2.2.
Le cycle de vie d'un logiciel

2.2. Le cycle de vie d'un logiciel

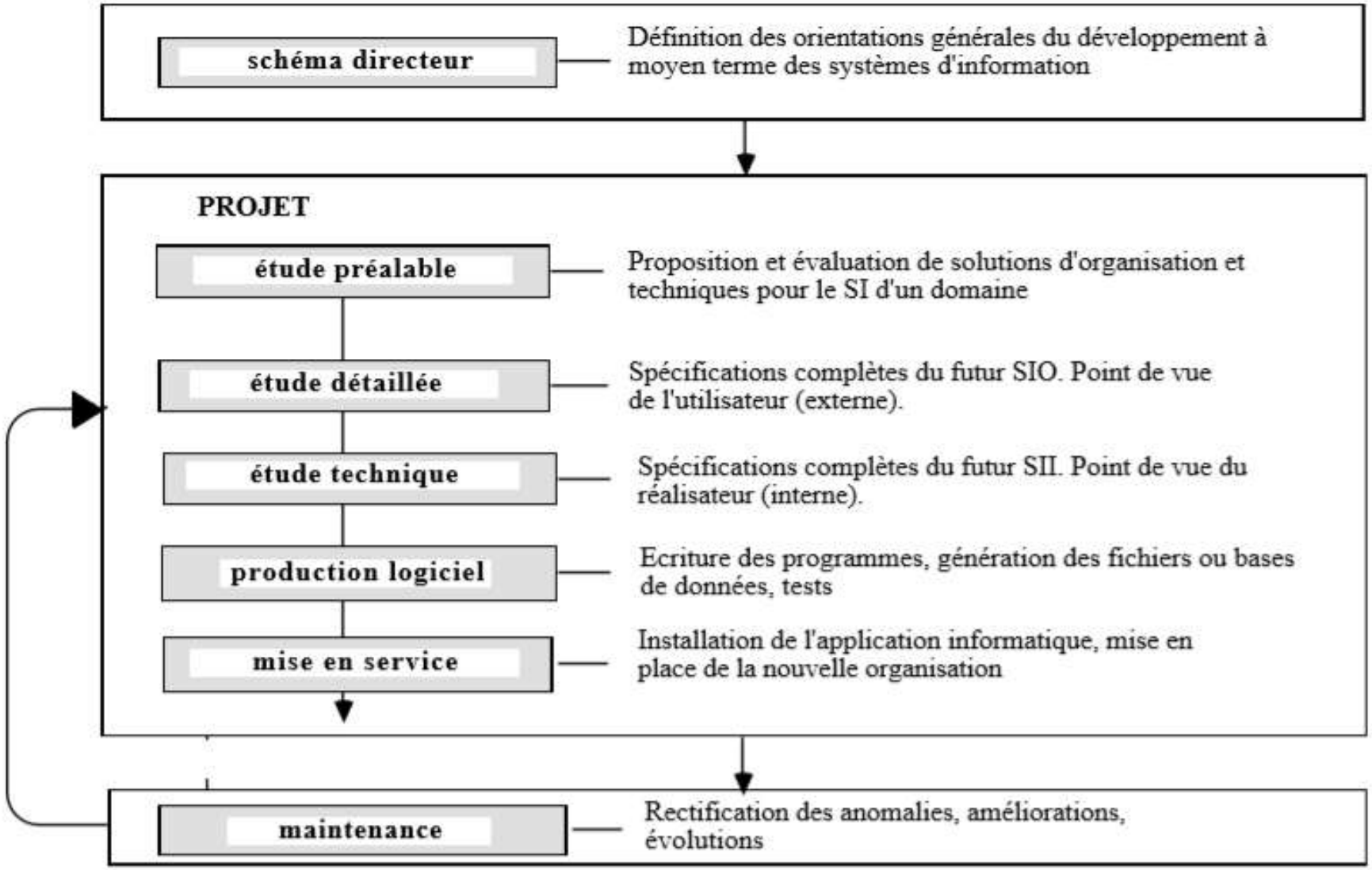
- **Le « cycle de vie d'un logiciel »** (*en anglais software* ***lifecycle***), désigne toutes les étapes du développement d'un logiciel, de sa conception à sa disparition.
- **L'objectif** d'un tel découpage est de permettre de définir des jalons intermédiaires permettant la validation du développement logiciel, c'est-à-dire la conformité du logiciel avec les besoins exprimés, et la vérification du processus de développement, c'est-à-dire l'adéquation des méthodes mises en œuvre.
- **L'origine de ce découpage** provient du constat que les erreurs ont un coût d'autant plus élevé qu'elles sont détectées tardivement dans le processus de réalisation. Le cycle de vie permet de détecter les erreurs au plus tôt et ainsi de maîtriser la qualité du logiciel, les délais de sa réalisation et les coûts associés.
- Le cycle de vie est très variable selon les projets.

- Plus classiquement, le cycle de vie se résume en une démarche comprenant les étapes successives suivantes :

1. **Analyse / Conception** (Schéma directeur, Etude préalable, Etude détaillée)
2. **La réalisation** (Etude Technique, Production Logicielle, Mise en service)
3. **La Maintenance**

2.2. Le cycle de vie d'un logiciel

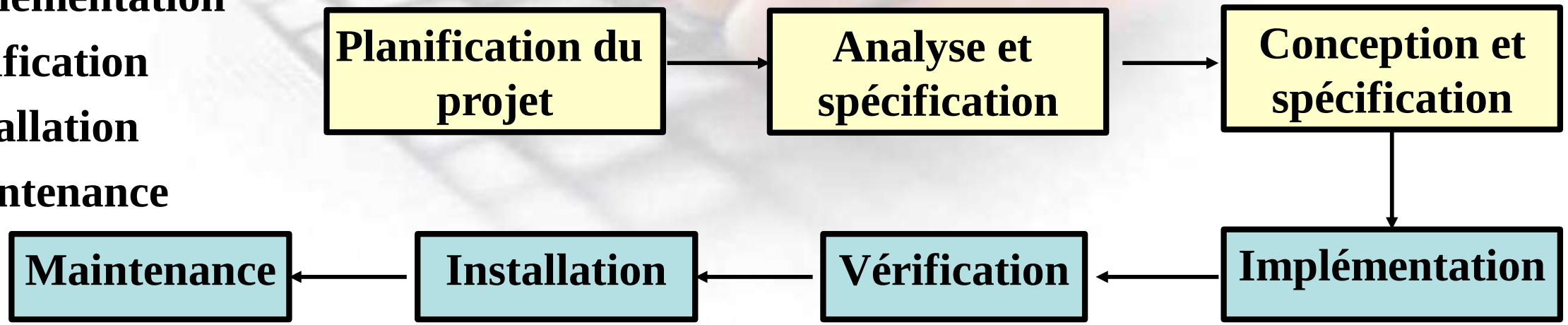
1. le schéma directeur
2. l'étude préalable
3. l'étude détaillée
4. l'étude technique
5. La production
6. la qualification
7. la maintenance



2.2. Le cycle de vie d'un logiciel

Pour une meilleure compréhension, nous allons retenir les activités suivantes comme composantes du cycle de vie. Ces activités constituent les principales activités de développement pour la réalisation d'un logiciel. Ce sont dans l'ordre chronologique:

1. **Planification du projet**
2. **Analyse et spécification**
3. **Conception et spécification**
4. **Implémentation**
5. **Vérification**
6. **Installation**
7. **Maintenance**



Remarque: la documentation, la validation, la vérification et la gestion constituent des activités continues

2.2. Le cycle de vie d'un logiciel

Etape 1: Planification (étude préliminaire) 	❖ Définition globale du problème ❖ Confirmer la faisabilité • évaluation des stratégies possibles • évaluation des ressources, coûts et délais ❖ Produire le calendrier du projet ❖ Trouver le personnel ❖ Lancer le projet	**Documents:** rapport de planification
Etape 2: Analyse des besoins 	❖ Recueille des informations ❖ Exigences fonctionnelles / qualités non fonctionnelles ❖ Spécification du système ❖ Construction de prototypes (pour élaborer la spécification) ❖ Prioriser les éléments de la spécification ❖ Produire et évaluer des solutions alternatives ❖ Examiner les recommandations avec le chef de projet et/ou le client…	**Documents:** cahier des charges, document de spécification (analyse), prototype, plan de test.

2.2. Le cycle de vie d'un logiciel

Etape 3: Conception 	1. Conception architecturale: décomposition et organisation de l'application en modules plus simples définis par une interface. Bases de données, environnement d'exploitation, interfaces. 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 (spécification), prototype, plan de test global et plan de test par module.
Etape 4: Implémentation	❖ Consiste à traduire la conception dans un langage de programmation ou la mise en œuvre en utilisant des outils de développement ❖ Construire les composantes logicielles	**Documents:** Dossiers de programmation, code source commenté, prototypes.

2.2. Le cycle de vie d'un logiciel

Etape 5: Vérification 	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.
Etape 6: Installation	❖ Mise en fonctionnement opérationnel chez les utilisateurs. ❖ Conversion des données. ❖ Parfois restreint (dans un 1er temps) à des utilisateurs sélectionnés.	
Etape 6: Maintenance	❖ Consiste à adapter le SI aux évolutions de l'environnement (correction des anomalies, amelioration, évolution) • Maintenance corrective (corriger erreurs) • Maintenance adaptative (modifications) • Maintenance perfective (améliorations) • *Aussi: maintenance préventive (pour faciliter les* opérations de maintenance à venir).	

2.2. Le cycle de vie d'un logiciel

2.2.8. Répartition des activités

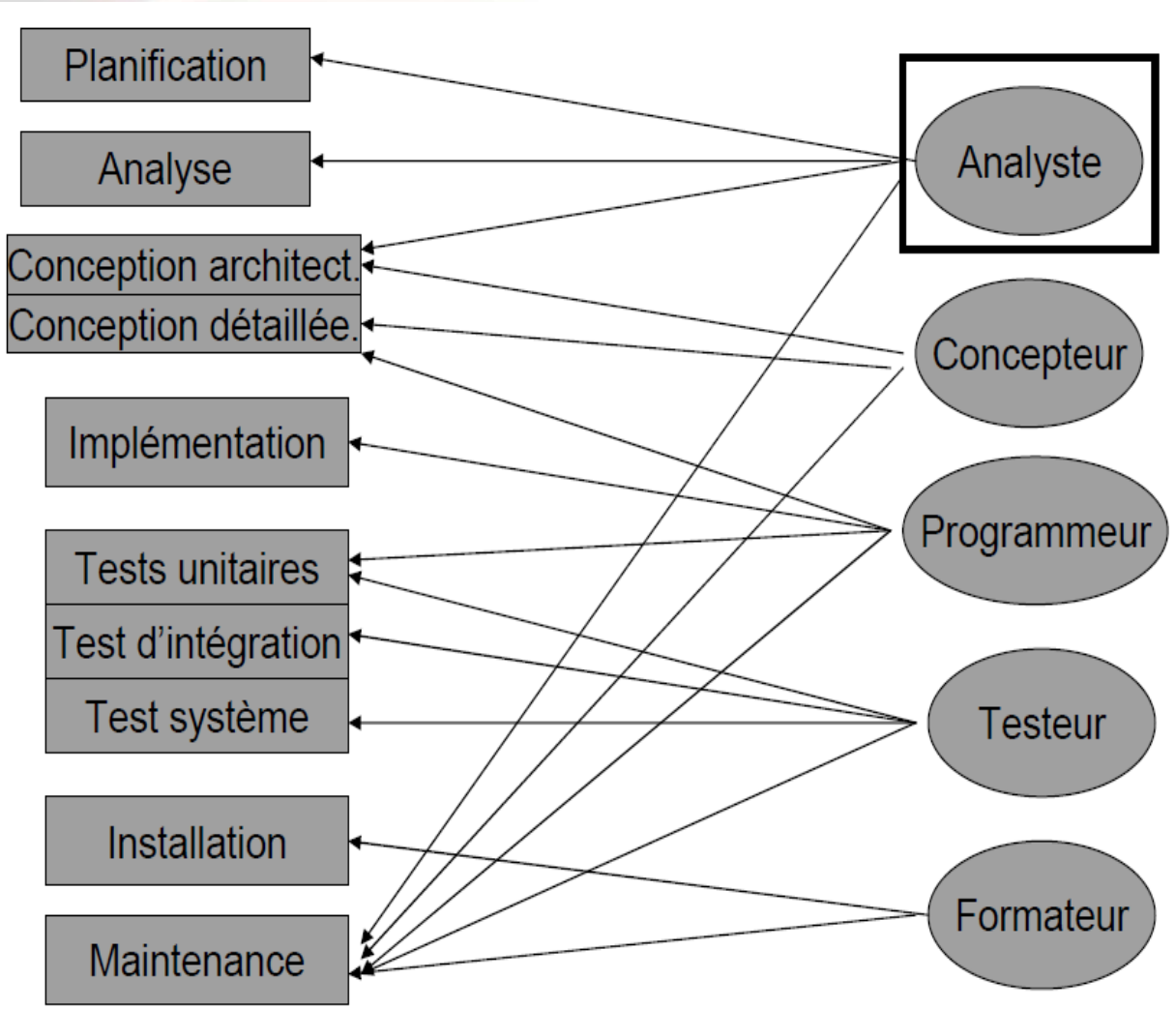
Dans un **projet logiciel** bien conduit, on a:

40 % **Analyse, Spécification, Conception**

20 % **Programmation**

40 % **Intégration, Vérification, Validation**

2.2.9. Rôle d'un analyste

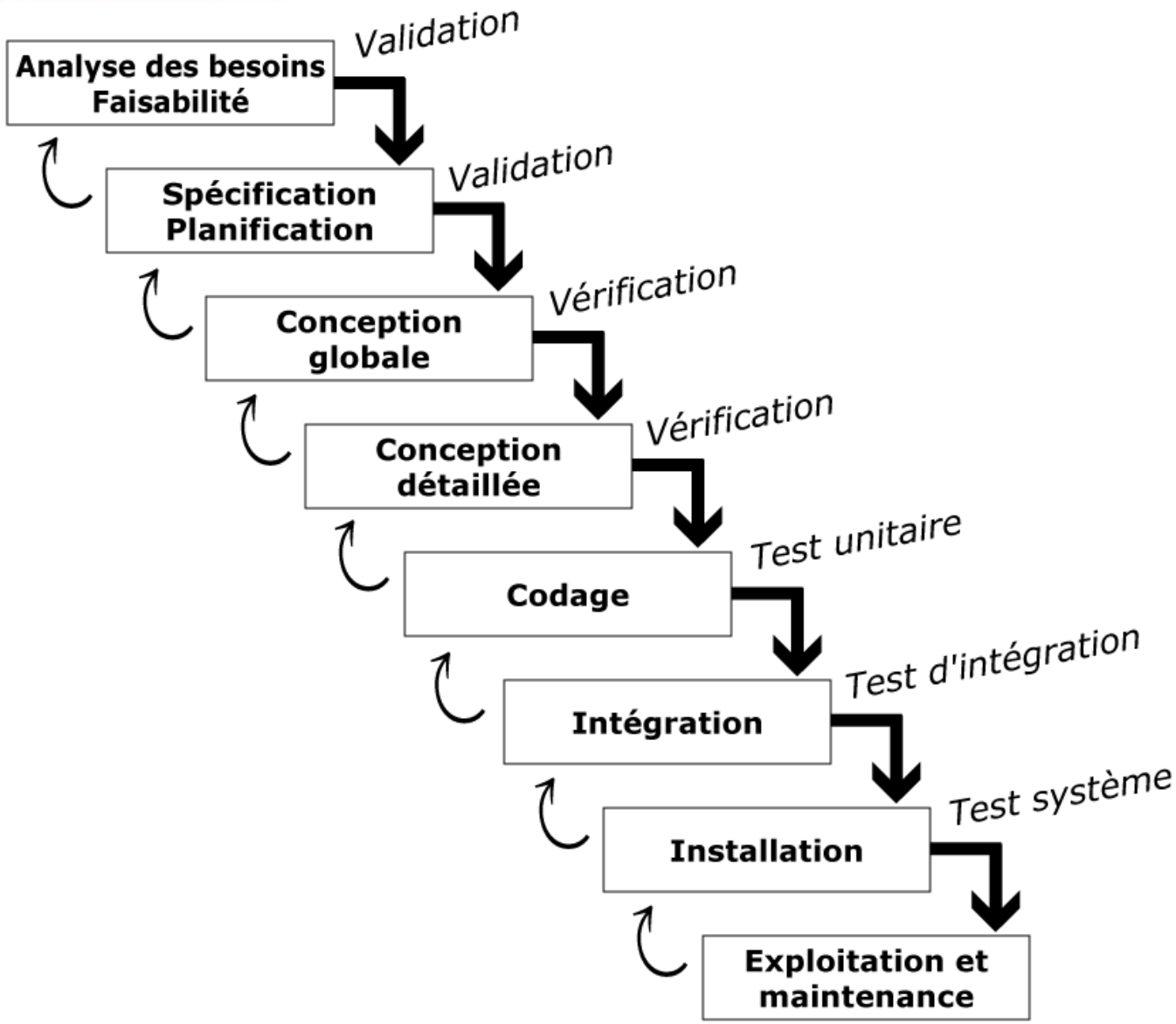


2.3.
Les différents modèles de cycles de vie

2.3. Les différents modèles de cycles de vie

1. Cycle de vie en cascade (ou Waterfall en anglais)

- Dans le cycle de vie en cascade, les étapes se succèdent dans le temps. Chaque étape produit des documents qui sont utilisés pour en vérifier la conformité avant de passer à la suivante.
- Auteur: Herbert D. Benington en 1956
- Le cycle de vie en cascade est souvent schématisé de la manière suivante :



2.3. Les différents modèles de cycles de vie

1. Cycle de vie en cascade (ou Waterfall en anglais)

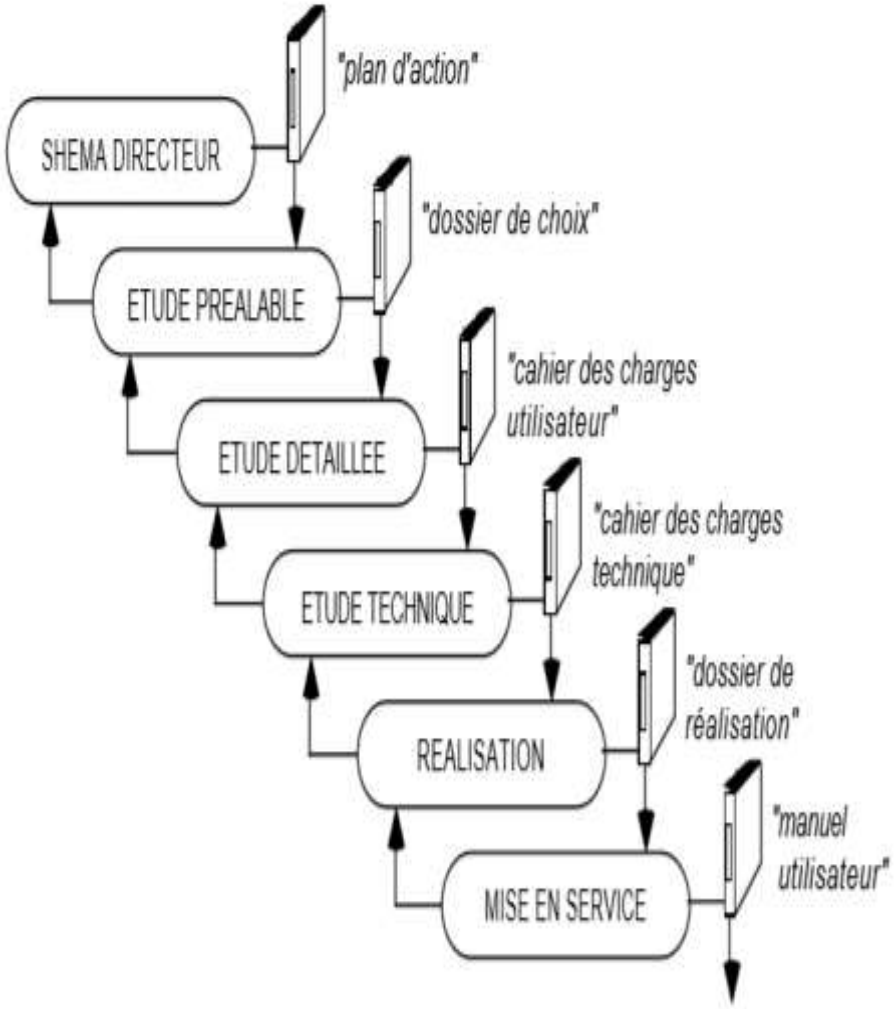
- Avantages:

- sa simplicité et sa souplesse
- offrent une vision très précise des délais de réalisation des projets

- Inconvénients:

- Dès qu'une modification nécessitant la reprise d'une étape, c'est tout le processus qui est impacté
- chaque étape ne validant que l'étape précédente, on prend un risque de voir le projet s'écarter petit à petit de son objectif initial, juste un peu à chaque étape, pour aboutir à un résultat final trop éloigné pour être satisfaisant.

Ce risque est bien réel. C'est la raison pour laquelle a été créé le Cycle en V, avec son système élaboré de validations croisées. Sauf que ces validations prennent un temps conséquent, et on perd alors encore plus de souplesse.



2.3. Les différents modèles de cycles de vie

2. Cycle de vie en V

C'est un modèle de gestion de projet qui part du principe que les procédures de vérification de la conformité du logiciel aux spécifications doivent être élaborées dès les phases de conception.

Il découle du modèle en cascade. Il a été Développé dans les années 1980.

- Avantages:

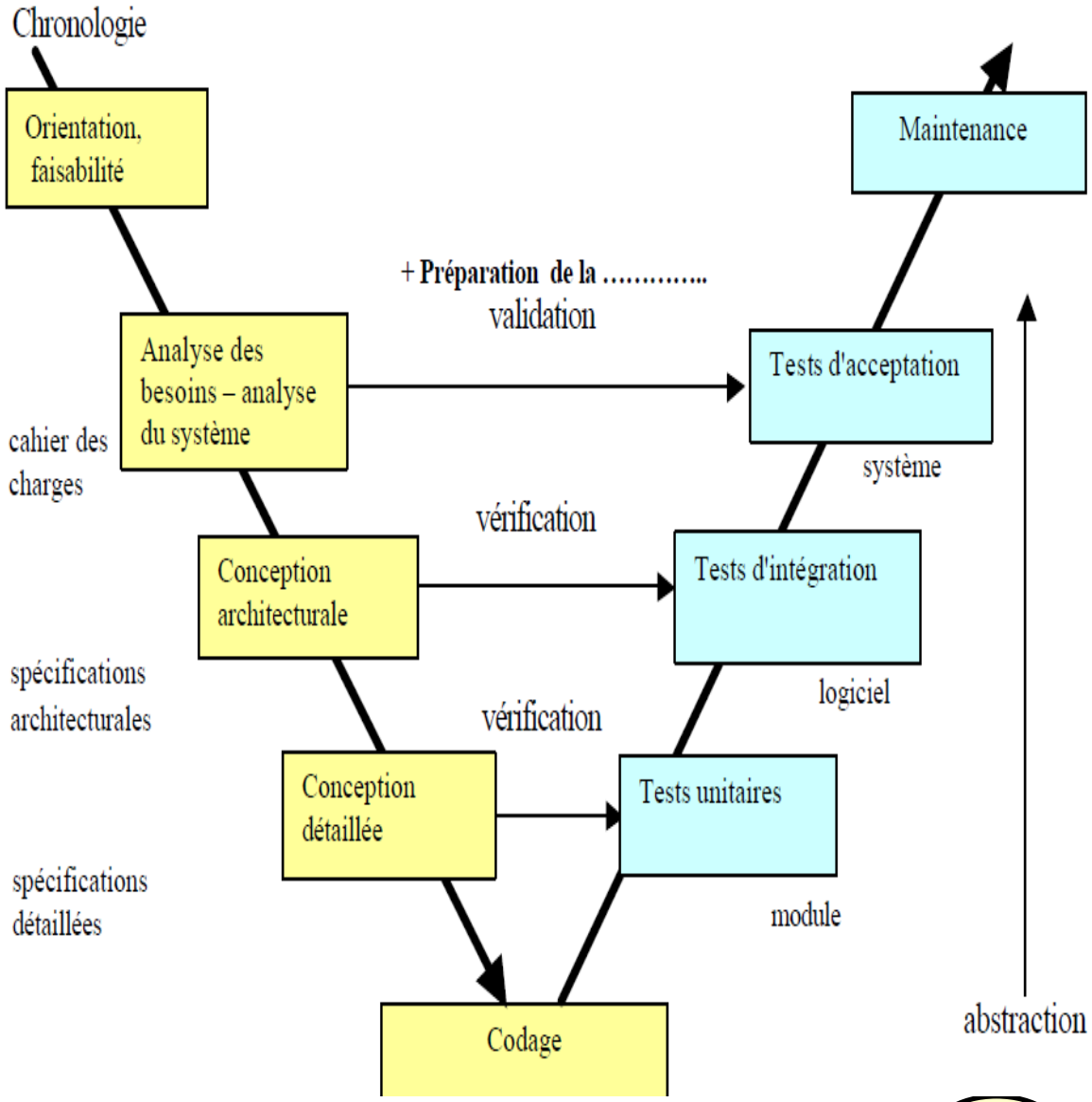
- évite de revenir en arrière incessamment pour redéfinir les spécifications initiales
- facile à mettre en œuvre

- Inconvénients:

- L'inconvénient principal du cycle en V se résume

en deux mots : l'effet tunnel.

Après l'expression de besoin, le client ne voit pas ce qui est fait avant la livraison en recette. Il peut donc rester plusieurs mois sans "voir" le résultat. Il n'a que les comités de pilotage pour l'informer de l'avancement.



2.3. Les différents modèles de cycles de vie

3. Cycle de vie en spirale

C'est un modèle générique de cycle de vie évolutif qui a été proposé par Barry W. Boehm en 1984. Ce modèle, axé sur la maîtrise et la réduction des risques, comporte 4 grandes phases et plusieurs boucles de spirale permettant:

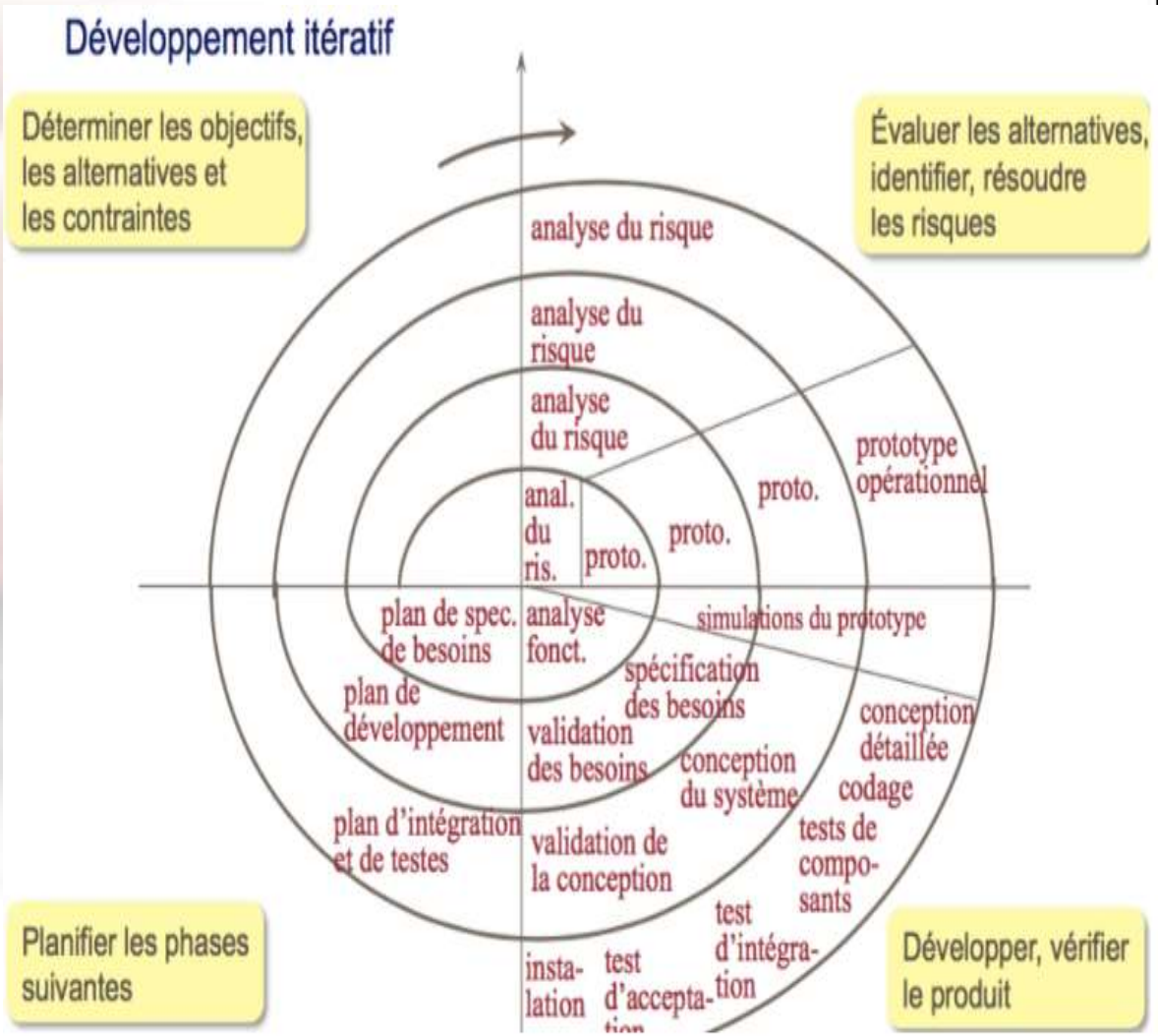
- d'identifier les objectifs propres de la boucle
- les moyens alternatifs pour atteindre les objectifs
- les contraintes de chaque alternative

- Avantages:

- Le client est plus impliqué dans la vie du projet que dans les modèles en V et en cascade.
- Ce modèle favorise plus les interactions entre les équipes du projet

- Inconvénients:

- Les décisions régulières peuvent retarder le processus de développement
- Inadapté aux petits projets aux risques raisonnables
- Effort de gestion important



4. Cycle de vie itératif et incrémental

Il permet de construire progressivement le système souhaité.

Il répète les étapes du cycle en cascade ou en V jusqu'à l'obtention du produit souhaité.

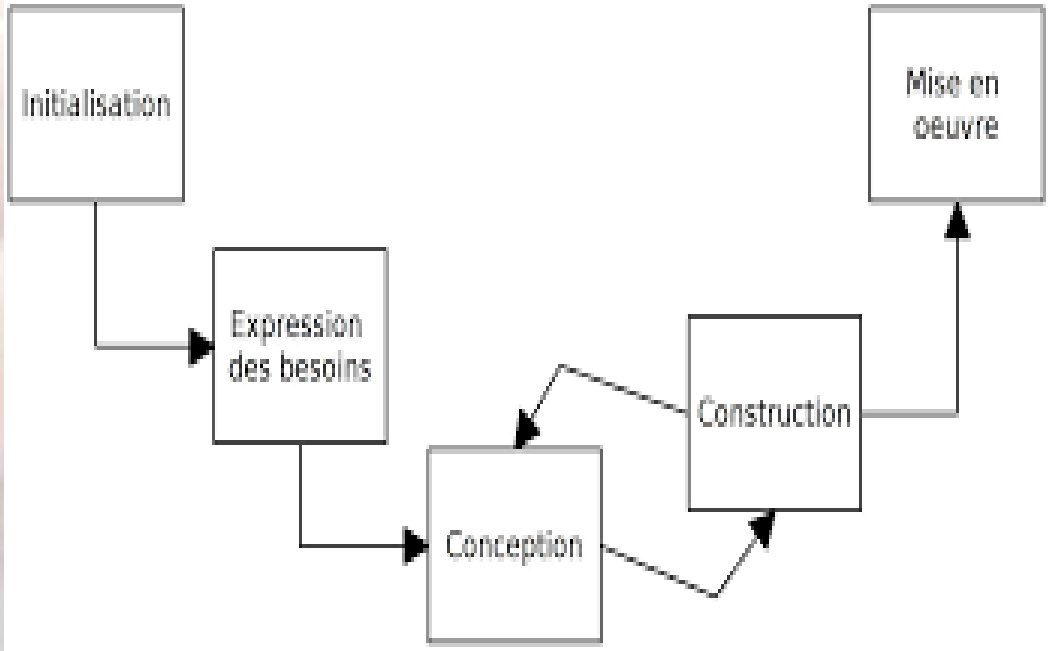
A chaque itération, il y'a une réalisation d'un prototype qui corrige ou complète son précédent. (comme dans le cycle spiral)

L'utilisation d'un prototype constitue une base de travail concrète et mesurable dans le cycle itératif et incrémental.

Développement

5. Cycle de vie en RAD

RAD (Rapid Application Développement) est un modèle qui conjugue le modèle en cascade et le modèle itératif. Son but est de livrer rapidement un minimum de fonctions viables pour assurer un retour sur investissement rapide et éviter l'effet tunnel.



Il se base sur l'utilisation de prototype, une démarche participative de tous les intervenants du projet, une priorisation des délais.

Il se caractérise par:

- Un cycle de développement accéléré
- Des validations fréquentes
- Une approche par composant.

6. Les méthodes agiles

Les méthodes agiles ont bousculé les modes de gestion classique de cycle de vie de logiciel.

Elles tirent leur racine de la méthode RAD.

Les méthodes agiles, apparues dans les années 2000, prennent en compte dans leur modèle de cycle de vie 3 exigences:

- Une forte participation entre développeurs et utilisateurs
- Des livraisons fréquentes de logiciel
- Et une prise en compte de possibles changements dans les besoins des utilisateurs au cours du projet.

Exemples de méthodes agiles: **SCRUM, XP, DSDM** etc.