**Analyse et conception**

Table des matières

[Projet informatique 3](#_Toc75241129)

[Les acteurs du projet 3](#_Toc75241130)

[La gestion de projet prévisionnelle 3](#_Toc75241131)

[Le résumé pour les null 3](#_Toc75241132)

[Processus projet 4](#_Toc75241133)

[Cycle en cascade 4](#_Toc75241134)

[Cycle en V 4](#_Toc75241135)

[Agilité et notion de valeur ajoutée 5](#_Toc75241136)

[Le manifeste Agile 5](#_Toc75241137)

[Le développement incrémental 5](#_Toc75241138)

[Le développement itératif 6](#_Toc75241139)

[Le developpement adaptatif 6](#_Toc75241140)

[Communication Agiles : le regles 6](#_Toc75241141)

[Management visuel 7](#_Toc75241142)

[XP et SCRUM 7](#_Toc75241143)

[Approche TDD 8](#_Toc75241144)

[Scrum : vue global 8](#_Toc75241145)

[Scrum : les roles 8](#_Toc75241146)

[Scrum : les evenements 9](#_Toc75241147)

[Scrum les artefacts 9](#_Toc75241148)

[Planning poker 9](#_Toc75241149)

[Equipe et velocité 9](#_Toc75241150)

[Introspection reguliere 10](#_Toc75241151)

[UML – Unified Modeling language 10](#_Toc75241152)

[Diagramme de structure 11](#_Toc75241153)

[Diagramme de comportement 12](#_Toc75241154)

[Diagramme de cas d’utilisation (use case) 12](#_Toc75241155)

[Diagramme d’état 13](#_Toc75241156)

[Diagramme de sequence 13](#_Toc75241157)

[Les relations dans les shémas 14](#_Toc75241158)

[Phase de lancement 14](#_Toc75241159)

[Le besoin 15](#_Toc75241160)

[Modele métier : diagramme d’activités 15](#_Toc75241161)

[Les acteurs 16](#_Toc75241162)

[Les exigences 16](#_Toc75241163)

[Les packages 16](#_Toc75241164)

[Documents « finaux » 17](#_Toc75241165)

[Itération 17](#_Toc75241166)

[Analyse de données 17](#_Toc75241167)

# Projet informatique

Dans un projet il faut éviter la **dette technique**, qui dit selon Ward Cunningham que quand on code au plus vite de manière non optimale on contracte une dette technique que l’on rembourse tout au long de la vie du projet sous forme de temps de développement de plus en plus long et de bugs de plus en plus fréquent

Dans le cadre de cette dette technique **l’ingénierie logicielles** et les **ateliers de génie logiciel** (AGL) sont indissociable et mette en œuvre des techniques qui permette de réduire le cout de la dette technique en préparant au mieux le **projet**.

## Les acteurs du projet

Une image contenant texte

Description générée automatiquement

## La gestion de projet prévisionnelle



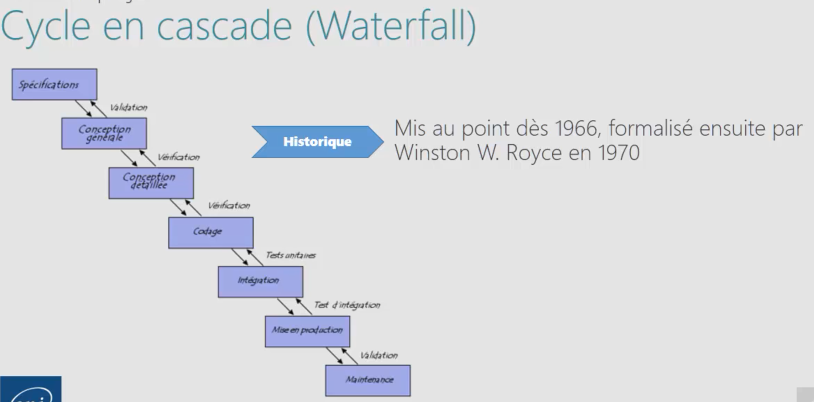
## Le résumé pour les null

**Pour résumer simplement** : Pour éviter la dette technique on doit mettre en œuvre des méthodes d’analyse avant conception réaliser par différente personnes (propriétaire, architecte, chef de projet) afin que le corps de métier puisse réaliser les tâches dans un cadre définis ne laissant place qu’a peu d’improvisation

# Processus projet

Le processus projet (méthodologie) est définie par le chef de projet (si cela n’a pas été déjà décidé par l’architecte ou le propriétaire). Ils existent plusieurs méthodologie, avec chacune leur spécificité

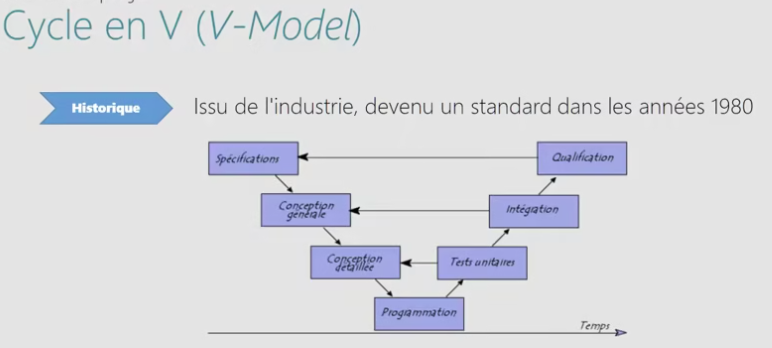
## Cycle en cascade



Les étapes s’enchainent les une après les autres, il est linéaire et aucune tache n’est exécuter en parallèle.

* **Avantages** : une planification claire et un suivi simplifié
* **Inconvénient** : très faible tolérance aux erreurs, peu de possibilités de modifier les spécifications, le client ne voit le résultat qu’une fois la solution livrée

## Cycle en V

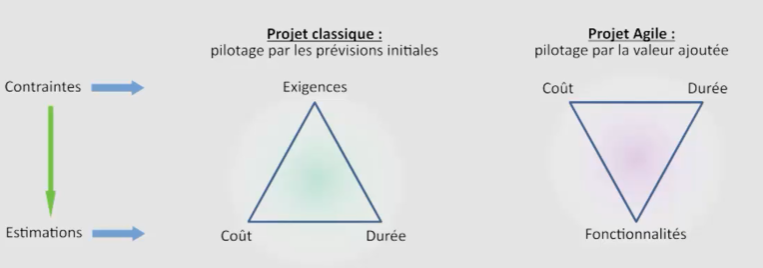


Le cycle en V est une amélioration du cycle en cascade et permet d’exécuter au plus tôt les tests.

## Agilité et notion de valeur ajoutée

Le problème rencontré par les deux précédents cycles est le faible manque de réactivité au changement. L’agilité au sens informatique est la capacité d’une organisation à s’adapter au fur et à mesure au besoin du client.

Sur une application seulement 20% des fonctionnalités sont essentiel et constamment utilisé, il est important que ce soit celle-ci qui sois codé en premier. On s’assure ainsi que le projet convient au client et qu’il sera satisfait.



Dans un projet classique on part des exigences a fin d’estimé le cout et la durée de la mise en œuvre d’un projet tandis qu’en agilité on part du cout et de la durée pour estimer quelles fonctionnalités pourront être faite.

### Le manifeste Agile

Une image contenant texte

Description générée automatiquement

### Le développement incrémental



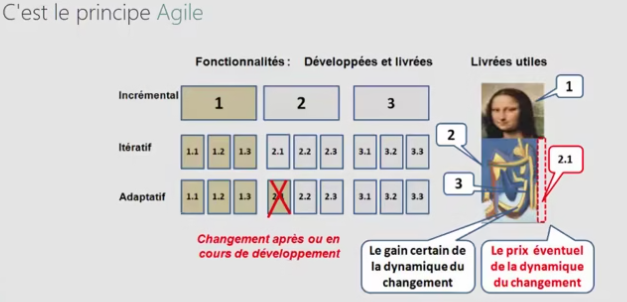
Le développement incrémental est une suite de « brique » que l’ont va développer et ajouter au fur et à mesure afin de terminer notre projet

### Le développement itératif



Dans le cas ou les besoin des le départ ne sont pas définis completement, le developpement itératif permet de s’adpater au fur et à mesure jusqu’à obtention de la satisfaction client.

### Le developpement adaptatif



Le développement adaptatif est une fusion entre le développement itératif et incrémental, il permet de bénéficier à la fois de travailler étape par étape et de répondre au changement efficacement. C’est la méthode utilisé en général.

### Communication Agiles : le regles

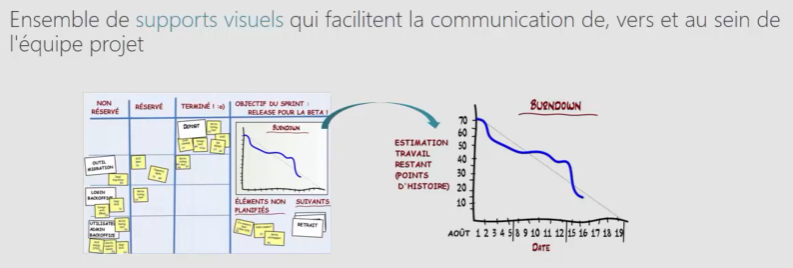
La communication est une des 4 valeurs agile. La communication en vis-à-vis est la référence pour son instantanéité, sa simplicité, son efficacité et la réactivité qu’elle offre.

Une image contenant texte

Description générée automatiquement

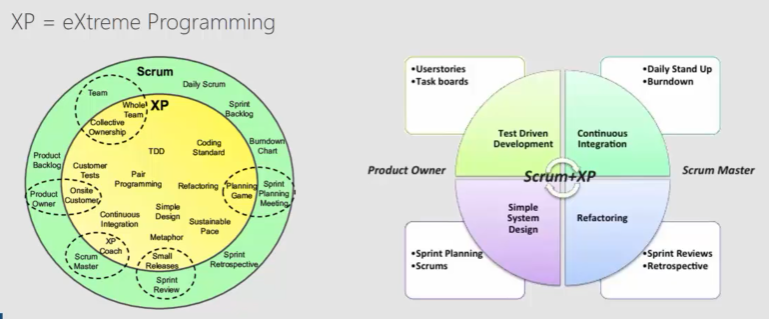
### Management visuel

D’un simple coup d’œil permet de connaitre l’état de l’avancement et permet de présenté de manière simple le reste à faire.



### XP et SCRUM

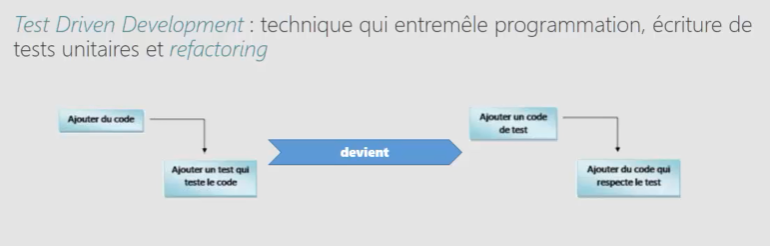
**XP** : e**x**trem **p**rogramming | **SCRUM** permet de gérer le fonctionnement d’XP



Une image contenant texte

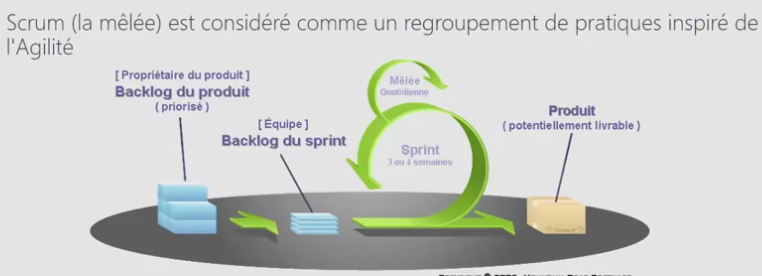
Description générée automatiquement

### Approche TDD



Traditionnellement on ajoute du code puis on test, tandis qu’avec l’approche TDD on créer d’abord le test et on adapte le code ensuite

### Scrum : vue global



Scrum est un cadre de travail qui se concentre sur les but commun a l’équipe et au client et en livrant des produit de la plus grande valeur ajoutés possible. Un Sprint est une décomposition qui est réalisable relativement rapidement afin de pouvoir livré un produit potentiellement livrable.

### Scrum : les roles

Une image contenant texte

Description générée automatiquementUne image contenant texte

Description générée automatiquement

### Scrum : les evenements

Une image contenant texte

Description générée automatiquement

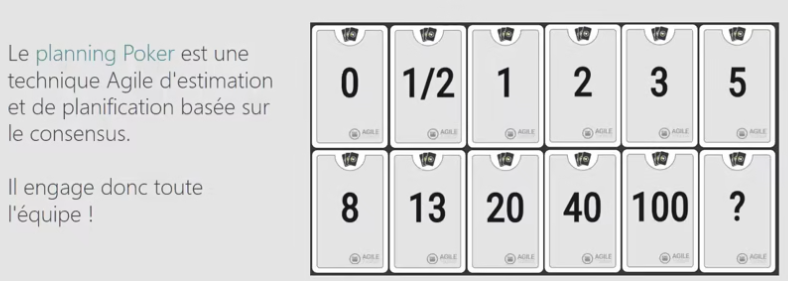
### Scrum les artefacts

Une image contenant texte

Description générée automatiquementUne image contenant table

Description générée automatiquementLes artefacts sont les produit issue des sprint de scrum.

### Planning poker



Le planning poker est un outil de communication qui permet d’estimer et de planifier. Plusieurs membres de l’équipe définissent leur estimations de la complexité et du risque d’une tache secrètement, pois tout le monde dévoile son « jeu » et les plus petit valeur discutent avec les plus grande valeurs pour comprendre au mieux collectivement la tâche.

### Equipe et velocité

Une image contenant texte

Description générée automatiquementDans un projet Agile nous somme contraint par la durée et le budget, en fonction de la durée nous allons pouvoir déterminer quel fonctionnalité nous allons pouvoir écrire. Elle est recalculer en fin de chaque sprint

### Introspection reguliere

Une image contenant texte

Description générée automatiquement

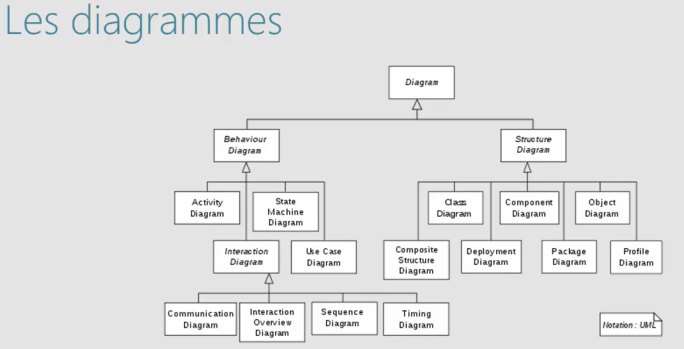
# UML – Unified Modeling language

**UML**, le langage de modélisation unifié (**U**nified **M**odeling **L**anguage) est un langage de modélisation graphique à base de pictogrammes conçu pour décrire et visualiser la conception d’un system, notamment en conception orienté objet

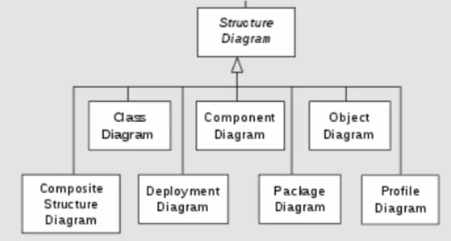
Une image contenant texte

Description générée automatiquement

Il existe des logiciels libres tel que **UMLet** qui permettent de créer des diagrammes.



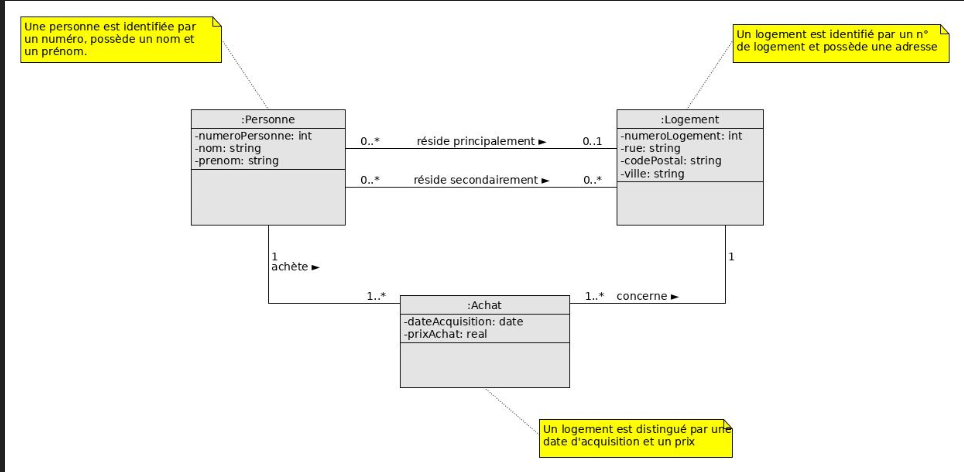
## Diagramme de structure



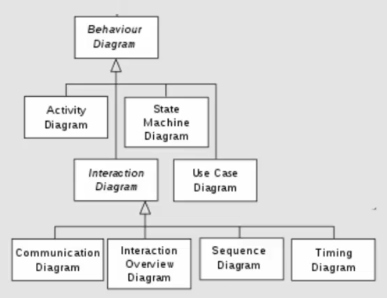
On retrouve dans ce groupe principalement le diagramme de classe utilisé par les AGL (atelier génie logiciel) pour la génération du code.

Une image contenant texte

Description générée automatiquement*Exemple de création de classe sur UMLet*

*Un diagramme de classe*

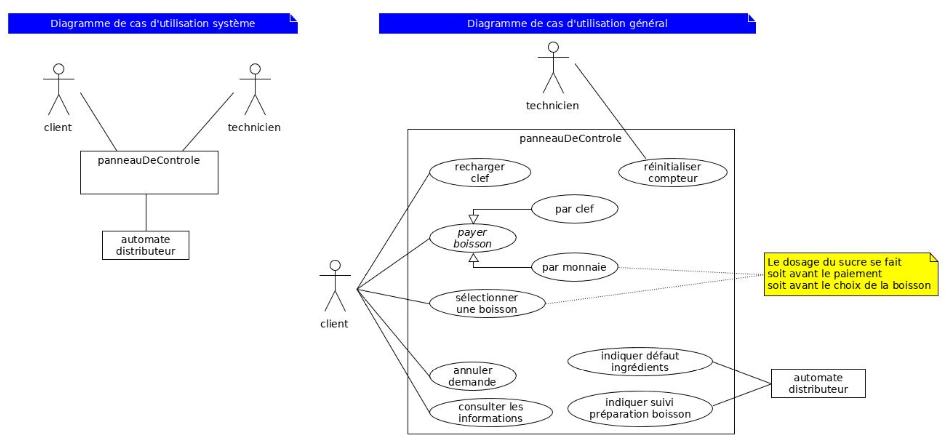
## Diagramme de comportement



On retrouve dans ce groupe les diagrammes de cas d’utilisation, d’état, de séquences et autres.

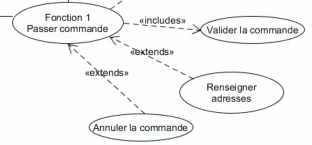
### Diagramme de cas d’utilisation (use case)

*Exemple de diagramme d’utilisation*

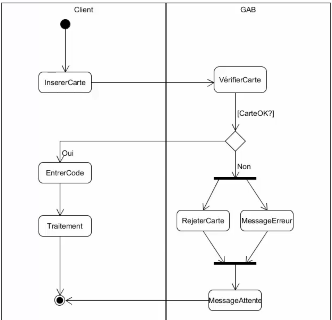


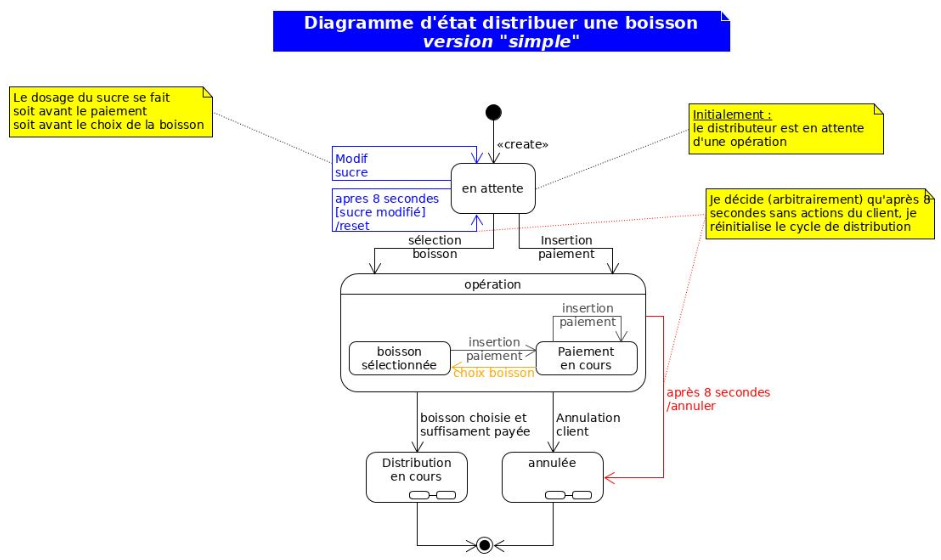
Un cas d’utilisation ne peut être lié qu’a un seul acteur (client, techniciens etc.) mais un utilisateur peut être lié à plusieurs cas d’utilisation. Si on veut pouvoir ajouter plusieurs utilisateurs sur un seul cas d’utilisation il faudra créer un héritage entre les acteurs.

Eviter de croiser les liens, il faut soigner la présentation afin de la rendre la plus compréhensible possible.

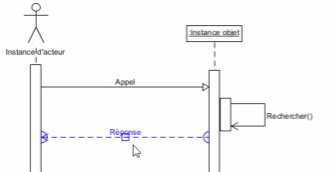
On peut représenter des relations obligatoires avec des « include » et des actions optionnelles avec des « extends »

### Diagramme d’état

Le diagramme d’activité ou diagramme d’état permet de détailler les étapes d’un processus. On part d’une position initiale (le point noir) pour arrivé à un état final (le point noir cerclé de noir en bas). Les branchements conditionnels sont représentés par les losanges.

*Exemple d’un diagramme d’état pour un distributeur de boisson*

### Diagramme de sequence

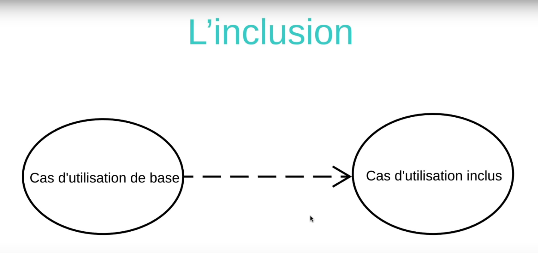
Une image contenant table

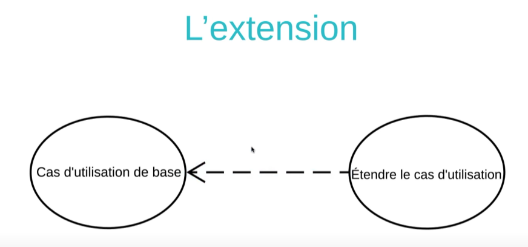
Description générée automatiquement

## Les relations dans les shémas

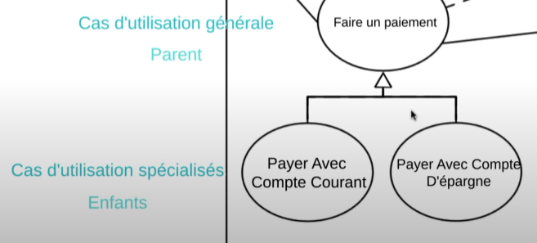
Une image contenant table

Description générée automatiquement

L’**inclusion** signifie que le cas d’utilisation à droite sera **toujours** appelé



L’**extension** signifie que le cas d’utilisation à gauche sera **parfois** appelé



La **généralisation** est la représentation de l’héritage entre les cas, ici par exemple payer avec un compte courant ou payer avec un compte épargné sont des « enfant » du cas faire un paiement.

# Phase de lancement

La phase de lancement est la phase de recueil des besoins du client

Le maitre d’ouvrage (MOA) est le commanditaire, le client, il connait le besoin metier mais ne sais pas réaliser le système.

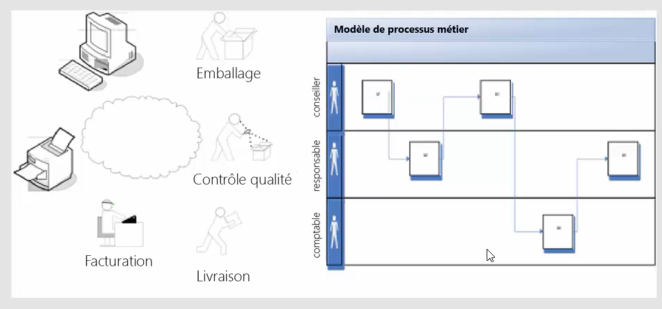
Le maitre d’œuvre (MOE) est le prestataire qui construit le system.

Les deux doivent travailler de concert afin de réaliser un objectif commun.

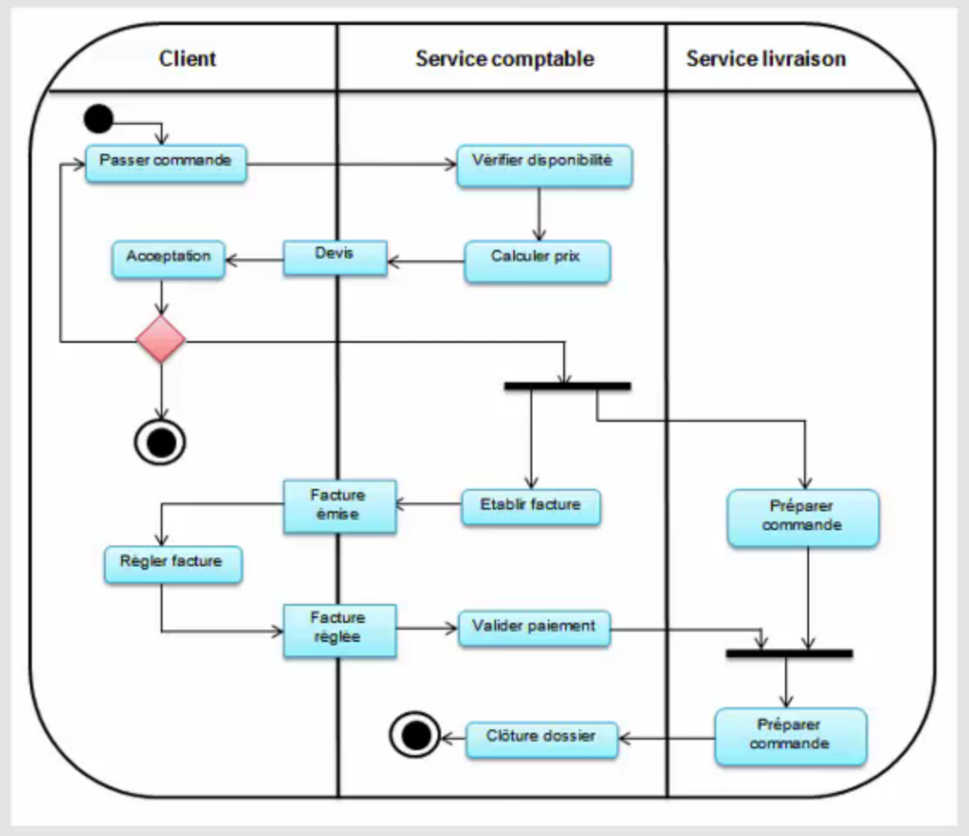
## Le besoinUne image contenant texte Description générée automatiquement

Ces besoins sont décomposés et analysés afin d’en extraire des exigences produit. Pour faire le tour complet des besoins d’un client on utilise les « 5 pourquoi » sur l’expression de ses besoins afin d’en comprendre l’essence.

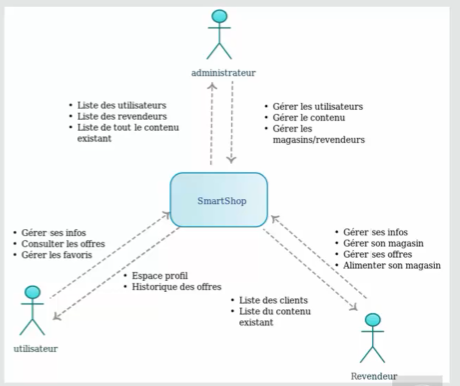
## Modele métier : diagramme d’activités



Il faut comprendre le contexte du client et ses besoins, pour cela on utilise un diagramme d’activité qui permet de définir les différents acteurs et leurs rôles.

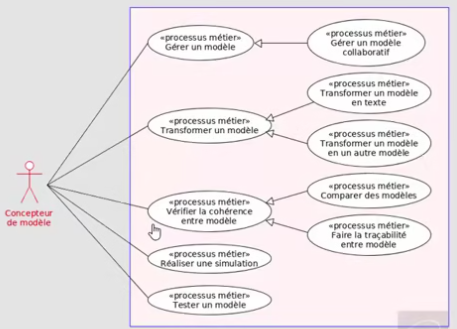


## Les acteurs

Les acteurs sont des personnes ou sous-système qui interagisse avec le system concerné.

Pour représenter tous ces acteurs on utilise un diagramme de cas d’utilisation système tel que ci-contre.

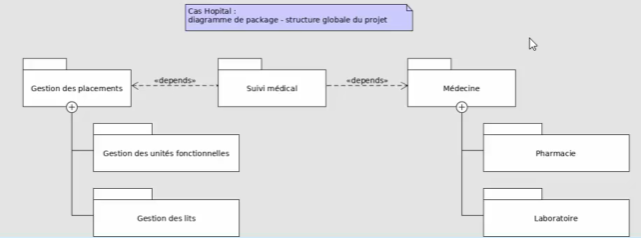
## Les exigences

Pour modéliser les exigences du projet on va utiliser un diagramme de cas d’utilisation général et/ou détailler.

Une image contenant texte, horloge

Description générée automatiquementChaque use case (les ovale) sont relié à un et un seul acteur, si plusieurs acteurs peuvent utiliser un use case il faudra utiliser d’héritage entre les acteurs afin de lier les acteurs entre eux et non les use case.

## Les packages

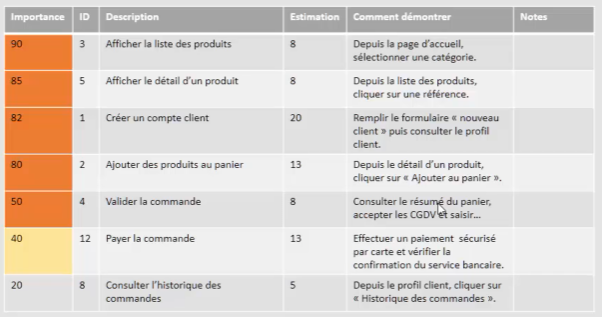


Afin de représenter les fonctionnalités (les packages) on utilise un diagramme de package. Chaque package peut faire l’objet d’un développement autonome et représente donc une brique du projet.

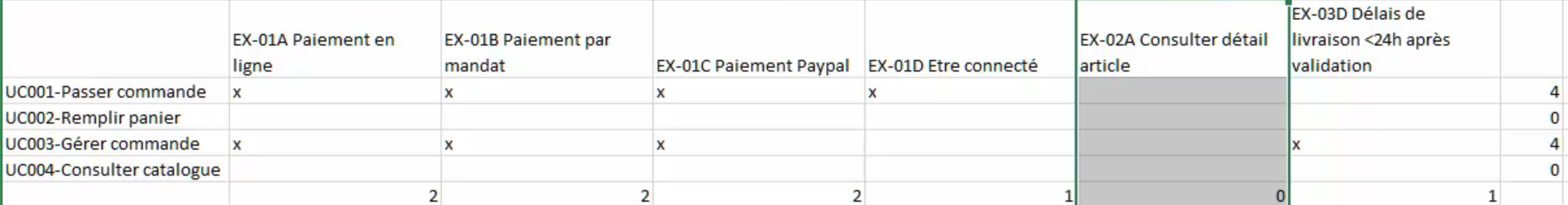
## Documents « finaux »

La liste des exigences (product backlog) vie durant tout le projet, les estimations et les durées peuvent être modifier en cours de projet, généralement elle est constituée des cas d’utilisation déterminé précédemment.

Chaque item de cette liste est noté par ordre d’importance et le temps est estimé.



La matrice de traçabilités est un tableau mettant en relation les exigences du client (en haut) et les fonctionnalités qui ont été déterminer (a gauche). Cela permet de déterminer si toute les exigences on été réaliser et si toute les fonctionnalités ont une utilité (qu’elle sont rattacher à un besoin client)



# Itération

# Analyse de données