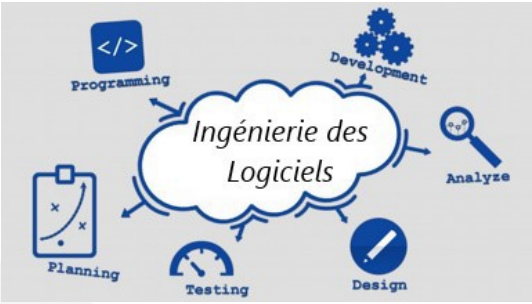
Ingénierie des Logiciels



Dr. Said MEGHZILI

Centre Universitaire de Mila

Département de Mathématique et Informatique

Email: s.meghzili@univ-mila.dz

1.0

Février 2022

Table des matières

I Chapitre 01 : Introduction au génie logiciel

1. Introduction

Attention

- Les Défaillances des Systèmes informatiques à peu prés: 80 % de logiciel et 20 % de matériel.
    - Les problèmes liés à l'informatique sont essentiellement des problèmes de Logiciel.
- La « Crise du logiciel » : Étude sur 8 380 projets[2]* (Standish Group, 1995)
    - ***Succès*** : 16 %
    - ***Problématique*** : 53 %
    - ***Echec*** : 31 % (abandonné)
- Introduction de l'expression: « ***Génie Logiciel*** » (Software Engineering)
    - Comment faire des logiciels de qualité ?
    - Quels sont les critères de qualité pour un logiciel ?

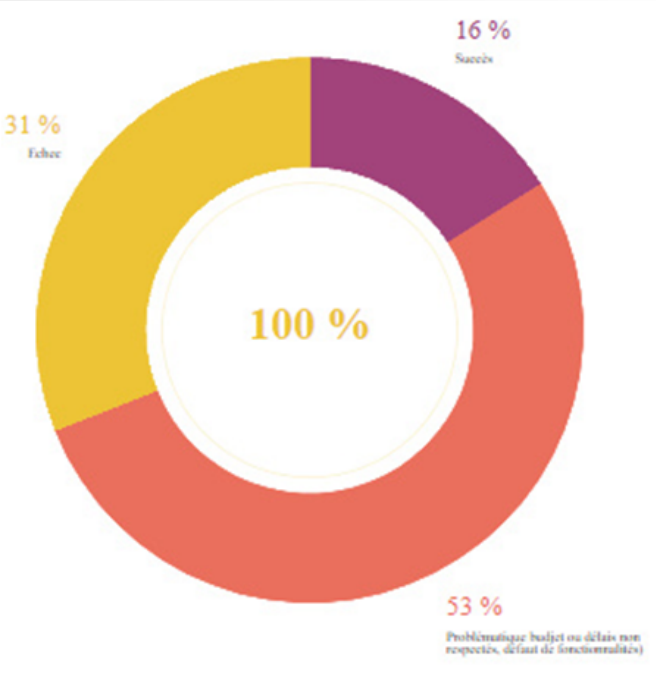


Figure 1.1 : Étude sur des projets informatique

2. Qualités attendues d'un logiciel

Fondamental

Il existe plusieurs qualités attendues d'un logiciel[2]*:

- Fiabilité: le logiciel fonctionne raisonnablement en toutes circonstances, rien de catastrophique ne peut survenir, même en dehors des conditions d'utilisation prévues.
- Interopérabilité: un logiciel doit pouvoir interagir avec d'autres logiciels
- Performance: un logiciel doit satisfaire aux contraintes de temps d'exécution
- Portabilité: Un même logiciel doit pouvoir fonctionner sur plusieurs machines
- Réutilisablité: 80% du code on retrouve partout 20% du code est spécifique
- Utilisabilité: Facilité d'apprentissage et Facilité d'utilisation
- Facilité de maintenance
- Maintenance corrective: Corriger les erreurs.
- Maintenance adaptative: Ajuster le logiciel pour qu'il continue à remplir son rôle compte tenu du l'évolution des Environnements d'exécution.
- Maintenance d'extension

3. Principes d'ingénierie pour le logiciel

Fondamental

Un certain nombre de grands principes se retrouvent dans toutes ces méthodes:

- Généralisation: regroupement d'un ensemble de fonctionnalités semblables en une fonctionnalité paramétrable (héritage...)
- Structuration : façon de décomposer un logiciel
- Abstraction :mécanisme qui permet de présenter un contexte en exprimant les éléments pertinents et en omettant ceux qui ne le sont pas
- Modularité :décomposition d'un logiciel en composants discrets
- Documentation : gestion des documents incluant leur identification, acquisition, production, stockage et distribution
- Vérification : détermination du respect des spécifications établies sur la base des besoins identifiés dans la phase précédente du cycle de vie

4. Composantes du cycle de vie d'un logiciel

Méthode

Ensemble d'activités successives, organisées en vue de la production d'un logiciel[2]*:

1. Analyse de besoins
    - Comprendre les besoins du client.
    - L'environnement et le contexte du futur système.
    - Ressources disponibles.

- Contraintes de performance.

2. Spécification
    - Établir une description claire de ce que doit faire le logiciel.
    - Clarifier le cahier des charges.
3. Conception
    - Conception générale
        - Conception architecturale : déterminer la structure du système.
        - Conception des interfaces : déterminer la façon dont les différentes parties du système agissent entre elles.
    - Conception détaillée : déterminer les algorithmes pour les différentes parties du système
4. Implémentation (codage)
    - Écrire le code du logiciel.
5. Tests
    - Validation
        - Assurer que les besoins du client sont satisfaits.
        - Concevoir le bon logiciel.
    - Vérification
        - Assurer que le logiciel satisfait sa spécification.
        - Concevoir le logiciel correctement.
6. Livraison
   Fournir au client une solution logicielle qui fonctionne correctement :
    - Installation : rendre le logiciel opérationnel chez le client.
    - Formation : apprendre aux utilisateurs à utiliser le logiciel.
    - Assistance : répondre aux questions des utilisateurs.
7. Maintenance
    - Mettre à jour et améliorer le logiciel pour assurer sa pérénité.
    - Pour limiter le temps et les coûts de maintenance, des efforts doivent être faits sur les étapes précédentes.

5. Modèles de cycle de vie d'un logiciel

5.1. Processus de Développement Séquentiels

Modèle en cascade

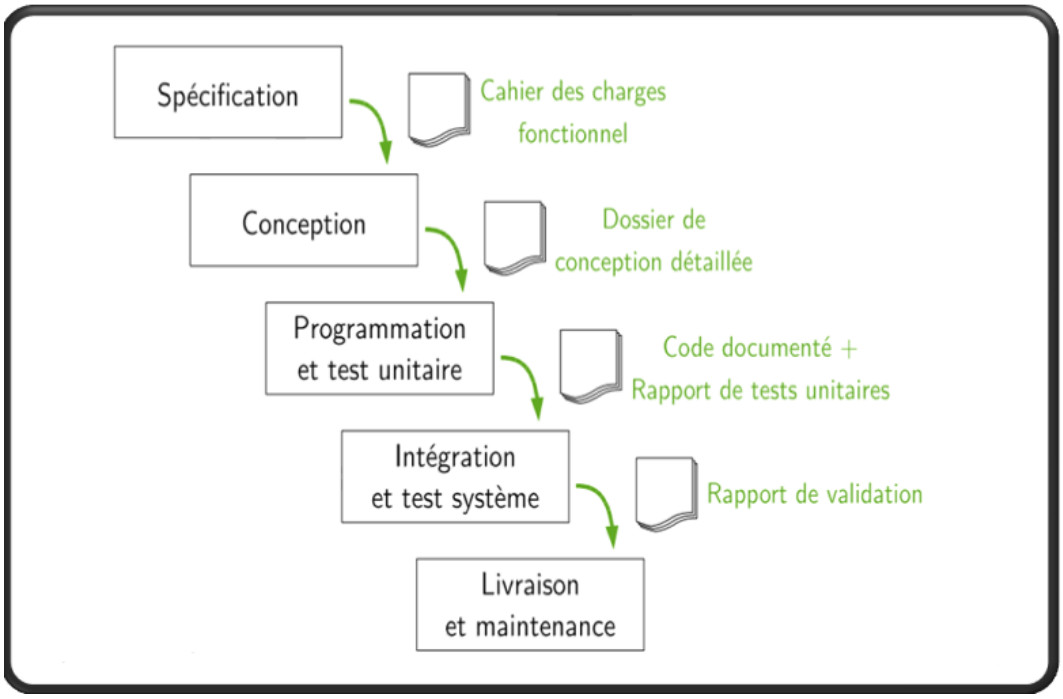


Figure 1.2 : Modèle en cascade

Caractéristiques :

- Hérite des méthodes classiques d'ingénierie.
- La découverte d'une erreur entraîne retour à la phase à l'origine de l'erreur et nouvelle cascade, avec de nouveaux documents..
- Coût de modification d'une erreur important.
- Mais pas toujours adapté à une production logicielle, en particulier si les besoins du client changeants ou difficiles à spécifier.

Modèle en V

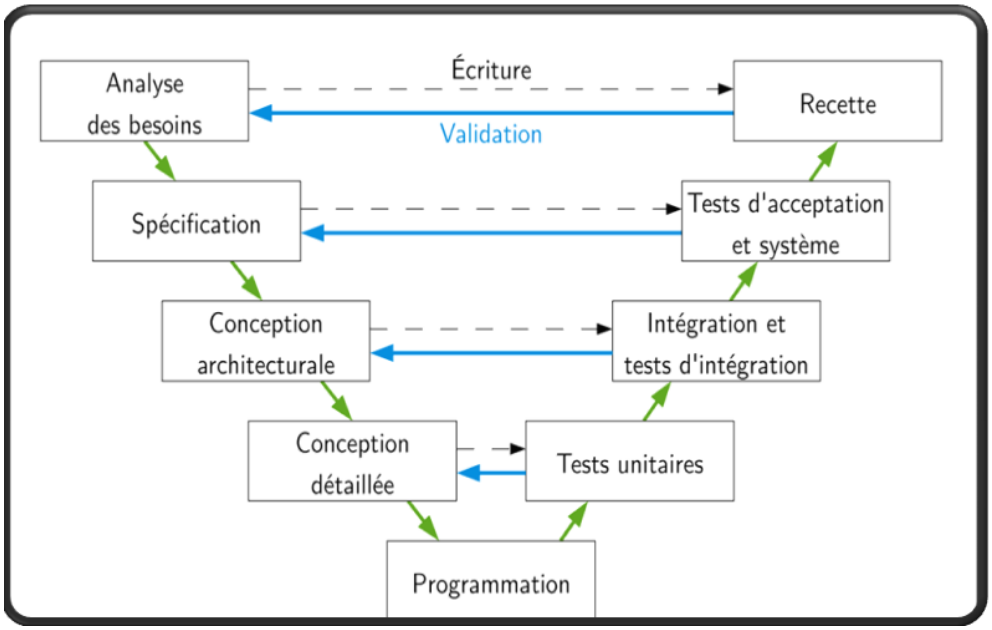


Figure 1.3 : Modèle en V

Caractéristiques:

- Variante du modèle en cascade.
- Mise en évidence de la complémentarité des phases menant à la réalisation et des phases de test permettant de les valider.

Inconvénients :

- Les mêmes que le cycle en cascade
- Processus lourd, difficile de revenir en arrière
- Nécessite des spécifications précises et stables
- Beaucoup de documentation.

5.2. Processus de Développement Itératifs

Modèle non linéaire

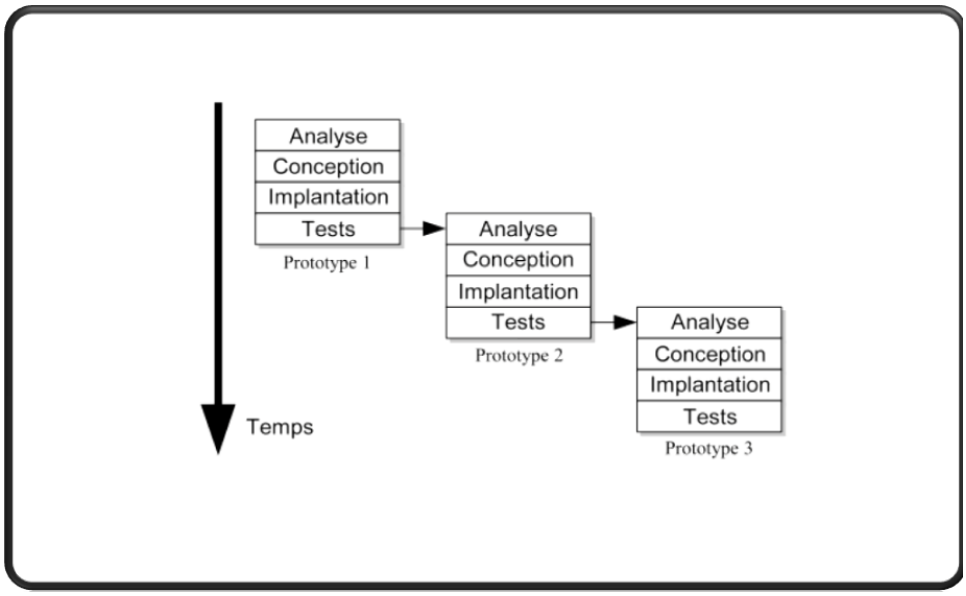


Figure 1.4 : Processus de développement itératifs

Avantages :

- Capture des besoins continue et évolutive.
- Détection précoce des erreurs.
- État d'avancement connecté à la réalité.
- Implication des clients/utilisateurs.

Inconvénients :

- Explosion des besoins.
- Difficile de définir la dimension d'un incrément.
- Nécessite une direction rigoureuse.

5.3. Processus Unifié

Caractéristiques

- Méthode itérative et incrémentale.
- Piloté par les cas d'utilisation.
- Centré sur l'architecture : Modélisation orientée objets, utilise la modélisation visuelle (UML) et Fondé sur la production et l'utilisation de composants.

Méthode : Les phases de développement d'un processus unifié

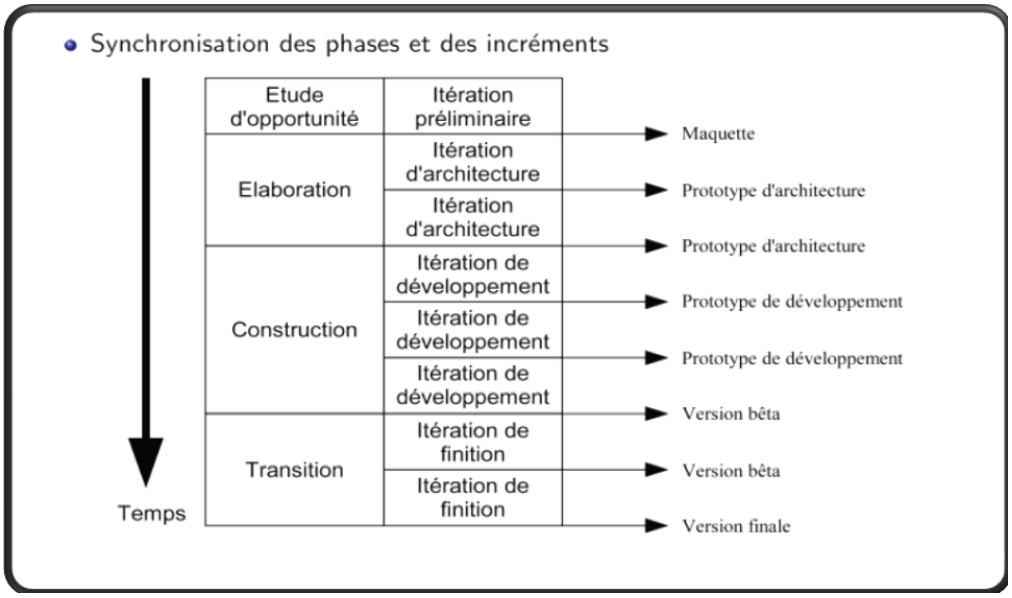


Figure 1.5 : Processus unifié

- Étude d'opportunité (préparation)
    - Étude de marché, estimation du coût.
    - Capture des besoins majeurs et analyse préliminaire.
- Élaboration
    - Capture et analyse des besoins.
    - Choix de l'architecture.
- Construction
    - Répartition des tâches sur plusieurs équipes.
    - Rédaction de la documentation finale.
- Transition
    - Fabrication, livraison.
    - Installation, formation.
    - Support technique.

5.4. Méthodes Agiles

Définition

Une méthode agile est une méthode de développement informatique permettant de concevoir des logiciels en impliquant au maximum le demandeur (client) :

- Plus grande réactivité à ses demandes.
- Plus pragmatiques que les méthodes traditionnelles.
- Recherche de la satisfaction réelle du besoin du client.

Exemple : Méthodes agiles

- XP (EXtreme Programming),
- DSDM (Dynamic Software Development Method),
- ASD (Adaptative Software Development),
- CCM (Crystal Clear Methodologies),
- SCRUM,
- FDD (Feature Driven development)

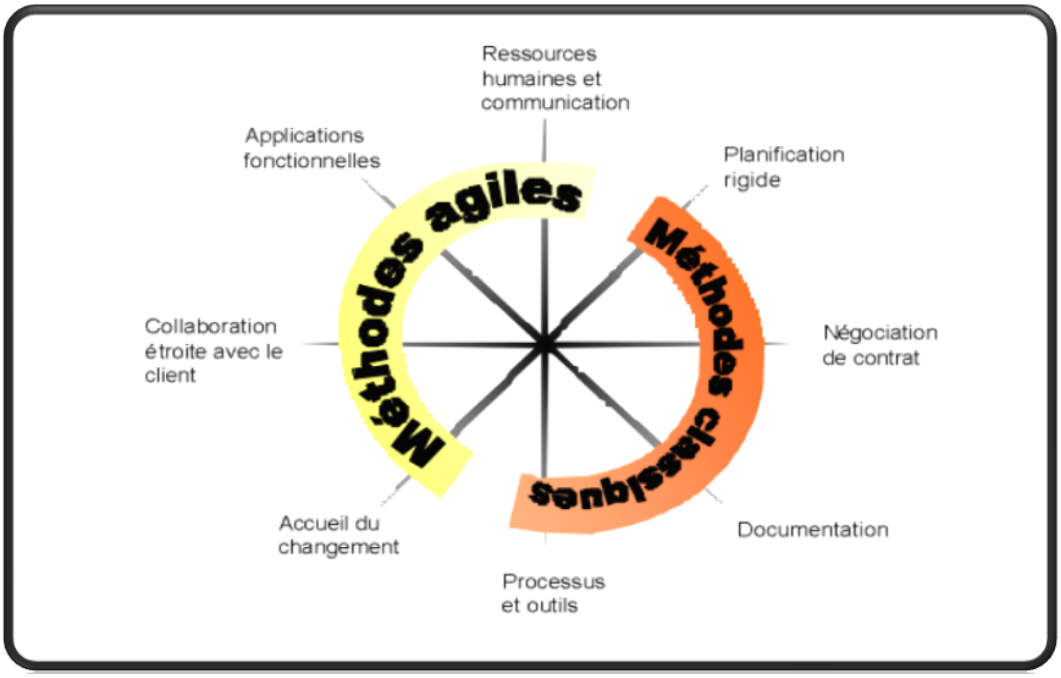


Figure 1.6 : Méthodes agiles

Caractéristiques :

- itératives à planification souple.
- Itérations très courtes.
- S'opposent à la procédure et à la spécification à outrance.

Les priorités d'une méthode agile :

1. Priorité aux personnes et aux interactions sur les procédures et les outils.
2. Priorité aux applications fonctionnelles sur une documentation pléthorique.
3. Priorité de la collaboration avec le client sur la négociation de contrat.
4. Priorité de l'acceptation du changement sur la planification.

Avantages/ Inconvénients :

- Développement rapide en adéquation avec les besoins.
- Mais pas de spécification, documentation = tests et maintenance.

6. Conclusion

Dans ce chapitre, nous avons présenté la notion générale de génie logiciel. Dans un premier temps, nous avons introduit les qualités attendues d'un logiciel. Ensuite, nous avons présenté les principes d'ingénierie du logiciel. Par la suite, nous avons abordé les composants du cycle de vie d'un logiciel. Enfin, nous avons présenté les modèles de cycle de vie d'un logiciel.

Références

2 Cours « Génie logiciels », Pierre Gérard, Université de Paris 13, 20072008.