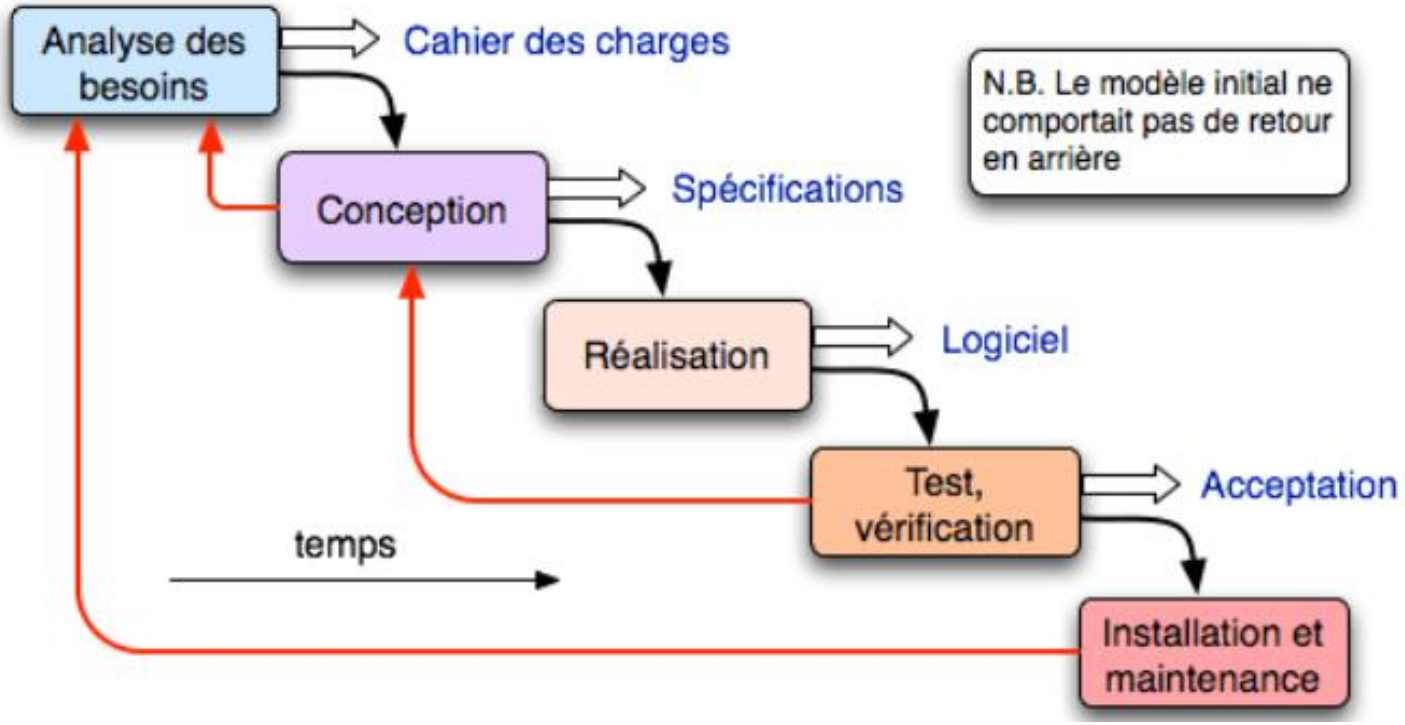
Génie Logiciel et Labo

- Dispensé par Dr. Msc. Ir. **MWAMBA KASONGO Dahouda**
  Docteur en génie logiciel et systèmes d'information
  Machine and Deep Learning Engineer

- Assisté par Master Yani KALOMBA

  - E-mail : dahouda37@gmail.com
    dahouda37@hanyang.ac.kr
    dahouda37@upl-univ-ac

  - Tel.: **+243 99 66 55 265**



Génie Logiciel

CHAPITRE 1. INTRODUCTION AU GENIE LOGICIEL

1.1. Historique

Le **génie logiciel** est une discipline relativement récente qui est apparue en réponse à ce qu'on appelle la "**crise du logiciel**" des années 1960 et 1970

- L'expression « **Génie Logiciel** » ou « **Software Engineering** » est introduit la première fois à une conférence en Allemagn e en 1968 pour répondre à la crise logiciel.

La crise du logiciel fait référence à la période où les coûts, les délais et la qualité des projets logiciels étaient souvent Insatisfaisants.

❖ **1968: La crise du logiciel**

- ✓ Lors d'une conférence de l'OTAN à Garmisch, en Allemagne, en 1968, le terme "**crise du logiciel**" a été introduit pour décrire les problèmes croissants liés au développement de logiciels.
- ✓ Les logiciels devenaient de plus en plus complexes, avec des retards, des dépassements de budget, des bugs non résolus, et des projets annulés.
- ✓ Cette crise a souligné la nécessité de nouvelles méthodes pour améliorer la qualité et la gestion des projets logiciels.

CHAPITRE 1. INTRODUCTION AU GENIE LOGICIEL

1.2. Objectifs

Le génie logiciel (GL) se préoccupe des procédés de fabrication de logiciel de façon à s'assuré que les quatre critères suivant soit satisfait: **Coût (C)**, **Qualité (Q)**, **Fonctionnalité (F)**, et **Délai (D).**

Les concepts de **Coût (C)**, **Qualité (Q)**, **Fonctionnalité (F)**, et **Délai (D)** sont des facteurs clés dans la gestion de tout projet de développement logiciel. Ils constituent les principaux critères d'évaluation du succès d'un projet, explication de chacun:

- ✓ **Coût (C)**: les coûts reste dans les **limites prévu au départ**.
  - Gérer les ressources de manière optimale pour respecter le budget sans compromettre la qualité ou la fonctionnalité.

- ✓ **Qualité (Q)**: La qualité correspond au **contrat du service initial**.
  - Fournir un produit fiable, utilisable, sécurisé et performant qui respecte les normes de qualité définies par les parties prenantes.

- ✓ **Fonctionnalité (F)**: Le système développé doit répondre aux **besoins des utilisateurs**.
  - Assurer que toutes les fonctionnalités requises soient correctement implémentées, et qu'elles répondent aux attentes des utilisateurs finaux.

- ✓ **Délai (D)**: Les délais correspondent au respect des **dates de livraison** prévues.
  - Terminer le projet à temps, tout en maintenant un équilibre entre qualité, coût et fonctionnalités

CHAPITRE 1. INTRODUCTION AU GENIE LOGICIEL

1.3. Principes de génie logiciel

❖ Rigueur

- Les principales sources de défaillances d'un logiciel sont d'origine humaine.

✓ À tout moment, il faut se questionner sur la validité de son action.

- Des outils de vérification accompagnant le développement peuvent aider à réduire les erreurs.

Cette famille d'outils s'appelle CASE (Computer Aided Software Engineering).

❖ Abstraction

- Extraire des concepts généraux sur lesquels raisonner, puis instancier les solutions sur les cas particuliers.

❖ Décomposition en sous problèmes

Traiter chaque aspect séparément, chaque sous-problème plus simple que problème global.

❖ Modularité

- Partition du logiciel en modules interagissant, remplissant une fonction et ayant une interface cachant l'implantation aux autres modules

CHAPITRE 1. INTRODUCTION AU GENIE LOGICIEL

1.4. Principes de génie logiciel

❖ Construction incrémentale

- Construction pas à pas, intégration progressive.

❖ Généricité

- Proposer des solutions plus générales que le problème pour pouvoir les réutiliser et les adapter à d'autres cas.
- Un logiciel réutilisable a beaucoup plus de valeur qu'un composant dédié.

❖ Anticipation des évolutions

- Liée à la généricité et à la modularité, prévoir les ajouts/modifications possibles de fonctionnalités.

Documentations

- Essentielle pour le suivi de projet et la communication au sein de l'équipe de projet.

CHAPITRE 1. INTRODUCTION AU GENIE LOGICIEL

1.5. Qualités attendues d'un logiciel

Les qualités attendues d'un logiciel sont des critères essentiels qui déterminent son succès et son utilité pour les utilisateurs. Ces qualités couvrent plusieurs aspects techniques et fonctionnels, garantissant que le logiciel est **fiable**, **performant**, et **adapté** à son environnement d'utilisation.

❖ Qualites Externe

- **La validité**: aptitude d'un produit logiciel à réaliser exactement les tâches définies par sa spécification.
- **La robustesse**: aptitude d'un logiciel à fonctionner même dans des conditions anormales.
- **L'extensibilité**: Facilité d'adaptation d'un logiciel aux changement de spécification.
- **Réutilisabilité**: aptitude d'un logiciel à être réutiliser en tout ou en partie pour des nouvelles applications.
- **La compatibilité**: aptitude d'un logiciel à pouvoir être combiner les un avec les autres.

CHAPITRE 1. INTRODUCTION AU GENIE LOGICIEL

1.5. Qualités attendues d'un logiciel

D'autres facteurs de qualité du logiciel sont moins cruciales:

❖ Qualites Externe

- **L'efficacité**: bonne utilisation des ressources du matériel.
- **La portabilité**: facilité aux laquelle le produit être adapté à des différents environnements matériel ou logiciel.
- **Vérifiabilité**: facilité de préparation des procédures de recette et de certification (test).
- **L'intégrité**: aptitude des logiciels à protéger leurs différents composants contre accès et des modifications non autorisés.
- **Facilité d'utilisation**: facilité avec lesquelles les utilisateurs d'un logiciel peuvent apprendre comment l'utiliser, comment le faire fonctionner, comment préparer les données, mais aussi comment interpréter les résultats et les effets en cas d'erreur.

CHAPITRE 1. INTRODUCTION AU GENIE LOGICIEL

1.6. Cycle de vie d'un logiciel

Le cycle de vie du logiciel modélise l'enchaînement des différentes activités du processus technique de développement du logiciel.

Une activité comprend: des tâches, des contraintes, des ressources, une façon d'être réalisée.

Les grandes activités sont:

- ✓ **Analyse des besoins**
- ✓ **spécification des besoins**
- ✓ **Conception architecturel et détaillé**
- ✓ **Programmation**
- ✓ **Gestion de configuration et d'intégration**
- ✓ **Validation et vérification**
- ✓ **Livraison et maintenance**

CHAPITRE 1. INTRODUCTION AU GENIE LOGICIEL

1.6. Cycle de vie d'un logiciel

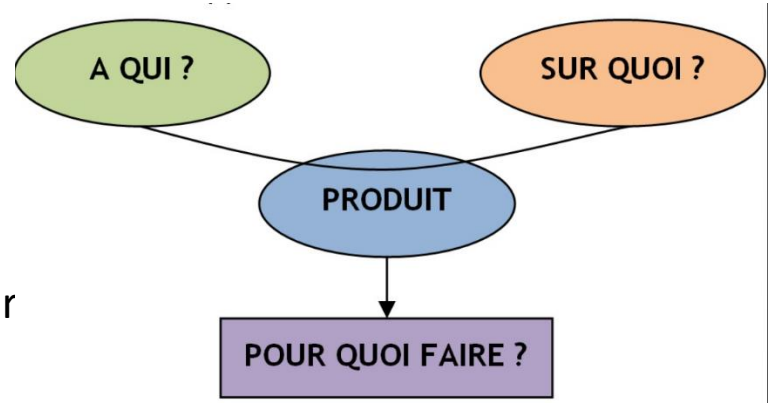
1.6.1. Analyse des besoins

- **Objectif** : Comprendre et définir précisément ce que le client ou l'utilisateur final attend du système.

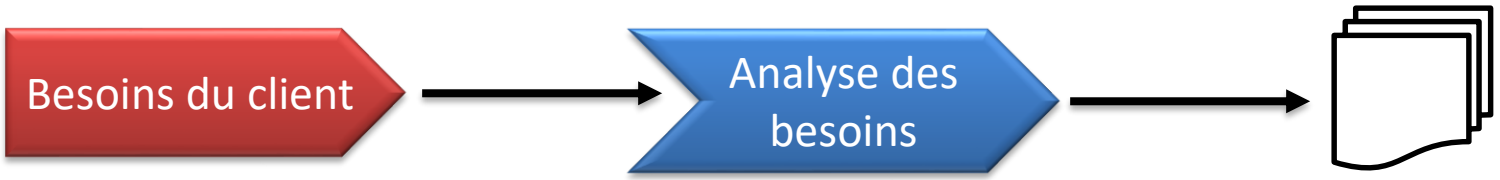
Cette étape implique la collecte, la documentation, et la priorisation des besoins fonctionnels et non-fonctionnels.

- **Actions** :

✓ Identifier les **exigences fonctionnelles** (ce que le logiciel doit faire).
✓ Identifier les **exigences non fonctionnelles** (performance, sécurité, fiabilité, etc.).
✓ Communiquer avec les **parties prenantes** (utilisateurs, clients, responsables).
✓ Créer des **cas d'utilisation** pour illustrer le fonctionnement attendu du système.



Output : Document de spécification des besoins, diagrammes de cas d'utilisation, cahier



- **Experts du domaine d'application**
- **Future Utilisateurs du système**

- **Entretien**
- **Questionnaire**
- **Observation**
- **Etude de situation similaire**

- **Cahier des Charges (CC)**
- **Manuel d'utilisation préliminaire**

CHAPITRE 1. INTRODUCTION AU GENIE LOGICIEL

1.6. Cycle de vie d'un logiciel

1.6.2. spécification des besoins

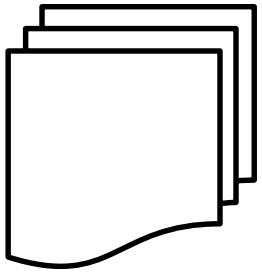
- **Objectif** : Traduire les besoins recueillis lors de l'analyse en spécifications détaillées et globales du système.

C'est la transition entre ce que le **client** **souhaite** et ce que **l'équipe de développement** peut **concevoir**.

- **Actions** :

✓ Formaliser les **exigences** dans un document plus technique.
✓ Définir les **interfaces utilisateur (IHM)** et les interactions principales.
✓ Élaborer des spécifications de haut niveau pour les **modules** du système.
✓ Valider les exigences auprès des **utilisateurs** et des **parties prenantes** pour s'assurer qu'elles sont complètes et cohérentes.

Output : Document de spécification technique globale, maquettes d'interfaces utilisateur, diagrammes UML de haut niveau.

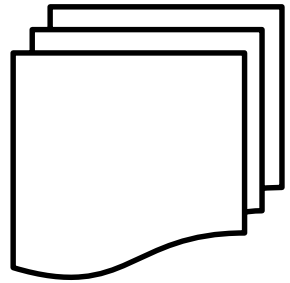


- **Cahier des Charges (CC)**
- **Manuel d'utilisation préliminaire**

spécification des besoins

1- Spécification en langage naturel
2- Spécification semi-formelle
3- Spécification formelle

- **Modèle E/A**
- **Diagrammes d'etats transition**



- **Cahier des Charges fonctionnel**
- **Technique et faisabilité informatique**

CHAPITRE 1. INTRODUCTION AU GENIE LOGICIEL

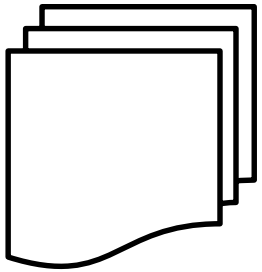
1.6. Cycle de vie d'un logiciel

1.6.3. Conception architecturel et détaillée

- **Objectif** : Concevoir la structure technique du système, en définissant comment les différentes parties du système interagiront entre elles (architecture) et les détails techniques de chaque composant (conception détaillée).

✓ **La conception architecturale** (ou conception globale) a pour but de décomposer le logiciel en composants plus simples, déf inis par leurs interfaces et leurs fonctions (les services qu'ils rendent).

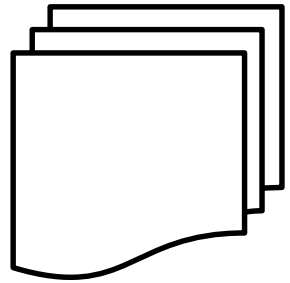
✓ **La conception détaillée** a pour but de détailler les spécifications de chaque composant.
Créer des diagrammes UML comme les diagrammes de classes, de séquence, ou d'état pour illustrer le fonctionnement interne.



- **Cahier des Charges (CC)**
- **Manuel d'utilisation préliminaire**

Conception
La conception architecturale
La conception détaillée

- **Conception fonctionnelle et conception orienté object**
- **Pour chaque composant, ecrire les algorithmes**



- **Dossier de conception (Architecture et Algorithme)**

CHAPITRE 1. INTRODUCTION AU GENIE LOGICIEL

1.6. Cycle de vie d'un logiciel

1.6.4. Programmation

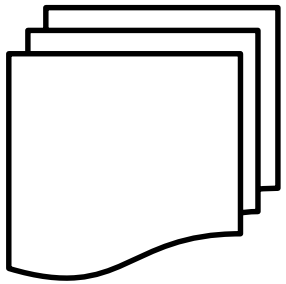
- **Objectif** : Développer le code correspondant aux spécifications techniques et à l'architecture définie.

Cette étape implique la **réalisation** concrète du système à travers l'écriture du code.

- **Actions** :

✓ Implémenter les fonctionnalités conformément à la **conception détaillée**.
✓ Suivre des **standards de codage** et des pratiques comme **l'intégration continue** pour tester et intégrer fréquemment du nouveau code.
✓ Effectuer des **tests unitaires** et des **tests d'intégration** pendant le développement.
✓ Utiliser des outils de **gestion de version** (Git) pour suivre l'évolution du code.

- **Output** : Code source, composants logiciels fonctionnels, documentation technique du code.



- Dossier de conception (Architecture et Algorithme)

- Choix de l'environnement
- Choix du langage de programmation

- Code source
- Manuel final d'utilisation
- Documentations

CHAPITRE 1. INTRODUCTION AU GENIE LOGICIEL

1.6. Cycle de vie d'un logiciel

1.6.5. Gestion de configuration et d'intégration

- **Objectif** : Gérer les différentes versions du système, s'assurer que tous les composants sont intégrés correctement, et coordonner les évolutions du code.

➢ **Gestion de configuration** :

Elle a pour but de maîtriser l'évolution et la mise à jour des composants tout au long du processus de développement.

✓ Suivre les modifications du code et des configurations.
✓ Maintenir un historique des versions pour identifier les changements apportés.

➢ **Intégration** :

Elle a pour but de réaliser un ou plusieurs systèmes exécutables à partir des composants (Combiner les composants).

✓ Rassembler les différents composants développés dans une seule version cohérente.
✓ Automatiser l'intégration grâce à des outils d'intégration continue.

- **Output** : Système complet intégré, journal de version, documentation de configuration

CHAPITRE 1. INTRODUCTION AU GENIE LOGICIEL

1.6. Cycle de vie d'un logiciel

1.6.6. Validation et vérification

- **Objectif** : S'assurer que le logiciel répond aux besoins initiaux (validation) et que chaque partie du logiciel fonctionne correctement selon les spécifications techniques (vérification).

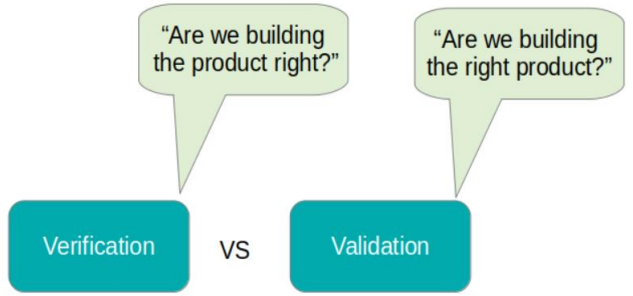
- **La Validation** a pour but de répondre à la délicate question : a-t-on décrit le bon système, celui qui répond à l'attente des utilisateurs.

✓ Valider que le logiciel répond aux attentes des utilisateurs en fonction des exigences.
✓ Faire des tests utilisateurs ou des tests d'acceptation pour vérifier que le produit correspond bien aux attentes.

- **Vérification** répond à la question : le développement est-il correct par rapport à la spécification globale ?

Ce qui consiste à s'assurer que les description successives et le logiciel lui-même satisfont la spécification.
✓ Tester chaque composant (tests unitaires) et l'ensemble du système (tests d'intégration).
✓ Utiliser des tests automatiques et manuels pour valider les performances, la sécurité, et la fiabilité.

- **Output** : Rapports de tests, corrections des bugs identifiés, documentation de validation.



CHAPITRE 1. INTRODUCTION AU GENIE LOGICIEL

1.6. Cycle de vie d'un logiciel

1.6.7. Livraison et maintenance

- **Objectif** : Livrer le logiciel au client/utilisateur final et garantir sa maintenance sur le long terme pour corriger les bugs, ajouter des fonctionnalités, ou améliorer les performances.

Il s'agit d'apporter des modifications à un logiciel existant. C'est la phase la plus coûteuse (70% du coût total).

- **Livraison** :

✓ Déployer le logiciel dans l'environnement de production.
✓ Former les utilisateurs et fournir la documentation d'utilisation.

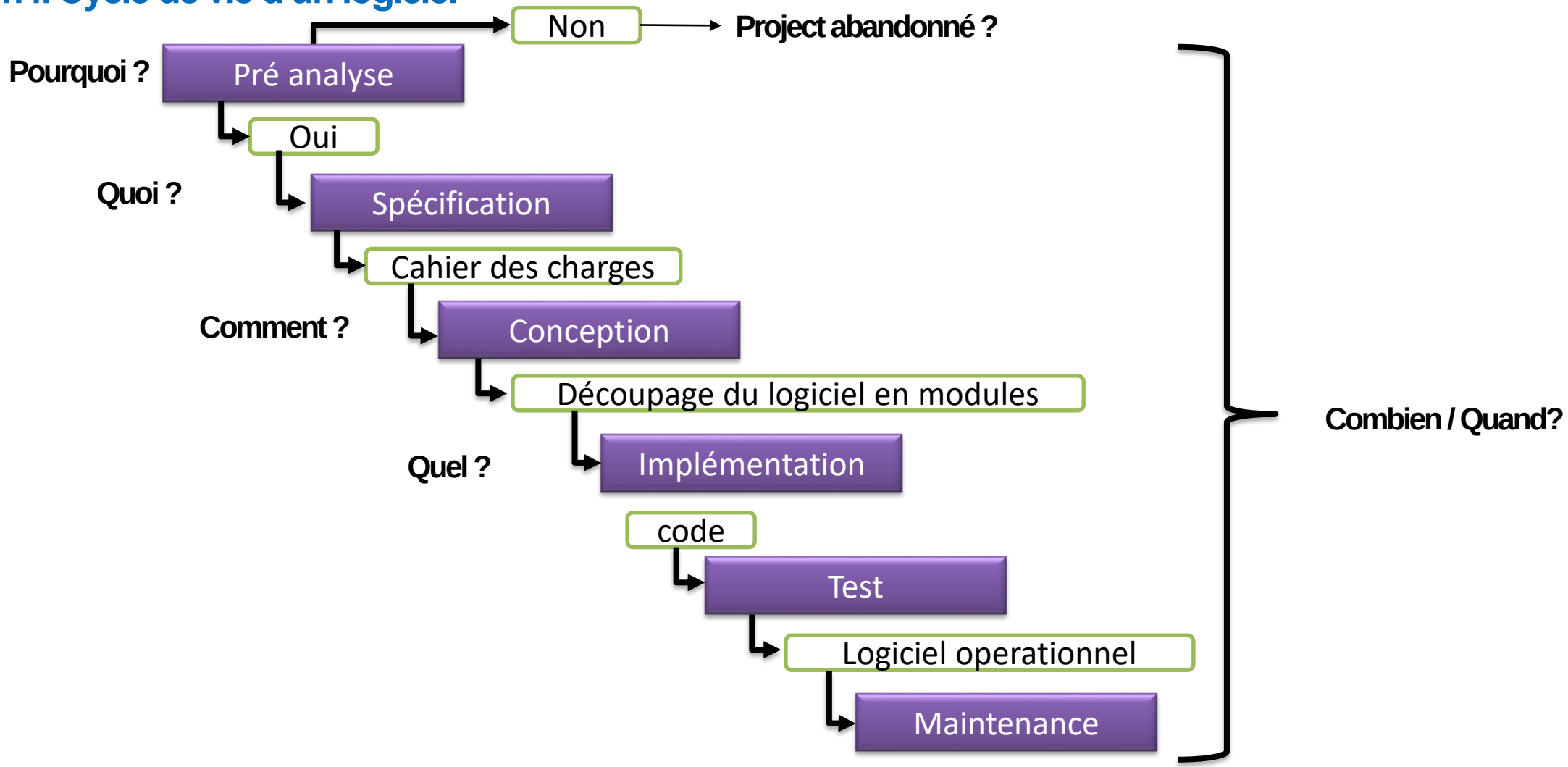
- **Maintenance** :

✓ Corriger les bugs identifiés après la mise en production.
✓ Effectuer des mises à jour pour ajouter des nouvelles fonctionnalités ou optimiser le système.
✓ Assurer la maintenance corrective, évolutive, et préventive.

- **Output** : Version finale du logiciel livrée, documentation utilisateur, plan de maintenance.

CHAPITRE 1. INTRODUCTION AU GENIE LOGICIEL

1.4. Cycle de vie d'un logiciel



CHAPITRE 1. INTRODUCTION AU GENIE LOGICIEL

1.4. Cycle de vie d'un logiciel

Documents:

Manuel utilisateur ----------> final Implémentation
Dossier de conception architecturale -----> Conception
Code source ----------> Implémentation
Cahier des charges ----------> Analyse des besoins
Manuel utilisateur préliminaire ----------> Analyse des besoins
Dossier de conception détaillée ----------> Conception
Rapport des tests ----------> Tests
Documentation ----------> Implémentation
Cahier des charges fonctionnel ----------> Spécification

Labo

- Utitisation de AGL (UML)
- Implementation des modeles des Machine Et Deep Learning

CHAPITRE 2. LABO : Machine Et Deep Learning

2.1 Outils et Bibliothèques

- ❑ Principes de base de Python
- Python pour la science des données
- ❑ Bibliothèques pour Machine Learning
  - Numpy (tableaux),
  - Pandas (manipulation de données)
  - Scikit-Learn pour la création de modèles simples
  - Matplotlib et Seaborn (visualization de données)
- ❑ IDE pour Machine Learning
  - Anaconda (Jupyter Nootbook, JupyterLab)
  - Google Colaboratory,

Problème a résoudre

- ❑ Régression Linear Simple :
- ❑ Régression Linear Multiple
- ❑ Classification

CHAPITRE 2. LABO : Machine Et Deep Learning

1.8 Outils et Bibliothèques [Lab]

❑ Installations

1.Créer un dossier portant comme nom « Votre nom complet » a la racine C

2. Créer un environement conda avec Anaconda prompt : conda create –n prenom_env python=3.10

3. Activer votre environment : conda activate prenom_env

4. Verifier la liste des environement : conda env list

5. Verifier la liste des libraries déjà installed : pip list

6. Installler les libraries suivants :

pip install numpy
pip install pandas
pip install matplotlib
pip install seaborn
pip install scikit-learn –U
pip tensorflow

Fin