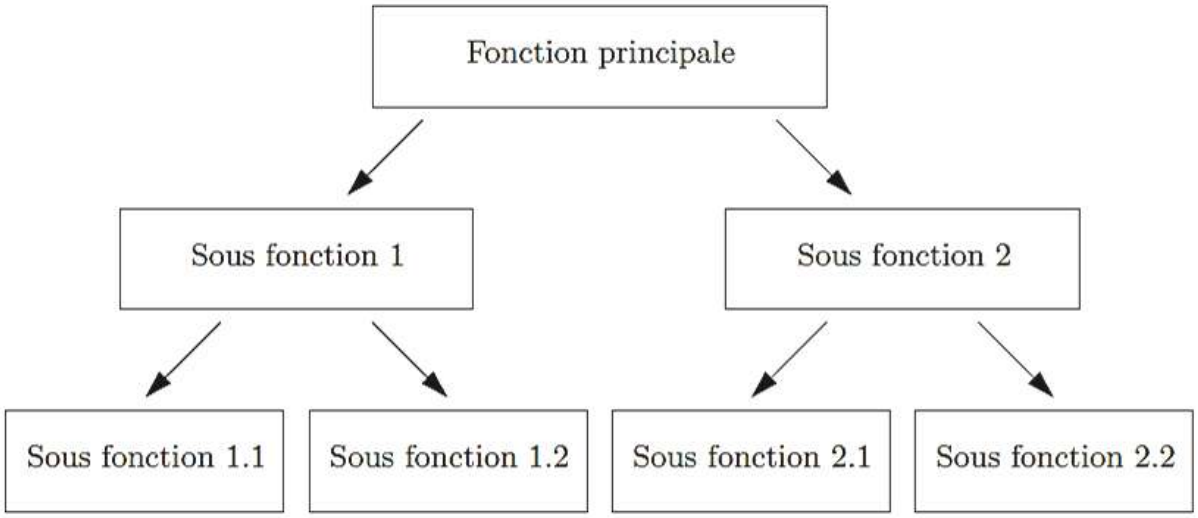
CHAPITRE 3 : LES METHODES DE DEVELOPPEMENT LOGICIEL

Une méthode définit une démarche en vue de produire des résultats. *Une méthode* permet d'assister une ou plusieurs étapes du cycle de vie du logiciel Il existe différentes manières pour classer ces méthodes :
Dans le cadre de cours, nous distinguerons alors quatre (4) types des méthodes à savoir :

les méthodes fonctionnelles, basées sur les fonctionnalités du logiciel ;
les méthodes objet, basées sur différents modèles (*statiques, dynamiques et fonctionnels*) de développement logiciel.
Les méthodes adaptatives ou Agiles, basées sur le changement des besoins ;
Les méthodes spécifiques, basées sur les découpages temporels particuliers.

1. LES METHODES FONCTIONNELLES

Les méthodes fonctionnelles ont pour origine la programmation structurée.
Cette approche consiste à décomposer une fonctionnalité (*ou fonction*) du logiciel en plusieurs sous fonctions plus simples. La programmation peut ensuite être réalisée soit à partir des fonctions de haut niveau (***développement « top-down »),*** soit à partir des fonctions de bas niveau (développement « ***bottom-up*** »).



2. LES METHODES OBJET

Les approches objet sont basées sur une modélisation du domaine d'application.
Les « *objets* »sont une abstraction des entités du monde réel. De façon générale, la modélisation permet de réduire la complexité et de communiquer avec les utilisateurs

Dans les méthodes objet, on distingue trois aspects :

 Un aspect statique : Dans lequel, on identifie les objets, leurs propriétés et leurs relations ;

 Un aspect dynamique : Dans lequel, on décrit les comportements des objets, en particuliers leurs états possibles et les évènements qui déclenchent les changements d'état ;

 Un aspect fonctionnel : qui, à haut niveau, décrit les fonctionnalités du logiciel, ou, a plus bas niveau, décrit les fonctions réalisées par les objets par l'intermédiaire des méthodes.

Les intérêts des approches objet sont les suivants :

 Les approches objet sont souvent qualifiées de « *naturelles* » car elles sont basées sur le domaine d'application. Cela facilite en particulier la communication avec les utilisateurs.

  Ces approches supportent mieux l'évolution des besoins car *la modélisation est plus stable*, et *les évolutions fonctionnelles ne remettent par l"architecture du système en cause*.
Les approches objet facilitent la réutilisation des composants

3. LES METHODES ADAPTATIVES

Les méthodes dites « adaptatives » sont subdivisées en 2 parties notamment :
les méthodes prédictives et *les méthodes agiles (adaptatives)*.

3.1. LES METHODES PREDICTIVES

Ce sont des méthodes qui correspondent à un cycle de vie du logiciel en cascade ou en V, sont basées sur une planification très précise et très détaillée, qui a pour but de réduire les incertitudes liées au développement du logiciel. Cette planification rigoureuse ne permet pas d'évolutions dans les besoins des utilisateurs, qui doivent donc être figes dès l'étape de définition des besoins.

3.2. LES METHODES AGILES

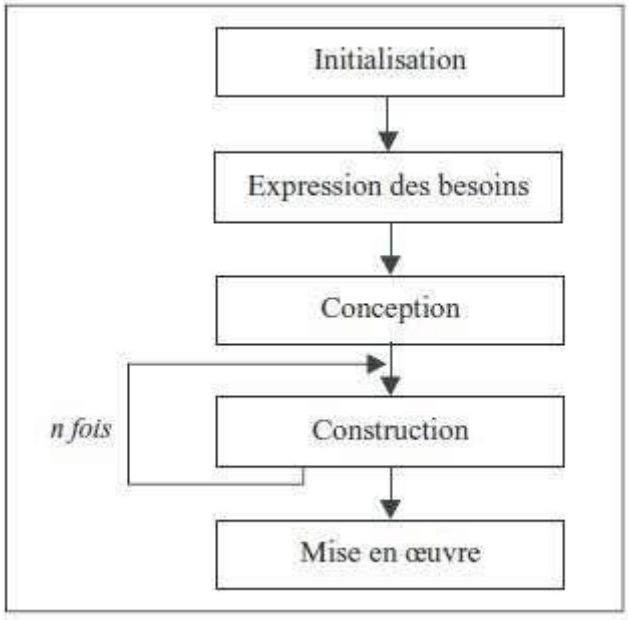
Ce sont des méthodes qui correspondent à un cycle de vie itératif, qui considèrent que les changements (*des besoins des utilisateurs, mais également de l"architecture, de la conception, de la technologie*) sont inévitables et doivent être pris en compte par les modèles de développement. Ces méthodes privilégient la livraison de fonctionnalités utiles au client à la production de documentation intermédiaire sans intérêt pour le client.

elles préconisent en général des durées de cycle de vie
des projets ne dépassant pas un an. Parmi les méthodes agiles, les plus usuelles ; on peut citer :

- La méthode RAD (*Rapid Application Development*) ;
- La méthode DSDM (*Dynamic Systems Development Method*);
- La méthode XP (*Programmation eXtrême*);
- La méthode SCRUM.

2.1. La méthode RAD

La méthode RAD, est un modèle linéaire (présentoir), structuré en cinq phases, et dont le modèle itératif intervient à la phase Construction du logiciel en vu de la séquencer en plusieurs modules Successivement livrés.



La participation des utilisateurs est placée au cœur du cycle. En effet, le déroulement d'une phase comprend une ou plusieurs sous-phases, et chaque sousphase

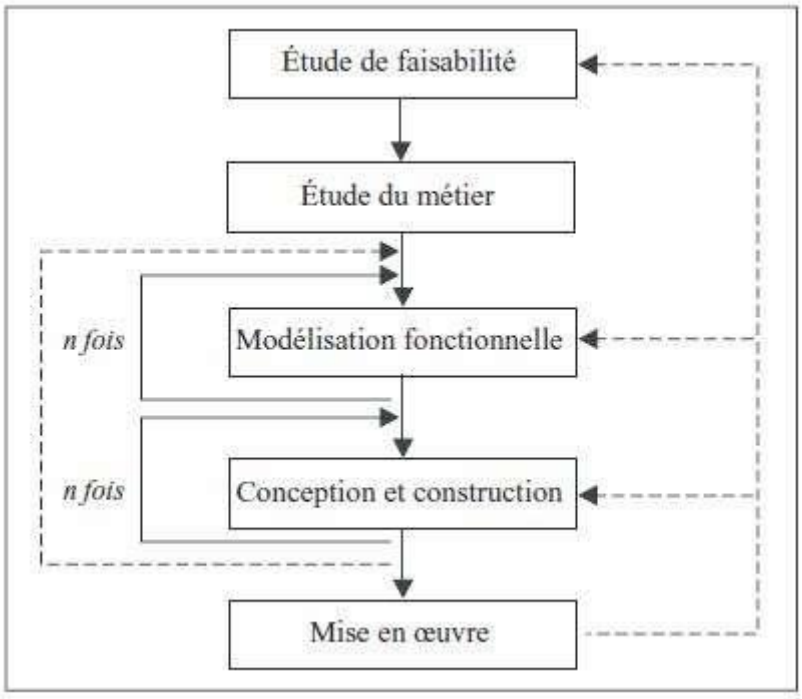
présente une structure à trois temps, dans laquelle la tenue d'une session participative joue un rôle central. Des travaux préparatoires rassemblent et construisent le matériau (modèle ou prototype) qui sera ensuite discuté par les différents acteurs et ajusté.

2.2. La méthode DSDM

La méthode DSDM a évolué au cours des années. L'actuelle version distingue *le cycle de vie du système* et *le cycle de vie du projet*. Le premier comprend, outre les phases du projet lui-même, une phase de pré-projet qui doit conduire au lancement du projet et une phase post-projet qui recouvre l'exploitation et la maintenance de

l’application.

Le cycle de vie du projet comprend cinq phases, dont deux sont cycliques. Les flèches pleines indiquent un déroulement normal. Les flèches en pointillé montrent des retours possibles à une phase antérieure, soit après la phase Conception et construction, soit après celle de Mise en œuvre.

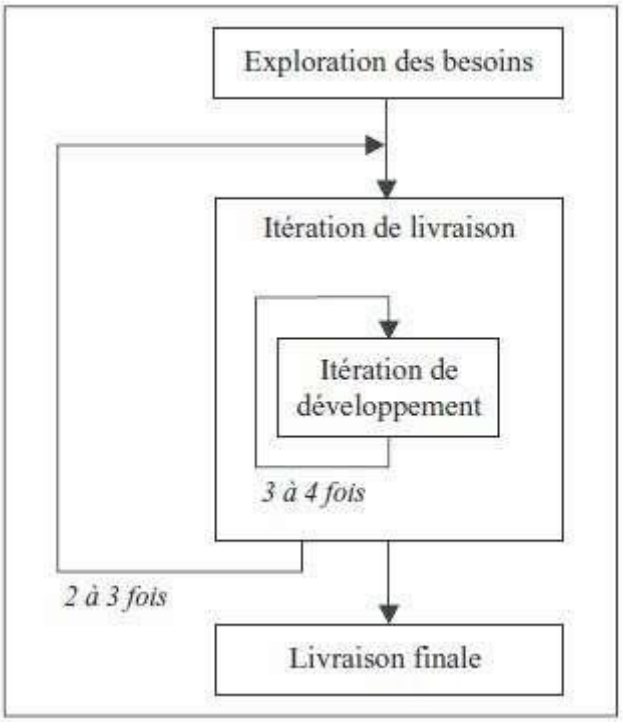


Après une Étude de faisabilité, la phase Étude du métier permet, à travers des ateliers (*workshops*) entre équipe de projet et manageurs, de définir le périmètre du projet, avec une liste d’exigences prioritaires et une architecture fonctionnelle et technique du futur système.

La phase Modélisation fonctionnelle est une suite de cycles. Chacun permet de définir précisément les fonctionnalités souhaitées et leur priorité. L’acceptation par toutes les parties prenantes d’un prototype fonctionnel, sur tout ou partie du périmètre, permet de passer à la phase Conception et construction. L’objectif de cette phase est de développer un logiciel testé, par des cycles successifs de développement/acceptation par les utilisateurs.

2.3. Le méthode XP

La méthode XP, focalisée sur la partie programmation du projet, propose un modèle itératif avec une structure à deux niveaux : d'abord des itérations de livraison (*release*), puis des itérations de développement. Les premières conduisent à livrer des fonctionnalités complètes pour le client, les secondes portent sur des éléments plus fins appelés scénarios qui contribuent à la définition d'une fonctionnalité.

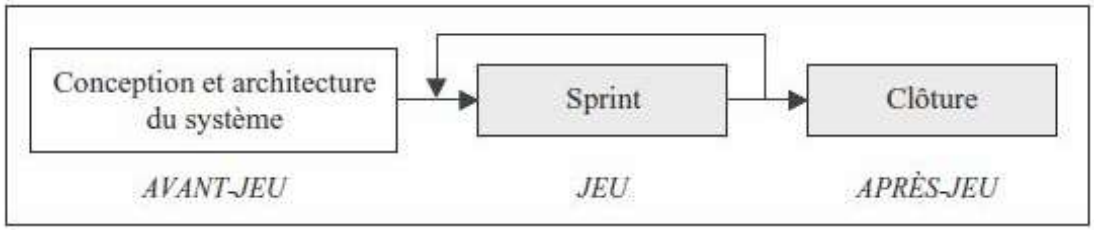


Après une phase initiale d'Exploration des besoins, un plan de livraison est défini avec le client. Chaque livraison, d'une durée de quelques mois, se termine par la fourniture d'une version opérationnelle du logiciel. Une itération de livraison est découpée en plusieurs itérations de développement de courte durée (deux semaines à un mois), chacune donnant lieu à la livraison d'une ou plusieurs fonctionnalités pouvant être testées, voire intégrées dans une version en cours.

De façon plus détaillée, chaque itération de développement commence par l'écriture de cas d'utilisation ou scénarios (*user stories*), c'est-à-dire des fonctions simples, concrètement décrites, avec les exigences associées, qui participent à la définition d'une fonctionnalité plus globale. Utilisateurs et développeurs déterminent ensemble ce qui doit être développé dans la prochaine itération. Une fonctionnalité est ainsi découpée en plusieurs tâches. Les plans de test sont écrits, les développeurs sont répartis en binôme ; ils codent les tâches qui leur sont affectées, puis effectuent avec les utilisateurs des tests d'acceptation. En cas d'échec, on revoit les scénarios et on reprend la boucle. Sinon, on continue jusqu'à avoir développé tous les scénarios retenus. Une version livrable est alors arrêtée et mise à disposition, ainsi que la documentation.

2.4. La méthode SCRUM

La méthode SCRUM emprunte au vocabulaire du jeu le qualificatif des trois phases du cycle préconisé.



 La phase d'Avant-jeu (*pre-game*), Conception et architecture du système, se déroule de façon structurée, en général linéaire, et permet de déterminer le périmètre, la base du contenu du produit à développer et une analyse de haut niveau.

 La phase de Jeu (*game*) est itérative et qualifiée d'empirique, dans la mesure où le travail effectué ne fait pas l'objet d'une planification. Une itération, dont la durée oscille entre une et quatre semaines, est appelée un Sprint, en référence à ces poussées rapides et fortes que les joueurs de rugby peuvent effectuer sur le terrain. Un Sprint est découpé en trois sous-phases :

 Développement (*develop*) : il s'agit de déterminer l'objectif visé au terme de l'itération, de le répartir en « paquets » de fonctions élémentaires, de développer et tester chaque paquet.

 Emballage (*wrap*) : on referme les « paquets » et on les assemble pour faire une version exécutable. • Revue (*review*) : une revue élargie permet de faire le point sur les problèmes et l'avancement.

 Ajustement (*adjust*) : ajusté le travail restant.

 La phase d'Après-Jeu (*postgame*), Clôture, vise à livrer un produit complet et documenté. Comme dans la première phase, on peut en planifier les tâches et les dérouler de façon linéaire.

4. LES METHODES SPECIFIQUES

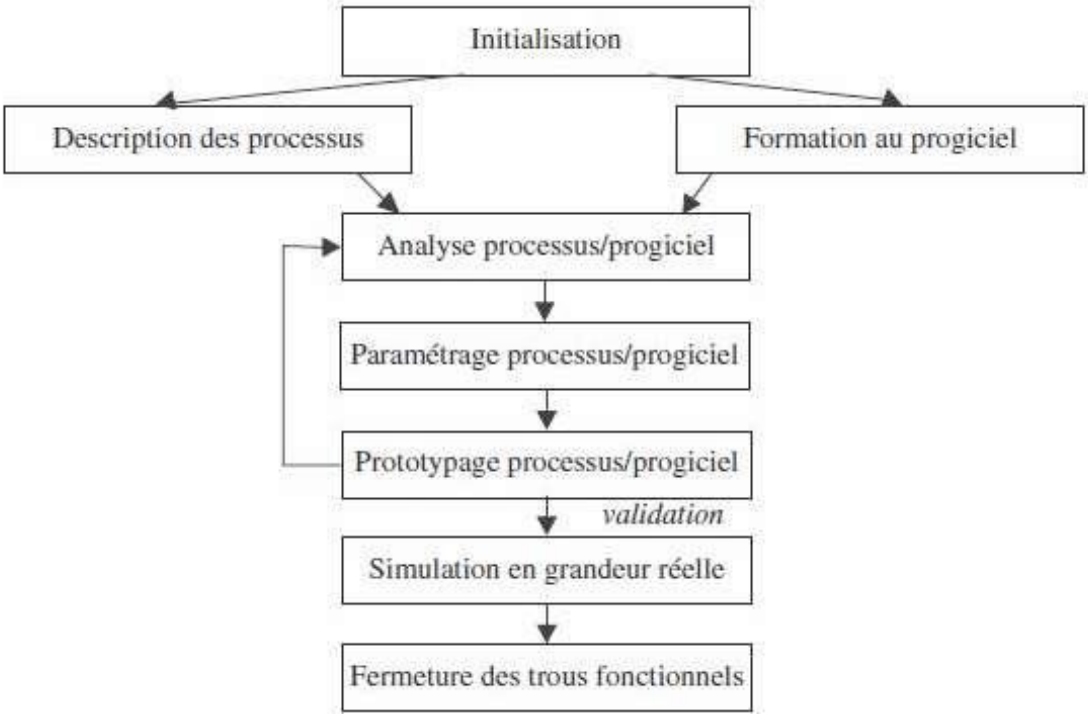
Certains découpages temporels sont liés soit à une méthode, soit à un type de projet bien particulier. Nous en proposons deux exemples : le découpage préconisé pour mettre en place un progiciel intégré et le modèle RUP proposé par la société Rational Software. A ce stade, nous citerons les méthodes telles que :

 La méthode ERP ;

 La méthode RUP ;

4.1. Le cycle ERP

La mise en place d'un progiciel de gestion intégré, souvent appelé du terme anglo-saxon ERP (*Enterprise Resource Planning*) s'appuie sur un découpage spécifique.



En effet, il s'agit de construire, en tirant le meilleur parti du progiciel, un système améliorant la performance de l'entreprise. Deux étapes doivent donc être menées en parallèle : Description des processus et Formation au progiciel. Ensuite, il y a autant de *cycles d"analyse — paramétrage — prototypage* qu'il y a de processus. La validation par le Comité de pilotage permet une simulation en grandeur réelle. Il faut alors prendre en compte ce qui est resté en dehors du champ couvert par le progiciel.

4.2. La méthode RUP

Le modèle RUP (*Rational Unified Process*) est représentatif d'une approche combinant plusieurs modèles. Sa structure fait l'objet d'un assez large accord, notamment parmi les praticiens. Il peut être lu de la façon suivante :

- Le cycle est constitué de quatre phases principales, que l'on retrouve globalement dans toutes les approches descendantes : étude préalable (opportunité), conception de la solution détaillée (élaboration), développement de la solution (*construction*) et mise en œuvre (transition).

- Il existe six types de tâches qui, au lieu d'être affectées exclusivement à une

phase, se retrouvent à des degrés divers dans chacune des phases. Par exemple, l'étude des besoins peut apparaître jusqu'à la fin du projet, mais la plus grande partie est effectuée dans les deux premières phases. L'implémentation (développement) a principalement lieu dans la phase de construction, mais on peut réaliser un prototype dès la première phase. Certaines tâches, comme la direction de projet, s'effectuent sur toute la durée du projet.

- Certaines phases peuvent être menées de façon cyclique. Ainsi, l'élaboration se fait en deux cycles, conduisant par exemple à la production de spécifications externes (vision utilisateur) et spécifications techniques (vision développeur). La construction est itérative et incrémentale. De plus, l'ensemble du modèle représente un tour de spirale, dans le cas d'une approche globale en spirale.