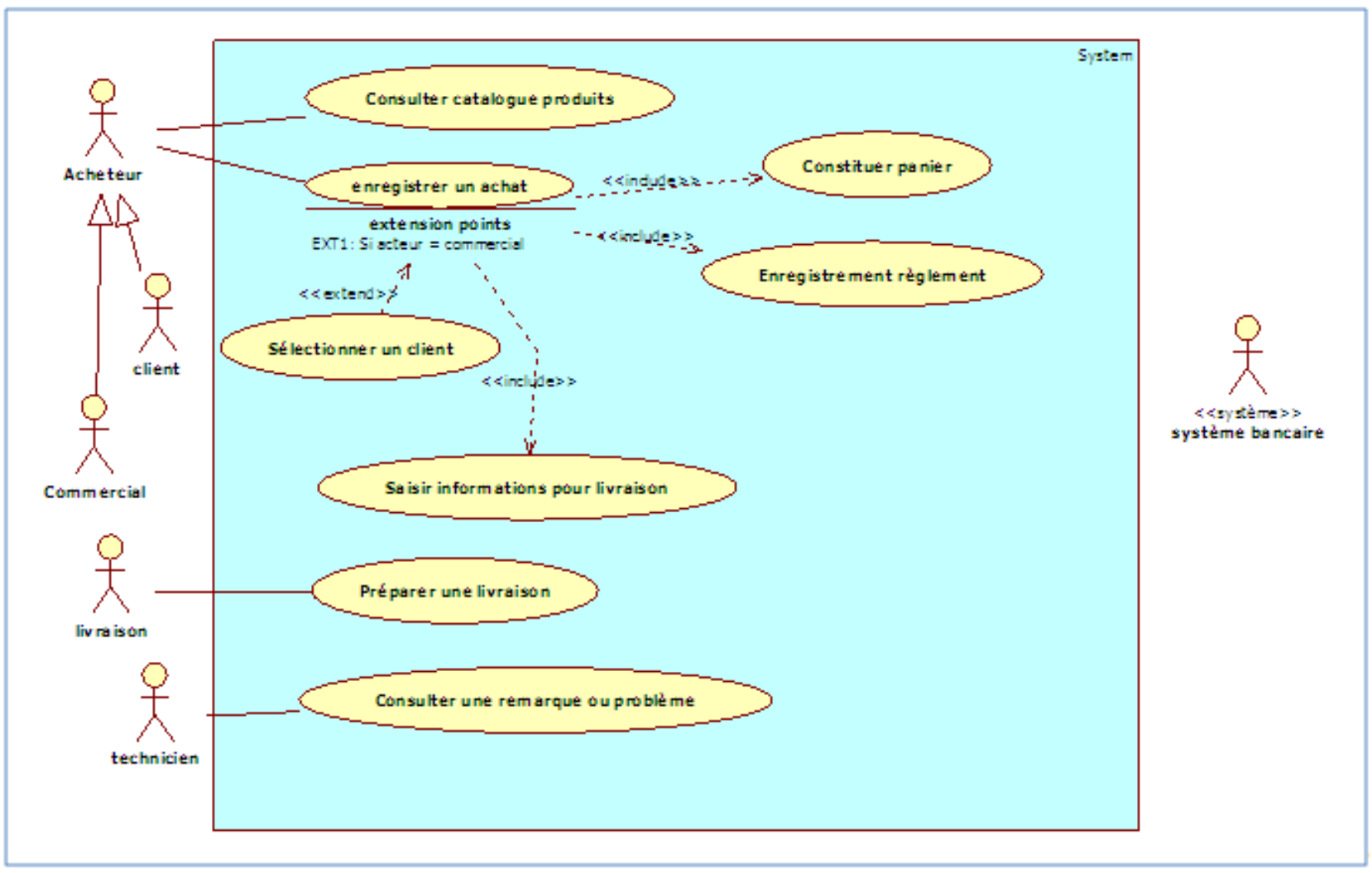


Diagramme de cas d'utilisation, package « Gestion des achats »

Et cette fameuse spécialisation des cas d'utilisation ?

Et oui, je vous avais promis une autre spécialisation, mais cette fois-ci non

pas au niveau des acteurs, mais des cas d'utilisation.

Souvenez-vous de notre description des lots d'action du cas d'utilisation « Enregistrer un achat ».

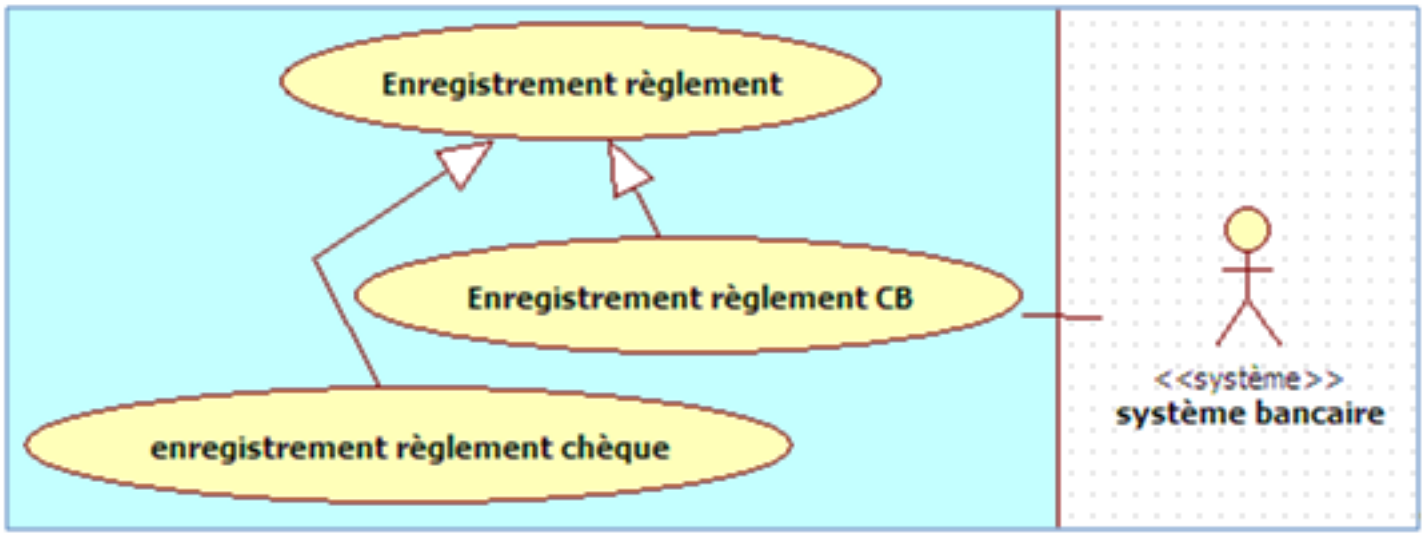
Actions ou lots d'actions pour le cas d'utilisation « Enregistrer un achat »:

Le client doit…	**Le commercial doit…**
1. S'authentifier	1. S'authentifier
- soit en se connectant	*- en se connectant*
- soit en s'inscrivant comme nouveau client	2. Sélectionner le client pour lequel on réalise l'achat
2. Constituer un panier	3. Constituer un panier
- sélectionner le produit	*- sélectionner le produit*
- valider le panier	*- valider le panier*
3. Saisir les informations de livraison	4. Saisir les informations de livraison
4. Enregistrer le règlement de l'achat	5. Enregistrer le règlement de l'achat
- par carte bancaire	*- soit par carte bancaire*
	- soit par chèque

Jusque-là, nous nous sommes intéressés aux lots d'action principaux numérotés. Il nous faut maintenant réfléchir aux éléments qui sont indiqués en italique. Regardons plus particulièrement le cas d'utilisation « Enregistrer un règlement ».

Prêt ? C'est parti !

Voici donc de détail du cas d'utilisation « Enregistrement règlement ».



Le complément pour le cas d'utilisation enregistrement règlement

Je vous explique. 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 ».

Ah, je vous avais bien dit qu'on aurait besoin de la notion de spécialisation entre cas d'utilisation !

Pourquoi s'agit-il d'une spécialisation et non d'une relation stéréotypée ?

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.

En effet, l'enregistrement d'un règlement par chèque n'est possible que lorsqu'un commercial enregistre l'achat. Notre site web commercial ne serait pas très sûr si on acceptait qu'un client enregistre un règlement par chèque lui-même. Il pourrait ne pas nous faire parvenir le chèque ! Nous acceptons donc que seul un commercial puisse enregistrer ce type de

règlement, après avoir reçu le chèque.

Par contre, l'enregistrement d'un règlement par carte bancaire est possible autant pour un client que pour un commercial. Le client pourra faire cela en ligne, mais le commercial peut également le faire pour le client lors d'une vente sur site.

Enfin, autre remarque : l'acteur « Système bancaire » n'est utile que dans un des deux cas. Cet acteur n'interviendra que pour l'action « Enregistrement règlement CB » car seul pour les CB, cet acteur secondaire vérifiera l'exactitude des informations fournies.

Est-ce que vous voyez pourquoi il s'agit bien de cas d'utilisation et non d'actions uniques ?

Chacun des cas d'utilisation spécialisés nécessite plusieurs actions. Le cas d'utilisation « Enregistrement règlement CB » comprendra très probablement :

- le choix du type de carte (visa, Mastercard, Maestro…) ;
- la saisie du numéro de la carte ;
- la saisie de la date de validité ;
- la saisie du cryptogramme ;
- l'envoi des informations au système bancaire ;
- la réception de la réponse du système bancaire et le traitement du règlement.

Le cas d'utilisation « Enregistrement règlement chèque» pourrait contenir aussi les actions suivantes :

- la saisie de la banque et du guichet ;
- la saisie du numéro de chèque ;

- la saisie de la date de réception du chèque.

Il y a donc une grande différence entre une action unique (tel qu'un clic sur un bouton « Valider » ou un bouton « Annuler ») et un lot d'actions qui composent un cas d'utilisation.

Voici, sur la figure suivante, le diagramme de cas d'utilisation complété.

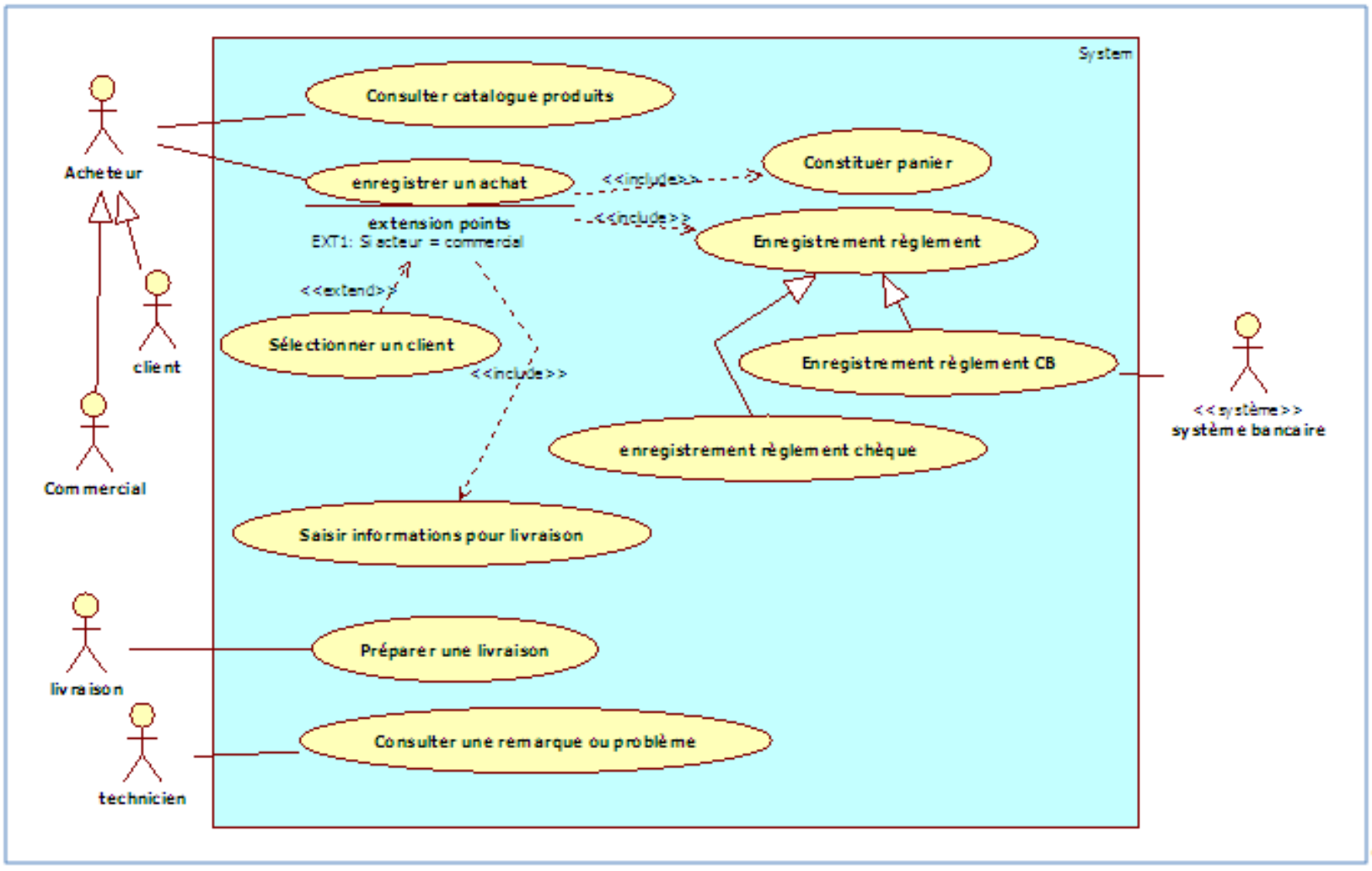


Diagramme de cas d'utilisation, package « Gestion des achats »

Vous avez vu comment notre diagramme a évolué ? En réfléchissant de façon méthodique à ce qui doit être possible de faire, nous avons découvert une multitude de cas d'utilisation. Encore heureux que nous avions décomposé notre système en packages. Sinon le diagramme serait devenu illisible.

Pour information, nous pourrions encore aller plus loin dans le détail de ce diagramme, en explicitant par exemple les cas d'utilisation « S'authentifier », « Constituer un panier » ou encore « Saisir les informations pour livraison ». Mais pour ce premier cours, je vais vous épargner cela ! 😉

Voilà, nous venons de voir comment réaliser des diagrammes pour illustrer

le besoin de nos utilisateurs. L'un de nos diagrammes contient des cas d'utilisation internes.

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**.

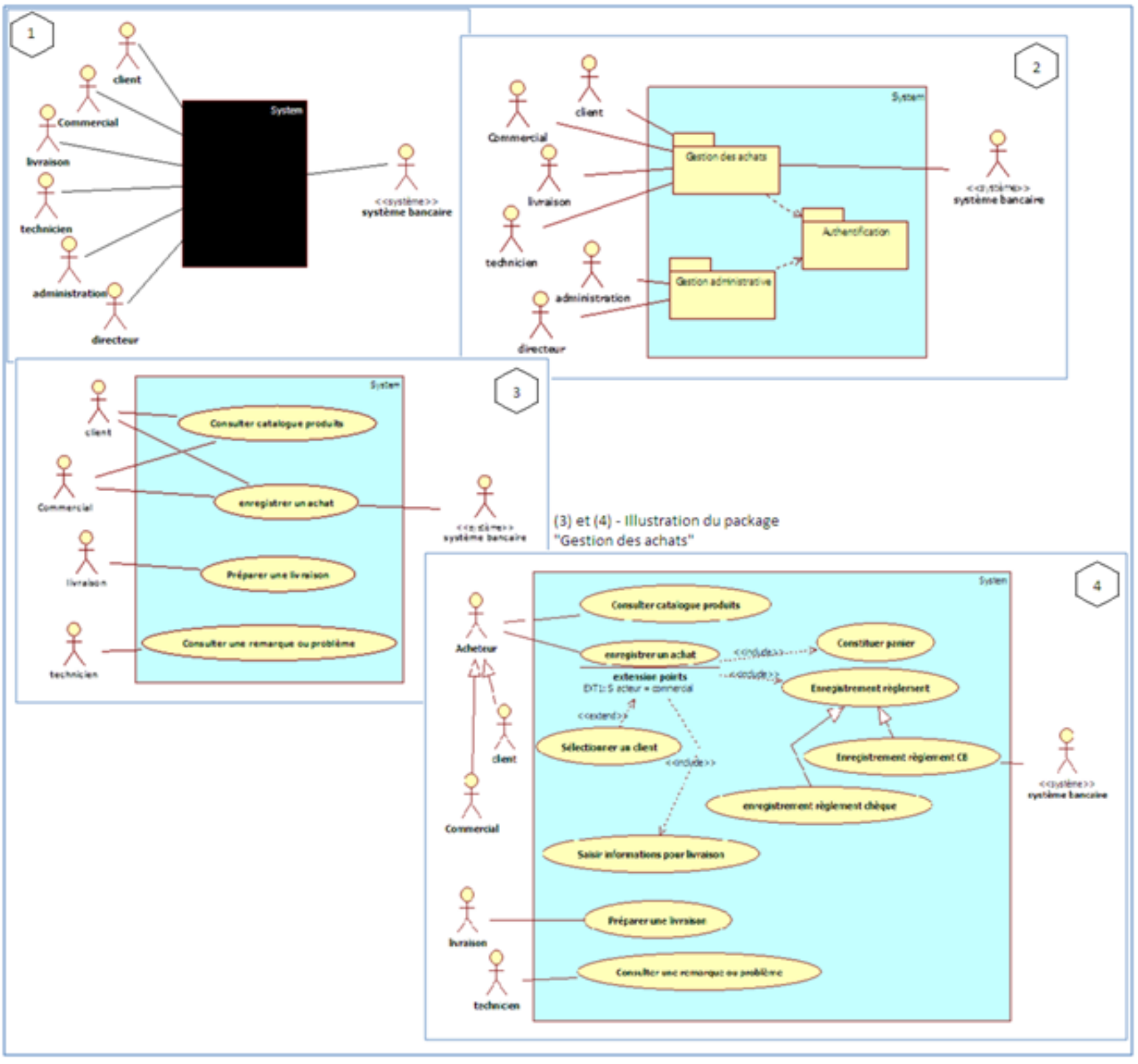
2. En étudiant les différents besoins de nos acteurs sur le logiciel, nous avons pu repérer 3 grandes familles de fonction : Gestion des achats, Gestion administrative et Authentification. Elles sont illustrées à l'aide d'un **diagramme de package**.

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)

En voici la synthèse visuelle.



Légende :

(1) Le diagramme de contexte

(2) Diagramme de package

(3) Diagramme de cas d'utilisation initial

(4) Diagramme de cas d'utilisation détaillé (avec include/ extend/ spécialisation)

Comme vous pouvez le constater, nos diagrammes s'imbriquent les une aux autres, telles des poupées russes. Au fil de l'analyse de besoins, nous avons décrit et détaillé les différents acteurs, leur rôle, leurs grandes fonctionnalités, les lots d'actions ainsi que les relations entre elles.

En résumé

- Les cas d'utilisation principaux sont complétés par des cas d'utilisation internes. Les cas d'utilisation internes sont des précisions d'autres cas d'utilisation.
- Les cas d'utilisation internes sont reliés au cas initial par des relations stéréotypées de type « include » ou « extend », ou par une relation de spécialisation.
- Les spécialisations peuvent se faire au niveau des acteurs et/ou au niveau des cas d'utilisation.
- Une relation « include » permet d'indiquer qu'un cas d'utilisation a toujours besoin du cas d'utilisation lié.
- Une relation « extend » c'est une relation qui est soumise à une condition. Le cas d'utilisation initial n'utilisera les actions du cas d'utilisation lié que dans certaines circonstances. On doit toujours expliciter la condition qui doit être rencontrée pour que le cas d'utilisation lié soit utile.

Cette partie est maintenant terminée. N'oubliez pas de faire vos exercices avant de passer à la partie suivante. Vous trouverez les liens des exercices (quiz et/ou activité) dans le plan principal du cours ICI. À vous de jouer!

La description textuelle d'un cas d'utilisation

Dans ce chapitre, nous allons découvrir l'utilité de décrire les scénarios des cas d'utilisation, ainsi que les éléments nécessaires à la description du déroulement des d'actions.

Définition

Les diagrammes réalisés jusqu'à maintenant (diagramme de contexte, diagramme de packages, diagramme de cas d'utilisation) nous ont permis de découvrir petit à petit les fonctionnalités (appelées aussi des cas d'utilisation) que l'on devrait avoir dans le futur logiciel.

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.

Les descriptions peuvent aider à découvrir d’autres cas d’utilisation que l’on pourrait ajouter. Il s’agit, dans ce cas, d’une nouvelle itération sur les diagrammes de cas d’utilisation.

Décrire le déroulement des actions pour un cas d’utilisation peut vous paraître simple, mais c’est un travail qui demande beaucoup de réflexion et de questionnement. On commence souvent par une première description, basée sur les informations que l’on a obtenu auprès du client et/ou des futurs utilisateurs. Lors de la réalisation de cette première description, on découvre souvent des questions auxquels nous n’avons pas de réponse.

Par exemple :
Si une personne n’est pas encore « cliente » sur la boutique en ligne et qu’elle souhaite réaliser un achat :

- Doit elle s’incrire avant de commencer la procédure d’achat ?

- Peut-elle s’inscrire pendant la procédure d’achat, par exemple juste avant de régler ?

Nous pouvons, bien sûr, avoir une idée sur la question. On se base d’ailleurs souvent sur des situations déjà vues ou vécues pour faire des propositions au client et aux utilisateurs.

Oui, j’ai bien dit : **faire des propositions**. Il ne s’agit pas de décrire ce que nous pensons être mieux, mais plutôt ce que le client et les utilisateurs souhaitent. Nos idées ne sont donc que des possibilités qu’il faut soumettre aux personnes concernées afin d’engager une communication, dans le but de découvrir les réels besoins.

On réalise alors une **fiche descriptive** pour chaque cas d’utilisation, de

façon à raconter **l'histoire** du déroulement des différentes actions.

Cette fiche descriptive doit comporter 4 volets :

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

Le schéma suivant illustre la relation entre le diagramme des cas d'utilisation et la description textuelle d'un cas d'utilisation.

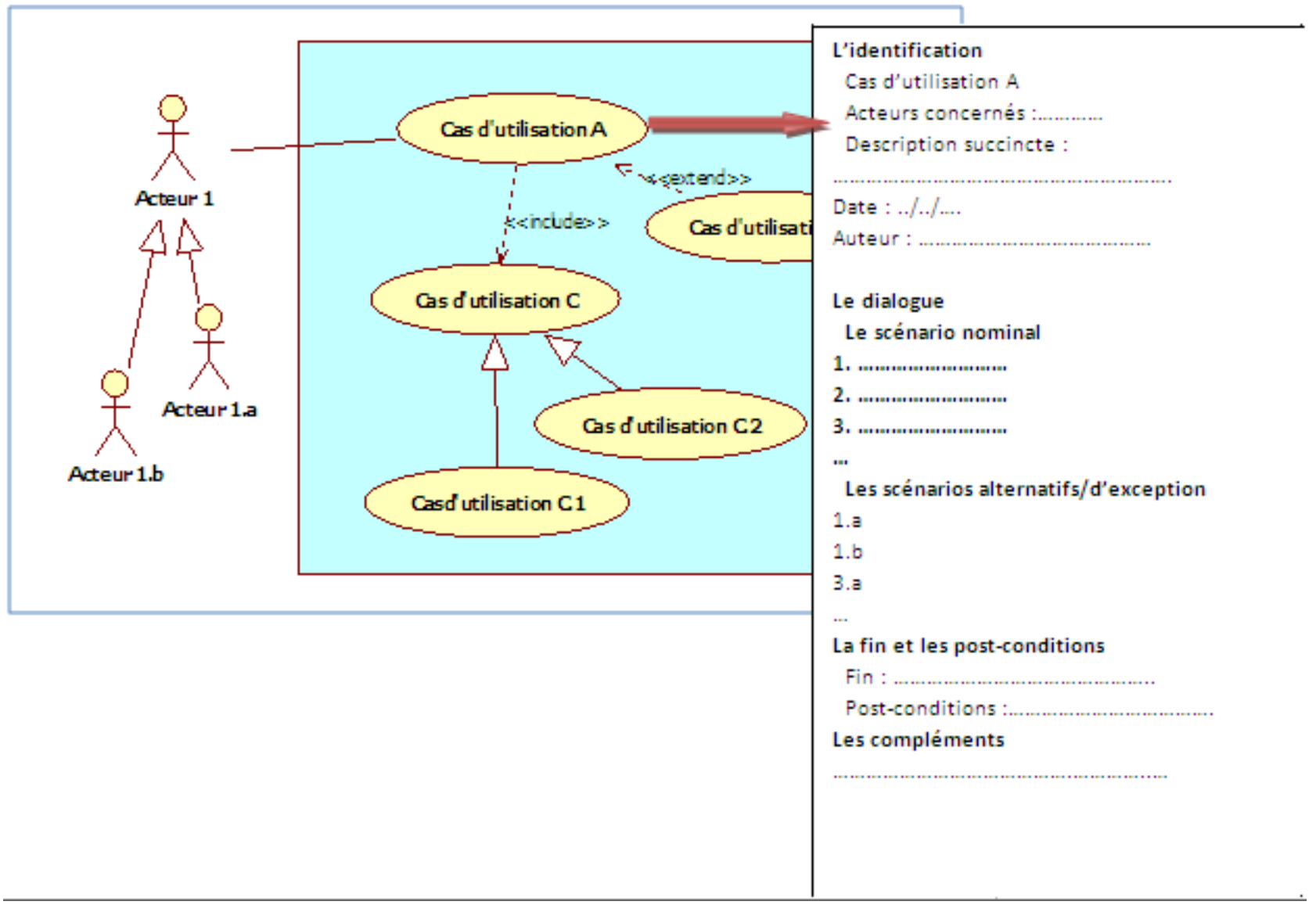


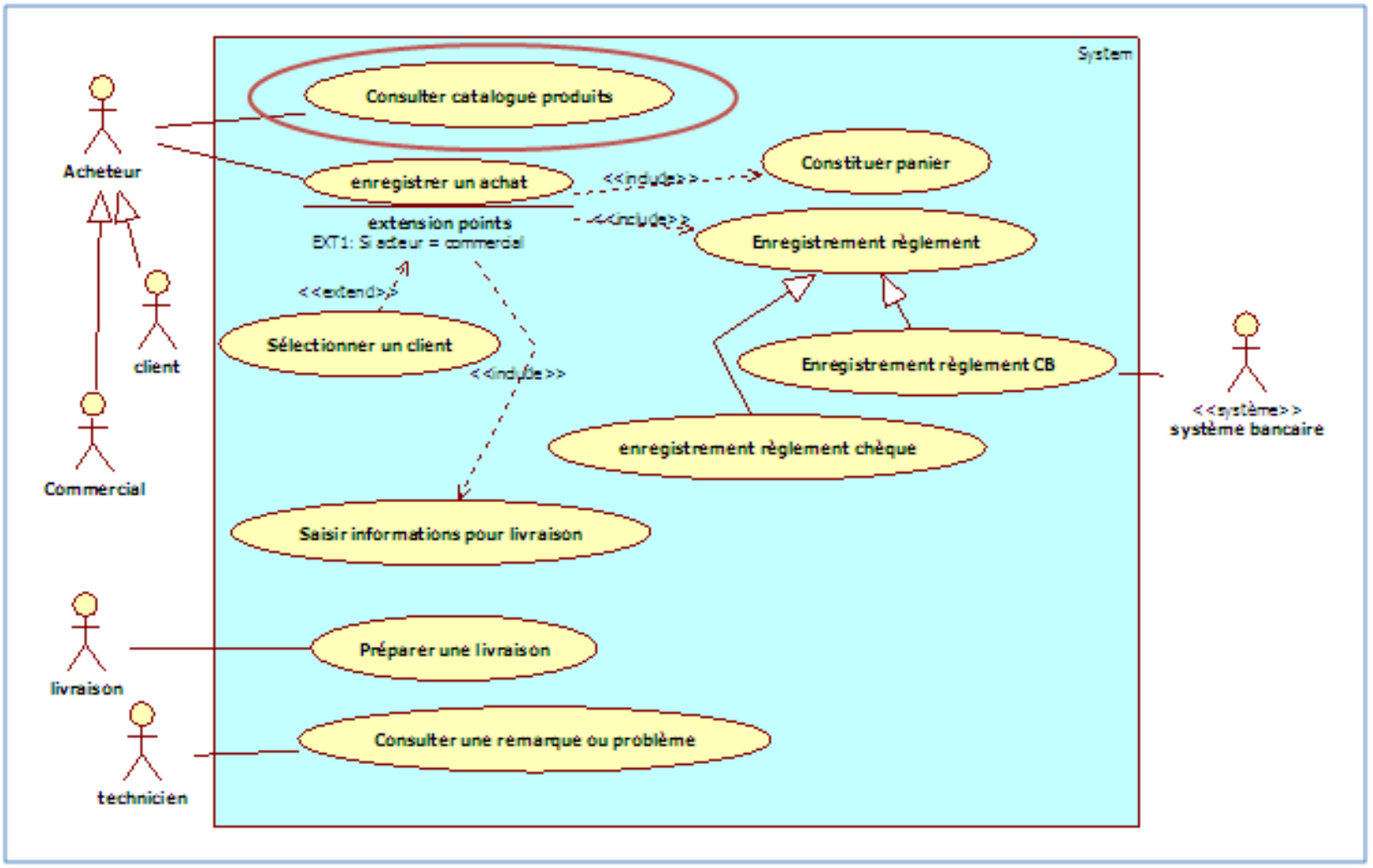
Diagramme des cas d'utilisation + Description textuelle

Description textuelle d'un cas d'utilisation simple

Nous allons d'abord nous intéresser à un cas d'utilisation simple (sans relation avec d'autres cas d'utilisation), par exemple : « Consulter catalogue produits ». À partir de la dernière version du diagramme de cas

d'utilisation du package « Gestion des achats », nous verrons comment cela devrait se faire en pratique.

Prêt ? Allons-y, réalisons ensemble la fiche descriptive du cas d'utilisation « Consulter catalogue produits » visible ci-dessous.



Le diagramme de cas d'utilisation du packages « Gestion des achats »

Volet 1 : L'identification

Dans le volet identification, on indique :

- Le numéro du cas d'utilisation, de façon aléatoire. Cela permet ensuite de les classer plus facilement ;
- le nom du cas d'utilisation (avec indication du package) ;
- l'acteur (ou les acteurs) s'il s'agit d'un cas d'utilisation principal ou le nom du cas d'utilisation principal lorsqu'il s'agit d'un cas d'utilisation interne ;
- une description succincte du cas d'utilisation ;

- la date de rédaction de la fiche et l'auteur, voire éventuellement les dates de mise à jour et les auteurs successifs ;

- les pré-conditions : il s'agit des conditions obligatoires au bon déroulement du cas d'utilisation. Par exemple, nous avons vu dans le chapitre précédent qu'il fallait obligatoirement s'authentifier avant même de pouvoir « Consulter le catalogue produit ». Dans le cas, le package « Authentification » est une pré-condition ;

- les événements à l'origine du démarrage du cas d'utilisation ;

Voici donc comment la fiche descriptive débuterait pour notre cas d'utilisation « Consulter catalogue produit » :

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 »

Volet 2 : La description des scénarios (ou dialogue)

Nous allons maintenant décrire les scénarios qui explicitent la chronologie des actions qui seront réalisées par l'utilisateur et le système. 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.

Les scénarios mettent en évidence les interactions entre les actions de l'utilisateur et le système (le logiciel). Cela peut se faire par une liste numérotée d'actions ou sous forme de tableau qui démontre clairement ce qui est réalisé par l'utilisateur et ce qui est du ressort du système.

La représentation de ces scénarios vous est personnelle. Ici, nous avons choisi de vous proposer une description textuelle, en listant le déroulement du scénario nominal par des chiffres (1, 2, 3…) et les scénarios alternatif et d'exception par des lettres rattachées aux numéros de l'action principale (1a, 1b, 2a, 2b…).

Exemple : Description détaillée du cas d'utilisation « Consulter catalogue produits » (Package « Gestion des achats »).

Pour rappel, la description initiale du projet indiquait qu'un acheteur doit avoir la possibilité de consulter le catalogue des produits. Aucune information supplémentaire n'avait été fournie quant à cette consultation. Il nous appartient donc de poser les bonnes questions au client et aux éventuels futurs utilisateurs pour décrire le scénario nominal. Nous pouvons supposer que notre site de vente en ligne doive permettre une consultation du catalogue de produits, par catégorie.

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.

Les scénarios alternatifs

L'utilisateur doit avoir la possibilité de mettre fin à la consultation du catalogue. C'est d'ailleurs le cas dans de nombreuses boutiques en ligne que vous connaissez. Nous ne devons pas encore décider de quelle façon cela se fera (le bouton « Echap » du clavier, bouton « Quitter » dans la page, un lien « Retour aux catégories », etc.), toutefois, nous pouvons le prévoir dans le scénario alternatif.

Volet 3 : La fin et les post-conditions

Le 3e volet d'une description détaillée d'un cas d'utilisation concerne :

- la fin du cas d'utilisation ;
- les post-conditions.

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.

Par exemple, ici il n'y en a pas car la consultation du catalogue ne donne pas lieu à l'enregistrement d'informations dans un fichier ou une base de données.

Voici le volet 3 de notre cas d'utilisation « Consulter catalogue produits ».

Fin : Scénario nominal : aux étapes 2, 5 ou 7, sur décision de l'utilisateur
Post-conditions : Aucun

Oui, il est vrai que ce cas d'utilisation est tellement simple que nous n'ayons pas grand-chose à indiquer. Il n'y a pas de post-conditions, puisqu'il ne restera pas de preuve tangible du bon fonctionnement du cas d'utilisation. Mais ne vous inquiétez pas, nous verrons un exemple un peu plus fourni dans pas longtemps.

Volet 4 : Les compléments

Les compléments peuvent porter sur des sujets variés :

- l'ergonomie ;
- la performance attendue ;
- des contraintes (techniques ou non) à respecter ;
- des problèmes non résolus (ou questions à poser au client et aux futurs utilisateurs).

Vous avez une petite idée des compléments qu'on pourrait indiquer pour le cas d'utilisation « Consulter catalogue produits » ? Voyons si nous avons eue la même idée.

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) ?

Il se peut que nous ayons plusieurs idées de scénarios. Nous choisissons le déroulement qui nous semble le plus adéquat et nous marquons les autres idées dans la partie « Problèmes non résolus ». Ceci devrait donner lieu à une discussion avec le client et/ou les futurs utilisateurs. Le scénario serait alors à revoir, en fonction de la décision prise.

Bon, nous avons vu les 4 volets d'une fiche descriptive d'un cas d'utilisation. Voici la synthèse de notre première fiche.

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) ?

Je pense que nous sommes prêts à découvrir comment on décrit un cas d'utilisation qui a de nombreuses relations avec d'autres cas d'utilisation. C'est ce que nous allons faire dans la prochaine section.

Les cas d'utilisation complexes

Lors de la description textuelle de cas d'utilisation, vous aurez parfois des cas plus complexes que ce que nous venons de voir. Il s'agit dans ce cas souvent de cas d'utilisation dont le scénario renvoie vers d'autre cas d'utilisation. Les fameux cas d'utilisation internes !

Je vous propose d'illustrer ce point par le cas suivant : « enregistrer un achat ». La particularité de ce cas d'utilisation est qu'il soit lié à plusieurs autres cas d'utilisation avec des relations « include » et « extend ».

L'identification

Cas n° 2

Nom : Enregistrer un achat (package « Gestion des achats »)
Acteur(s) : Acheteur (client ou commercial)
Description : L'enregistrement d'un achat doit pouvoir être utilisé en ligne, par un client ainsi que par les commerciaux de l'entreprise. L'enregistrement comprend les produits demandés et le règlement de l'achat.
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 « Enregistrer des achats »

Description des scénarios

Description détaillé de « Enregistrer un achat » du package « Gestion des achats ».

Le scénario nominal

1. **Le système** vérifie le type d'utilisateur connecté (si commercial ou client)
2. Si l'utilisateur est le commercial, **le système** fait appel au cas d'utilisation interne « sélectionner un client »
3. **Le système** affiche des informations concernant le client

4. **Le système** fait appel au cas d'utilisation interne « Constituer panier » 5. **Le système** fait appel au cas d'utilisation interne « Saisir information pour livraison» 6 **Le système** fait appel au cas d'utilisation interne « Enregistrer le règlement» 7. **Le système** enregistre définitivement l'achat 8. **Le système** affiche le récapitulatif de l'achat.
Les scénarios d'exception 2.a **Le système** n'affiche aucun utilisateur sélectionné. Il affiche « Veuillez sélectionner le client concerné par l'achat » (retour à l'étape 2) 6.a L'enregistrement du règlement n'a pas réussi. **Le système** récapitule les informations dans un message qui est envoyé au département commercial. (Arrêt du cas d'utilisation) 7.a L'enregistrement définitif de l'achat n'a pas réussi. **Le système** récapitule les informations dans un message qui est envoyé au département commercial. (Arrêt du cas d'utilisation)

Les points 6.a et 7.a sont un exemple de ce qui pourrait être demandé par un client ou un utilisateur. Le fait de récapituler les informations d'un achat qui n'a pas abouti pourrait faire objet d'une vérification ultérieure afin de déterminer s'il s'agit d'une erreur au niveau du logiciel, s'il s'agit d'un problème au niveau du règlement ou si cela est dû à une volonté de l'acheteur.

Nous pensions que ce cas d'utilisation était complexe, à cause des différentes relations avec les autres cas d'utilisation. En réalité, la description est assez simple. Il a suffi d'indiquer l'ordre dans lequel les autres cas d'utilisation seront utilisés.

Nous pouvons constater que l'utilisateur n'intervient pas du tout dans le cas d'utilisation principal. Les interactions entre l'utilisateur et le système seront précisées dans les descriptions des cas d'utilisation internes.

On a beaucoup parlé des relations de type « include » et une relation de type « extend », mais je ne les vois pas clairement dans la description. Comment on les distingue ?

Rappelez-vous qu'une relation de type « **extend** » est soumise à une condition. Nous l'avions d'ailleurs indiqué dans le diagramme de cas

d'utilisation : le point d'extension.

Dans la fiche descriptive, cela se traduit par une indication de la condition et du cas d'utilisation lié : **Si l'utilisateur est commercial, le système fait appel au cas d'utilisation « Sélectionner client ».** Cela signifie que les actions du cas d'utilisation « Sélectionner un client » seront déroulées avant de poursuivre avec les autres étapes, uniquement si l'utilisateur connecté est commercial.

Une relation de type « **include** » est indiquée par une simple indication du cas d'utilisation lié. Par exemple : **Le système fait appel au cas d'utilisation « Constituer panier**.
Les actions du cas d'utilisation « Constituer panier » seront donc toujours déroulées à cet endroit, avant de poursuivre avec les autres étapes.

Fin et post-conditions

Fin • Scénario nominal : sur décision de l'utilisateur, après le point 8 (affichage du récapitulatif de l'achat) • Scénario d'exception : après le point 6 ou 7, si l'enregistrement du règlement ou de l'achat définitif ne réussit pas.
Post-conditions • Scénario nominal : l'achat et son règlement ont été enregistrés en base de données. • Scénario d'exception : l'achat a été récapitulé dans un message et a été envoyé au service commercial de l'entreprise.

La fin et les post-conditions sont ici différentes dans le scénario nominal et dans les scénarios d'exception.

On constate que dans le scénario nominal, le cas d'utilisation aura produit un résultat tangible : l'achat et le règlement auront été enregistrés en base de données.
Pour les scénarios d'exception, le résultat serait un envoi de message au

service commercial avec un récapitulatif de l'achat qui n'a pas abouti.

Les compléments

Ergonomie L'enregistrement d'un achat doit pouvoir se faire avec un maximum de 3 pages. Les éventuels messages aux utilisateurs doivent être fournis à l'aide de fenêtres pop-up.
Problèmes résolus Nous avons décrit le cas où un utilisateur est soit un commercial, soit un client connu (indiqué par la pré-condition). Est-ce bien ainsi que cela devra fonctionner ? Serait-il envisageable de dérouler l'ensemble des actions lié à la constitution du panier avant de s'enregistrer comme client ?

Je ne vous dis pas que nous avons vu tous les cas de figure qui peuvent se présenter.
Chaque cas d'utilisation sera particulier et nous devons faire preuve d'imagination, de proposition et surtout d'écoute pour arriver à des descriptions qui reflètent le réel besoin des utilisateurs.

Voici la synthèse de notre deuxième fiche.

Cas n° 2 Nom : Enregistrer un achat (package « Gestion des achats ») **Acteur(s) :** Acheteur (client ou commercial) **Description :** L'enregistrement d'un achat doit pouvoir être utilisé en ligne, par un client ainsi que par les commerciaux de l'entreprise. L'enregistrement comprend les produits demandés et le règlement de l'achat. **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 « Enregistrer des achats »
DESCRIPTION **Le scénario nominal** 1. **Le système** vérifie le type d'utilisateur connecté (si commercial ou client) 2. Si l'utilisateur est le commercial, **le système** fait appel au cas d'utilisation interne « sélectionner un client »

3. **Le système** affiche des informations concernant le client
4. **Le système** fait appel au cas d'utilisation interne « Constituer panier »
5. **Le système** fait appel au cas d'utilisation interne « Saisir information pour livraison»
6. **Le système** fait appel au cas d'utilisation interne « Enregistrer le règlement»
7. **Le système** enregistre définitivement l'achat
8. **Le système** affiche le récapitulatif de l'achat.

Les scénarios d'exception

2.a **Le système** n'affiche aucun utilisateur sélectionné.
Il affiche « Veuillez sélectionner le client concerné par l'achat » (retour à l'étape 2)
6.a L'enregistrement du règlement n'a pas réussi.
Le système récapitule les informations dans un message qui est envoyé au département commercial. (Arrêt du cas d'utilisation)
7.a L'enregistrement définitif de l'achat n'a pas réussi.
Le système récapitule les informations dans un message qui est envoyé au département commercial. (Arrêt du cas d'utilisation)

Fin :

- Scénario nominal : sur décision de l'utilisateur, après le point 8 (affichage du récapitulatif de l'achat)
- Scénario d'exception : après le point 6 ou 7, si l'enregistrement du règlement ou de l'achat définitif ne réussit pas.

Post-conditions :

- Scénario nominal : l'achat et son règlement ont été enregistrés en base de données.
- Scénario d'exception : l'achat a été récapitulé dans un message et a été envoyé au service commercial de l'entreprise.

COMPLEMENTS

Ergonomie

L'enregistrement d'un achat doit pouvoir se faire avec un maximum de 3 pages. Les éventuels messages aux utilisateurs doivent être fournis à l'aide de fenêtres pop-up.

Problèmes résolus

Nous avons décrit le cas où un utilisateur est soit un commercial, soit un client connu (indiqué par la pré-condition). Est-ce bien ainsi que cela devra fonctionner ? Serait-il envisageable de dérouler l'ensemble des actions lié à la constitution du panier avant de s'enregistrer comme client ?

Voilà, nous avons constitué les fiches descriptives de deux cas d'utilisation pas à pas. Vous pouvez consulter les fiches complètes en annexe 1.

Je vous entends penser « Mais, il faut faire cela pour tous les cas d'utilisation ? C'est énormément de travail ! ».

Laissez-moi de nouveau vous répondre par analogie. Si l'architecte en charge de votre projet de construction d'une maison se contentait de faire seulement une partie des plans avant de lancer les travaux, seriez-vous satisfait ?

Je parie que non !

Le diagramme d'activité

Dans ce chapitre, nous allons brièvement voir l'alternative visuelle des descriptions détaillées des cas d'utilisation. Il s'agit du diagramme d'activité.

Description

La fiche descriptive d'un cas d'utilisation peut contenir plusieurs scénarios alternatifs et/ou d'exception. Il est alors difficile d'avoir une vision de l'ensemble des actions. Le diagramme d'activité est un moyen graphique pour donner cette vision d'ensemble.

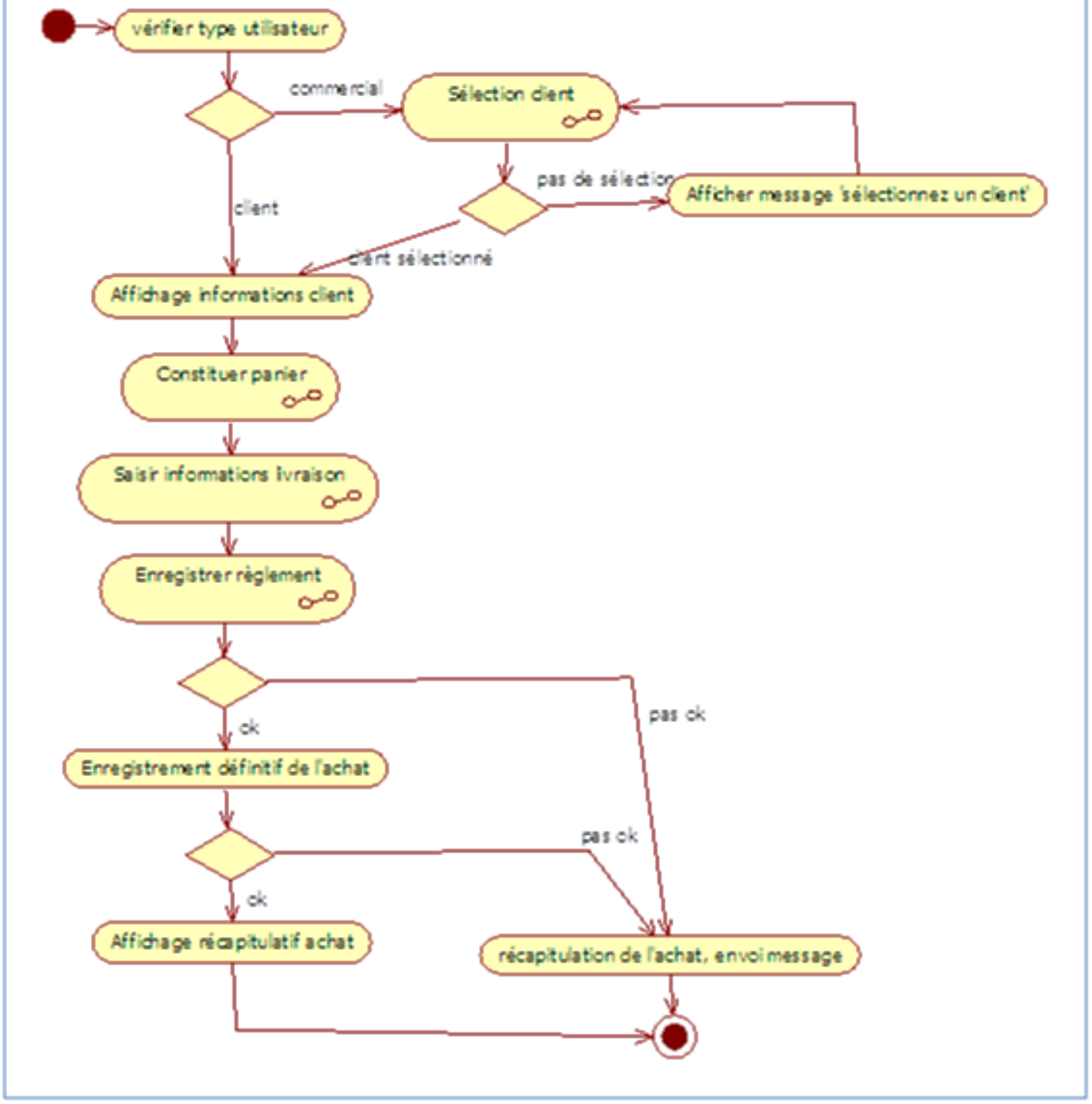
Certaines personnes préfèrent le diagramme d'activités à la description textuelle. Pour ma part, je préfère commencer par une description textuelle qui donne des précisions que nous n'aurons pas dans le diagramme d'activité (des informations telles que les pré-conditions, le démarrage, les post-conditions,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

Ensuite, pour les cas d'utilisation les plus complexes, un diagramme d'activité peut aider à y voir un peu plus clair. Cela peut même aider à trouver de nouvelles questions auxquelles on n'avait pas pensé jusque-là.

La représentation graphique du diagramme

Vous trouverez ci-dessous la légende des différents types de représentation qui apparaissent sur un diagramme d'activité.

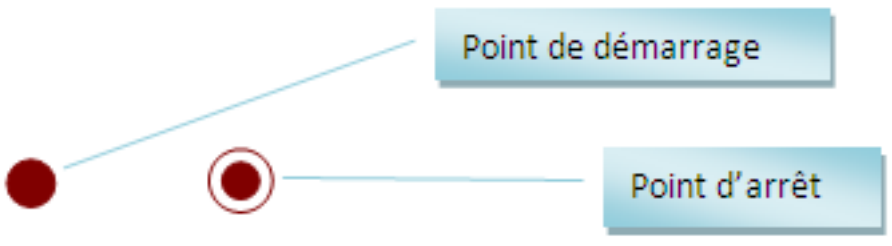
Voici le diagramme d'activité du cas d'utilisation « Enregistrer un achat (package gestion des achats) » :



Exemple de diagramme d'activité

Point de démarrage et d'arrêt

Le diagramme est composé d'un point de démarrage, d'un point arrêt et d'action, qui sont représenté par des cercles rouges.

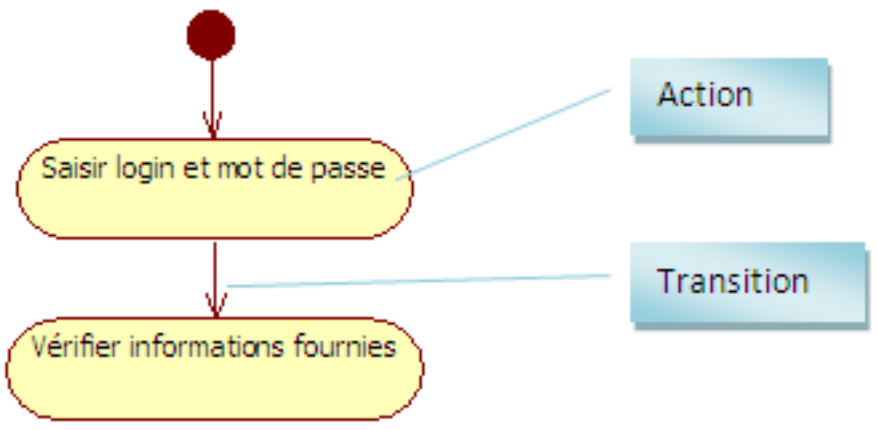


Point d'arrêt et point de démarrage

Les actions et les transitions

Je vous rappelle qu'un diagramme d'activité est une formalisation graphique des actions qui sont réalisées dans un cas d'utilisation.

Le diagramme est donc organisé en actions réalisées soit par un acteur, soit par le système, relié par une flèche indiquant l'enchaînement des actions.



Action / transition

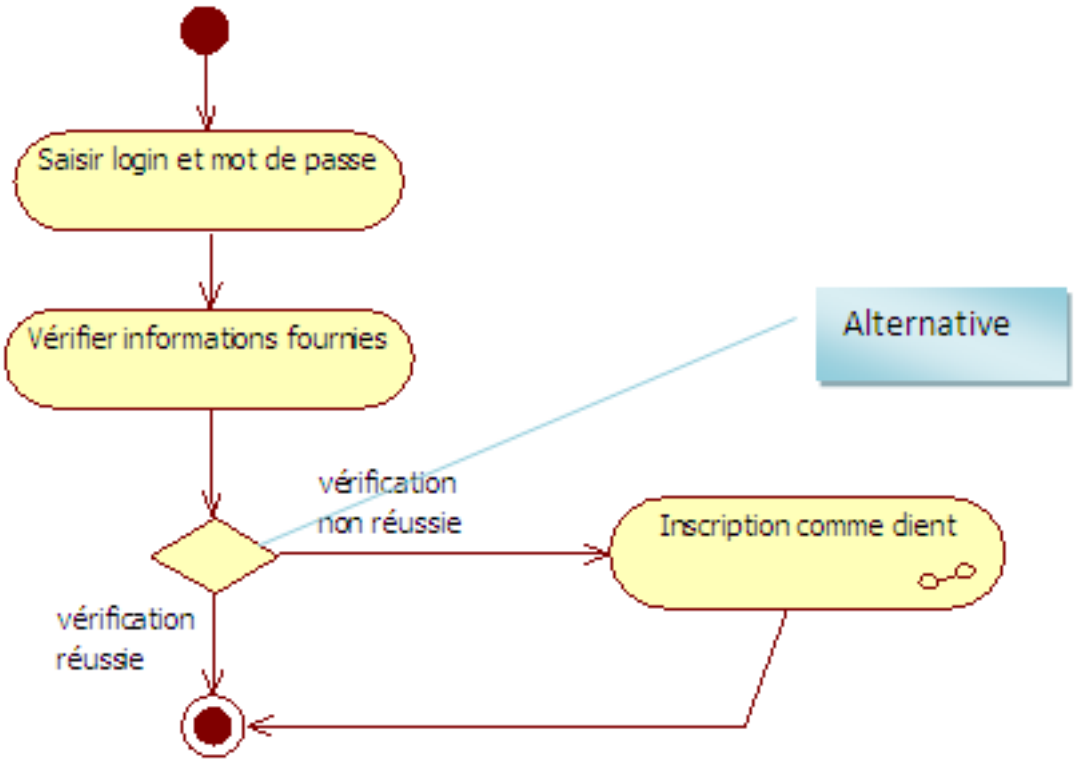
Le lot d'actions, ou autre cas d'utilisation

Si une action du cas d'utilisation correspond à l'appel d'un cas d'utilisation interne (lié par une relation de type « include » ou « extend ») ; elle est représentée par une action contenant un signe spécial : deux cercles reliés par un trait.

Cas d'utilisation inclus

L'alternative

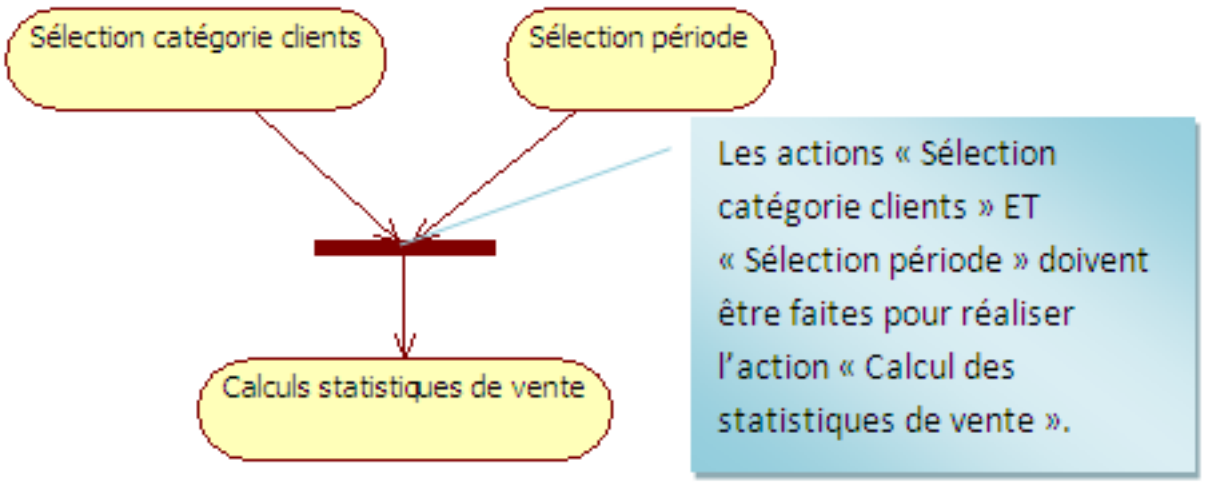
Elle permet d'indiquer les différents scénarios du cas d'utilisation dans un même diagramme.
Dans l'exemple, il s'agit de la condition d'après laquelle le cas d'utilisation « Inscription comme client » serait appelé.



L'alternative

La synchronisation

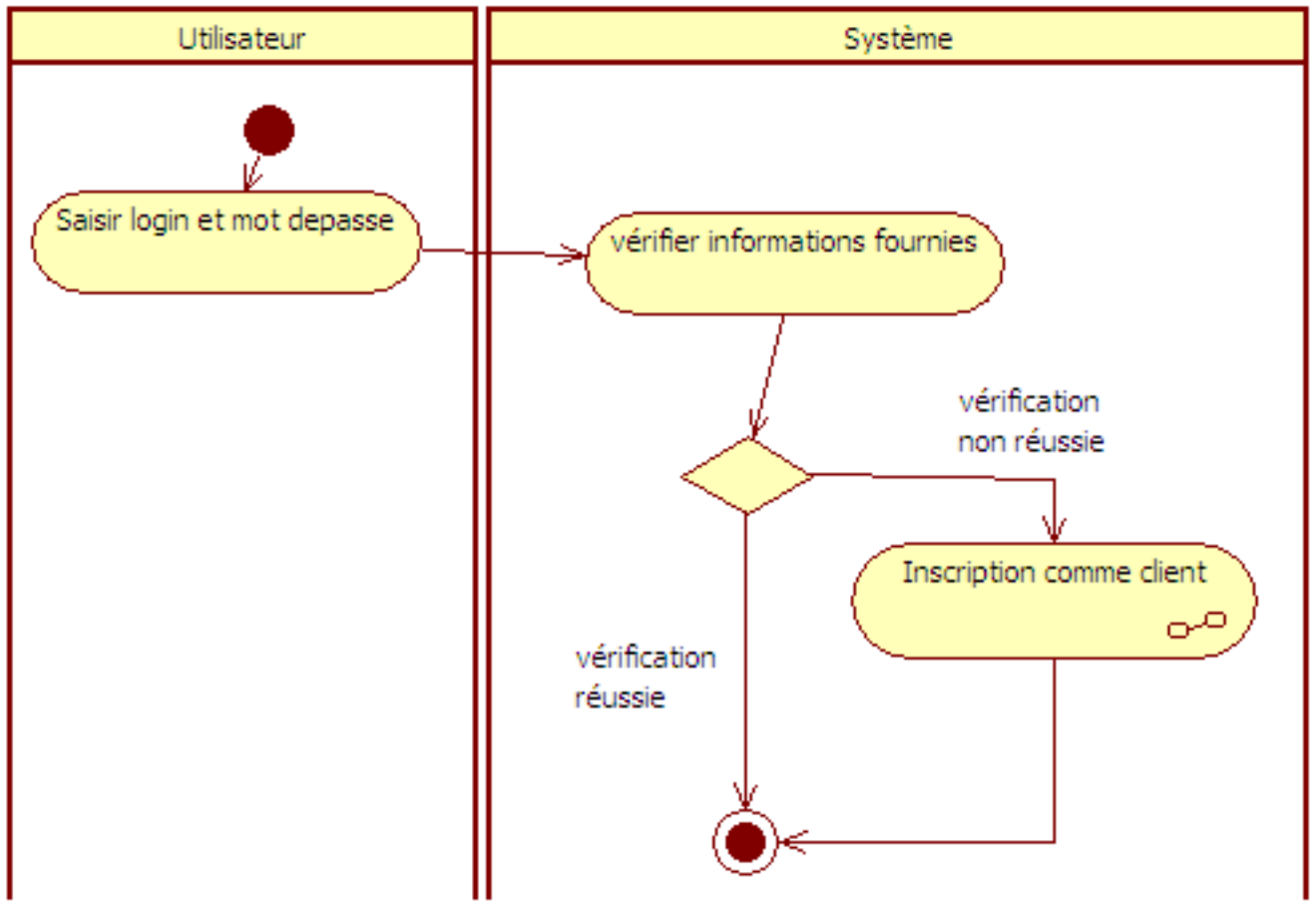
Elle indique qu'il faut avoir réalisé deux actions pour pouvoir réaliser la troisième en-dessous.



La synchronisation

Les couloirs (dit « swimlanes » en anglais)

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



Les 'swimlanes'

Un diagramme d'activité est donc un bon complément à la fiche descriptive d'un cas d'utilisation complexe. Si un cas d'utilisation contient de nombreux scénarios, le diagramme d'activité permet de donner une vision globale de l'ensemble des scénarios possibles.

Fiches descriptives