Introduction à la méthode

Bertrand LIAUDET

USAGE du document :

les paragraphes précédés d'un (pour UML) sont ceux qui sont traités dans l'introduction du cours UML.

Le titre des paragraphes complets pour UML est entièrement surligné en jaune.

Dans un paragraphe (pour UML), les sous paragraphes (pas pour UML) sont en rose.

En bleu, les définitions et les phrases clé :

La méthode c'est la suite des opérations mises en œuvre pour arriver à un but.

SOMMAIRE

Edition maj mars 2025

INTRODUCTION A LA METHODE

Il est facile de décrire la méthode encore que
son application exige à coup sûr savoir et pratique.
La méthode est dénuée de sens tant qu'elle est déconnectée du rapport au savoir.

USAGE du document :
les paragraphes précédés d'un (pour UML) sont ceux qui sont traités dans l'introduction du cours UML.
Le titre des paragraphes complets pour UML est entièrement surligné en jaune.
Dans un paragraphe (pour UML), les sous paragraphes (pas pour UML) sont en rose.
En bleu, les définitions et les phrases clé :
La méthode c'est la suite des opérations mises en œuvre pour arriver à un but.

1. (pour UML) Généralités sur la méthode

Définition

La méthode c'est la **suite des opérations mises en œuvre pour arriver à un but**.

Le trio impossible : temps, coût, qualité

Les objectifs centraux de toute méthode

⇒ Bien répondre aux besoins et même aider à les exprimer : **meilleure qualité fonctionnelle** (« bien » externe).

⇒ Répondre au **meilleur coût**.

⇒ Répondre avec les **meilleurs délais.**

⇒ Répondre avec une architecture qui permet de **faciliter les évolutions** (évolutions au meilleur coût et dans les meilleurs délais) = répondre avec la **meilleure qualité technique** (« bien » interne).

Le trio impossible : Vite - Bien - pas Cher

⇒ Vite et Bien = > Cher

⇒ Vite et pas Cher = > pas Bien

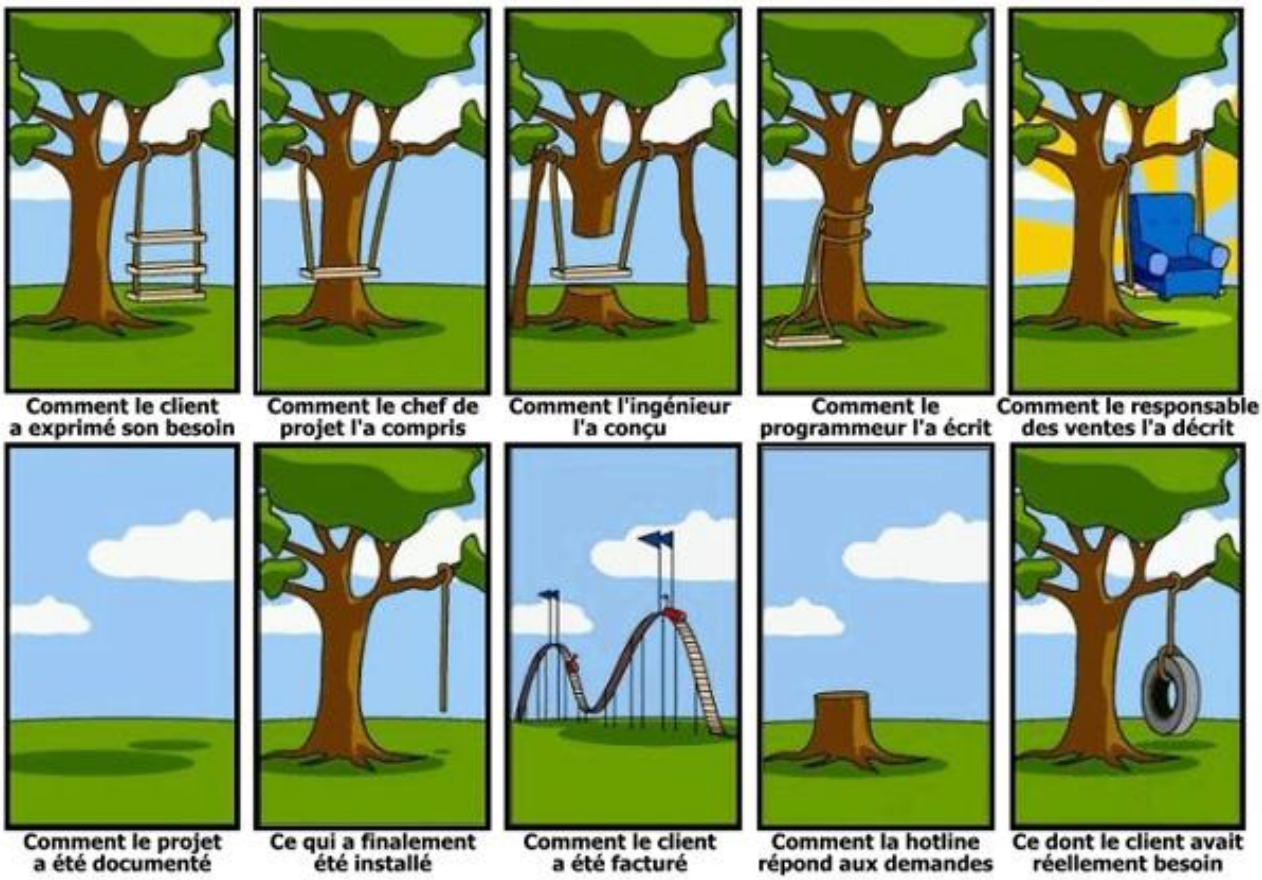
⇒ Bien et pas Cher = > pas Vite

Conclusion

⇒ La méthode s'adapte le plus souvent aux contraintes de temps et d'argent.

Les risques entre le besoin et sa réalisation

- Schémas classiques montrant les difficultés de la bonne compréhension du cahier des charges.
- Ces schémas font apparaître différents acteurs du projet : client, chef de projet, ingénieur, programmeur, commercial, hotline, utilisateur.
- Il montre 3 problèmes classiques :
  - ⇒ A qui s'adresse ce que l'on fait ?
  - ⇒ Comment est-ce compris ?
  - ⇒ Comment est-ce traduit ?
- Les **4 premiers dessins** et le **dernier** sont le plus classiques.



2. (pour UML) Développement d'un logiciel : les 5 distinctions capitales (3 pour UML)

(pour UML) Les 5 distinctions (les les 3 pour UML)

- **1** (pour UML) **:** Développement = Conception + Réalisation
- **2** (pour UML) : Conception = analyse fonctionnelle + analyse technique
- **3** (pas pour UML de base)**:** Analyse = architecture des sous-systèmes + analyse par sous-système
- **4** (pas pour UML de base)**:** Analyse d'un sous-système = analyse des structures + analyse détaillée
- **5** (pour UML) : Données versus Traitements : l'analyse des données.

(pour UML) Vocabulaire – opposition: < et synonymie: =

Pour tout projet : 2 étapes

- La conception
- La réalisation

Pour toute conception : 2 parties

- Le fonctionnel = l' externe = le quoi, le pourquoi
- Le technique = l' interne = l'organique = le comment

Analyse et architecture

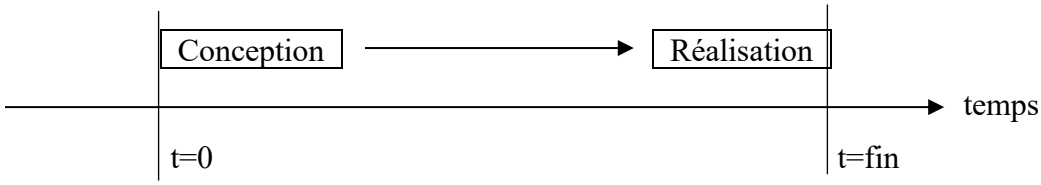
- L' analyse = décomposition et regroupement : découper et classer
  - Analyse fonctionnelle
  - Analyse technique
- L'architecture = le résultat de l'analyse = analyse

(pour UML) 1ère distinction : Développement = Conception + Réalisation

- La première distinction s'applique au développement en général.
- Le développement se compose de **2 activités** qu'on peut distinguer : la **CONCEPTION** et la **REALISATION**.
  - ⇒ **La conception** consiste à **comprendre** et à **décrire** ce qu'il a à faire.
  - ⇒ La **réalisation** consiste à **faire concrètement ce qu'il y a à faire**.

- La distinction entre la conception et la réalisation **permet d'organiser la division du travail**.

  Le premier principe de la méthode consiste à **considérer ces deux activités comme deux étapes successives**. Le projet se déroule dans le temps : il **commence avec la conception**, il se termine avec la réalisation.



- Chaque étape de la conception ou de la réalisation pourra se subdiviser en conception et réalisation à son tour.

(pour UML) 2ème distinction : Conception = Analyse fonctionnelle + Analyse technique

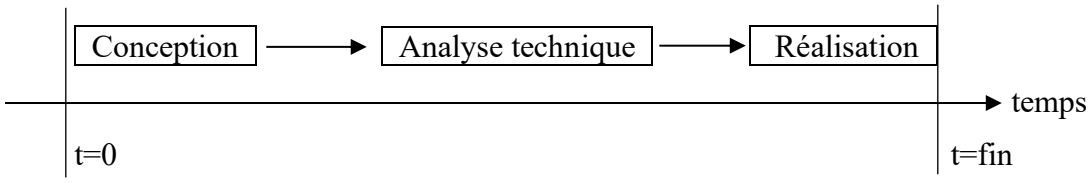
Principes

- La deuxième distinction s'applique à la conception uniquement.
- La CONCEPTION se divise en deux parties : le POURQUOI et le COMMENT : le fonctionnel et le technique.

Précision – 1 : le pourquoi puis le comment

- **Le POURQUOI = l'analyse fonctionnelle** (ou Le fonctionnel ou les spécifications fonctionnelles) **d'abord**.
  - ⇒ L'analyse fonctionnelle s'occupe des fonctionnalités (ou des services) que le système offre à ses utilisateurs.
  - ⇒ Autrement dit elle analyse **POURQUOI** faire le système et aussi **POUR QUI ?**
  - ⇒ Elle s'occupe aussi des relations que les fonctionnalités ont avec d'autres systèmes (analyse des interfaces et des dépendances). Elle répond alors à la question **AVEC QUI et AVEC QUOI.**

- **Le COMMENT = l'analyse technique** (ou Le technique ou l'analyse organique) **ensuite**. L'analyse technique s'occupe de la façon dont sera construit le système pour répondre aux attentes de l'analyse fonctionnelle.



Précision – 2 : right system et system right

Analyse fonctionnelle (= Le FONCTIONNEL)	**Analyse technique (= Le TECHNIQUE)** (= analyse organique)
EXTERNE	**INTERNE**
Le **POURQUOI** POUR QUI – AVEC QUOI – AVEC QUI	Le **COMMENT**
Point de vue de l'utilisateur et du client, le maître d'ouvrage (**MOA**) : celui qui commande le logiciel	**Point de vue de l'informaticien** et du maître d'œuvre (**MOE**) : celui qui réalise le logiciel
Build the right system	Build the system right

- Avec cette distinction, on fait apparaître :
  - ⇒ le **point de vue de l'utilisateur** : le maître d'ouvrage (l'utilisateur, le client, la MOA)
  - ⇒ le **point de vue de l'informaticien** : le maître d'œuvre (le technicien, la MOE)
- **Pour l'utilisateur, ce qui compte** : c'est l'usage du système (à quoi sert le système).
- **Pour l'informaticien, ce qui compte :** c'est l'architecture interne du système (comment est fait le système)
- **L'analyse fonctionnelle** garantit qu'on va **faire ce qui est demandé** : répondre aux exigences du client.
- L'analyse technique (ou organique) garantit que ce qu'on va faire, on va bien le faire.

(pas pour UML) Précision – 3 : Généralisation : API – package - boite noire - responsabilités

- En informatique, on trouve la notion d'**API** : Application Programming Interface
  - ⇒ Une API définit la partie fonctionnelle d'un service : la partie externe pour les utilisateurs qui veulent utiliser le service.
  - ⇒ C'est la même chose que de définir les **méthodes utilisables une fois un package installé**.
- On trouve aussi la notion de « **boite noire** ». Une boite noire est un service dont le contenu interne est invisible mais dont les interfaces, les entrées et les sorties, sont les parties externes qui permettent l'utilisation du service.
  - ⇒ Les fonctions et les méthodes informatiques sont des boites noires utilisées par les développeurs.
- On trouve aussi la notion de « **responsabilité** » associée à une « classe » de POO. La « responsabilité » d'une classe, c'est comme une API ou les méthodes utilisables une fois un package installé. La « responsabilité » d'une classe, ce sont les méthodes et les attributs accessibles par l'utilisateur programmeur de la classe.

- On constate donc que la distinction entre « fonctionnel » et « technique » s'applique à tous les niveaux.
  - ⇒ Pas seulement au niveau « B to C » (Business to Customer, autrement dit Technique vers Fonctionnel)
  - ⇒ Mais aussi aux différents niveaux « B to B » (Technique vers technique).

⇒ C'est la **distinction entre INTERNE et EXTERNE.**

(pas pour UML de base) 3ème distinction : Analyse = Architecture des sous-systèmes + Analyse par sous-système

Principes

- La troisième distinction s'applique aux niveaux fonctionnel et technique.
- Un **système** (d'un point de vue fonctionnel ou technique) peut, le plus souvent, se décomposer, **se découper en sous-systèmes autonomes**. On parle d'architecture des sous-systèmes ou plus simplement **d'architecture système**.
  - ⇒ Découpage veut dire : plusieurs morceaux de puzzle.
- Le découpage en sous-système peut de faire au niveau fonctionnel et/ou au niveau technique.
- Chaque sous-système autonome peut ensuite s'étudier de façon autonome.

Analyse fonctionnelle = Architecture des sous-systèmes fonctionnels + Analyse par sous-système fonctionnel

- L'analyse fonctionnelle se divise en deux parties :
  - ⇒ **L'architecture des sous-systèmes fonctionnels :** elle s'occupe de l'organisation des sous-systèmes fonctionnels, autrement dit des **POSTES de TRAVAIL**. Cette analyse répond à la question **OU ?** Par exemple, dans une bibliothèque, la borne automatique d'emprunt, la borne de consultation et de recherche, le poste du bibliothécaire, le site internet sont autant de sous-systèmes fonctionnels ou postes de travail permettant l'accès à des fonctionnalités de la bibliothèque.
  - ⇒ **L'analyse fonctionnelle par sous-système fonctionnel :** on peut analyser les fonctionnalités au niveau de chaque poste de travail.

Analyse technique = Architecture des sous-systèmes logiciels et matériels + Analyse par sous-système logiciel et/ou matériel

- L'analyse technique (ou organique) se divise en deux parties :
  - ⇒ **L'architecture des sous-systèmes logiciels et/ou matériels** : Elle analyse la division du logiciel en **matériels et programmes distincts**.
    - ⇒ Par exemple, une architecture WEB peut mettre en œuvre une **architecture 3-Tiers** avec un serveur de BD sur une machine, un serveur WEB sur une autre machine et un navigateur sur le poste client.
    - ⇒ Une architecture WEB peut aussi mettre en œuvre une **architecture N-Tiers** en ajoutant des machines dédiées au traitement de certaines fonctionnalités (et se comportant comme une **API privée** pour l'application).
    - ⇒ Pour une application, une API peut s'appeler le « **middelware** ». Les API sont aujourd'hui un composant important des applications. Cf : ici (article de 2018 : legacy = vieux système hérité toujours utilisé et à maintenir). Complément sur le middelware : ici (2022). Complément sur la notion d'intégrateur : ici.
  - ⇒ **L'analyse technique par sous-système logiciel :** L'analyse technique par sous-système s'occupe de la façon dont sera construit chaque sous-système pour répondre aux attentes de l'analyse fonctionnelle.

(pas pour UML de base) 4ème distinction : Analyse d'un sous-système = analyse des structures + analyse détaillée

Principes

- La quatrième distinction s'applique aux niveaux fonctionnel et/ou technique.
- Tout système (ou sous-système), qu'il soit fonctionnel ou technique, peut s'analyser de façon générale par une **analyse des différentes structures (ou des différentes architectures)** qui vont le constituer et qui peuvent être partagées, ou pas, par les différents sous-systèmes. On a dont plusieurs architectures pour un même système, selon le point de vue. Par exemple :
  - ⇒ L'architecture des cas d'utilisation,
  - ⇒ L'architecture des données,
  - ⇒ L'architecture des fichiers et des dossiers,
  - ⇒ L'architecture logicielle (MVC, couches, librairies, etc.)
  - ⇒ L'architecture des classes,
  - ⇒ etc.
- L'analyse des structures peut aussi être appelée : analyse générale.
- **L'analyse détaillée** permet de préciser les contenus de façon détaillée chaque composant des différentes structures.
- Les analyses générale et détaillée peuvent s'appliquer au niveau du système complet au niveau des sous-systèmes.
- Ces principes s'appliquent aux niveaux fonctionnel et technique.

() Analyse générale et détaillée au niveau fonctionnel

- L'analyse fonctionnelle générale consiste à analyser les utilisateurs et les cas d'utilisations en organisant des regroupements adaptés (les généralisations). C'est l'architecture des cas d'utilisation matérialisée par les héritages en UML.
- Elle consiste aussi à analyser, si nécessaire, les **processus métier** (succession de UC correspondant à un usage métier pour le sous-système). Les **diagrammes à travées UML** permettent cela.

- L'analyse fonctionnelle détaillée permet de préciser le contenu des cas d'utilisation. L'UML offre des outils pour cela : les inclusions (include et extend UML), les interfaces (les acteurs passifs), les diagrammes de séquence système, diagramme d'activités, diagrammes d'état-transition.

() Analyse générale et détaillée au niveau technique (organique)

- **L'analyse technique générale (= les architectureS transversales)** s'occupe des différentes **architectureS** qu'on trouve dans le système et qui sont plus ou moins partagées par les sous-systèmes :
  - ⇒ **Architectures logicielles :** MVC, Dossiers, Fichiers, Bibliothèques, Classe, Package, etc. C'est fonction des langages et des usages.
  - ⇒ **Architecture des données :** organisation des données. Structures des données, base de données, classes. Elle reprend la 5ème et dernière distinction.

- **L'analyse technique détaillée** (ou spécifications détaillées) : elle s'occupe du découpage en **fonctions informatiques** ou en **classe avec des méthodes** pour chacun des sous-systèmes logiciels. A ce niveau vont apparaître les en-têtes des fonctions, leurs modes d'emploi, leurs principes de résolution, voir leurs pseudo-codes.

(pour UML) 5ème distinction : données versus traitements : l'analyse des données.

Principes

- Les **distinctions précédentes** sont centrées sur la **question des traitements** (c'est-à-dire des fonctionnalités).
- Seul le point d'architecture des données de l'analyse technique générale aborde la question des données.
- La dernière distinction est la distinction entre les données et les traitements.
- Les **données sont analysées pour elle-même**, indépendamment des traitements qu'on leur appliquera.
- Les données peuvent être analysées à deux niveaux :
  - ⇒ L'analyse **fonctionnelle** : c'est l' **approche base de données** des classes métier.
  - ⇒ L'analyse **technique** : c'est l' **approche orientée objet**.
    - ⇒ Ces façons de faire, globalisantes, sont contradictoires avec l'approche incrémentale des méthodes agiles.
    - ⇒ C'est pourquoi elles sont moins utilisées.

Approche « base de données »

- Dans une approche « base de données », l'analyse concerne les données de la BD.
- Selon le niveau de l'analyse, l'analyse change :
  - ⇒ **Niveau conceptuel (quoi)** : l'analyse des données se fait indépendamment de toute architecture.
  - ⇒ **Niveau organisationnel (qui et où)** : l'analyse des données se fait en fonction des sous-systèmes
  - ⇒ **Niveau logique (avec quel langage)** : l'analyse des données se fait en fonction du modèle des données utilisé (modèle relationnel en général)
  - ⇒ **Niveau physique (avec quelles optimisations)** : l'analyse des données se fait en fonction de choix concret de réalisation en vue d'une optimisation des usages.
- Ainsi :
  - ⇒ Le niveau conceptuel (quoi) correspond au fonctionnel.
  - ⇒ Le niveau organisationnel (qui et où) à l'architecture des sous-systèmes.
  - ⇒ Les niveaux logique et physique à l'analyse technique détaillée.
- Usages concrets :
  - ⇒ Dans la pratique, les niveaux sont souvent mélangés. En effet :
    - ⇒ Les bases de données SQL s'étant généralisées, le niveau logique est toujours « relationnel » et très proche du code SQL donc proche du niveau physique.
    - ⇒ Les informaticiens ayant tendance à préférer le code à la conception, ils ont tendance à proposer un modèle physique pas complètement abouti (c'est-à-dire un modèle proche du code).
  - ⇒ Il y a toutefois toujours une différence entre le modèle conceptuel et le modèle logique : c'est la question des clés étrangères et des « classes d'association ».

Approche « orientée objet »

- Dans une approche « orientée objet », il n'y a pas de découplage entre les données et les traitements.

  ⇒ **L'analyse des données (le diagramme de classes)** se situe donc uniquement **au niveau de l'analyse technique** et pas du tout au niveau de l'analyse fonctionnelle (les cas d'utilisation).

- Cependant, on pourra distinguer un niveau intermédiaire entre le fonctionnel et le technique : le niveau des **classes « métier » qui reprend l'organisation de la BD** et dans lesquelles seront réparties les responsabilités (fonctionnalités) mises au jour par l'analyse fonctionnelle.

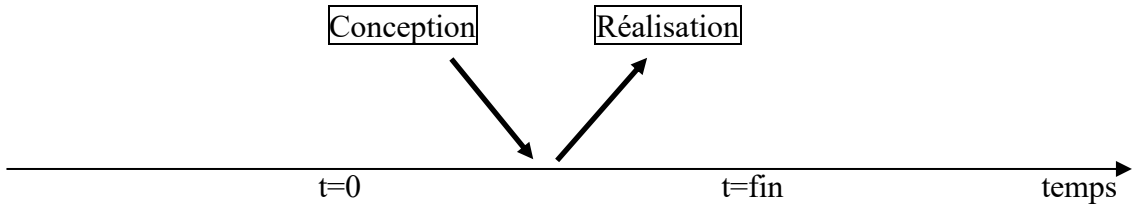
- Au niveau technique, on entrera dans le détail du diagramme des classes avec les méthodes détaillées.

- Les classes « métier » correspondent aux classes du « modèle » dans une architecture MVC.

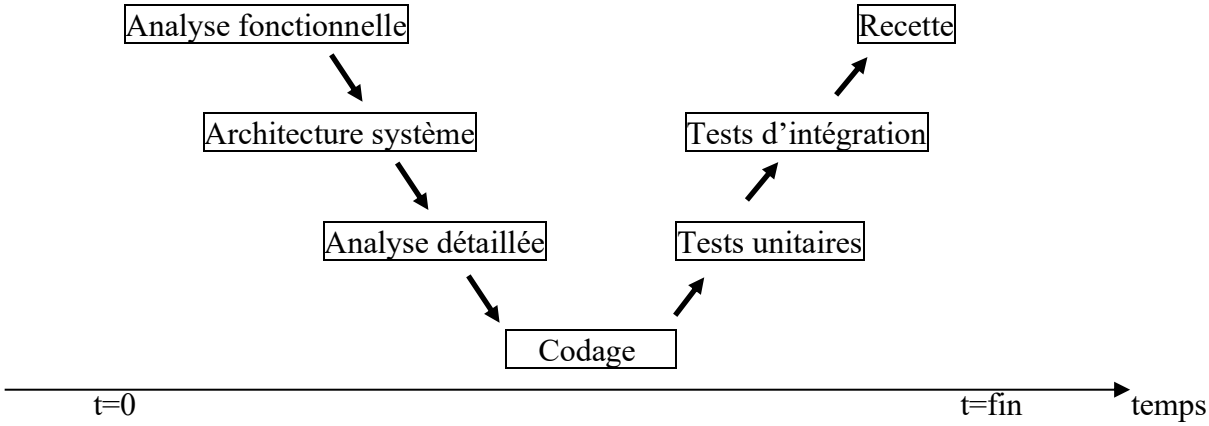
3. (pour UML) Le cycle en V

(pour UML) Construction du cycle en V

- Le cycle en V c'est une méthode classique de développement du logiciel.
- la conception et la réalisation forment les deux branches du cycle en V :



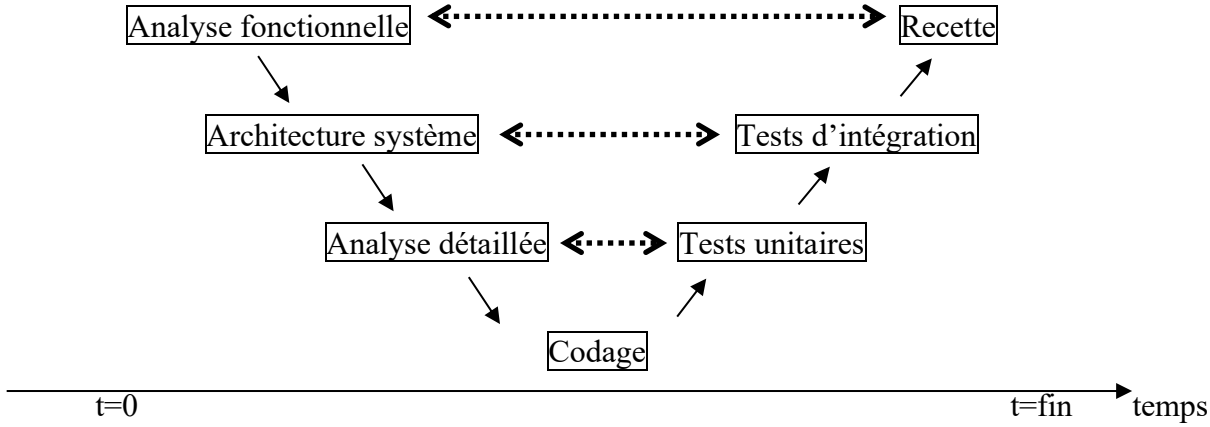
- On reprend les 3 premières distinctions abordées précédemment et en ajoutant des distinctions dans la réalisation :



- **Conception** = Analyse fonctionnelle + Architecture système + Analyse détaillée.
- **Réalisation** = Codage + Tests unitaires + Tests d'intégration + Recette.

(pour UML) Logique du cycle en V

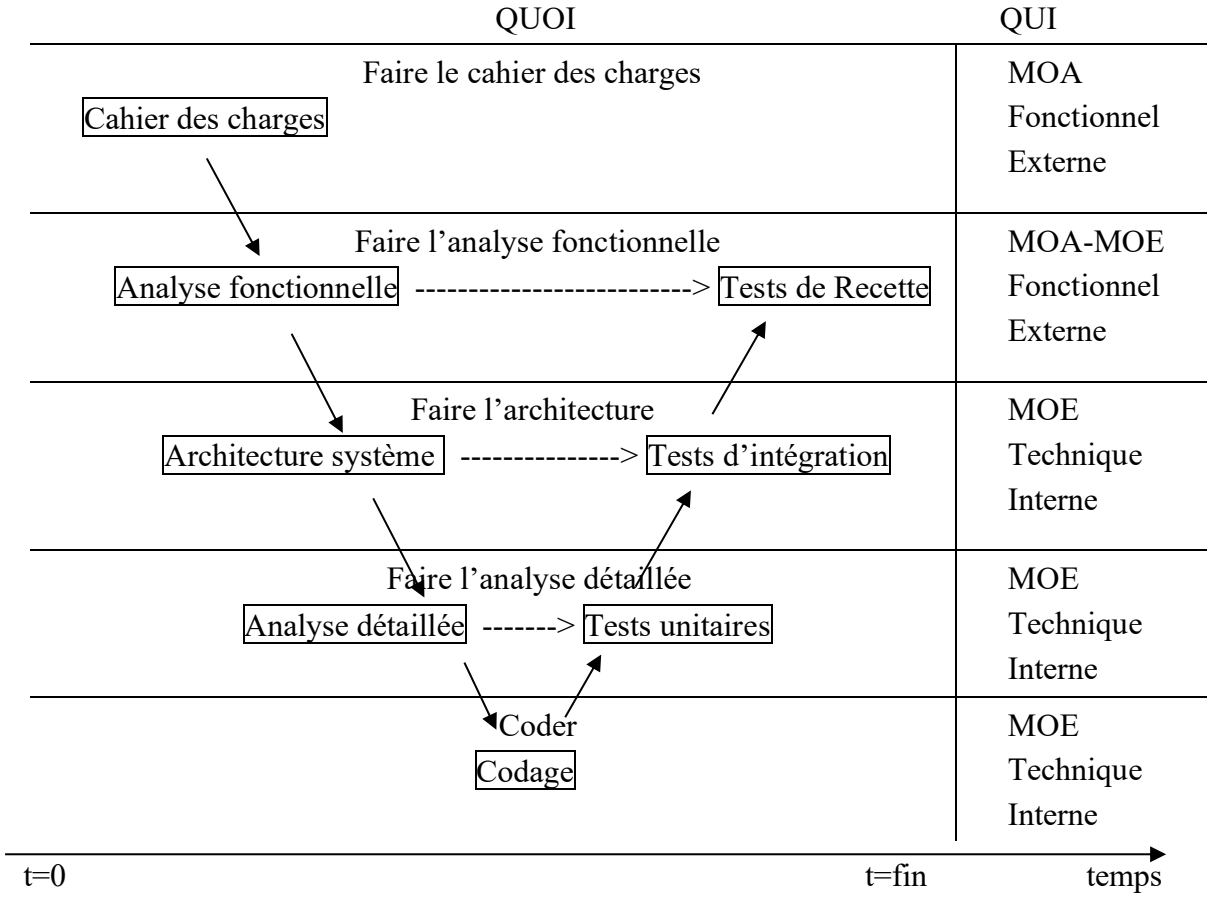
- C'est le lien entre les étapes de chaque branche qui justifie le cycle en V :
  - ⇒ Quand on fait l'analyse fonctionnelle, on peut préparer la procédure de recette.
  - ⇒ Quand on fait l'architecture système, on peut préparer les tests d'intégration des sous-systèmes.
  - ⇒ Quand on fait l'analyse détaillée, on peut préparer les tests unitaires.



- Cette méthode permet de simuler le système et les sous-systèmes au fur et à mesure de l'analyse.
- Cette méthode permet aussi, en cas de problème de test (unitaire, d'intégration ou de recette), de revenir facilement à la partie de la conception à laquelle le problème correspond.

(pas pour UML) Relations cycle en V, MOA, MOE, fonctionnel, technique

- La MOA produit le cahier des charges qui est en entrée du cycle en V.
- MOA = maîtrise d'ouvrage
- MOE = maitrise d'œuvre. C'est celui qui réalise le projet à partir du cahier des charges.



(pas pour UML) Qui fait quoi

- Le cycle en V correspond aussi à un cycle de consommation et de production des documents et de réalisation du logiciel par différents acteurs.
- La question qui se pose toujours quand on produit un document doit être :

  ⇒ **A qui s'adresse ce document ? Pour qui est-il ?**

- De façon schématique, on peut présenter le « qui fait quoi pour qui » suivant :

QUI réalise	**QUOI**	**Pour QUI**
Le maître d'ouvrage (la MOA, le client)	Exprime les besoins dans un **cahier des charges**.	**Le maître d'œuvre** (la MOE)
L'analyste - MOE	Comprend les besoins exprimés dans le cahier des charges. Produit un **document d'analyse fonctionnelle** avec des diagramme de cas d'utilisation, de séquence, d'activités, etc. Produit un **document de recette**.	**L'architecte – MOE** **La MOA** **Le testeur fonctionnel**
L'architecte - MOE	Comprend les besoins exprimés dans le **document d'analyse fonctionnelle**. Produit un **document d'architectureS** avec l'architecture des sous-système, l'architecture MVC, des diagrammes de classes et autres diagrammes. Produit un **document de tests d'intégration.**	**Le programmeur** **Le testeur fonctionnel**
Le programmeur	Comprend les besoins exprimés dans un **document d'architecture.** Réalise les besoins conçus. Fait les tests unitaires. Produit du code pour le testeur technique.	
Le testeur technique	Comprend et met en œuvre les besoins exprimés dans le **document de tests d'intégration et de recette**. Récupère les codes des programmeurs. Produit un document de validation des tests	**Le testeur** **La MOE**
Le testeur fonctionnel	Comprend et met en œuvre les besoins exprimés dans le **document de recette**. Produit un document de validation de la recette.	**La MOA**

(pour UML) Production documentaire

- L'étape d'analyse fonctionnelle conduit à écrire :
  - ⇒ un **document d'analyse fonctionnelle**
  - ⇒ un **document de recette**.
- C'est particulièrement sur cette étape qu'on va travailler dans ce cours :
  - ⇒ L'analyse fonctionnelle sera divisée en plusieurs étapes (les données, les usages, le MVC, la simulation pour la recette).

- Pour produire les documents, on utilise des logiciels variables selon les attentes :
  - ⇒ Traitement de texte classique : **Word,** etc.
  - ⇒ Traitement de texte partagé : **Google docs,** etc.
  - ⇒ Traitement de texte scientifique : LaTex
  - ⇒ Texte avec calcul Python : NoteBook

  - ⇒ Les usages dépendent des entreprises et des applications réalisées.

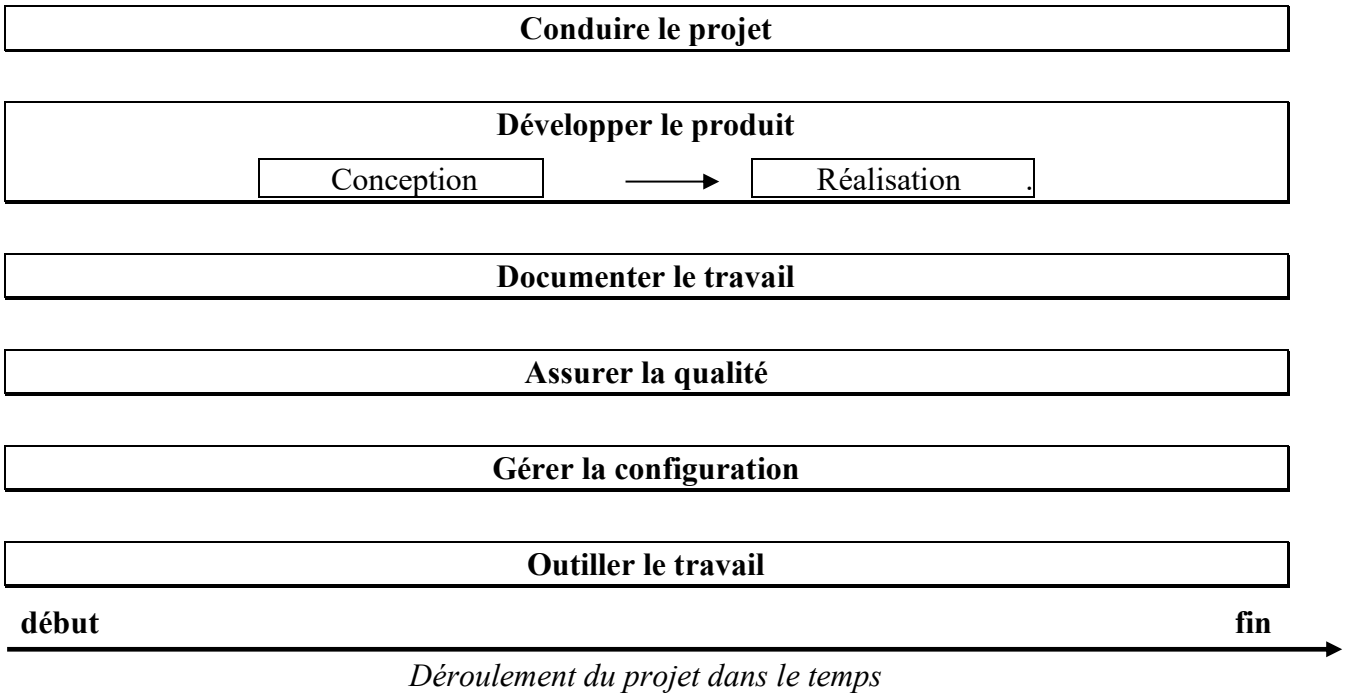
- Un document technique est un **document qu'on doit pouvoir lire « en diagonale » :**
  - ⇒ En utilisant la **table des matières** pour accéder aux chapitres qui nous intéressent et qu'on lira plus ou moins complètement.
  - ⇒ En utilisant le **volet de navigation** qui relie table des matière et contenu si on a une version pdf. Ca s'utilise directement dans le navigateur.
  - ⇒ En utilisant la **recherche directe de mots** si on a une version électronique du document pour accéder aux thèmes qui nous intéressent et qu'on lira plus ou moins complètement.

- **Table des matières :**
  - ⇒ Il faut maitriser la gestion de la table des matières, quel que soit l'outil utilisé.
  - ⇒ La table des matières c'est le plan d'organisation du texte : elle est fondamentale !
  - ⇒ Aujourd'hui, les « pdf » sur accessible sur internet avec Chrome (par exemple) permettent de circuler dans la table des matières : c'est à utiliser !

- **Document partagé** :
  - ⇒ Selon les environnements, on peut aussi utiliser des outils de partage de documents et de gestion de version (google docs)

(pas pour UML) Les activités dans le projet côté MOE

- Le passage du cahier des charges produit par la MOA au produit réalisé par la MOE correspond au projet côté MOE. La méthode va consister à organiser dans le temps les différentes activités de l'ingénierie informatique.
- On peut présenter les choses ainsi :



Déroulement du projet dans le temps

Conduire le projet

C'est le premier travail du chef de projet : la gestion des ressources (humaines, matérielles, financières) et la gestion du planning. Les techniques de planification, d'estimation des charges et des coûts sont présentées un peu plus loin.

Développer du produit

Son phasage classique correspond à celui du **cycle en V**. D'autres phasages ou modèles de développement peuvent être utilisés. Ils sont présentés un peu plus loin.

Documenter le travail

Chaque étape du travail donne lieu à de la production documentaire. Les plans type des principaux documents sont présentés un peu plus loin.

Assurer la qualité

Le suivi de la qualité du travail produit, qu'il y ait ou non un plan d'assurance qualité est une tâche du chef de projet.

La gestion de configuration

C’est la gestion des différentes versions des documents et des logiciels produits. Selon la taille du projet, elle peut être plus ou moins complexe à prendre en compte.

Outiller le travail

Tous les points précédents se font avec des outils : outils de gestion de projet, de développement, de gestion de configuration, etc. Ces outils doivent être recensés.

4. Du cycle en V aux méthodes « agiles » : cycle de vie itératif

Avant le V : cycle de vie en tunnel et cycle de vie en cascade

Le cycle de vie en tunnel

- Il part de l'analyse du cahier des charges pour arriver à la mise en production en passant par un « tunnel » : aucune étape intermédiaire n'est planifiée !

Le cycle de vie en cascade

- Il part de l'analyse du cahier des charges pour arriver à la mise en production en planifiant chaque étape mais sans faire les corrélations du cycle en V.

Avantages du V : la simulation

- L'analyse de la recette au niveau de l'analyse fonctionnelle permet de simuler le résultat final. On peut faire des maquettes du résultat attendu.
- Ainsi, le client valide l'analyse fonctionnelle mais aussi la recette déjà prévue.

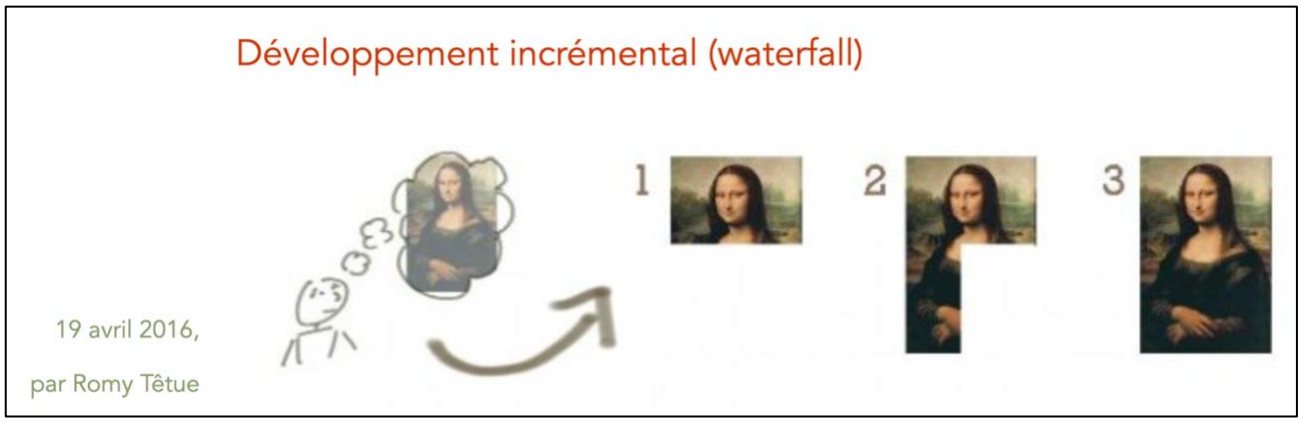
Principal défaut du cycle en V : le résultat apparaît à la fin

- Si le cycle en V est appliqué sur tout le projet, on ne verra le résultat final qu'au cours de la recette, à la dernière étape du projet.
- Cela risque de créer des déceptions !

Après le V : cycle de vie itératif et incrémental

Principes

- Le cycle en V est une méthode rigide et formelle qui est efficace pour des gros développements industriels.
- La méthode peut être assouplie en ajoutant les **notions d'itération et d'incrément.**
- **Une itération** c'est une première **esquisse** grossière du projet complet qui sera affinée à l'itération suivante. **Un incrément** c'est un **morceau** du projet.
- Les méthodes post cycle en V sont **itératives** (du général au détaillé) **et / ou incrémentales (**brique par brique) : http://romy.tetue.net/mona-lisa-agile.



Principes d'une méthode itérative et incrémentale

- Dans une **méthode itérative et incrémentale**, on réalise une partie des fonctionnalités selon le cycle en V, puis une autre, etc. jusqu'à la réalisation de toutes les fonctionnalités (**incrément successifs**), le tout dans un cadre générale qui s'affine au fur et mesure (**itération**).
- Différents modèles sont apparus dès les années 80 comme évolutions du cycle en V : cycle en W, spirale de Boehm, etc.
- Du fait du **caractère systémique du paradigme objet**, et non pas hiérarchique comme celui du paradigme procédural classique, **la programmation objet se prête bien à cette méthode** puisqu'elle consiste concrètement à faire croître le diagramme de classes par **ajout de nouveaux composants logiciels** (classes et paquetages).
- L'avantage de cette méthode est que **le client peut « voir » des résultats concrets au fur et à mesure** du développement. La méthode permet aussi de faciliter les évolutions du cahier des charges.

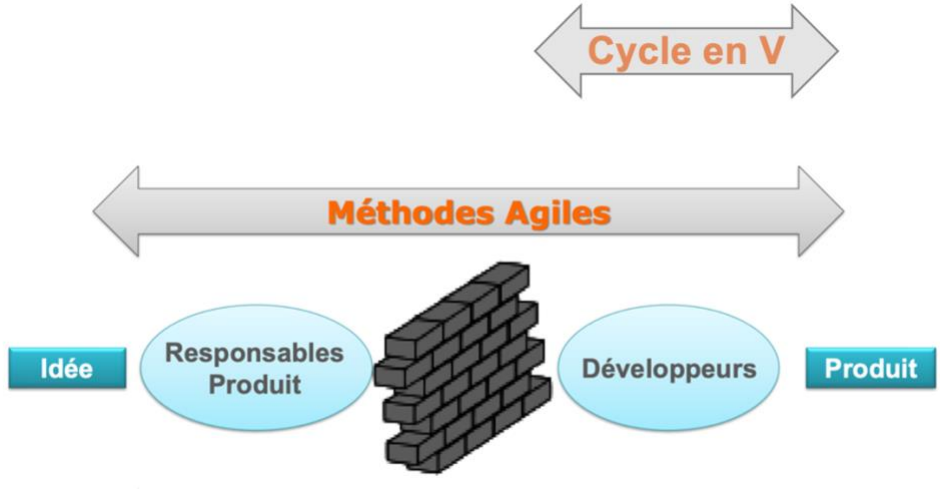
Méthodes agiles

Présentation

- La méthode agile est une méthode incrémentale.
- Il existe plusieurs méthodes agiles.
- La notion de méthode « agile » prend en compte le fait que le client n'est pas forcément capable dès le début de fixer son cahier des charges. Celui-ci peut donc évoluer au fur et à mesure du développement qui sera itératif et incrémental permettant ainsi de pouvoir commencer à tester, voir utiliser en production le produit en cours de développement.
- Ce type de situation correspond particulièrement aux projets web.

Principes des méthodes agiles

- Incrémental
- Adaptable en itérative si nécessaire.
- Adaptatif
- Livraisons régulières de versions opérationnelles (les « sprints » SCRUM)
- Accepter le changement



Les deux écueils à éviter : l'effet tunnel et l'usine à gaz

L'effet tunnel

- L'effet tunnel consiste à rentrer dans un tunnel en début de projet et à n'en ressortir qu'à la fin. Entre temps, personne ne sait ce qui se passe.
- L'effet tunnel risque de conduire à ne pas répondre aux besoins.
- Il faut produire des incréments pour éviter ça : régulièrement, on doit pouvoir mesurer et critiquer l'état d'avancement ainsi que la réalisation.

L'usine à gaz

- « L'usine à gaz », c'est une architecture de « patch » : on ajoute des petits bouts sur d'autres petits bouts pour arriver au résultat final.
- L'agilité permanente risque de conduire à une architecture d' «usine à gaz » : les incréments architecturaux peuvent finalement rendre la structure très difficile à faire évoluer.
- Il faut avoir une vision globale de l'architecture même si on avance par incréments successifs. C'est très évident pour la base de données.
- **L'utilisation de framework** évite cet écueil : il offre une architecture itérative.

5. Des méthodes agiles au DevOps

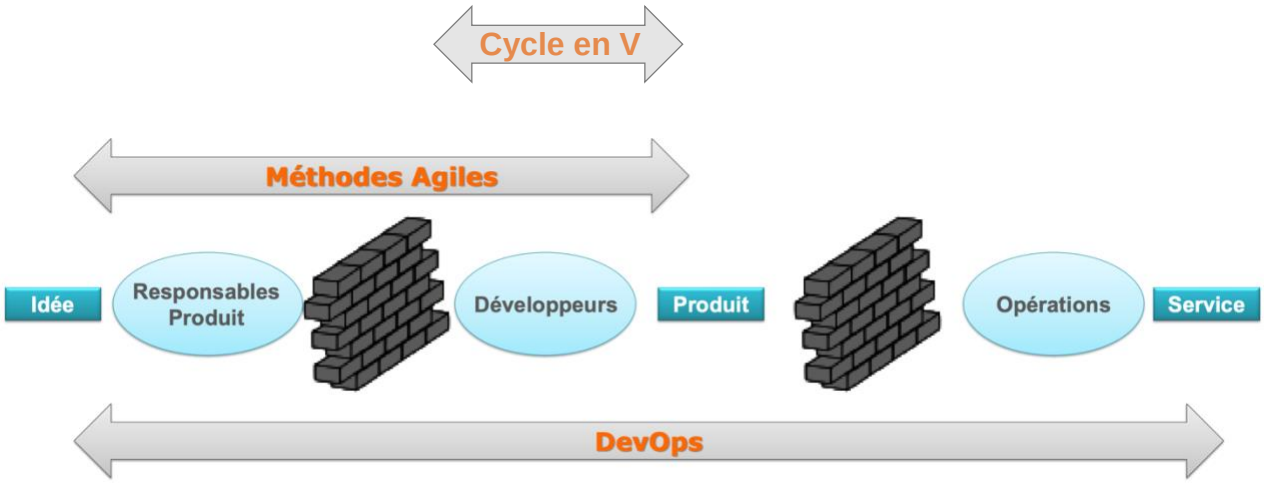
Dev et Ops

- On distingue classiquement :
  - ⇒ Les équipes de développement
  - ⇒ Les équipes opérationnelles
    - ⇒ L'équipe opérationnelle s'occupe de la mise en service et de l'exploitation du produit développé par l'équipe de développement.

Méthode DevOps

- La méthode DevOps consiste à penser le développement :
  - ⇒ des besoins du client (donc de l'utilisateur final)
  - ⇒ jusqu'à l'utilisation concrète par l'utilisateur final.

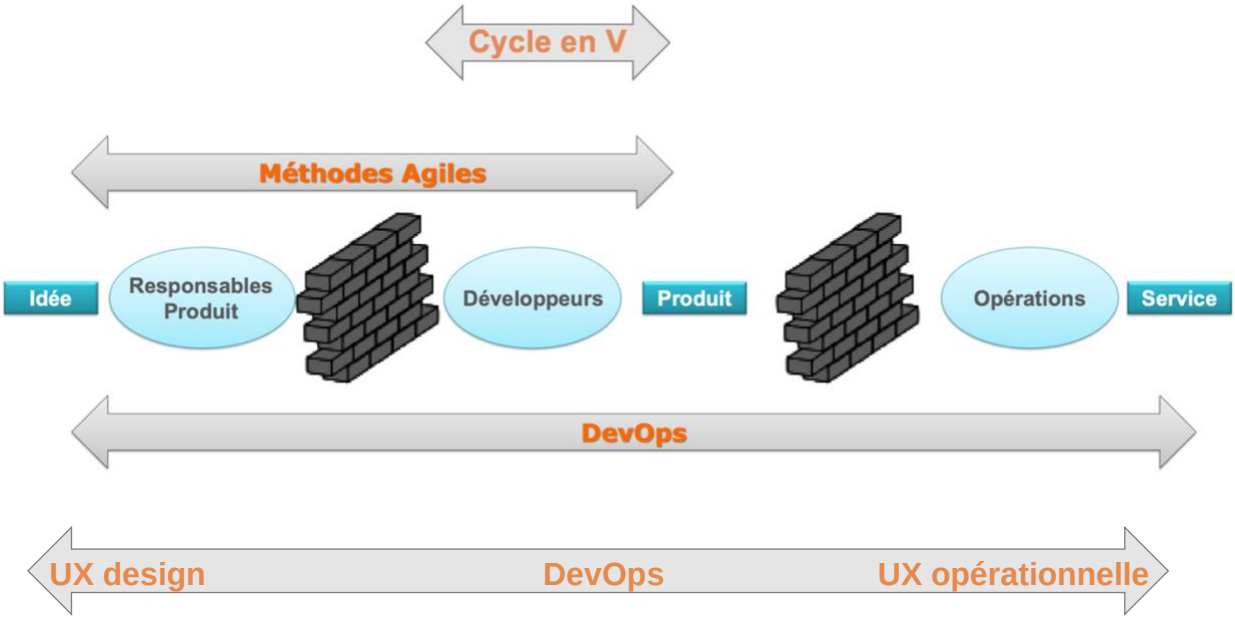
https://streaming-canal-u.fmsh.fr/vod/media/canalu/documents/jdev/jdev2020.t6.keynote.devops.presentation.generale.et.bien.plus._56703/devops4null.2020.pdf



6. UX : l'expérience utilisateur comme méthode

- UX design (conception de l'expérience utilisateur), qui est un aboutissement de l'évolution des méthodes vers une encore meilleure prise en charge des besoins des utilisateurs.
- ↗ UX design
  ↘ UI design

  ⇒ L'UX design s'intéresse à la conception de l'expérience utilisateur (UX design) plus qu'à la création visuelle de l'interface utilisateur (UI design).
- En intégrant l'UX au DevOps, on va de la conception de l'expérience utilisateur à la pratique opérationnelle de l'expérience utilisateur.



7. Le cahier des charges fonctionnel

Plan type d'un cahier des charges fonctionnel

- Contexte :
  - ⇒ Quel est le besoin ? Qu'est-ce qui justifie le besoin ?
  - ⇒ Qu'est-ce qui existe déjà qui répond au besoin (la concurrence) ?
- Quoi :
  - ⇒ Quels sont les services, les fonctionnalités, les usages qu'on veut offrir.
  - ⇒ UC (Use Case) : ancienne approche d'analyse des usages, pas centrés sur l'utilisateur. Les UC peuvent être associées à l'analyse des données.
  - ⇒ US = UX (User Story et User eXperience) : nouvelle approche d'analyse des services en passant par l'expérience de l'utilisateur.
- Qui :
  - ⇒ service pour qui : l'utilisateur. Il a des besoins. On peut mesurer sa satisfaction (UX).
  - ⇒ intérêt de qui : le commanditaire = le bénéficiaire. Il a aussi des besoins. On peut mesurer sa satisfaction. C'est une première approche du modèle économique.
  - ⇒ dépendant de qui : c'est tout ce qui permet de fonctionnement de l'application (les interfaces dont l'application dépend).
- Contraintes :
  - ⇒ techniques : quels sont les matériels et logiciels imposées ?
  - ⇒ humaines : quelles sont les contraintes d'organisation, d'argent et de délai imposées ?
  - ⇒ normatives : quelles sont les lois et les règlements qu'on doit prendre en compte ?
    - ⇒ Norme : ce qui est habituellement (l'usage) OU ce qui doit être. Le devoir être peut être légal (loi / public) ou réglementaire (règlement / privé).
    - ⇒ Normatif : relatif à la norme
- Les données du système :
  - ⇒ On peut analyser les données du projet en tant que SI dans le cahier des charges.
  - ⇒ Cette analyse permet d'accéder à des sous-domaines fonctionnels.
  - ⇒ C'est de la « Business Analyse ».

  Le chapitre suivant reprend cette question.
- On voit que le cahier des charges fonctionnel, qui est du niveau de la MOE, rejoint les problématiques de la MOA. L'AMOA, Assistance à la MOA, consiste à aider la MOA à faire un cahier des charges (une analyse du business). L'AMOA permet à la MOA d'être mieux phaser avec la MOE.

8. La modélisation des données

Où on en est :

- On a passé en revue les méthodes associées aux 4 premières distinctions :
  - ⇒ 1ère distinction : Développement = Conception + Réalisation
  - ⇒ 2ème distinction : Conception = Analyse fonctionnelle + Analyse technique
  - ⇒ 3ème distinction : Analyse = Architecture des sous-systèmes + Analyse par sous-système
  - ⇒ 4ème distinction : Analyse d'un sous-système = analyse des structures + analyse détaillée
- Il reste à revenir sur la 5ème distinction :
  - ⇒ 5ème distinction : données versus traitements : l'analyse des données.

UX vs modélisation des données

- On a vu que les méthodes évoluent vers une prise en compte beaucoup plus fine des utilisateurs :
  - ⇒ On aboutit à l'UX et au DevOps.
- Il reste une autre approche qui existe : celle par les données. C'est la 5ème distinction présentée.
  - ⇒ La partie « fonctionnelle » de cette approche permet d'analyser l'ensemble des données du système, indépendamment de toute contrainte technique.
    - ⇒ C'est ce qu'on appelle la « **modélisation des données** ».
    - ⇒ C'est l'approche « MCD-MLD-MPD ».

La modélisation des données

- La modélisation des données consiste fondamentalement à :
  - ⇒ organiser toutes les données du système dans des tables de type EXCEL,
  - ⇒ reliées entre elles par des « clés étrangères »,
  - ⇒ en évitant toute duplication d'information.
- Cette approche reste pertinente car :
  - ⇒ Les datas sont stables : cette stabilité fait la pérennité de la méthode.
  - ⇒ Le recueil des données est facile à réaliser.
  - ⇒ C'est rapide.
  - ⇒ Ca permet de simuler le fonctionnement général de l'application.
- Cette approche a 2 défauts majeurs :
  - ⇒ En tant que méthode d'analyse globale, elle est contradictoire avec l'approche incrémentale des méthodes agile.
  - ⇒ Les modèles d'analyse ne sont pas standardisés : il y a une véritable jungle des modèles et des syntaxes (MCD, MLD, MPD, MR, MEA, UML, etc.).

9. Méthodologie

La méthodologie c'est la « science des méthodes » :

- La méthodologie consiste à présenter les principes généraux des méthodes en général :
  - ⇒ C'est ce qu'on a abordé en présentant les 5 distinctions principales.
- La méthodologie consiste aussi à présenter un panorama des **principales méthodes actuellement en vogue** en rapport avec les principes généraux :
  - ⇒ C'est ce qu'on a abordé en parcourant les principes du **Cycle en V**, des **méthodes agiles**, du **DevOps** et de l'**UX**, ainsi que le plan type d'un **cahier des charges fonctionnelle** qui est le point d'entrée dans tout projet.
  - ⇒ C'est aussi ce qu'on a abordé en présentant les **principes de la modélisation des données**.
- Enfin, la méthodologie consiste aussi à **étudier les différentes méthodes** qui existent ou ont existées.
  - ⇒ Une méthode est une façon de faire qui change en fonction des cultures, des habitudes, des outils, des modes.
  - ⇒ Il y a des **méthodes actuellement utilisées : cycle en V, agile, DevOps, UX, modélisation des données**.
  - ⇒ Il y a de nombreuses méthodes passées de mode et entrées dans le **grand cimetière des méthodes** ! Elles présentent pourtant toujours certains aspects intéressants.

    ⇒ La suite du document va détailler certaines de ces méthodes, vivantes ou enterrées.

Les fondamentaux

- Le **cycle en V** de base avec la distinction fonctionnel vs technique.
- **Les méthode agiles** (par exemple SCRUM)
- Le DevObs
- L'UX
- **La modélisation des données** par les niveaux d'abstraction de la méthode MERISE (**MCD-MLD-MPD**) avec les 3 niveaux : conceptuel, logique et physique.

Le cimetière des méthodes

- Le cycle en Y qui traite en même temps l'architecture fonctionnelle et l'architecture technique.
- La spirale de Boehm, méthode itérative et incrémentale.
- La méthode RUP, qui était une méthode incrémentale et itérative de la fin des années 90, début des années 2000, directement associée au développement de l'UML.

10. UX design (méthode vivante)

UX et UI

Définition

- L'UI c'est la User Interface : l'interface utilisateur.
- L'UX c'est la User eXperience : l'expérience utilisateur.

Design

- UI design = design d'interface.
  - ⇒ Design veut alors dire **création visuel** (design classique) plus que conception
  - ⇒ L'UI est du côté de la réalisation.
    - ⇒ A noter que l'UI peut être en mode **GUI** (Graphical User Interface) ou **CLI** (Command Line Interface).
- UX design = design d'expérience.
  - ⇒ Design veut alors dire **conception** plus que création visuelle.
  - ⇒ L'UX est du côté de la conception.
  - ⇒ L'UX designer intervient en amont de l'UI.

Exemple

- Expérience d'utilisateur : manger des céréales au petit déjeuner.
  - ⇒ Désigner cette expérience, c'est désigner le bol et la cuillère nécessaires pour l'expérience.

Principes

- Il faut toujours partir des besoins réels, des attentes : c'est centré utilisateur.
  - ⇒ L'UI peut être magnifique, mais si personne ne l'utilise, c'est inutile.
  - ⇒ L'UX s'intéresse à ce qui est utile pour l'utilisateur.

Synthèse

UX : Expérience	**UI :** Interface
Le fond	La forme
L' « interne »	L'externe
Le fonctionnel	Le technique
Conception centré utilisateur	Réalisation
L'utilité	L'esthétique
Se demande si le bouton est utile	S'intéresse à la couleur du bouton pour qu'il soit visible

Entrainement

- Sur tout site : qu'est-ce qui est UI, qu'est-ce qui est UX.
  - ⇒ La difficulté, c'est que l'UX passe par une bonne maitrise des besoins de l'utilisateur.
  - ⇒ Exemple de site : paris librairies

Méthodologie pour une démarche ergonomique

Les étapes

1 : analyse du contexte

2 : liste des objectifs et des attentes-besoins-objectifs des utilisateurs.

3 : conception fonctionnelle (arborescence et wireframes)

4 : les tests utilisateurs (règles et bonnes pratiques de l'ergonomie web)

1 : le contexte

- Si le site existe déjà :
  - ⇒ Analyse des usages du site.
  - ⇒ Recueil de feedback.
  - ⇒ Benchmark concurrentiel.
- Si le site n'existe pas déjà :
  - ⇒ Analyse des usages sans site.
  - ⇒ Benchmark concurrentiel.

2 : les utilisateurs et leurs besoins : persona, scenario et storyboard

- Il faut d'abord définir les personas et les fonctionnalités par recueil des usages.
- **Persona** : on liste les objectifs et les attentes-besoins-objectifs des utilisateurs.
  - ⇒ Définition d'utilisateur-type : les personas
  - ⇒ Les personas permettent de modéliser les besoins réels des utilisateurs et leurs parcours.
  - ⇒ On leur donne une identité (nom, photo, âge) pour nous les rendre plus « sensibles ».
- **Scenario** : Pour répondre aux besoins et pour innover, il faut concevoir des « **scénarios d'expérience utilisateur** » ou « **scénario d'usage** ».
  - ⇒ Il faut observer les **utilisateurs** : les clients de nos clients
    - ⇒ C'est un travail d'anthropologue.
    - ⇒ La conception concerne d'abord l'utilité (pour les utilisateurs !)
    - ⇒ Pour innover, il faut savoir sortir des usages et des bonnes pratiques.
  - ⇒ Les scénarios d'expérience utilisateur sont le cœur du futur cahier des charges.
  - ⇒ Un scénario c'est :
    - ⇒ Comment Jean-Luc va faire ses courses. Il va au supermarché et avec sa voiture et il remplit son coffre. Une voiture, c'est une solution de mobilité pour un utilisateur. Ce que veux Jean-Luc, c'est aller d'un point à un autre en transportant des choses ou des personnes.
    - ⇒ Comment Mathilde utilise une application mobile pour acheter un billet pour un atelier de design alors qu'elle se rend au travail.
- Scenario vs **User Story** ?
  - ⇒ La méthode agile utilise la notion de User Story (US) :
    - ⇒ En tant que <qui>, je veux <quoi> afin de <pourquoi>
  - ⇒ Un scénario est à peu près identique : il introduit en plus un contexte :
    - ⇒ En tant que <qui>, je veux <quoi> afin de <pourquoi> dans tel <contexte>
  - ⇒ https://uxplanet.org/user-personas-scenarios-user-stories-and-storyboards-whats-the-difference-cf00315f0799
  - ⇒ https://storiesonboard.com/blog/its-all-about-ux-through-user-story-mapping
- Le **storyboard** c'est la description plus ou moins dessiné du scénario. C'est un concept cinématographique.

3 : la conception fonctionnelle : l'arborescence des scénarios et les wireframes

Présentation

- Une fois qu'on a les scenarios et les personas :
  - ⇒ On organise les scénarios dans une arborescence :
    - ⇒ La conception concerne aussi la structure.
  - ⇒ On crée des wireframes pour chaque scénario
- La création de l'UI viendra après.

L'arborescence des scénarios

- On organise les scénarios dans une arboresence (un menu)
  - ⇒ Il organise scénarios, donc les attentes-besoins-objectifs des utilisateurs.
  - ⇒ En général : une page d'accueil, des rubriques, des pages par rubrique.
- Pour ça, on va :
  - ⇒ Catégoriser les scénarios : les regrouper par thèmes
  - ⇒ Structurer les scénarios : trouver les relations entre eux
  - ⇒ Anticiper la manière de chercher des utilisateurs
  - ⇒ Aider à la compréhension de l'arborescence par les utilisateurs : l'arborescence doit permettre que l'utilisateur trouve facilement ce qu'il veut.

Les wireframe

- Wireframe = maquette fonctionnelle = maquette « fil de fer »
- Le wireframe c'est la traduction en écran du scénario (ou du storyboard).
  - ⇒ Un wireframe est un schéma au graphisme simplifié de l'UI dessiné sur papier pour un scénario.
- Un bon Wireframe aide à visualiser :
  - ⇒ L'agencement de la page
  - ⇒ L'architecture de l'information
  - ⇒ Les parcours utilisateurs
  - ⇒ Les fonctionnalités essentielles.
- Un bon Wireframe va mettre en œuvre tout ce qui facilite l'utilisation (sera donc ergonomique).

Entrainement

- Concevez l'arborescence d'un site de magasin d'instrument de musique

4 : les tests utilisateurs

- L'objectif des tests est de valider les concepts, en termes d'interface et d'expérience utilisateur, avant de transmettre le projet aux développeurs.
- Les tests peuvent être faits soit sur des wireframes interactifs, soit par de la simulation avec des jeux de wireframe.
- Le but de l'ensemble du processus est d'identifier les besoins réels des utilisateurs.

Entrainement :

- Concevoir une application de réservation de repas en ligne.
  - ⇒ Quelle démarche adopter pour rendre l'application ergonomique ?
  - ⇒ En mettant l'utilisateur au centre de la conception d'un produit, on rend plus simple des fonctionnalités complexes.

Vocabulaire UX : persona

- Un personna est un utilisateur type pour un usage donné

  On lui donne : prénom, photo, métier, age (+ histoire, situation, contexte, comportement)

- On se dote de plusieurs pesonnas.
- On définit 2 types de persona :
  - ⇒ Les personnas primaires : utilisateur les plus fréquents.
  - ⇒ Les personas secondaires : utilisateur moins se sert moins souvent du système ou avec des exigences moindres.
- Les personas servent à :
  - ⇒ Synthétiser les entretiens "anthropo" avec les utilisateurs.
  - ⇒ Partager une vision commune au sein de l'équipe de conception.
  - ⇒ Générer de l'affect pour les utilisateurs (prénom, photo, histoire).
  - ⇒ Aider à la prise de décision dans la conception.

- Entrainement :
  - ⇒ Concevez les personas d'un site de magasin d'instrument de musique.

Vocabulaire UX : scénario = scénario d'expérience utilisateur = scénario d'usage = SU

- Un scénario d'usage (SU), c'est ce que vient faire l'utilisateur : il vient réaliser un objectif.
  ⇒ Connaitre les objectifs est donc crucial pour optimiser l'expérience utilisateur.
- Un bon SU est :
  ⇒ Court, pertinent, précis.
  ⇒ Il prend en compte le contexte.
  ⇒ Il répond à l'objectif.
- Le contexte, c'est
  ⇒ D'où vient l'utilisateur : d'expérience passée, du hasard, de recommandation.
  ⇒ Où est l'utilisateur : dans la rue, sur son canapé
    ⇒ Le contexte influence fortement le scénario d'usage.
- Un SU décrit le processus suivi par l'utilisateur en vue d'atteindre son objectif.
- L'ojectif est variable : on peut aller sur un site pour acheter ou pour comparer les prix.
  ⇒ Comparer n'est pas acheter.
  ⇒ Pour comparer il faut préciser des "variables" (taille, gamme de prix, marques, etc.)
- Ecrire un SU permet d'être au plus près des besoins réels des utilisateurs.

Entrainement :

- Pourquoi j'utilise un site ? Pourquoi suis-je là ? Quel est mon objectif ? Comment suis-je là ?
  ⇒ Répondre à ces questions c'est esquisser un premier SU.
  ⇒ On peut faire ça sur n'importe quel site.

Principes généraux d'ergonomie des applications numériques

UX = ergonomie = utilisabilité

- L'expérience utilisateur (UX) tend à remplacer les notions d'ergonomie ou d'utilisabilité du logiciel.
- Les 7 points suivants concernent l'ergonomie :
  - ⇒ 1 : Utilité et utilisabilité
  - ⇒ 2 : Trop de choix ou trop d'arborescence noie l'information
  - ⇒ 3 : Méthode experte et méthode participative
  - ⇒ 4 : Usage standard de navigation - lois de proximité et de similarité
  - ⇒ 5 : Améliorer l'affordance
  - ⇒ 6 : Conventions de localisation, de vocabulaire et de comportement
  - ⇒ 7 : Gérer les erreurs des utilisateurs

1 : Utilité et utilisabilité

- Un site doit être efficace, efficient et satisfaisant.

Objectif		**Caractère**	**Exemple : une librairie**
Utilité	Utile	Efficace	Trouver
Utilisabilité (ou usabilité)	Facile à utiliser	Efficient (capacité de rendement)	Trouver vite
	Agréable à utiliser	Satisfaisant (qui ouvre agréablement sur autre chose)	Flaner

➢ ***Entrainement***

- Sur tout site : analyser les caractéristiques d'utilité et d'usabilité.
  - ⇒ La difficulté, c'est que l'UX passe par une bonne maitrise des besoins de l'utilisateur.
  - ⇒ Exemple de site : paris librairies

2 : Trop de choix ou trop d'arborescence noie l'information

- C'est le design de l'UI qui induit le trajet de l'œil : on ne lit pas dans l'ordre.
  ⇒ L'important, c'est l'architecture de l'information
  ⇒ La difficulté, c'est toujours que ça passe par une bonne maitrise des besoins de l'utilisateur.
- Trop de choix ou trop d'arborescence noie l'information
  ⇒ Règle des 3 clics max (arborescence limitée)
  ⇒ Ne faut pas noyer l'information (limiter le choix)

3 : Méthode experte et méthode participative

- Méthode experte : on passe par l'expert = l'UX designer.
- Méthode participative : avis des utilisateurs.
  ⇒ La démarche ergonomique couple les méthodes expertes et participatives.

4 : Usage standard de navigation - lois de proximité et de similarité

- L'utilisateur utilise ses expériences précédentes pour comprendre le fonctionnement d'un site.
  ⇒ Le comportement d'un site doit dont être standard.
- Loi de proximité : ce qui est proche visuellement est considéré comme proche conceptuellement.
  ⇒ On doit donc rapprocher ce qui va ensemble.
- Loi de similarité : ce qui est similaire est considéré comme proche conceptuellement.
  ⇒ On doit donc représenter de la même façon ce qui va ensemble.

- **Entrainement** : sur tout site : analyser les proximités et les similarités.

5 : Améliorer l'affordance

- Une UI fournit des indices de la manière dont on peut l'utiliser.
- L'UI permet de savoir si c'est pour valider, pour aller quelque part ou pour faire quelque chose.
- Usages :
  - ⇒ Un texte souligné = un lien.
  - ⇒ Un cadre autour d'un texte centré = bouton. La couleur est vive.
  - ⇒ Un rectangle blanc = un champ de formulaire => animation au passage de la souris.
- **L'affordance** : c'est la capacité d'un objet à suggérer sa propre utilisation.
  - ⇒ L'affordance passe par un comportement standard.
  - ⇒ Il faut éviter les affordances erronées (un texte souligné non cliquable).

- **Entrainement** : sur tout site : cherchez les affordances erronées

6 : Conventions de localisation, de vocabulaire et de comportement

- Les utilisateurs apprennent des autres sites : ils prennent des habitudes.
  ⇒ Un site doit donc suivre les habitudes des autres sites : ce sont des conventions.
- Conventions de localisation :
  ⇒ En haut à gauche : logo, menu
  ⇒ En haut à drote : moteur de recherche, connexion, panier
- Convention de vocabulaire :
  ⇒ Accueil pour la page d'accueil
- Convention de comportement :
  ⇒ Logo qui ramène à l'accueil, confirmation, image cliquable, etc.

- **Entrainement** : sur tout site : réfléchissez à vos propres habitudes de navigation

7 : Gérer les erreurs des utilisateurs

- Faites-en sorte que l'utilisateur ne commette pas d'erreur.
- Il faut bien faire comprendre les erreurs : aider à les corriger.

- **Entrainement** : sur tout site : notez la façon dont les erreurs sont gérées

11. Agile et SCRUM (méthodes vivantes)

Généralités sur la méthode agile

- La méthode agile est une méthode incrémentale.
- Elle introduit la notion de **« User Story » (US) :**
  - ⇒ En tant que <qui>, je veux <quoi> afin de <pourquoi>
  - ⇒ https://www.atlassian.com/fr/agile/project-management/user-stories
- Il existe plusieurs méthodes agiles.
  - ⇒ https://fr.wikipedia.org/wiki/Méthode_agile

La méthode SCRUM : notion de « sprint »

- La méthode SCRUM est une méthode agile.
- Elle introduit la notion de **« sprint ».**
  - ⇒ https://fr.wikipedia.org/wiki/Scrum_(développement)

12. MERISE - partie datas (méthodes vivantes)

Les cycle d'abstraction de la méthode MERISE

- MERISE est une méthode systémique de conception de systèmes d'information
- La méthode MERISE, on définit le cycle d'abstraction, qui correspond à la partie systémique de la méthode.
- Le cycle d'abstraction est découpé en **4** niveaux d'abstraction : **conceptuel, organisationnel, logique et physique.**
- **L'analyse fonctionnelle** concerne les **niveaux conceptuel et organisationnel**. On y intègre les sous-systèmes et donc potentiellement les classes-métiers.
- **L'analyse technique** concerne les **niveaux logique** (choix des langages) **et physique** (réalisation concrète).
- De plus, dans **MERISE**, il y a une division entre les données et les traitements.
  - ⇒ **La partie sur les données reste « vivante »** avec les **MCD, MLD et MPD.**
  - ⇒ La partie sur les **traitements** est au **cimetière** !

Tableau synthétique du cycle d'abstraction de la méthode MERISE

		Données	**Traitements (cimetière)**
SIO Système d'information organisationnel ANALYSE FONCTIONNELLE	Niveau conceptuel	**M C D** Modèle conceptuel des données Signification des informations sans contraintes techniques, organisationnelle ou économique	**M C T** Modèle conceptuel des traitements Activité du domaine sans préciser les ressources et leur organisation
	Niveau organisa-tionnel	**M O D** Modèle organisationnel des données Signification des informations avec contraintes organisationnelles et économiques	**M O T** Modèle organisationnel des traitements Fonctionnement du domaine avec les ressources utilisées et leur organisation
SII Système d'information informatisé ANALYSE TECHNIQUE	Niveau logique	**M L D** Modèle logique des données Description des données tenant compte de leurs conditions d'utilisation par les traitements et des techniques de mémorisation	**M L T** Modèle logique des traitements Fonctionnement du domaine avec les ressources et leur organisation informatique
	Niveau physique	**M P D** Modèle physique des données Description de la (ou des) base(s) de données dans la syntaxe du SG de données (SGF ou SGBD)	**M P T** Modèle physique des traitements Architecture technique des programmes

ISIM, p. 37

13. Méthodes mortes : le cimetière des méthodes !

MERISE - partie traitement

- MERISE est une méthode systémique de conception de systèmes d'information
- Dans **MERISE**, il y a une division entre les données et les traitements.
  - ⇒ **La partie sur les données reste « vivante »** avec les **MCD, MLD et MPD.**
  - ⇒ La partie sur les traitements est au cimetière !

Ingénierie des systèmes d'information *vs* Génie Logiciel : théorique

- La distinction entre l'ingénierie des systèmes d'information et le génie logiciel correspond à celle entre le fonctionnel et l'organique (le technique).

Génie logiciel - Software

- Le terme de génie logiciel (*software engineering*) est né en Europe à la fin des années 60.
- Le G.L. regroupe l'ensemble des
  - ⇒ Méthodes (organisation du travail)
  - ⇒ Techniques (langages de programmation, documentation des programmes)
  - ⇒ Outils de développement du logiciel (compilateurs, systèmes de gestion de la documentation)
- Le G.L vise à transformer les besoins et attentes des utilisateurs en une application informatique.
- Besoins et attentes ⟶ Application informatique
- Quoi : software
- **Qui** : les informaticiens.

Ingénierie des Systèmes d' Information - Brainware

- Le terme d'ingénierie des systèmes d'information (*requirement engineering*) est né au début des années 90.
- **L' I.S.I.** vise à transformer les besoins et attentes des utilisateurs en spécifications formalisées d'une future application informatique.
- Besoins et attentes ⟶ Spécifications formalisées
- Quoi : brainware
- **Qui** : les informaticiens, les gestionnaires et les autres utilisateurs du système d'information

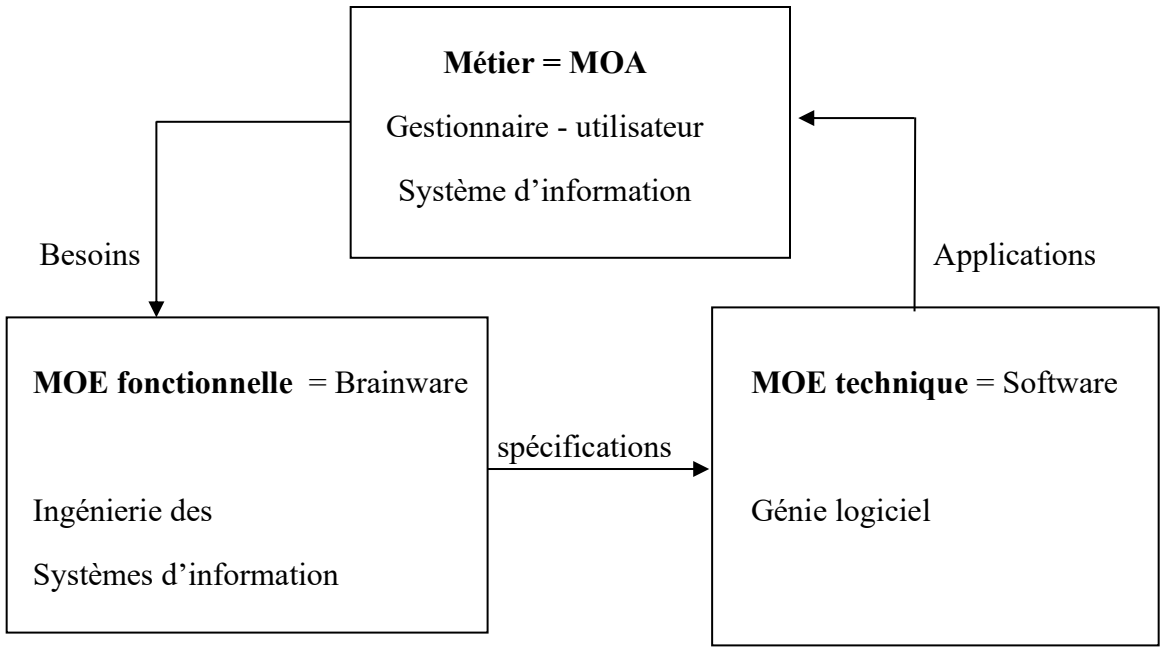
- L' I.S.I. regroupe des méthode, des techniques et des outils utilisés pour le développement des spécifications :
  - ⇒ Des **METHODES** d'organisation du travail de spécification : MERISE, RUP, Cycle en V, Méthode itérative, Méthode spécifique d'une entreprise, etc.
  - ⇒ Des **TECHNIQUES** de modélisations : MEA, Diagramme des flux, MCT, UML, etc.

⇒ Des **OUTILS** spécification, de modélisation, de génération de code, ateliers de génie logiciel : Win'Design, power AMC, IBM-Rational Rose, Visio, etc. Et, au niveau des bases de données : Oracle Modeler, MySQL workbench, etc.

Le brainware

Le concept de brainware, très peu usité, a été introduit par Tosio Kitagawa en septembre 1974 dans le n°39 des Research Report of Research Institute of Fundamental Information Science.

Relations entre software engineering et brainware engineering



Critiques

C'est une autre approche du cycle en V, avec ses avantages et ses défauts.

Le cycle en Y : cimetière

- Le cycle en Y traite en même temps la partie fonctionnelle et l'architecture technique (les 2 bras du Y) puis passe à la conception détaillée : le pied du Y.
- La bonne idée est d'introduire l'architecture dans le fonctionnelle.
- Les limites sont dans la non prise en compte d'aspects itératifs et incrémentaux, et donc de remise en cause du fonctionnel.

La spirale de Boehm, méthode itérative et incrémentale : cimetière

- Une itération c'est une répétition ou l'action qui est répétée.
- Un incrément, c'est une quantité ajoutée à chaque itération (au sens d'action répétée).
- Les méthodes itératives et incrémentales reproduisent le cycle en V à chaque itération.
- Chaque itération propose un incrément du produit final : on peut dont rapidement proposer une première version du produit final (un incrément).

La méthode RUP, méthode itérative et incrémentale : cimetière

Le méthode RUP

- La méthode RUP (Rational Unified Process) est une méthode adaptée à la programmation objet qui est née avec l'UML.
- Elle est aujourd'hui détrônée par les méthodes agiles.

Le cycle est articulé en 4 phases

- Le RUP répète des **cycles** constituant la vie du système et donnant naissance à une **version**. Tout cycle se conclut par la livraison d'une version du produit.
- Un cycle se divise en **phases**.
  - ⇒ **La phase de création** (étude préalable, étude d'opportunité) : elle pousse l'étude de rentabilité jusqu'au stade propre à lancer le projet. Il s'agit d'obtenir l'accord de toutes les parties concernées.
  - ⇒ **La phase d'élaboration :** elle finalise l'analyse des d'utilisation et met en place les fondations de l'architecture.
  - ⇒ **La phase de construction** (développement) : elle complète la description des cas d'utilisation, elle modifie l'architecture si nécessaire, implémente les sous-systèmes de l'incrément et les teste, puis les intègre au système et le teste.
  - ⇒ **La phase de transition :** elle finalise l'installation de l'incrément en intégrant les conditions du fonctionnement opérationnel et en validant la satisfaction du client.

Il existe 6 tâches réparties sur les 4 phases

- Le RUP définit des tâches à répartir sur les phases.
  1. **La modélisation par métier :** elle a pour principal objectif le développement d'un modèle du métier, constitué d'un modèle de cas d'utilisation métier et d'un modèle objet métier (soit à peu près un MCT et un MCD).
  2. **L'étude des besoins (fonctionnel) :** elle a pour principal objectif de définir une vision du système par un modèle des cas d'utilisation qui constituera le cahier des charges logiciel, et de construire un prototype de l'interface utilisateur.

3. **L'analyse et conception (organique) :** son objectif est de traduire l'ensemble des exigences en une spécification qui décrit comment réaliser le logiciel.
4. L'implémentation
5. Les tests
6. Le déploiement

Notion d'itérations et incréments

- Une **phase** divise en **itérations**. **L'itération** désigne une étape de l'enchaînement des **activités**.
- **L'incrément** désigne un **stade de développement** du projet dans sa globalité.

Bibliographie

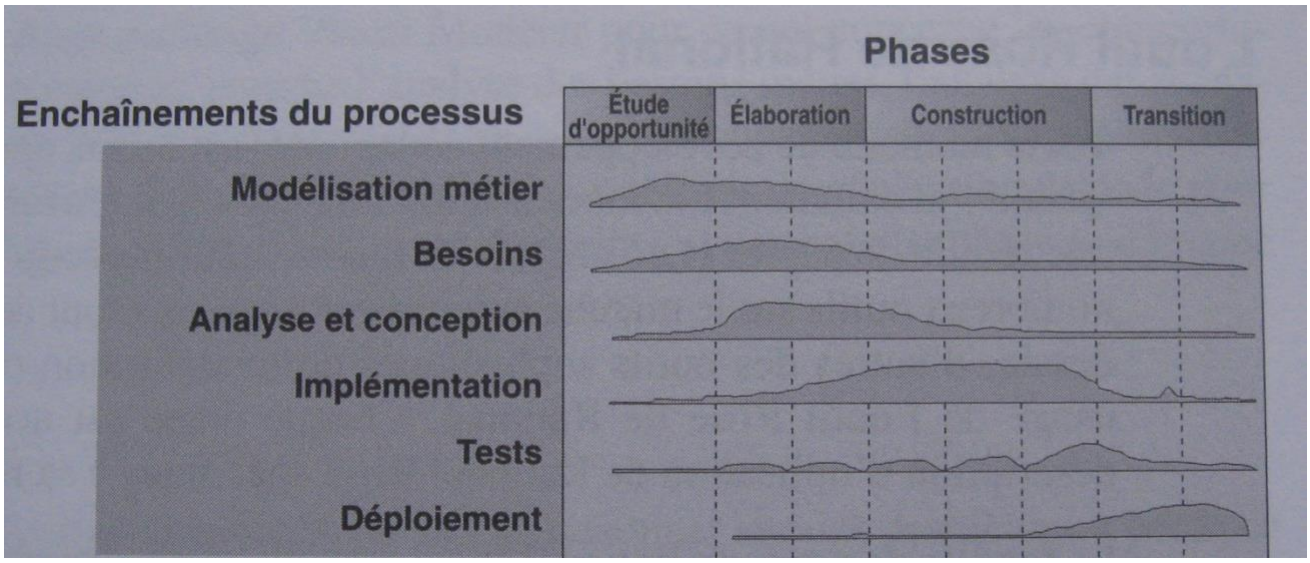
- **Introduction au RUP,** Philippe Kruchen, Eyrolles 2000.

  Le RUP vu par Rational Rose, l'éditeur de référence de l'outil de modélisation UML (racheté par IBM).

- **Le processus unifié de développement logiciel**, Grady Booch, Ivan Jacobson et James Rumbaugh, Eyrolles, 1999 (RUP)

  La bible du RUP

Répartition temporelle



Les 6 tâches peuvent se retrouver à chaque phase, même si certaines phases sont plus dédiées à certaines tâches. Ainsi : en étude préalable, on peut coder un prototype pour valider la faisabilité. Pendant la phase d'élaboration, on peut coder quelques éléments. Pendant le codage, on peut revenir à de l'élaboration, etc.

Critique

La méthode reste trop rigide et adaptée aux « gros projets ».