CHAPITRE I
LES MÉTHODES AGILES

I - INTRODUCTION

Tout projet informatique, dès lors qu'il dépasse quelques jours, se doit de s'inscrire dans une méthode de conduite de projet. Depuis les années 1970, des recherches et des méthodes ont été proposées.

Derrière ces approches, se cache se que l'on nomme le génie logiciel. La volonté d'utiliser l'informatique dans les entreprises comme outil de tous les jours implique un besoin d'acquisition ou de production de logiciels. Cette activité fut longtemps artisanale. Actuellement, face à la grande demande en logiciel fiable et peu coûteux, le besoin de passer à une production industrielle se fait sentir. Les buts du génie logiciel vont dans le sens d'une réduction des coûts de production, de maintenance et d'utilisation du logiciel. Ces trois aspects sont différenciés selon un point de vue temporel.

Pour maîtriser le développement et réduire les coûts et assurer la qualité, le génie logiciel propose des méthodes (façon de faire quelque chose), des outils (programmes, langages et formulaires de documentation qui peuvent aider à utiliser une méthode) et des méthodologies (gestion et application d'un ensemble de méthodes et d'outils). Certaines méthodologies sont mieux adaptées à un domaine d'utilisation particulier. La nature de l'optimisation dépend du contexte d'utilisation du logiciel. Si l'on vise la réduction des coûts, il faut s'assurer qu'elle ne se fait pas au détriment des objectifs dont la définition précise doit avoir été faite au début du travail.

La plus ancienne des approches consistait à écouter les besoins du demandeur, programmer, tester et revenir sur la programmation pour gérer les bugs inévitables. Cette approche, simple mais très inefficace pour les projets de taille moyenne (quelques semaines) ou plus conséquent (quelques mois) a très vite laissé la place à des méthodes dites en « V » ou en « W ».

Le modèle dit en « V » part d'une conception globale et se décompose en sous-ensembles (découpage structurel) qui seront développés et testés séparément.

Le modèle dit en « W » reprend l'esprit du modèle en V en ajoutant des étapes d'élaboration de maquettes qui permettent de valider la conception.

Toutefois, même si ces deux approches étaient intéressantes, elles n'évitaient pas ce que l'on nomme « l'effet tunnel ». Derrière ce terme, se cache le plus grand écueil de toutes les anciennes méthodes de conduite de projet : l'absence du demandeur dans les différentes phases du développement. En effet, ces méthodes posent le postulat que le besoin exprimé par le demandeur en début de projet est clair et immuable. Elles supposent aussi que les choix techniques établis en début de projets sont optimum.

Or l'expérience montre que ces postulats sont fragiles. La communication, entre le monde du demandeur et le monde de l'équipe informatique, est parfois délicate. De plus, les besoins du demandeur peuvent évoluer au cours du temps. Enfin, les choix techniques arrêtés en début de processus ne sont pas toujours les plus adaptés aux situations que les équipes informatiques n'avaient pas envisagées.

À partir de ce constat, des méthodes de développement rapides (R.A.D) ont vu le jour. Ces méthodes sont basées principalement sur la notion de cycles : une application se construit en couches et non en un seul bloc. La première couche ou la première itération débouche sur un outil très complet mais sur lequel le client peut déjà se prononcer. Au cours de la seconde itération, le produit est affiné et complété. Les itérations cessent quand le produit est conforme aux besoins du client.

Ces méthodes ont débouché, en 2001, sur le Manifeste Agile qui pose les bases des méthodes de mêmes noms.

Ce manifeste repose sur quatre éléments fondateurs :

- Les personnes et les interactions entre elles sont primordiales et beaucoup plus centrales que les processus et les outils des anciennes approches. Il s'avère que les différentes approches agiles reposent toutes sur ces liens humains, ce qui rend la gestion des relations humaines extrêmement importante dans le processus agile.
- Un produit opérationnel est la priorité du projet, pas la documentation exhaustive des anciennes approches. La satisfaction du client est le seul et unique but du projet et de l'équipe qui le développe. Les méthodes agiles ne s'attardent pas sur l'optimisation du code, si le client n'en exprime pas le besoin.
- Une collaboration continue avec le demandeur (le client) est préférable à la négociation d'un contrat qui fige les demandes et les attentes.
- L'adaptation aux changements sera toujours préférée au suivi d'un plan établi et immuable. Cette adaptation permanente nécessite des cycles courts, chaque cycle étant composé d'une phase de conception, d'une phase de développement et d'une phase de test. C'est l'enchaînement (une itération) des cycles qui permettra de réagir en permanence à tout nouveau changement ou à toute difficulté.

Les méthodes dites « agiles » sont les méthodes qui respectent les fondamentaux du manifeste. En particulier, on peut citer SCRUM et xP (Extreme Programming).

II - L'APPROCHE SCRUM

Scrum est le fruit d'une réflexion entre Ken Schwaber et Jeff Sutherland en 1995 dévoilée lors de l'ACM conférence (Object-Oriented Programming, Systems, Languages and Applications) mais ce n'est qu'en 1996 qu'il naquit officiellement avec la publication du premier article définissant cette méthode.

Scrum est un processus Agile qui permet de produire un produit en totale adéquation avec les demandes du client dans un temps très court.

Le processus Scrum peut être schématisé ainsi :

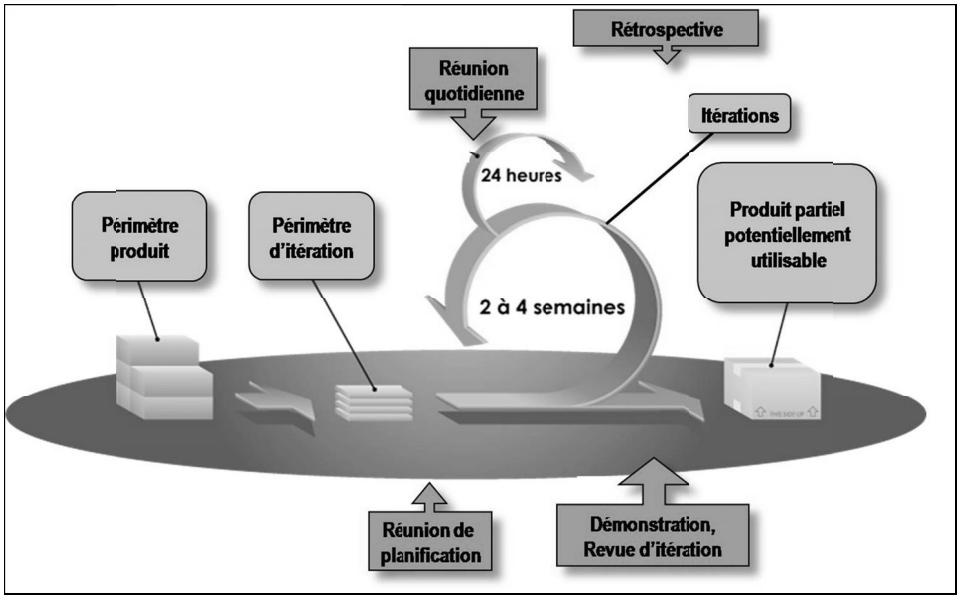

Figure 1 : Le processus Scrum

Les demandes du client sont capturées dans un catalogue nommé **B**acklog du produit ou périmètre du produit. **L**a demande globale du client est découpée en demandes unitaires et priorisées.

Comme toute méthode agile, Scrum est basée sur des cycles de développement de 2 à 4 semaines. Ces cycles sont appelés des Sprints. Au cours de chaque Sprint, une

des demandes est mise en œuvre, c'est le Backlog du Sprint. Le point essentiel de cette approche, tient à la composition et au management des équipes de travail. Ce sont des équipes qui s'autogèrent et où chacun joue un rôle bien déterminé. Ainsi, au cours d'un Sprint qui dure moins d'un mois, il y a quotidiennement une réunion qui se nomme une mêlée (Scrum en anglais) au cours de laquelle un point rapide est fait par tous les membres de l'équipe. En fin de Sprint, l'équipe aboutit à un produit potentiellement livrable, et donc évaluable par le client.

L'équipe de travail constitue le cœur de la méthode, elle peut être décomposée en une succession de rôles :

- Le Product Owner : il s'agit du client qui sera fortement impliqué tout au long du processus de développement. C'est lui qui a la charge de définir les fonctionnalités attendues du produit et qui choisit et ajuste les priorités dans le BackLog. C'est lui qui a le pouvoir d'accepter ou de rejeter les résultats obtenus en fin de Sprint.
- Le Scrum Master : il est responsable du management du projet. Ses responsabilités sont proches du chef de projets. Il a, en particulier, la charge du respect, par son équipe, des valeurs de la méthode. De plus, c'est lui qui doit assurer le bien-être de l'équipe, en éliminant les obstacles du quotidien et les interférences extérieures : une équipe pleinement focalisée sur le produit sera ainsi plus efficace. C'est ainsi que des crèches d'entreprises ou des services domestiques sont offerts aux salariés.
- Les développeurs, les architectes logiciels, les testeurs et tous les autres métiers de l'informatique nécessaires à un projet sont représentés par les autres membres de l'équipe. Une équipe est immuable le temps d'un Sprint et ne dépasse jamais une dizaine de personnes.

La méthode Scrum repose sur une équipe autogérée, cela signifie en particulier que :

- C'est l'équipe qui estime la difficulté de chaque BackLog de Sprint et qui les planifie.
- C'est l'équipe qui décide de la manière d'aborder et de développer chaque BackLog de Sprint.
- Chaque membre de l'équipe s'engage sur du travail qu'il choisit : rien n'est imposé par une tierce personne.
- L'estimation du « reste à faire », ou BurnDown, est actualisée quotidiennement et en fin de chaque Sprint par tous les membres de l'équipe.

- C'est l'équipe qui gère ses mêlées quotidiennes et qui tente de résoudre ses problèmes internes, qu'ils soient humains ou techniques. En particulier, au cours de ces mêlées, chacun doit répondre à 3 questions simples :
    - Qu'as-tu fait hier ?
    - Que vas-tu faire aujourd'hui ?
    - Y a-t-il un obstacle qui te freine ?

A la fin de chaque Sprint, l'équipe présente ce qu'elle a développé ou entrepris. Cette revue du travail effectué se fait toujours par une démonstration et est purement informelle. Chaque fin Sprint donne lieu également à une rétrospective afin de réfléchir régulièrement sur ce qui est opérationnel et ce qui ne l'est pas au sein de l'équipe ou au sein de l'architecture technique.

La méthode SCRUM s'appuie sur un certain nombre d'outils :

- Un tableau blanc et des post-it pour gérer le quotidien de l'équipe,
- Des outils libres comme OpenErp ou propriétaires comme Team Foundation Server de Microsoft qui sera étudié dans cet ouvrage.

Cette méthode seule ne suffit pas car c'est une méthode de gestion de projet, et elle se doit d'être complétée par des techniques d'ingénierie logicielle comme eXtreme Programming.

III – EXTREME PROGRAMMING

eXtreme Programming (XP) est un processus agile né en 1999 qui a pris tout son essor au début du siècle grâce à l'avènement des sites Web. Comme toutes les approches agiles, ce processus a pour unique but de fournir rapidement au client un produit fonctionnel, tout en garantissant le bien-être de l'équipe de développement.
Le processus est construit autour d'un comité de pilotage qui dirige les aspects stratégiques du projet et définit les objectifs et alloue les moyens nécessaires. Son interlocuteur principal est naturellement le client (ou maîtrise d'ouvrage) qui co-pilote le projet, et ce grâce à des cycles itératifs courts (une à deux semaines). A la fin de chaque cycle, un produit fonctionnel est fourni au client.

On distingue un certain nombre de profils qui interviennent dans une équipe XP :

- Le client qui est un membre à part entière de l'équipe et dont la présence physique est imposée dans l'équipe tout au long du développement. C'est lui qui spécifie les fonctionnalités à implémenter et les tests fonctionnels à mettre en place. Ce rôle peut être tenu par une ou plusieurs personnes.

- Le testeur qui est l'assistant du client et, le plus souvent, un informaticien. Il code les tests de recette le plus tôt possible, valide le fonctionnement du code et vérifie la non régression. Les notions relatives aux tests seront détaillées lors du chapitre consacré aux projets de tests.
- Le manager qui est responsable de l'infrastructure dans laquelle l'équipe travaille. Il doit s'assurer de la non existence de problèmes étrangers au projet ou logistiques (espace de travail, outillage, documentation, …).
- Le coach qui s'assure de la bonne compréhension de la méthode XP et son application correcte par les différents acteurs du projet. En général, c'est un expert de la méthode et bon communicateur doublé d'un technicien crédible et respecté.
- Le tracker qui contrôle l'avancement des tâches à l'intérieur d'une itération. Il s'entretient fréquemment avec chaque programmeur pour s'enquérir des difficultés rencontrées et du travail déjà effectué. Il construit ainsi une vision fiable de l'avancement des tâches afin de détecter le plus tôt possible les dérives et de lancer une nouvelle phase si nécessaire.
- Le programmeur qui estime la charge nécessaire à l'implémentation d'un scénario dans le cadre du jeu de la planification. C'est donc lui qui implémente les scénarios en s'appuyant sur l'écriture de tests unitaires. Le programmeur est aussi analyste et concepteur. De manière plus classique, on assimile le programmeur à la partie pleinement opérationnelle de l'équipe.

Si ce processus est proche de SCRUM, il se positionne plus précisément sur la technicité et la manière de coder ensemble. Seuls des outils comme Visual Studio peuvent aider les équipes de développement à se situer dans le respect des valeurs de ce processus.

En particulier, XP véhiculent des concepts tournés clairement sur l'optimisation des temps de développement :

- Les compétences vont s'uniformiser tout au long du processus de développement, les savoirs se partageant et s'enrichissant mutuellement. Le niveau global des compétences va donc s'accroître tout au long du projet.
- L'équipe doit respecter des règles de codage précises afin que la manière de coder soit unique et non individualisée.
- Le développement se fera en binôme afin de se contrôler mutuellement, de diminuer les erreurs de conception et de se concentrer totalement sur son travail. Cela permet également de supprimer le risque de dépendance à un des membres de l'équipe.

Ce processus de développement, contrairement à SCRUM qui se positionne comme une méthode de conduite de projet, est encore assez confidentiel en France.

Cette approche est entièrement basée sur les bonnes relations humaines au sein de l'équipe toute entière, et sur une approche du développement qui tranche radicalement avec les pratiques plus classiques. Ses deux colonnes vertébrales sont, pour le moment, ses deux plus grands freins.

CHAPITRE II
VISUAL STUDIO

I – PRÉSENTATION

Le Basic est né en 1964. A l'origine, c'est un simple langage d'apprentissage qui sera très vite concurrencé par d'autres langages plus performants comme le Pascal (avec, en particulier Borland Turbo Pascal) et le C.

En 1980, Microsoft commercialise l'ancêtre des systèmes d'exploitation sur PC : le MsDos qui intègre le Basic. Extrêmement limité à ses débuts, le langage Basic fera néanmoins la joie de tous les programmeurs débutants. Toutefois, avec les différentes versions du MsDos, le langage Basic évoluera pour déboucher enfin sur les environnements visuels, contrairement à son environnement natif qui était le mode textuel.

En 1991, c'est une véritable révolution qui s'amorce avec la mise à disposition de Visual Basic 1 avec son interface graphique (puis la version 2 en 1992 et la version 3 en 1993). Cette révolution s'appuie sur quelques concepts novateurs :

- De simples glisser-déplacer suffisent à concevoir une fenêtre Windows.
- La notion d'évènements fait son apparition.
- L'accès aux bases de données via ODBC est enfin possible hors des Systèmes de Gestion de Bases de Données.
- En 1995, avec la mise sur le marché de Windows 95, apparaît la version 4 de Visual Basic. Cette version est qualifiée de version hybride car elle repose sur une architecture 16 et 32 bits. La nouveauté liée à cette version tient aux modules de classes qui permettent au programmeur de plonger encore plus aisément dans le monde de la programmation objet.
- En 1997, Microsoft sort la version 5 de Visual Basic, dix fois plus rapide que les versions précédentes. De plus, grâce à cette nouvelle version, le programmeur a la possibilité de mettre au point ses propres contrôles Ocx. De plus, la technologie ActiveX permet d'insérer ses applications (appelés Documents ActiveX) dans le navigateur Internet Explorer 4.
- En 1998, Microsoft souhaite rapprocher son produit d'Internet et développe sa technologie ADO qu'il implémente dans la sixième