

# chapitre 2

## Cycle de vie de logiciel

Année universitaire 2023-2024

## Raisons de la faible qualité des logiciels



Ce que le client a expliqué



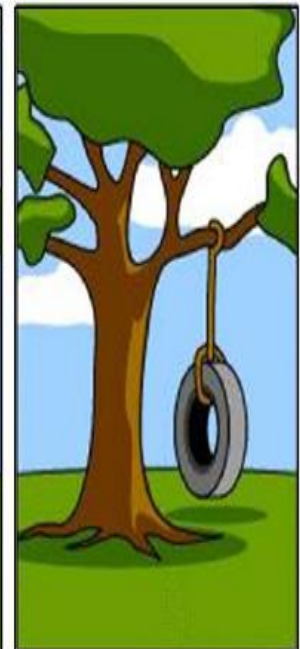
Ce que le chef de projet a compris



Ce que l'analyste a proposé



Ce que le programmeur a écrit



Ce dont le client avait vraiment besoin

# Le génie logiciel

## Génie :

Ensemble des connaissances et des techniques concernant la conception, la mise en œuvre et les applications de procédés, de dispositifs, de machines propres à un domaine donné (Petit Larousse Illustré).

## Génie logiciel :

Ensemble des activités de conception et de mise en œuvre des produits et des procédures tendant à rationaliser la production du logiciel et son suivi.



# Le génie logiciel

## Définition :

Ensemble des méthodes, des techniques et des outils dédiés à la **conception**, au **développement** et à la **maintenance** des systèmes informatiques

## Objectif :

Avoir des **procédures systématiques, rigoureuses et mesurables** pour des logiciels de **grande taille** afin que

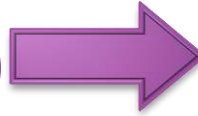
- la spécification corresponde aux **besoins réels** du client
- le logiciel respecte sa **spécification**
- les **délais** et les **coûts** alloués à la réalisation soient respectés



# Cycle de vie

Le cycle de vie du logiciel est la période s'étalant du début à la fin du processus du logiciel.

**Proposition ou décision  
de développer un logiciel**



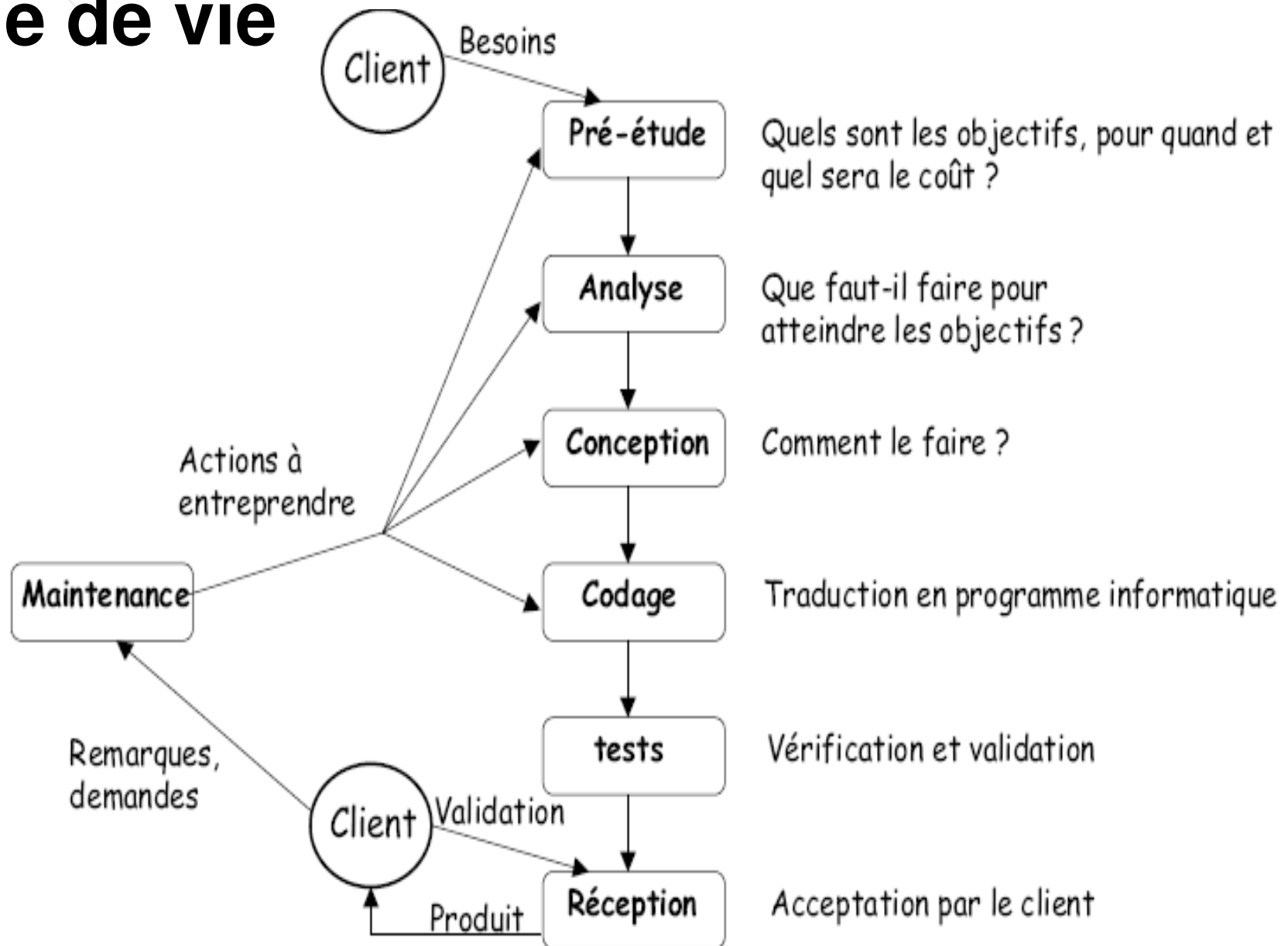
**Mise hors service**

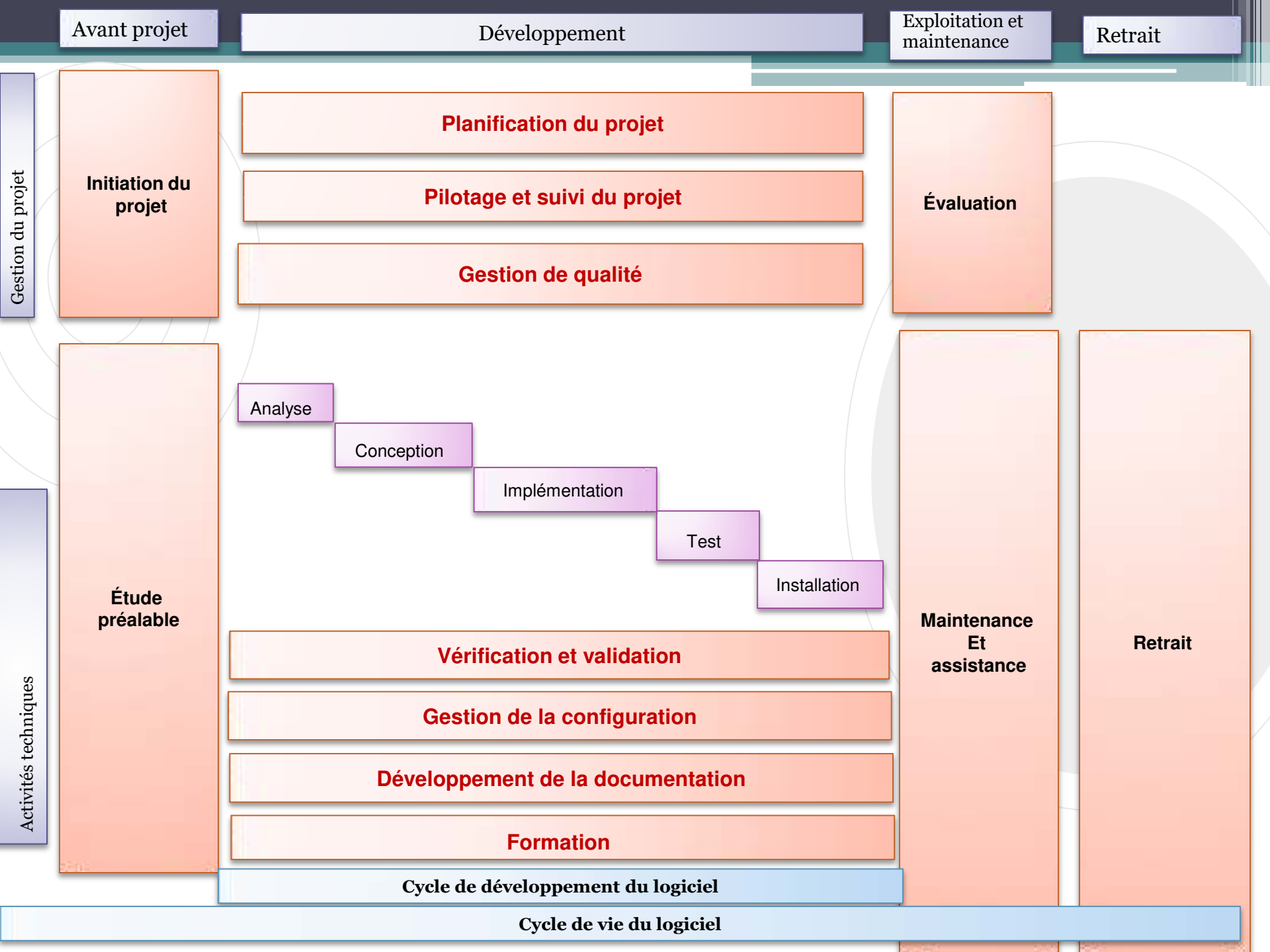
**Processus  
du logiciel**

Le développement et l'évolution d'un logiciel doivent être vu comme un processus. (processus logiciel )

Ce processus est composé de sous processus qui communiquent entre eux et dont certains évoluent en parallèle.

# Cycle de vie





# Phase préparatoire(1)

- On appelle aussi **avant projet**. C'est l'étude d'opportunité ou étude préalable.
- Cette phase répond aux questions suivantes:
  - ◆ pourquoi développer un logiciel?
  - ◆ Comment procéder pour faire ce développement?
  - ◆ Quels moyens faut il mettre en œuvre?

# Phase préparatoire(2)

Cette phase comprend des aspects techniques et de gestion.

Parmi **les tâches techniques**, groupés sous le terme étude préalable, on peut citer :

- ❑ Dresser un état de l'existant et faire une analyse de ses forces et faiblesses.
- ❑ Identifier les idées ou besoins de l'utilisateur.
- ❑ Formuler les solutions potentielles.
- ❑ Faire les études de faisabilité.
- ❑ Planifier la transition entre l'ancien et le nouveau système, s'il y a lieu
- ❑ Affiner l'énoncé des besoins de l'utilisateur.

# Phase préparatoire(3)

La préparation de la gestion du projet est appelé **initiation du projet** ,  
et comprend les tâches suivantes:

- ☐ Représenter les activités à entreprendre dans un modèle.
- ☐ Prévoir les ressources nécessaires au projet.
- ☐ Mettre en place les environnements de développement et de support.
- ☐ Identifier les procédures et normes spécifiques au projet et les ressources à mettre en place pour contrôler leur application.
- ☐ Planifier la gestion du projet.

## Phase d'exploitation et de maintenance(1)

- ◆ Une fois le logiciel a été développé, il sera mis en **exploitation**
- ◆ Cette dernière ait lieu après une phase de transition, appelée installation, qui est prise en charge par l'équipe de développement pour un logiciel fait sur mesure.
- ◆ Pour un logiciel à large diffusion, l'acheteur se charge de l'installation, avec l'assistance du service après vente du distributeur.

## Phase d'exploitation et de maintenance(1)

- Au cours de **la phase de maintenance**, le logiciel est employé dans un environnement opérationnel.
- Son comportement est surveillé et si nécessaire, il est modifié pour :
  - 📄 corriger les défauts
  - 📄 pour améliorer ses performances
  - 📄 pour adapter le logiciel à un nouvel environnement
  - 📄 répondre à des nouveaux besoins ou à des besoins modifiés.

## Phase de développement (1)

**La planification** du projet a pour but de :

- ❑ générer une décomposition du projet en activités et tâches plus élémentaires ayant des interactions minimales.
- ❑ produire une structure de référence pour le pilotage et le suivi du projet.

Elle définit les tâches, le calendrier, les ressources, l'allocation de ces ressources aux tâches et les procédures du projet.

# Phase de développement (2)

**Le pilotage et le suivi du projet** consiste à :

- ☐ enregistrer les faits sur l'avancement du projet.
- ☒ Comparer cet avancement avec la planification
- ☐ Entreprendre, si c'est nécessaire des mesures correctives.

**La gestion de la qualité du logiciel** comprend:

- ☐ l'ensemble des activités de gestion déployées pour garantir que le processus de développement engendre un produit de qualité.

# Phase de développement (3)

**La documentation** est un élément essentiel dans le développement d'un logiciel:

- ❑ elle fait partie du produit à réaliser
- ❑ elle matérialise l'avancement des travaux, puisque chaque phase est concrétisée par la production d'un ou plusieurs documents
- ❑ elle constitue un support de communication entre les différents intervenants du projet.

# Phase de développement (4)

**La gestion de configuration** a pour but de maîtriser l'évolution des différents éléments constituant le projet.

Parmi les entités et les composants qui peuvent jouer le rôle d'éléments de configuration, c'est-à-dire pris en charge par la gestion de configuration:

- ❑ les documents de gestion du projet
- ❑ les documents techniques de réalisation
- ❑ les manuels d'utilisation et d'exploitation
- ❑ les programmes sources et les moyens permettant de produire et de reproduire les programmes machine.

# Phase de développement (5)

Parmi les activités de **la gestion de configuration** :  
contrôler les modifications des éléments de configuration

- ▶ autoriser le changement
- ▶ vérifier que le nouvel état est complet et cohérent
- ▶ enregistrer le nouvel état
- ▶ annoncer la modification
- ▶ mettre à disposition le nouvel état

# Analyse



Quoi  
faire ?

## □ Analyse

### 1. Etude préliminaire:

- ✓ Définition globale du problème (diag. contexte)
- ✓ Evaluation des stratégies possibles
- ✓ Evaluation des ressources, coûts et délais

**Documents:** rapport d'analyse

### 2. Expression et analyse des besoins

- ✓ Exigences fonctionnelles attendues: opérations, etc.
  - ✓ Qualités non fonctionnelles attendues: performance, portabilité, etc.
- cahier des charges, document d'analyse (spécification) et plan de test du système

# Conception. • •

Comment  
faire ?

## □ Conception

1. Conception architecturale: décomposition et organisation de l'application en composants plus simples (modules) définis par leurs interfaces (fonctions et services offerts).
2. Conception détaillée: pour chaque module, description de la manière dont les services et fonctions sont réalisés:
  - algorithmes essentiels
  - structures de données utilisées, etc.

□ **Documents** : document de conception, plan de test global et plan de test par module.

# Programmation

- ❑ **Programmation**
  - Traduction de la conception dans un langage de programmation
  - **Documents:** dossiers de programmation, codes sources commentés.

# Test

## 1. Tests unitaires:

Tests avec les jeux d'essais par module selon le plan de test.

## 2. Tests d'intégration:

- Composition progressive des modules
- Tests des regroupements de modules

## 3. Test du système:

- Test en vraie grandeur du système complet selon le plan de test global.

□ **Documents:** rapport de vérification par test.

# Cycle de vie: autres activités

## Activités exécutées en continu

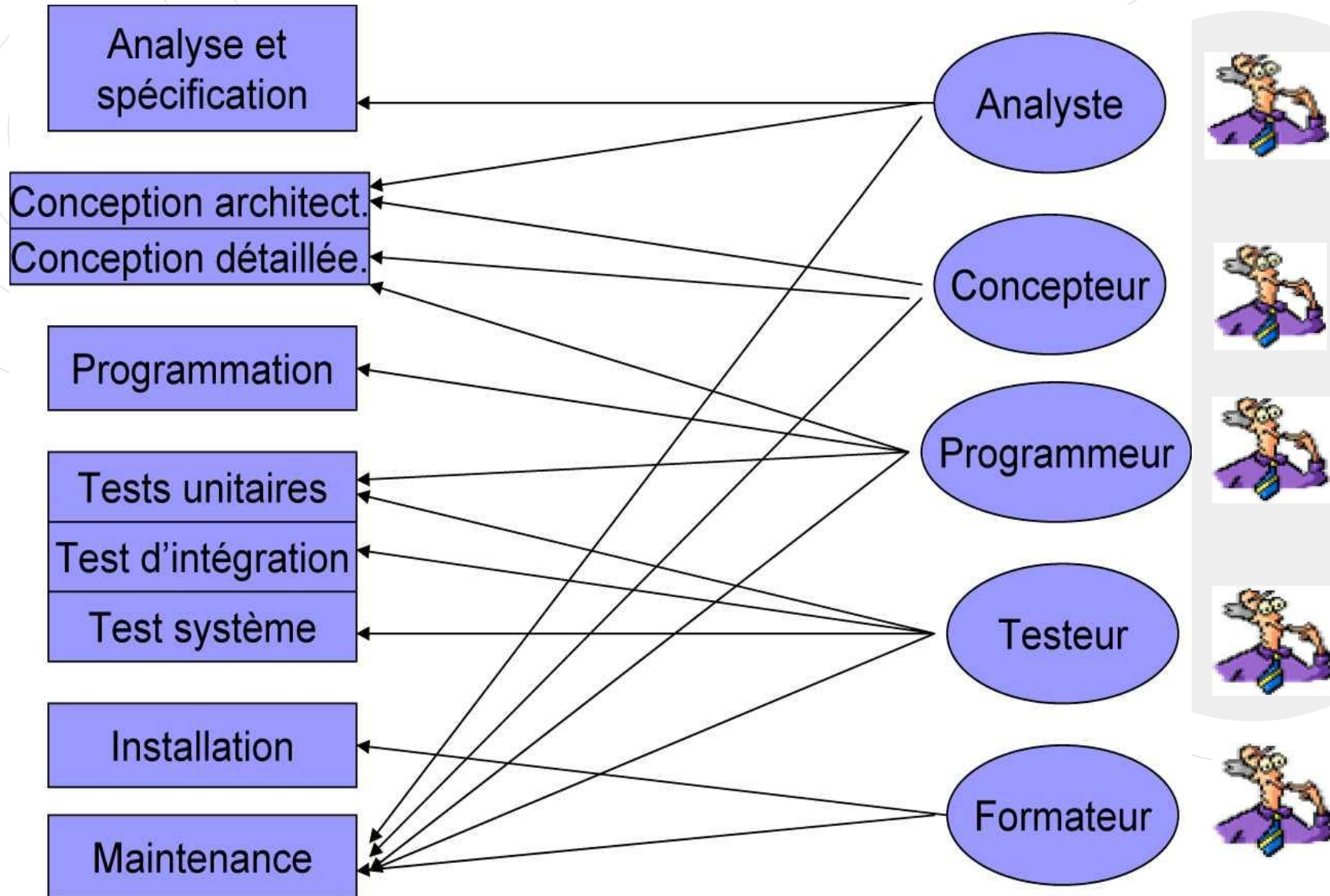
### □ Validation

- Le produit répond-il aux besoins du client? «Construit-on le bon produit?»
- S'assurer de répondre aux exigences du client.

### □ Vérification

- Le produit est-t-il correct (par rapport à la spécification)? «Construit-on le produit comme il faut?»
- S'assurer de la qualité du produit (révisions et inspections)

## L'équipe de développement: le rôle des membres dans le cycle de vie



# chapitre 3

# Processus de développement

Année universitaire 2023-2024

# Processus de développement

**Processus : Description abstraite et idéalisée des différentes manières d'organiser les activités du développement d'un logiciel).**

- Décrit un **ensemble de tâches ordonnées** impliquant
  - des activités (celles du cycle de vie)
  - des contraintes (sur le temps, ressources, etc.)
  - des ressources (humaines, matérielles, etc.)

# Processus de développement

- Le processus de développement est à définir en équipe, de préférence, car ce travail permet:
  - ☐ Compréhension commune des activités, ressources et contraintes.
  - ☐ Identification des incohérences, redondances et omissions dans le processus.
- ☐ Identification des activités permettant de réaliser les objectifs de qualité.
- ☐ Adaptation du processus aux besoins spécifiques du projet et compréhension de ses particularités.

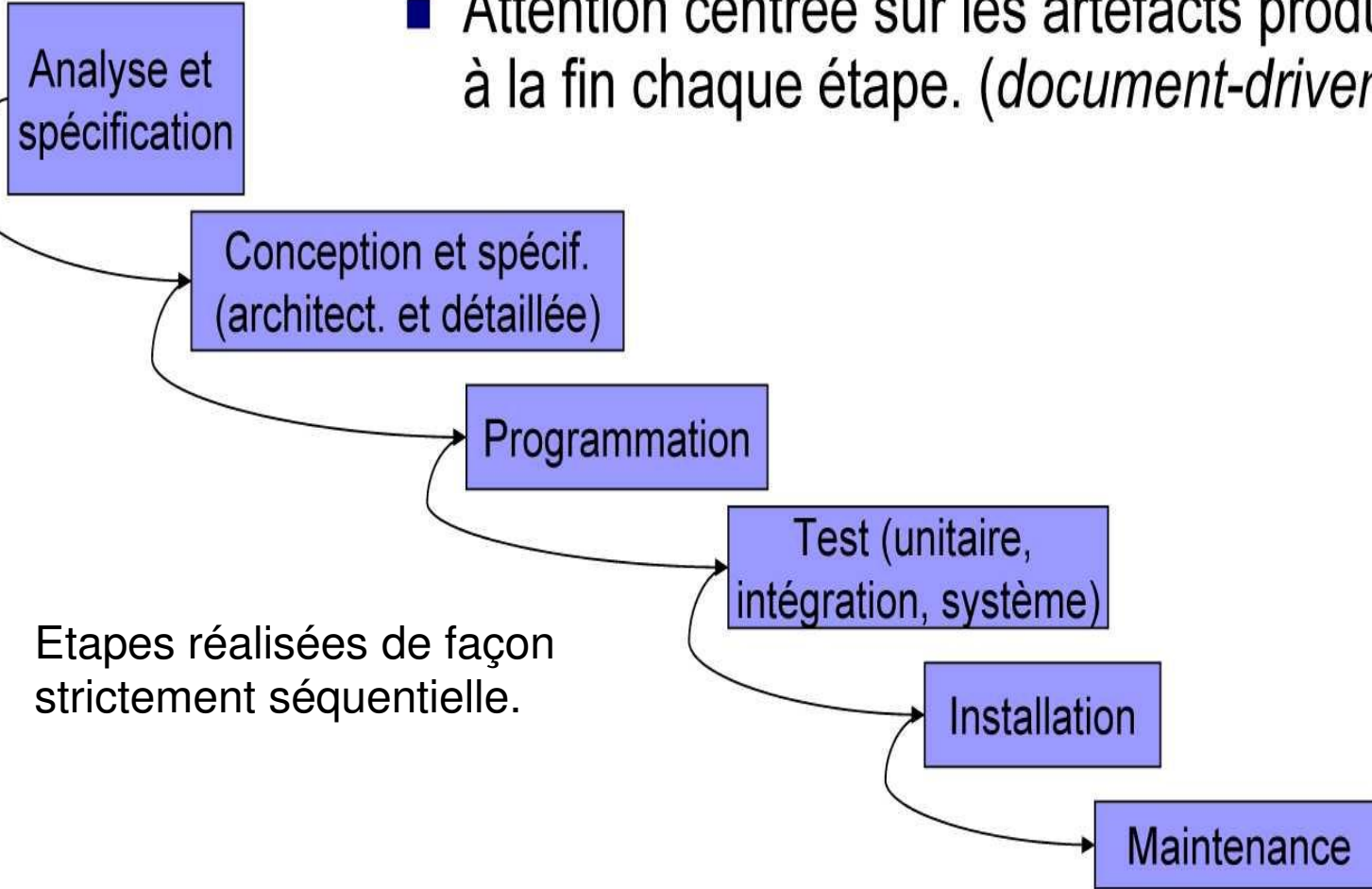
# Processus de développement

Quelques modèles existants ... à personnaliser

- Modèle en cascade
- Modèle en V
- Modèle par prototypage
- Modèles évolutifs: incrémental et itératif

# Modèle en cascade

- Attention centrée sur les artéfacts produits à la fin chaque étape. (*document-driven*)



# Modèle en cascade

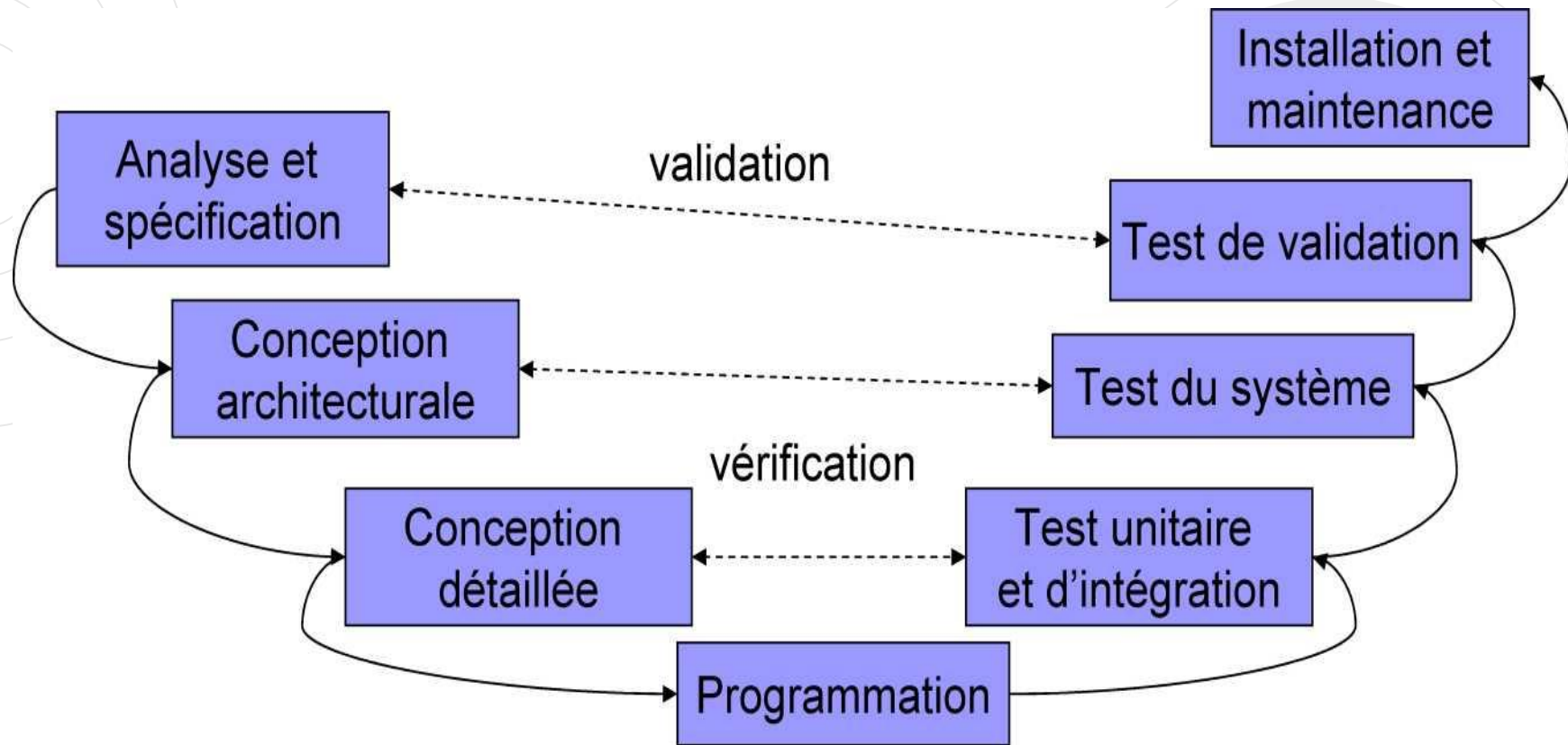
## Avantages

- Plan **simple** de ce qu'il faut faire.
- **Facile** à comprendre
- Cette première définition a permis la normalisation des cadres conceptuels et terminologiques des différentes activités.

## Inconvénients

- Hypothèses souvent **irréalistes** que l'on peut dès le départ définir complètement et en détail
  - ce qu'on veut réaliser
  - les résultats intermédiaires obtenus
- Ne reflète pas la façon dont le code est réellement développé.
- Trop **rigide**, manque de flexibilité pour imprévus.
- **Pas de «feedback»** avant la livraison au client....

# Modèle en V



Variation du modèle en cascade

- Attention centrée sur la correction: vérification et validation

# Modèle en V

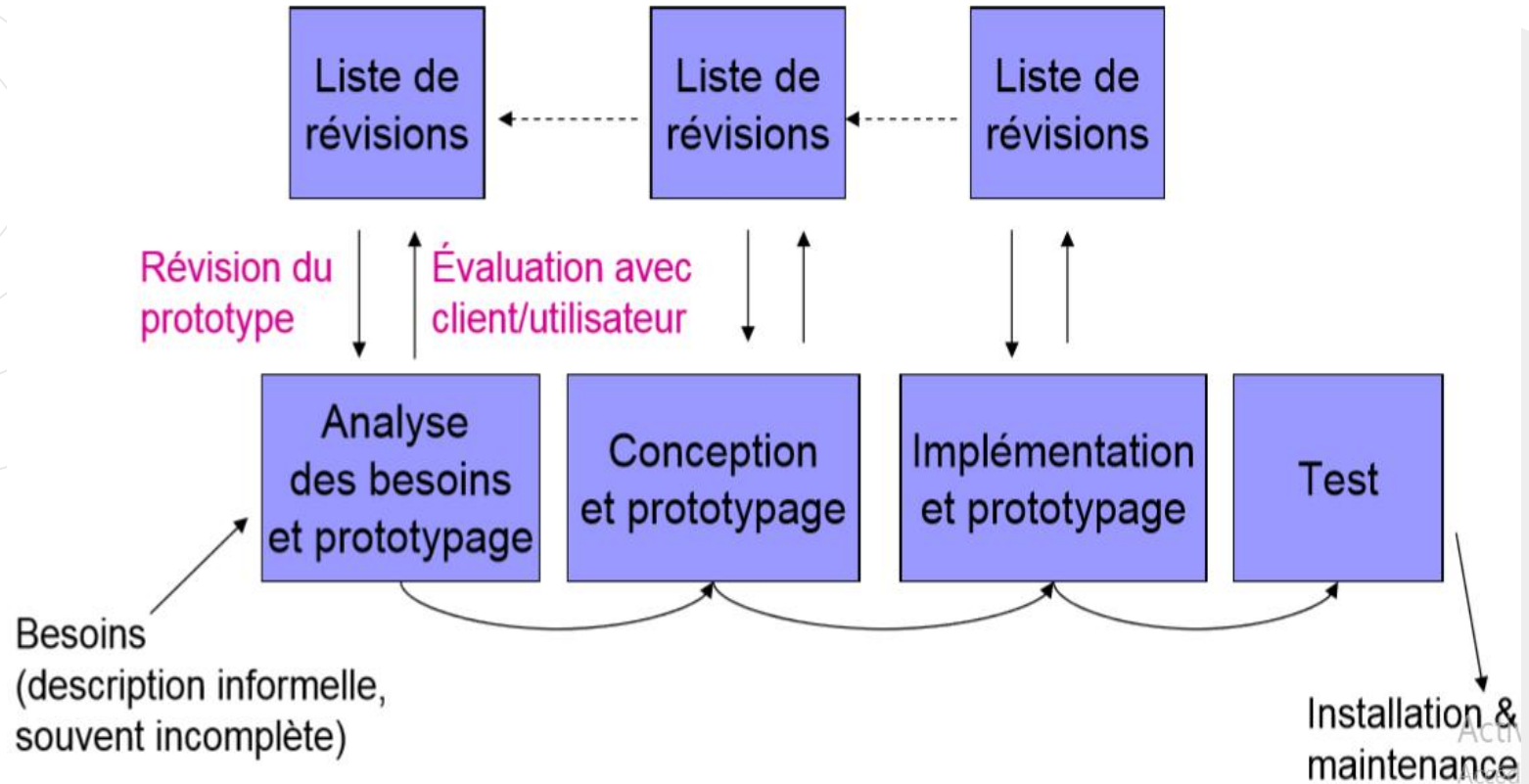
## Avantages

- Montre comment les activités de «test» **sont liées** à celles d'«analyse et de conception».

## Inconvénients

- **Manque de « feedback »**. Pas de résultats intermédiaires dont on peut discuter avec le client.

# Modèle par prototypage



- À chaque étape, un ou plusieurs prototypes sont soumis au client pour évaluation/révision.
- Permet d'examiner et d'explorer certains aspects du système pour évaluer et choisir les meilleures stratégies/solutions.

# Modèle par prototypage

## Types de prototypes

Prototype jetable: maquette exploratoire développée pour mieux comprendre les besoins du client, évaluer différentes solutions, etc. Développé rapidement et jeté ensuite.

Prototype évolutif (ou réutilisable): maquette destinée à être complétée/optimisée dans les prochaines étapes du développement jusqu'à l'obtention du produit final.

# Modèle par prototypage

## Avantages

- Améliore la compréhension mutuelle du problème entre client, développeur et utilisateur.
- Favorise les activités de vérification et de validation intermédiaires.

## Inconvénients

- Tentative d'abrégier le processus et de se contenter d'un prototype incomplet...

# Modèles évolutifs

- Le processus de développement d'un logiciel peut être très long (mois, années...)
- Le logiciel est un produit de nature évolutive.
- Pour être compétitif, il faut **réduire le temps de mise en marché** et pouvoir
  - produire versions partielles du système
  - ajouter graduellement de qualités et fonctionnalités.

## Solution

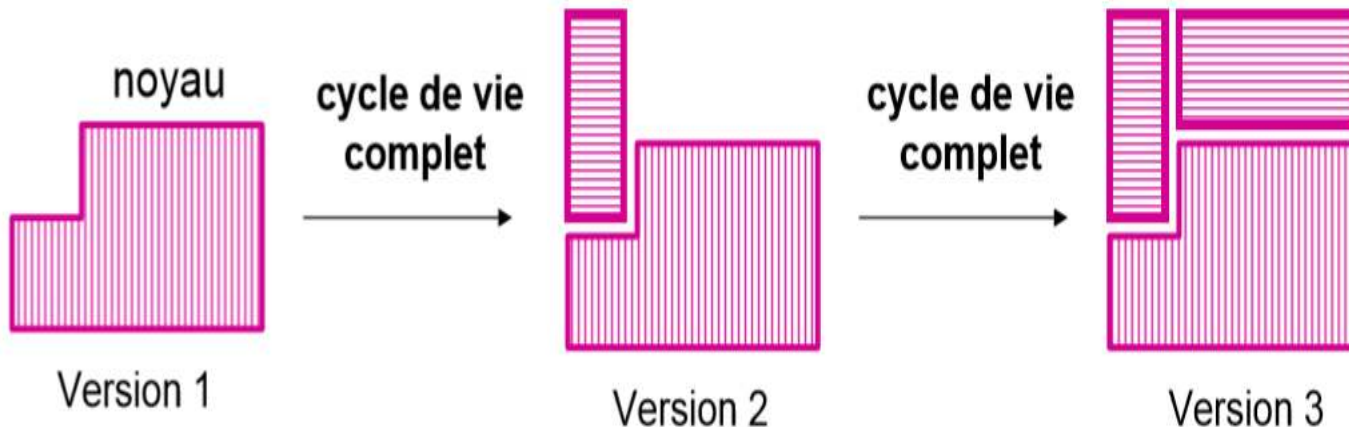
Modèle évolutif: développement au cours duquel des **versions** du logiciel **de plus en plus complexes** et **détaillées** sont développées.

- Deux versions du logiciel existent alors en parallèle
  - logiciel opérationnel (version utilisée actuellement)
  - logiciel en développement (prochaine version)

# Modèle évolutif - incrémental

Approche incrémentale

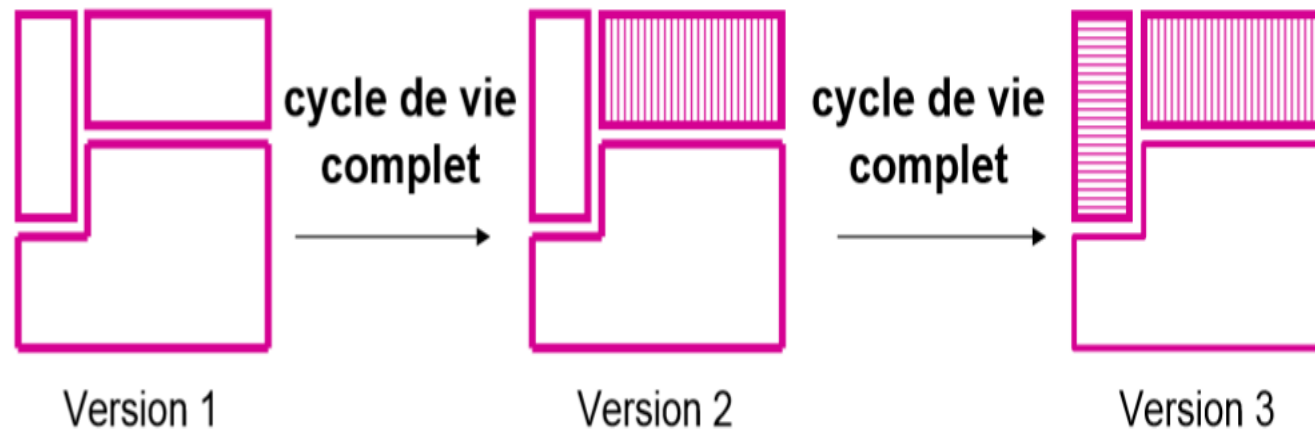
- division du système en sous-systèmes (un par fonctionnalité).
- 1re version: système partiel.
- Chaque nouvelle version (incrément) ajoute une nouvelle fonctionnalité.



# Modèle évolutif - itératif

## ■ Approche itérative

- division du système en sous-systèmes (un par fonctionnalité).
- 1re version: coquille complète du système.
- Chaque nouvelle version apporte une modification/amélioration à une fonctionnalité.



# Modèles évolutifs

## Avantages

- Formation précoce des utilisateurs. Réponse rapide possible.
- Création précoce de nouveaux marchés pour nouvelles fonctionnalités.
- Focus sur nouveau domaine d'expertise à chaque étape (version).
- Détection précoces des problèmes imprévus (correction immédiate du système en développement).

## Inconvénients

- Risque de la remise en cause du noyau (fonctionnalités de base) au cours du développement.