

# INTRODUCTION AU GENIE LOGICIEL

Objectif Général: Ce cours vise à présenter les grands enjeux et les bonnes pratiques (méthodes, outils) liés à l'activité de production des logiciels.

## INTRODUCTION

A la même vitesse que la taille et la complexité des systèmes informatiques s'accroissent, les exigences et attentes des organismes à l'égard des logiciels deviennent problématiques. En effet, l'intérêt apporté par les premiers programmes malheureusement écrits par des programmeurs travaillant seuls et pour leur propre utilisation, motiva les experts à faire de la production logiciel non plus seulement une affaire pour les programmeurs isolés mais d'équipe. Ainsi, avec l'augmentation de la complexité des projets logiciels, les équipes aussi ont grandi vite et un nouveau besoin a fait surface : l'amélioration de la gestion de ces projets informatiques. Cependant, en plus du coût très élevé de production logiciel par rapport au coût du matériel, s'ajoute le déclin de la qualité des produits logiciels et les échecs spectaculaires de beaucoup de projets, l'on assiste donc dans les années 1960 à ce qui s'est appelé la ***crise logiciel***. En réponse donc aux 2 problématiques engendrées par les logiciels à savoir :

- ✓ Comment produire des logiciels (*Cycle de vie*) ?
- ✓ Qu'attend-on réellement d'un logiciel (*critère de qualité*)?

est né lors de la Conférence de l'OTAN 1968 (Organisation de traité de l'Atlantique Nord) à Garmish en Allemagne la discipline du « Génie logiciel » ou « Software Engineering ». Rappelons que cette expression fût prononcée par la mathématicienne et informaticienne Margaret Hamilton auteur du système embarqué lié au programme Apollo.

Le génie logiciel est ainsi considéré comme un domaine scientifique de l'ingénierie axé sur l'ensemble des activités visant à la rationalisation, la production et le suivi du logiciel jusqu'à sa disparition. Cette discipline est une réponse aux défis posés par la complexification des logiciels et de l'activité qui vise à les produire.

Ce cours aidera l'apprenant à identifier et à utiliser des **méthodes**, des **pratiques** et des **outils** permettant de maximiser les chances de réussite d'un projet logiciel. Il s'agira donc pour l'étudiant :

- De comprendre la discipline du génie logiciel sous son aspect scientifique et artistique
- De maîtriser les enjeux visés par chacune des phases du cycle de vie du logiciel
- De pouvoir appliquer les méthodes et outils pour répondre à ces enjeux.

## I- CONCEPTS DU GENIE LOGICIEL

### 1- Logiciel

Le dictionnaire Larousse français définit le logiciel comme un ensemble de programme, procédés et de la documentation permettant le fonctionnement d'un ensemble de traitement automatique de l'information. Autrement dit, un logiciel est un ensemble d'entités nécessaires au fonctionnement d'un processus de traitement automatique de l'information (*Conf Génie Logiciel Avancé, Stefano Zacchiroli Laboratoire PPS, Février 2011*). Parmi ces entités, on retrouve:


- des programmes (en format *code source* ou exécutables) ;
- la documentation (*des informations d'installation, de configuration ou d'utilisation*);
- des contrôleurs matériels (*opérations d'interactions avec le matériel*),


Un logiciel peut donc être vu comme une partie intégrante d'un système informatique devant interagir avec d'autres entités.

Les logiciels en effet déterminent les tâches précises que peuvent effectuer la machine pour lui procurer son utilité fonctionnelle. Les séquences d'instructions qu'accomplissent de façon unique chacune de ces tâches sont appelées programmes et sont ordinairement structurées en fichiers. Donc : **Logiciel = programme + documentation**

#### a- Catégories de logiciels

En pratique on distingue deux principales catégories de logiciels : **les logiciels applicatifs et les logiciels système.**

 **les logiciels système** destinés à effectuer des opérations viables en rapport immédiat avec le matériel informatique. Il est interface de communication entre le matériel et les autres logiciels ; Il facilite l'interdépendance entre les logiciels applicatifs vis-à-vis du matériel ; Il donne au matériel son utilité fonctionnelle. *Exple : les OS ; les logiciels embarqués (bios)...*

 **les logiciels applicatifs** destiné à aider les utilisateurs à effectuer certaine tâche complémentaires à celles qu'offrent les logiciels systèmes. On y retrouve :

- les logiciels utilitaires : outils d'administration systèmes, outils de sécurités (antivirus), les outils de bureautiques...
- les outils de développement d'application

## b- les types de logiciels




- ✓ Logiciels commerciaux ou propriétaire
- ✓ Les sharewares qui sont des logiciels d'essais mis gratuitement à la disposition des utilisateurs pendant une durée définie appelé période d'essai mais payant pour une utilisation régulière après la période d'essai.
- ✓ Les freewares mis à disposition gratuitement et peuvent être librement utilisés sans contribution
- ✓ Les logiciels libres (Opensources) qui sont fournis avec leurs codes sources pour d'éventuelle modification par le public.

## 2- Le Génie logiciel

Dans chaque métier de l'ingénierie, s'appliquent des principes, des méthodes, des outils dont la mise en commun crée un consommable satisfaisant à sa cible. Le **génie logiciel** (software engineering) constitue ce métier de l'informatique où sont appliqués des principes d'ingénierie nécessaires à la création ou production des logiciels. C'est dans ce sens que *Stefano Zacchioli (Conf Génie Logiciel Avancé Laboratoire PPS, Février 2011)* définit Le **Génie logiciel** comme un domaine des sciences de l'ingénieur dont l'objet d'étude est la conception, la fabrication et la maintenance des systèmes informatiques. Autrement dit, *Le génie logiciel* est l'ensemble des activités de conception et de mise en oeuvre des produits et des procédures tendant à rationaliser la production du logiciel et son suivi (*Source : Wikipédia*).

## 3- Enjeux du Génie logiciel

Le génie logiciel vise à rationaliser et à optimiser le processus de production d'un logiciel. Son enjeu principal est de définir et résoudre les problèmes posés par le développement de systèmes complexes, c'est-à-dire nécessitant des moyens humains importants et dont le déroulement est relativement long. Pour y parvenir le Génie logiciel:

-  Repose sur un ensemble de principes et de normes
-  Définit une démarche ou des processus mettant en communs ses métiers
-  fournit des méthodes de conception des systèmes pour
  - Une adéquation aux des besoins du client
  - Une démarche qualité (*maximisation des performances, de la fiabilité...*)

- Une organisation du travail en équipe soumise à des exigences contradictoires et difficilement conciliables (triangle [coût-délai-qualité](#))

✚ Le GL fournit des outils et une meilleure utilisation à l'instar

- Des outils de conception architecturale ( framework...)
- Des outils de modélisation, de développement

## II- CRITERES DE QUALITE D'UN LOGICIEL

La **qualité** est l'ensemble des caractéristiques intrinsèques d'une entité qui lui confèrent l'aptitude à satisfaire des besoins exprimés par des futurs utilisateurs (AFNOR Juillet 1982). Pour qu'un logiciel soit de qualité, celui-ci doit intégrer des caractéristiques bien définies afin de répondre aux attentes du client.

### 1- Critères externes : front office (côtés utilisateur)

✚ **La Conformité ou Validité** : c'est la capacité du logiciel à répondre aux besoins des utilisateurs tels que les conventions préétablies soient respectées.

✚ **La convivialité** c'est l'aisance avec laquelle des personnes présentant des formations, des compétences et des anatomies différentes pourront apprendre à utiliser le logiciel. D'une manière générale, elle renvoie à l'ergonomie visant à l'adaptation des machines et logiciel à l'homme (dans tous ses aspects physiologique, psychologique ...) pour une utilisation avec un maximum de confort et de sécurité. Elle recouvre également la facilité d'installation, d'opération et de contrôle.

✚ **Sécurité**: intégrité des données et gestion des droits d'accès ou privilèges

✚ **Fiabilité** : tolérance aux pannes ou à certains manquements. Autrement dit, le logiciel doit être capable d'assurer la continuité dans les services qu'elle vient rendre.

✚ **Performance** : le logiciel doit répondre dans les délais courts aux requêtes, avoir un bon débit, fluidité...

✚ **Efficacité** capacité qu'aura le logiciel à remplir ou à retourner le résultat exact des tâches lui ayant donné naissance avec une utilisation minimales de ressources avec lesquelles il doit interagir.

## 2- Critères interne (Côté concepteur) : back office

✚ **Modularité** : Aptitude d'un logiciel à être structuré en composants ou modules indépendants. Ceci revient à juger pertinence de la fonction de chaque module et de ses interactions avec les autres modules.

✚ **Portabilité** : facilité avec laquelle des produits logiciels peuvent être transférés d'un environnement logiciel ou matériel à l'autre.

✚ **L'Interopérabilité** c'est la capacité pour un logiciel à pouvoir interagir avec d'autres entités partageant le même environnement.

✚ **Robustesse** capacité qu'offrent des systèmes logiciels à réagir de manière appropriée lorsqu'une condition anormale (*fausse manipulation de la part de l'utilisateur*) survient.

✚ **La réutilisabilité** est la capacité des éléments logiciels à servir à la construction de plusieurs autres applications différentes.

✚ **L'extensibilité ou flexibilité** est la facilité d'**adaptation** des produits logiciels aux changements de spécifications.

✚ **Lisibilité** ceci renvoi à la clarté, c'est-à-dire l'organisation donnée au code source afin qu'il soit compréhensible et facilement modifiable.

## III- PRINCIPES DU GENIE LOGICIEL

Les principes et techniques du génie logiciel s'applique mieux au projet de grandes tailles (regroupant plusieurs participants, devant fournir plusieurs version et généralement de longue durée). Ceci met en évidence les différences claires entre la programmation et le génie logiciel :

- ✓ Programmation = activité personnelle
- ✓ Génie logiciel = activité d'équipe

Bien que les principes du GL s'appliquent selon la méthode choisit cette partie liste sept principes proposés par Carlo Ghezzi dont 5 fondamentaux car couramment appliqués : **rigueur**, **séparation des problèmes ou structuration** (« separation of concerns »), **modularité**, **abstraction**, **généricité**, construction incrémentale, anticipation du changement.

### 1- La rigueur

Partant du constat fait que « les principales sources de défaillances d'un logiciel sont d'origine humaine », la production logiciel est une activité créative qui impose d'être conduite avec une certaine rigueur. En effet, le résultat d'une activité de création pure doit être rigoureusement évalué, avec des critères précis. Le niveau maximum de rigueur est la *formalité*, c'est à dire le moment où les descriptions et les validations s'appuient sur des notations et lois mathématiques. Sachant qu'il faut construire une description formelle (pas du tout évidente) à partir de connaissances non formalisées, il existe certaines techniques formelles utiles.

Ce principe important du génie logiciel renvoi :

- ✓ Au suivi (vérification validation) des activités liées au processus adapté. Des outils de vérification accompagnant le développement peuvent aider à réduire les erreurs. Cette famille d'outils s'appelle CASE (*Computer Aided Software Engineering*)
- ✓ A l'utilisation correcte des techniques et normes adaptées
- ✓ La fourniture des livrables prévus (CDC, modèles, guide, code...)
- ✓ L'attitude professionnelle en équipe

Ingénieur = rigueur + précision

## 2- Généricité

Il est parfois avantageux de remplacer la résolution d'un problème spécifique par la résolution d'un problème plus général. Encore appelé généralisation ou héritage, elle stipule le regroupement d'un ensemble de fonctionnalités semblables en une seule fonctionnalité paramétrable ou adaptable. Ce principe aide à rendre le système modulable et réutilisable pour plus de performance. Un logiciel est *générique* lorsqu'il est facilement adaptable.

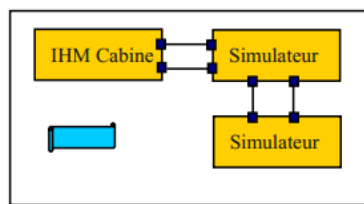
## 3- Structuration

C'est le principe général du « diviser pour mieux régner » qui vise à décomposer et organiser le futur logiciel sous ses aspects structurels, conceptuels et technique afin de le rendre extensible. Il concerne d'abord la structure générale d'un logiciel, puis sa subdivision en unités plus petites. Nous pouvons noter :

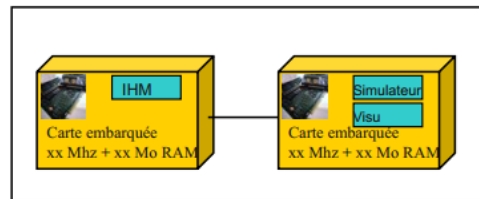
- Séparation des activités et ressources dans le temps avec la notion de cycle de vie du logiciel et les outils de planification ou de découpage du projet ;
- Séparation des qualités (fonctionnelles et non fonctionnelles) que l'on cherchera toujours à optimiser

- séparation du système en parties (un noyau, des extensions, ...) ;
- Séparation des vues que l'on peut avoir : Vue logique, dynamique ou physique. Il décrit une organisation de classes fréquemment utilisée pour résoudre un problème récurrent. Les patrons de conception ou d'architecture (design pattern) aident donc à ce principe car suggère un arrangement, une manière d'organiser des modules.

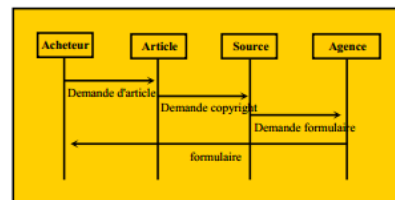
- Vue logique
- Vue dynamique
- Vue physique



Vue logique



Vue physique



Vue dynamique

#### 4- Abstraction

Mécanisme qui consiste à présenter un contexte en exprimant les éléments pertinents et en omettant ceux qui ne le sont pas. C'est encore une instance du principe de décomposition des problèmes. Il s'agit d'exhiber des concepts généraux regroupant un certain nombre de cas particuliers et de raisonner sur ces concepts généraux plutôt que sur chacun des cas particuliers. Une même réalité peut souvent être décrite à **différents niveaux d'abstraction** il est donc important de fixer le bon niveau de **granularité du détail** pour :

- ✓ pouvoir raisonner plus efficacement ;
- ✓ factoriser le travail en instanciant le raisonnement général sur chaque cas particulier.

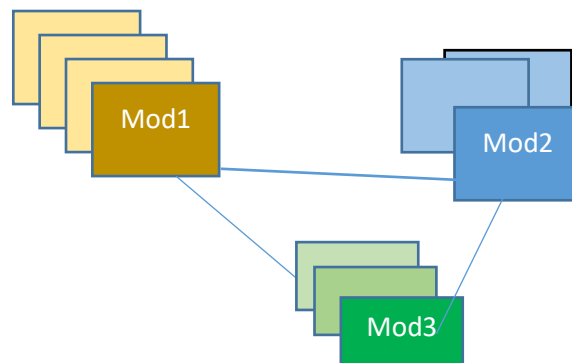
**Niveau de granularité** = Niveau de détails le plus fin contenus dans une unité d'information. Plus il y a de détails, plus bas sera le niveau de la granularité, (*je cherche jusqu'où je peux aller*).

**Niveau d'abstraction** = je sais jusqu'où je veux aller sans trop chercher à savoir « jusqu'où je peux aller » (*il est inclus dans le Niveau de granularité*)

## 5- La modularité :

La modularité est une propriété importante de tous les procédés et produits industriels. C'est une instance cruciale du principe de décomposition des problèmes. Il s'agit de partitionner le système initiale en **modules de moindres complexités tel que** :

- Un module vient aider à résoudre un sous problème du problème initial.
  - Chaque module est constitué de fonctionnalités paramétrables traitant d'une partie du problème avec une cohérence interne.
  - Ils soient compréhensible, indépendants les uns des autres de part leurs fonctionnalités.
  - Ils sont reliés entre-eux possèdent une **interface** ne divulguant sur le contenu du module que ce qui est strictement nécessaire aux modules clients. L'évolution de l'interface est **indépendante** de celle de l'implémentation du module.
- ✓ Les choix d'implémentation sont **indépendants** de l'utilisation du module.



## IV- LES PROCESSUS DU GL

### A- CONCEPTS GENERAUX

#### 1- Notion de Conduite de projet

##### a- Définitions

**Un projet** est un ensemble d'activités envisagés ou entrepris dans le but d'atteindre un objectif précis qui soit conforme à des exigences spécifiques, telles que les contraintes de délais, de coûts et de ressources (**Afnor X50-115**).



La **gestion de projet** est la mise en œuvre des activités liées au projet, mobilisant des ressources humaines et matérielles possédant un coût prévisionnel afin de produire les résultats espérés en réponse aux objectifs clairement définis dans des délais fixés (date début et date de fin). Plus simplement la **gestion de projet** est une démarche visant à organiser de bout en bout le bon déroulement d'un **projet**.

### **b- Caractéristiques du projet**

Un projet est défini par : son objectif ; Sa singularité (innovation apportée) ; Sa cible ou bénéficiaires ; Des exigences qui se veulent conciliables et équilibrées (coûts-délais-qualité).

### **c- Principaux acteurs d'un projet**

Le MOA : La maîtrise d'ouvrage est constituée de toute personne physique ou morale propriétaire de l'ouvrage. Elle représente les utilisateurs finaux à qui l'ouvrage est destiné. Elle est chargée d'exprimer leurs besoins, prédefinisant les objectifs, et les délais de réalisation. Cependant, elle n'a généralement aucune idée sur les contraintes techniques qu'elle impose aux développeurs.

Le MOE : La maîtrise d'œuvre est l'entité retenue par la MOA en qualité de chef de projet qui sera responsable des choix techniques. Le Maître d'œuvre est le garant du respect des engagements pris notamment sur les délais et les contenus des livrables. Il définit clairement les objectifs, le budget et les délais de réalisation.

La sous-traitance : réalise un sous-ensemble du projet en lien avec la maîtrise d'œuvre, surtout lorsque celle-ci ne possède pas les ressources internes nécessaires.

## **2- Notion de Processus**

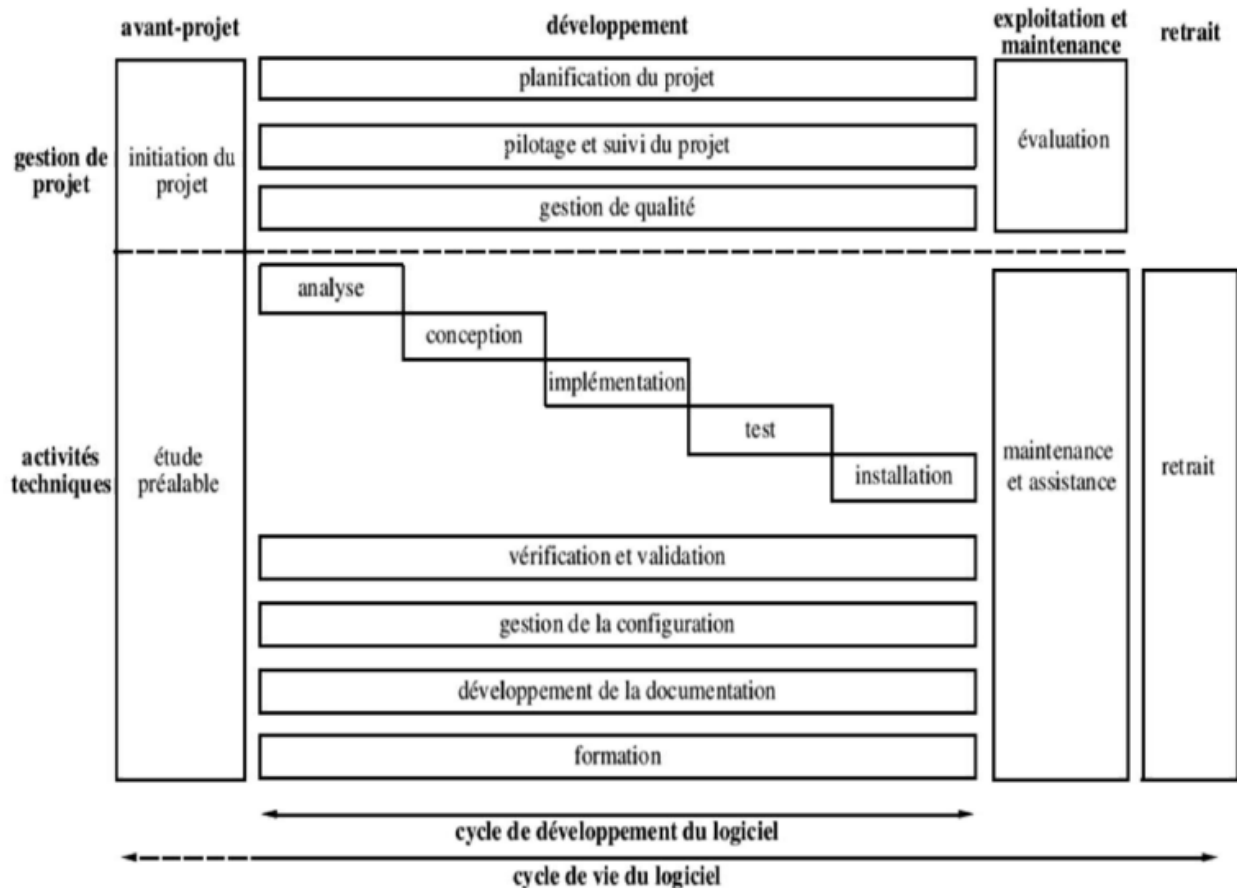
Un processus est un ensemble structuré de tâches, d'acteurs (codage, planification, ...) ; d'artefacts (CDC, exécutable, planning...) et de workflows tous mis en commun en vue de réaliser une activité.

Un processus regroupe un ensemble d'étapes centrées sur des activités :

- Séquentielles : Définition des besoins- Analyse- Conception- Implémentation- Validation et tests- Déploiement

- Permanentes (horizontales) : Produits intermédiaires ; Plan d'assurance qualité et documentation ; Gestion de projet et des ressources
- Permettant d'assurer le cycle de vie du logiciel

Tous les processus du GL s'organisent autour des 05 étapes du cycle de développement du logiciel : Analyse (des besoins et du système)- Conception- Implémentation- test et Déploiement.



## B- LES DIFFERENTES ETAPES D'UN PROCESSUS DU GL

### 1- L'analyse (des besoins et du système)

**Objectifs** : Comprendre les besoins du client « **Quoi ?** », Établir une description claire de ce que doit faire le logiciel « **Que fait le système ?** ».

#### a- L'analyse des besoins

#### Spécifications

- Recueil des besoins du client
- Abstraction et séparation des problèmes (informations manipulées, acteurs, types de besoins, architecture...)
- Décomposition modulaire
- Ressortir les trois contraintes de coût- délais- qualités :
  - *qualités fonctionnelles attendues en termes des services offerts : efficacité, fiabilité*
  - *qualités non fonctionnelles attendues : sécurité, convivialité (facilité d'utilisation,...)*
  - *qualités attendues du procédé de développement (ex : procédures de contrôle qualité).*

### **Sorties :**

- Cahier des charges techniques découlant du cahier des charges commerciales,
- plan d'assurance qualité.

### **Remarques :**

*Une fonctionnalité* est une action concrète rendue par le système. Elles sont mises en exergue par le diagramme des cas d'utilisation ou chacun des cas d'utilisation représente une fonctionnalité dans le système. Ces différents cas d'utilisation sont identifier à partir des besoins exprimés de façon implicite ou explicite. Ainsi l'on distingue :

- *les besoins fonctionnels encore appelés besoins utilisateur* sont les fonctionnalités du système venant directement répondre à une exigence exprimées par le client et dont l'exécution aidera à résoudre un sous problème identifié. Les besoins fonctionnelles induisent les **qualités fonctionnelles** attendues en termes de service offert.
- *les besoins non fonctionnels encore appelés besoins systèmes* sont des fonctionnalités sous-adjacentes venant aider à l'exécution optimales des besoins utilisateurs. Celles-ci bornent donc les besoins utilisateurs et sont nécessaires au bon fonctionnement du système même si généralement non accessible par les utilisateurs. Ceux-ci induisent les qualités non fonctionnelles.

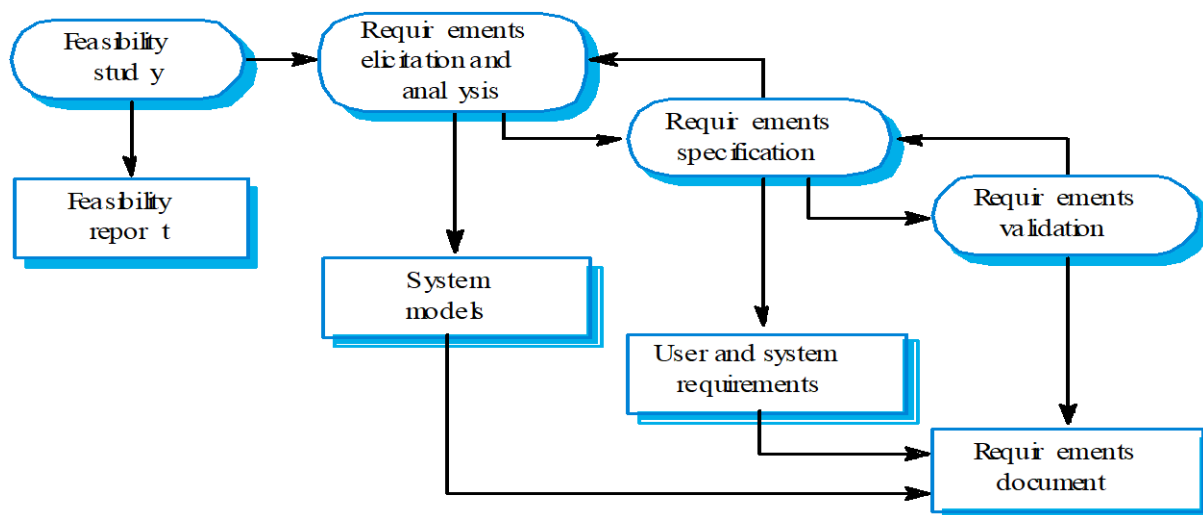
### **b- L'analyse du système**

### **Spécifications**

- Modélisation du domaine,
- modélisation de l'existant (éventuellement),
- définition d'un modèle conceptuel (ou spécification conceptuelle),

#### Sorties :

- Dossier d'analyse ;
- plan de validation.



#### Exemple d'Application de gestion d'une bibliothèque numérique:

Dans le cadre d'un projet portant sur la « Mise en place d'une application de gestion de sa bibliothèque numérique », l'entreprise « Soft-kipeu » fait appel à votre expertise et en voici une exigence extraite du CCC ; ***Exigence du client : l'abonné peut emprunter des ouvrages pour consultation mais ne peut procéder à une impression que s'il a payé les droits d'auteur du dit ouvrage.***

#### TAF :

- 1) Identifier 03 sous-problèmes pouvant ressortir de l'étude d'un tel projet. Formuler chaque sous-problèmes en modules en précisant pour chacun 3 fonctionnalités.
- 2) Etablir un questionnaire de 5 à 10 questions nécessaire à procéder à une spécification de cette exigence du client.
- 3) Spécifier les besoins du client en terme de qualité fonctionnelles et non fonctionnelles

#### 2) La conception

**Objectifs :** faire une proposition de solution au problème en Répondant au « **Comment faire le système ?** ».

### Activités :

- Organiser les modules de l'application et faire une description détaillée de ses modules avec les *algorithmes essentiels*
- Ressortir les différentes règles de gestions
- organisation de l'application en *modules et interface des modules*
- Définir l'architecture logique du logiciel
- Etablir une architecture physique du système
- Etablir une description claire de chaque constituant du logiciel (informations traitées, traitements effectués, résultats fournis.
- structuration des données.

**Sorties** : dossier de conception + plan de test global et par module.

### 3) Implémentation

**Objectif** : Produire un programme pour l'exécution automatique des tâches

#### **Spécifications**

- Choix de l'environnement de développement
- Traduction de l'algorithme dans un langage de programmation
- Définition des normes de développement,

**Sortie** : codes sources + dossiers de programmation décrivant la démarche de résolution du problème.

### 4) Les Tests

**Objectif** : Livrer un produit consommable borné par des activités de **Vérification et de validation**.

#### **Spécifications**

- Effectuer *des tests systèmes* pour assurer l'*activité de verification, « are we building the product right? »* (construisons-nous le produit correctement ?). Autrement dit, l'on s'assure de construire un *système conforme aux spécifications et aux normes*. Nous effectuons généralement:
  - Les tests unitaires (alpha testing) qui consiste à effectuer des cas de test sur chacun des modules indépendamment les uns des autres.
  - Tests d'intégrations : regroupements des modules et procéder à des test en vraie grandeur du système complet selon le plan de test global ('beta testing').

- Effectuer *des tests d'acceptations* pour assurer l'*activité de validation* « *are we building the right product ?* » (construisons-nous le bon produit ?) permettant de s'assurer que *les résultats produits par système répondent aux exigences du client*.

### Notes

- Au cours de la phase de test du système, le système logiciel est intégré avec d'autres systèmes et testé contre les exigences du logiciel / système. Les tests système sont habituellement effectués dans l'environnement de développement. Ainsi, des corrections internes du logiciel sont effectuées et ne concernent que les développeurs.
- Par contre lors de l'*activité de validation* : « *are we building the right product* » (construisons-nous le bon produit ?) l'on procède à une adaptation du logiciel vis à vis des besoins des utilisateurs, et concerne les utilisateurs).

*Le produit issu du test système est un système qui est prêt pour le déploiement et du test d'acceptation dans l'environnement cible du client. Pendant la phase de test d'acceptation, un analyste ou un consultant embauché par le client effectuera le test du système dans l'environnement cible du client pour veiller à ce que le système fonctionne correctement dans cet environnement.*

### Sortie :

- Scénario de test : un document spécifiant les séquences d'actions effectuées lors de l'exécution d'un cas de test.
- Stratégie de test : un document de haut niveau définissant, pour un programme, les tests à exécuter (pour un ou plusieurs projets similaires).

## **5) Le déploiement**

**Objectif** : Mettre le logiciel en production

### **Spécifications** :

- Rassembler les différents équipements (matériels u logiciel)
- Configurer et mettre en communication les équipements

**Sortie** : Guide d'installation et guide d'utilisation.

## **6) Maintenance**

**Objectif** : Assurer la continuité dans les services

**Spécifications** : Elle peut être

- Corrective : identifier et corriger des erreurs trouvées après la livraison
- Adaptative ou Evolutive: adapter le logiciel aux changements dans l'environnement (format des données, environnement d'exécution...) ou aux nouvelles technologies.
- Préventive : apporter des modifications visant à anticiper sur le comportement futur et estimer pertinent du système
- Perfective : améliorer la performance, ajouter des fonctionnalités,.....

**Sortie** : Rapport de maintenance

## **V- CYCLES DE VIE DU LOGICIEL**

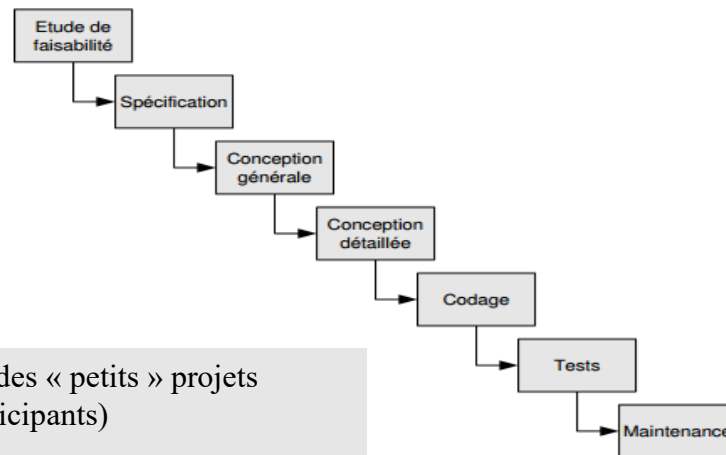
Le cycle de vie d'un logiciel est un ordonnancement des différentes étapes du processus développement allant des faisabilités jusqu'à la maintenance et la substitution du logiciel.

Comme pour toutes les fabrications, il est important d'avoir un procédé de fabrication du logiciel bien défini et explicitement décrit et documenté selon la pertinence du projet. Les modèles de cycle de vie du logiciel décrivent à un niveau très abstrait et idéalisé les différentes manières d'organiser la production.

### **A- LES MODELES LINEAIRES**

#### **1- Le cycle de vie en Cascade**

Même si elle n'est pas réaliste, cette représentation très simplifiée a permis de définir des cadres conceptuels (définition des différentes phases - cf. Fig. 3.4) et terminologiques, largement acceptés et normalisés par plusieurs organismes (ISO, AFNOR, IEEE,.. pour les applications militaires aux USA, ESA, etc.). Ceci facilite la gestion et le suivi des projets.



Peut être viable pour des « petits » projets  
(Taille + Nbre de participants)

- **Adapté pour des projets de petite taille, et dont le domaine est bien maîtrisé**

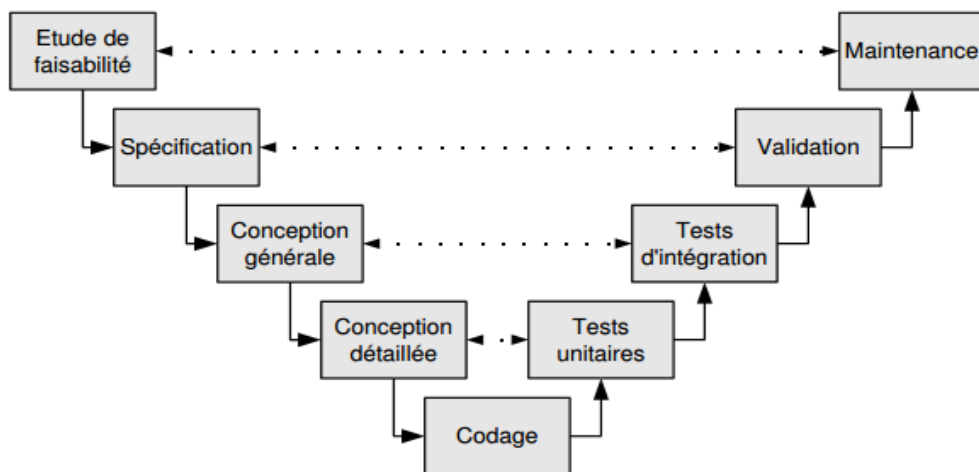
Principe :

- Pas de validation intermédiaire
  - Les résultats sont définis sur la base des interactions entre étapes et activités, ils sont soumis à une revue approfondie (on ne passe à la phase suivante que s'ils sont jugés satisfaisants)
- ❑ Limite : Haut risque : erreurs coûteuses ! ne comporte pas de possibilité de retour en arrière.

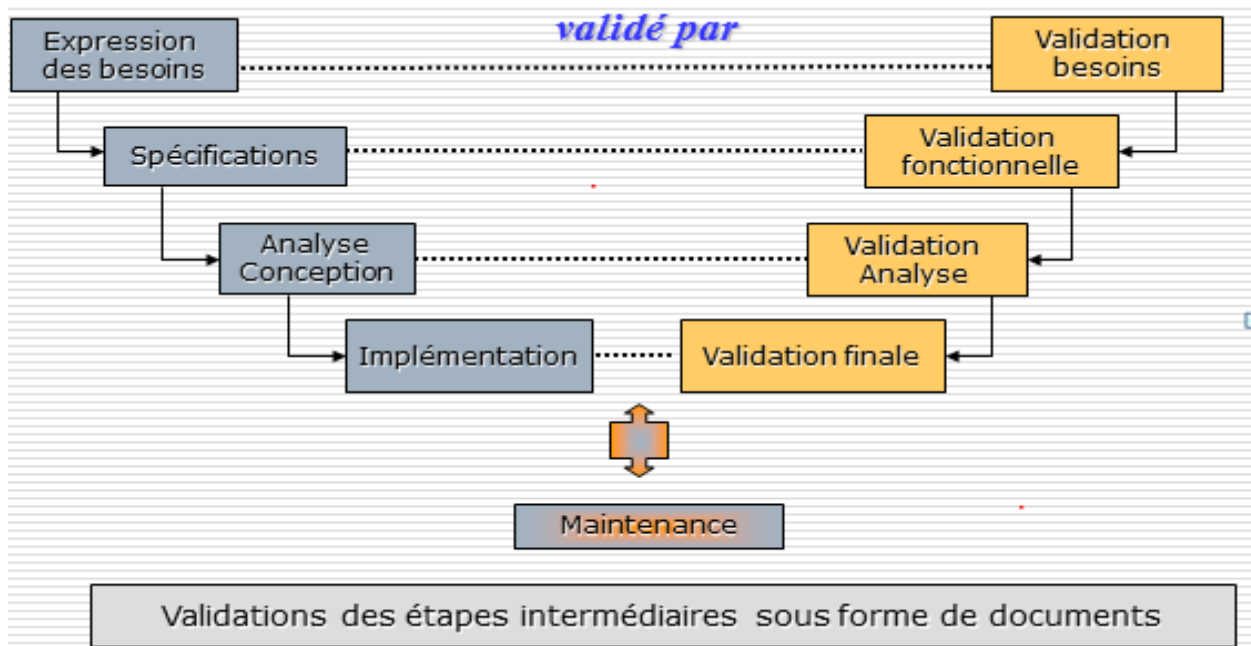
## 2- Le cycle en V

Le modèle en V est une autre façon de présenter une démarche qui reste linéaire, mais qui fait mieux apparaître les produits intermédiaires à des niveaux d'abstraction et de formalité différents et les procédures d'acceptation (validation et vérification) de ces produits intermédiaires. Le V est parcouru de gauche à droite en suivant la forme de la lettre : les activités de construction précèdent les activités de validation et vérification. Mais l'acceptation est préparée dès la construction (flèches de gauche à droite). Cela permet de mieux approfondir la construction et de mieux planifier la 'remontée'.





- Adapté pour des projets dont le domaine est bien maîtrisé



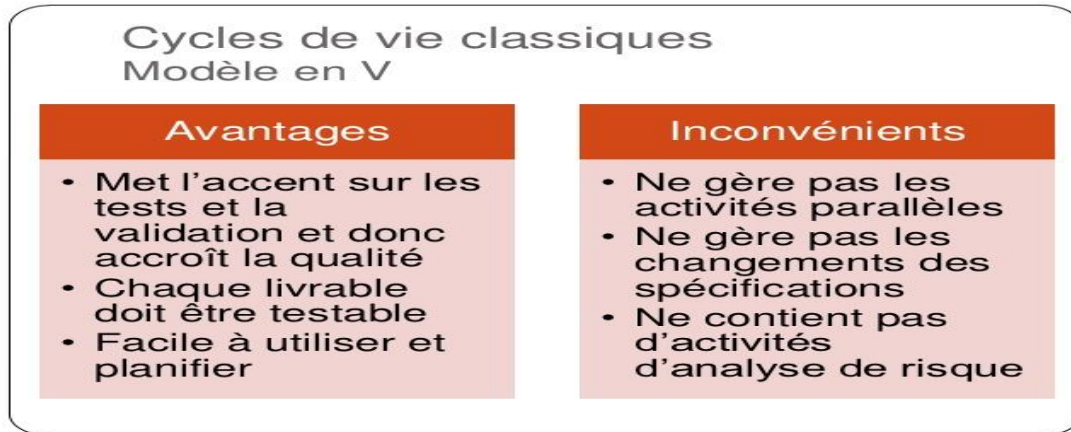
### Intérêts :

#### ➤ Validations intermédiaire

- bon suivi du projet : avancement éclairé et limitation des risques en cascade d'erreurs
- favorise la décomposition fonctionnelle de l'activité
- génération directs des documents et outils supports

➤ Modèle très utilisé et éprouvé

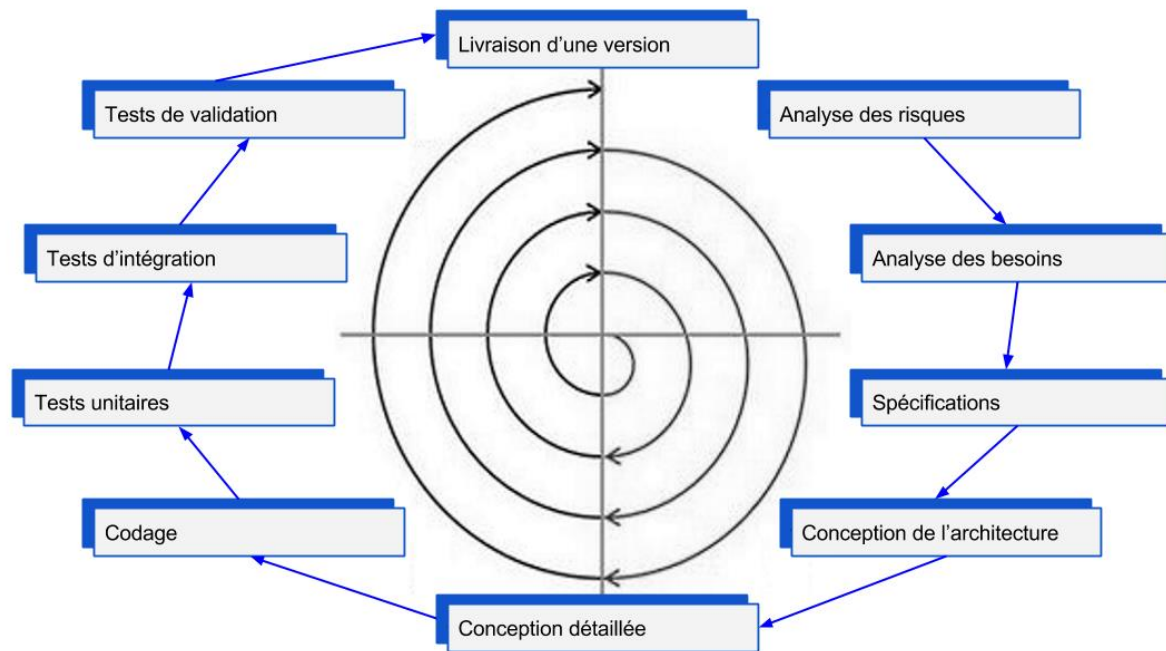
**Limite** : Difficultés extrêmes dans la gestion des versions car aucune garantie sur la non transmission d'erreurs et par conséquent logiciels à vocation temporaire.



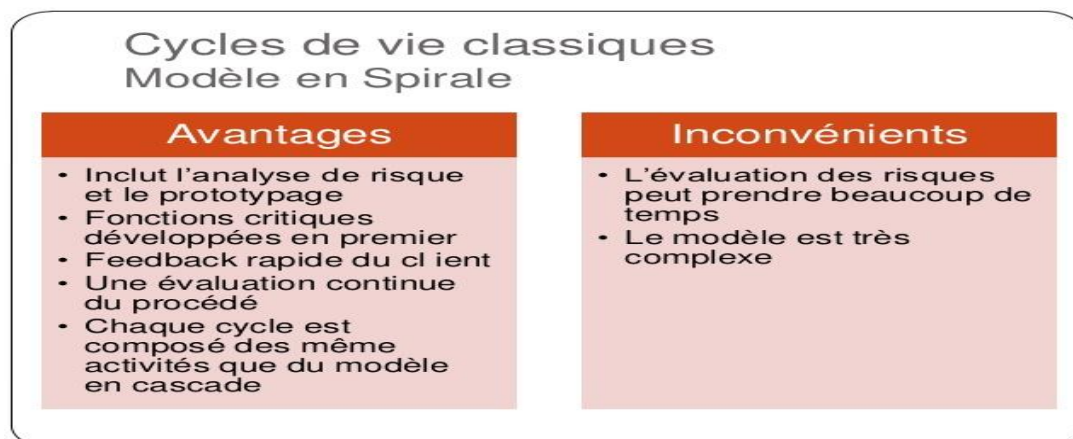
## **B- LES MODELES ITERATIFS OU INCREMENTALS**

Un procédé incrémental atteint son but par étapes en s'en approchant de plus en plus ; chaque résultat est construit en étendant le précédent. On peut par exemple réaliser d'abord un noyau des fonctions essentielles et ajouter progressivement les aspects plus secondaires. Ou encore, construire une série de prototypes 'simulant' plus ou moins complètement le système envisagé.

### **1- Cycle de vie en spiral**



**Principe :** Le modèle en spirale (spiral model) est un modèle de cycle de développement logiciel qui reprend les différentes étapes du cycle en V. Par l'implémentation de versions successives, le cycle recommence en proposant un produit de plus en plus complet et dur. Le cycle en spirale met cependant plus l'accent sur la gestion des risques que le cycle en V. *Identification rapide des risques*

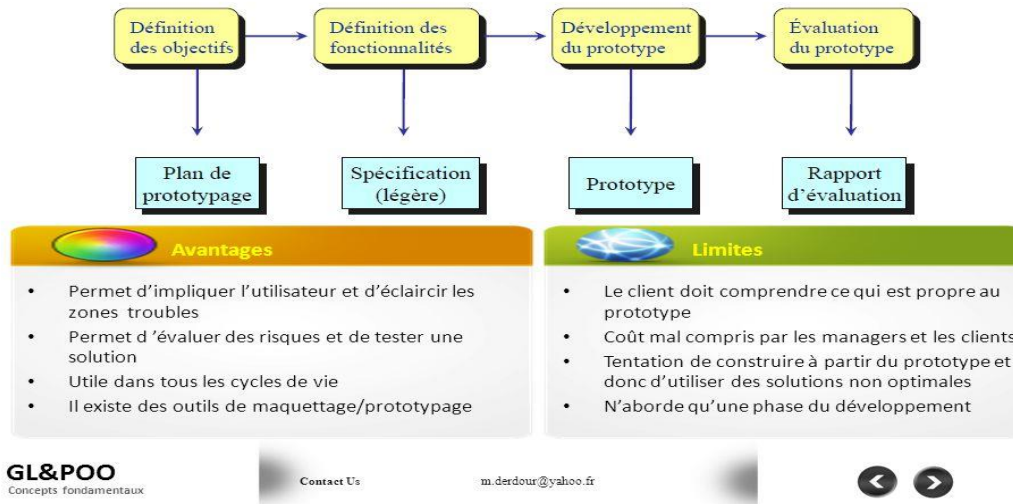


2- Cycle en Y du processus 2TUP

3- Cycle de vie par prototypage

# Prototypage

- Construire un prototype **jetable pour mieux** comprendre les points durs (exigences, technologies)



## Avantages

- ✓ Une application informatique est un excellent support de communication avec le Client
- ✓ On peut tirer du prototype d'autres enseignements : réflexion sur l'architecture et/ou une faisabilité technique, clarification des besoins, essai des IHM

## Inconvénients

- ✓ Le Client peut voir dans le prototype le système final (alors insatisfait des imperfections qui peut influencer sur sa décision à accepter ou non le futur produit)
- ✓ La technologie employée spécialement par le prototype donne une idée qui peut être fautive du développement futur Il faut avoir la détermination de jeter le prototype et le prototype ne doit implémenter que les parties qui nécessitent une réflexion

## BIBLIOGRAPHIE

- 1- *Génie Logiciel Avancé*, Stefano Zacchiroli Laboratoire PPS, Février 2011 ;
- 2- *Introduction au Génie Logiciel*, Guillaume Laurent, ENSMM 2007 ;
- 3- *C. Ghezzi. Fundamentals of Software Engineering*, Prentice Hall, 2nd edition, 2002;
- 4- *Génie Logiciel*, Pierre Gérard, Licence pro.FC 2007/2008 ;
- 5- *Le Génie Logiciel*, Lydie du Bousquet ;
- 6- *Génie Logiciel*, Jacques LONCHAMP, CNAM - Nancy 2003 ;