1.1 Notion de Processus

Rappel des objectifs des processus.

Un processus définit QUI fait QUOI, QUAND et COMMENT pour atteindre un certain objectif. Les principaux objectifs peuvent être :

- Construction des modèles d'un ou plusieurs systèmes,
- Organisation et gestion de la totalité du cycle de vie du projet,
- Gestion des risques,
- construction répétitive de produits logiciels de qualité constante.

Le cycle de vie du logiciel est un ensemble cohérent d'*activités* pour spécifier, concevoir, implémenter et tester des systèmes logiciels. Il y a alors différents *livrables* associés chaque activité. Les livrables peuvent être des cahiers de charges, des modèles d'analyse, des modèles de conception, du code source, des manuels d'utilisation ou autres.

1.1.1 Activités

a. - *Etude de faisabilité* : déterminer si le développement proposé est réalisable.

   - *Analyse du marché* : déterminer s'il y a un marché potentiel pour ce produit.

b. - *Expression des besoins* : déterminer quelles sont les fonctionnalités que le logiciel doit offrir.

   - *Recueil des besoins* : obtenir les besoins à partir des utilisateurs.

   - *Analyse du domaine* : déterminer quelles sont les tâches et les structures qui sont communes à ce problème.

c. - *Planification du projet* : déterminer comment développer le logiciel.

   - *Analyse des coûts* : déterminer les estimations du coût.

   - *Assurance qualité* : déterminer les activités qui permettront d'assurer la qualité du produit pour garantir la satisfaction du client (selon les objectifs contractuels).

   - *Structure work-breakdown* : déterminer les sous-tâches nécessaires pour développer le produit.

d. - *Conception* : déterminer comment le logiciel doit fournir la fonctionnalité.

   - *Conception architecturale* : concevoir la structure du système.

   - *Conception d'interfaces* : spécifier les interfaces entres les différentes parties du système.

   - *Conception détaillée* : concevoir les algorithmes pour les parties individuelles.

e. - *Implémentation* : construire le logiciel.

f. - *Test* : exécuter le logiciel avec des données pour permettre de s'assurer que le logiciel opère correctement.

   - *Test unitaire* : tester par le développeur original.

   - *Test d'intégration* : tester durant l'intégration du logiciel.

   - *Test du système* : tester le logiciel dans un environnement.

   - *Test d'acceptation* : tester pour satisfaire le client.

g. - *Livraison* : fournir au client une solution logicielle efficace.

   - *Installation* : mettre le logiciel disponible dans le site opérationnel du client.

   - *Formation* : former les utilisateurs à utiliser le logiciel et répondre à leurs questions.

h. - *Maintenance* : mettre-à-jour et améliorer le logiciel pour garantir une utilisation efficace continue.

1.1.2 Documents

Les résultats des différentes activités sont représentés par plusieurs types de documents, dont les plus importants sont :

- *Spécification des besoins logiciels* : décrit ce que doit faire le logiciel.

- *Modèle d'objet* : montre les principaux objets / classes.

- *scénarios de cas d'utilisation* : montrent les séquences des comportements possibles du point de vue utilisateur.

- *Plan du projet* : décrit l'ordre des tâches et estime les besoins en matière *temps* et *efforts*.

- *Plan de test du logiciel* : décrit comment le logiciel serait testé pour garantir un comportement correct.

- *Conception du logiciel* : décrit la structure du logiciel

- *Plan assurance qualité du logiciel* : décrit les activités à effectuer pour assurer la qualité.

- *Manuel d'utilisation* : décrit comment utiliser le logiciel final.

- *Code source* : le code du produit final.

- *Rapport du test* : décrit quels sont les tests effectués et quel était le comportement du système.

1.2 Modèles de cycles de vie

Un modèle d'un processus de développement est une représentation abstraite de ce processus. Il présente la description du processus d'une perspective particulière. Nous avons quatre principaux modèles de cycles de vie de logiciel :

1.2.1 Modèle séquentiel linéaire

Appelé aussi le modèle en cascade (*Waterfall model*) du fait que le diagramme ressemble à des séries de cascades. C'est un modèle adapté seulement quand tous les besoins sont bien déterminés à priori.

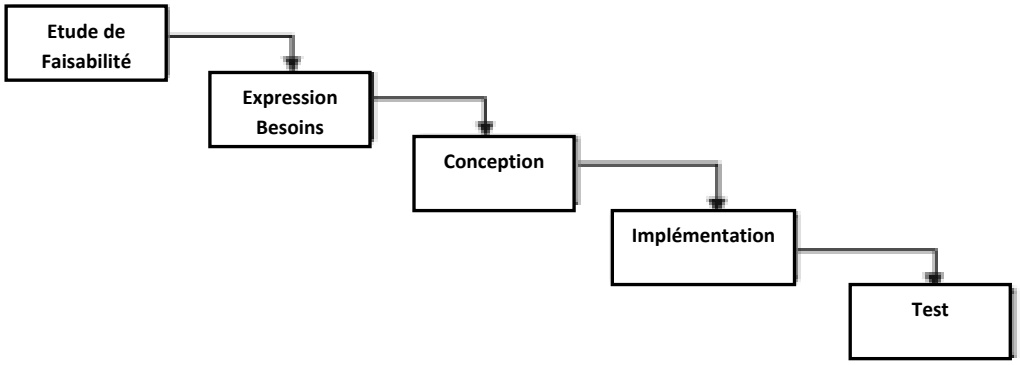

Fig. 1. Modèle en cascade.

Le modèle d'origine a été décrit initialement par Royce (1970), mais actuellement il y a plusieurs versions qui peuvent mettre l'accent sur certaines activités et négliger d'autres selon le besoins. Dans la version présentée par la figure 1, les activités de *planification du projet* sont inclues dans la phase expression des besoins. D'autre part, les phases *livraison* et *maintenance* ont été négligées dans cette version du modèle.

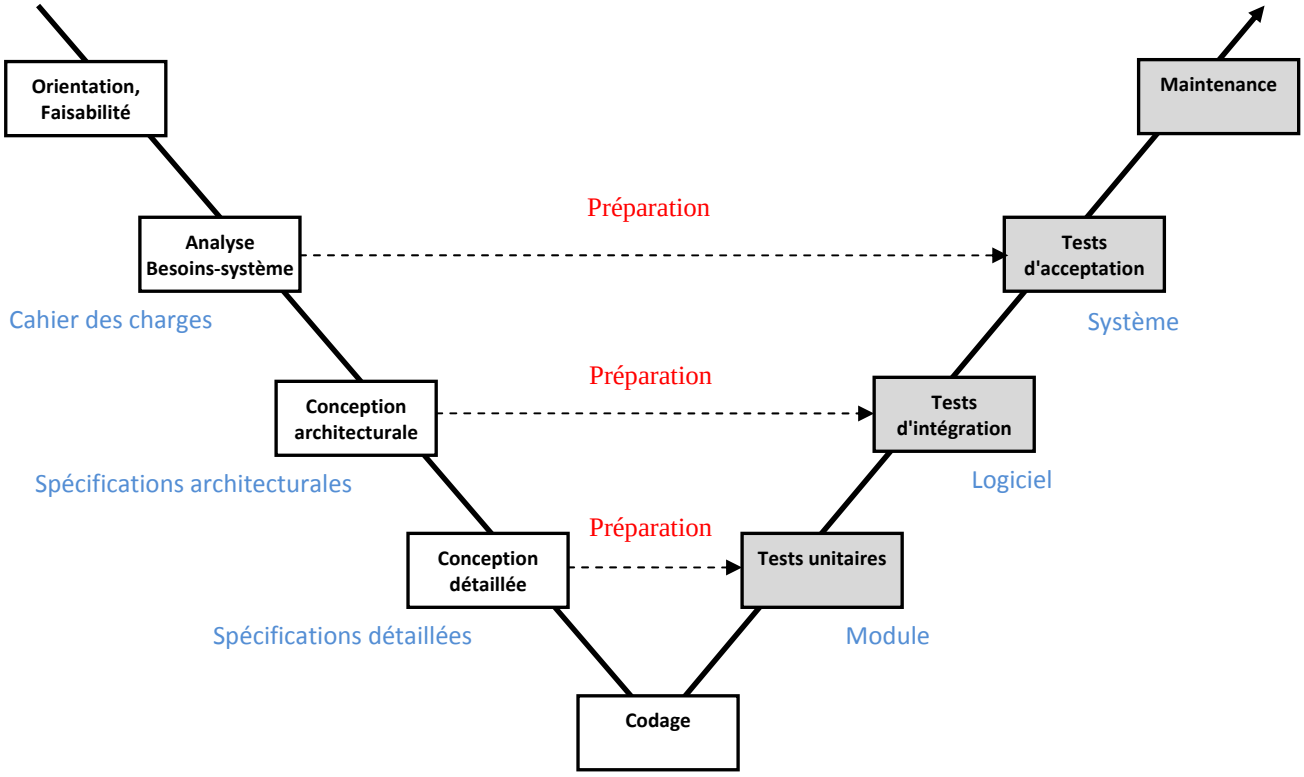

Fig. 2. Modèle séquentiel en V.

1.2.3 Modèle incrémental

Proposé par D. L. Parnas (1979) pour concevoir et livrer au client un sous-ensemble opérationnel du système global. Le processus continue d'itérer, comme le montre la figure 3, à travers le cycle de vie global avec des incréments additionnels.

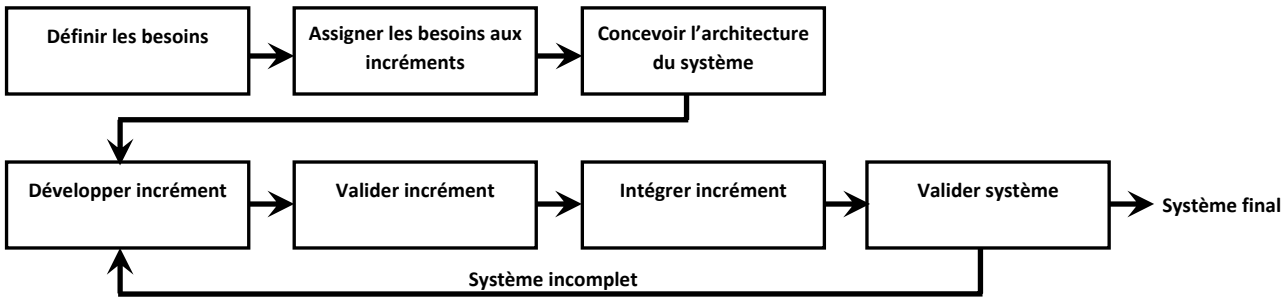

Fig. 3. Modèle incrémental.

1.2.2 Modèle de prototypage

Ce modèle de cycles de vie construit un prototype pour tester les concepts et les besoins. Après accord du client, le développement du logiciel se poursuit en passant toujours par les mêmes phases du modèle précédent.

1.2.4 Modèle en spirale

C'est un modèle qui a été introduit par B. Boehm (1988). L'image du modèle est une spirale qui commence au milieu et qui réitère continuellement les tâches de base (figure 4).

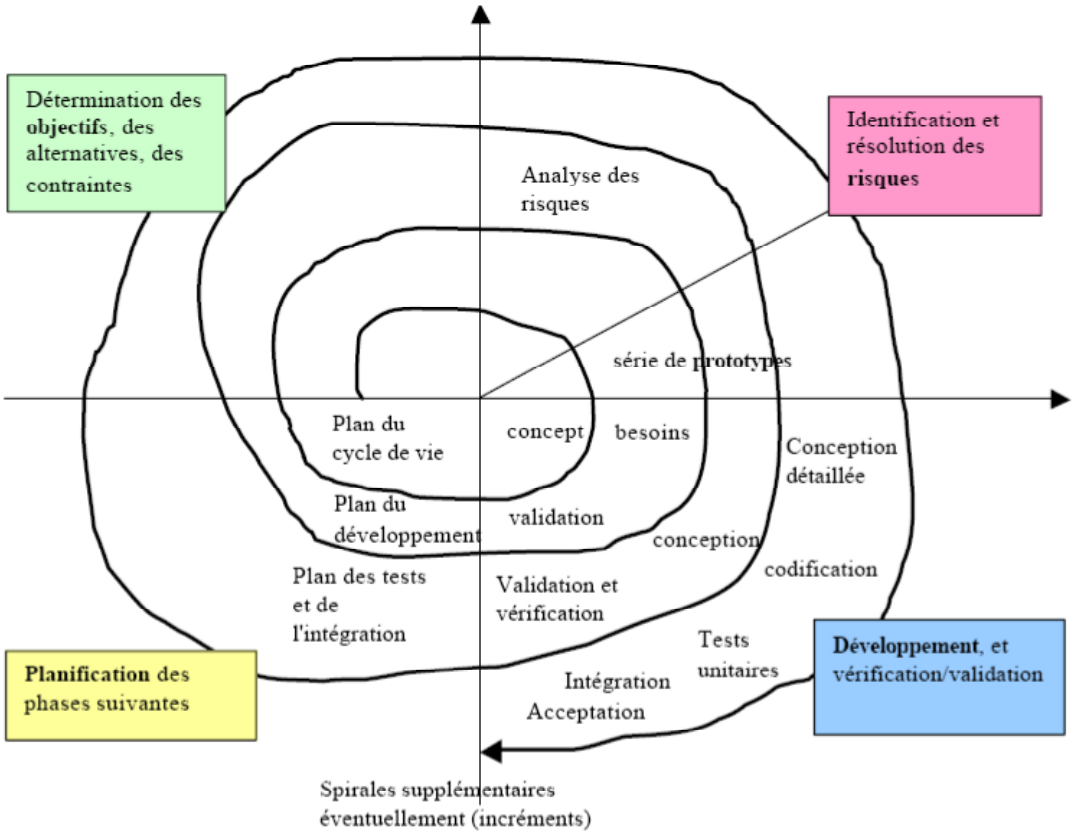

Fig. 4. Modèle en spirale.

Les principaux risques et leurs remèdes, tels que définis par Boehm, sont les suivants :

- Défaillance de personnel : embauches de haut niveau, formation mutuelle, leaders, adéquation profil/fonction, etc.
- Calendrier et budgets irréalistes : estimation détaillée, développement incrémental, réutilisation, élagage des besoins, etc.
- Développement de fonctions inappropriées : revues d'utilisateurs, manuel d'utilisation précoce, etc.
- Développement d'interfaces utilisateurs inappropriées : maquettage, analyse des tâches, etc.
- Produit "*plaqué or*" : analyse des coûts/bénéfices, conception tenant compte des coûts, etc.
- Volatilité des besoins : développement incrémental de la partie la plus stable d'abord, masquage d'information, etc.
- Problèmes de performances : simulations, modélisations, essais et mesures, maquettage,
- Exigences démesurées par rapport à la technologie : analyses techniques de faisabilité, maquettage, etc.
- Tâches ou composants externes défaillants : audit des sous-traitants, contrats, revues, analyse de compatibilité, essais et mesures, etc.

1.2.5 Autres modèles

Il y a plusieurs autres modèles ou variantes de modèles déjà existants tels que le modèle de développement des systèmes formels, le développement basé réutilisation, la programmation extrême, etc.