

Méthodes de développement

Méthodes agiles

1 - Introduction.....	2
2 -Le manifeste agile et les méthodes agiles	2
2.1 Le manifeste agile	2
2.2 Les méthodes agiles	3
3 - Caractéristiques communes des méthodes agiles	3
4 - Caractères particuliers de méthodes agiles	4
4.1 Rapid Application Development (RAD).....	4
4.2 SCRUM	4
4.3 Extreme Programming (XP).....	5
5 - Intérêt des méthodes agiles	5
6 - Les limites et les risques des méthodes agiles	6
7 - Planification du projet.....	6
8 - Conduite des activités	7
8.1 Analyse du besoin	7
8.2 Conception de l'architecture	8
8.3 Tests	9
9 - Conclusion	9

1 - Introduction

Un objectif important dans le développement de logiciel est de réduire les risques : risques de dépassement des coûts et délais, risque de mauvaise adaptation aux besoins du client.

L'utilisation de techniques itératives permet de bien réduire ces risques.

L'objet des méthodes agiles est de mieux garantir l'adaptation aux besoins du client. Elles s'appuient sur des techniques itératives avec une forte implication du client dans le développement.

Les méthodes dites agiles ont commencé à apparaître dans les années 90. En 2001 17 experts américains se sont réunis et ont établi le "manifeste agile" qui sert de référence pour les méthodes agiles.

2 -Le manifeste agile et les méthodes agiles

2.1 Le manifeste agile

Ce manifeste définit 4 valeurs fondamentales d'où sont déduits 12 principes généraux.

Les 4 valeurs fondamentales sont :

Les **individus et les interactions** doivent primer sur les processus et les outils

Le **développement logiciel** doit primer sur la documentation exhaustive

La **collaboration avec le client** doit primer sur la négociation contractuelle

L'ouverture au changement doit primer sur le suivi d'un plan rigide

Les 12 principes généraux sont :

- « Notre première priorité est de satisfaire le client en livrant tôt et régulièrement des logiciels utiles ».
- « Le changement est bienvenu, même tardivement dans le développement. Les processus agiles exploitent le changement comme avantage compétitif pour le client ».
- « Livrer fréquemment une application fonctionnelle, toutes les deux semaines à deux mois, avec une tendance pour la période la plus courte ».
- « Les gens de l'art et les développeurs doivent collaborer quotidiennement au projet ».
- « Bâissez le projet autour de personnes motivées. Donnez leur l'environnement et le soutien dont elles ont besoin, et croyez en leur capacité à faire le travail ».
- « La méthode la plus efficace pour transmettre l'information est une conversation en face à face ».
- « Un logiciel fonctionnel est la meilleure unité de mesure de la progression du projet ».

- « Les processus agiles promeuvent un rythme de développement soutenable. Commanditaires, développeurs et utilisateurs devraient pouvoir maintenir le rythme indéfiniment ».
- « Une attention continue à l'excellence technique et à la qualité de la conception améliore l'agilité ».
- « La simplicité - l'art de maximiser la quantité de travail à ne pas faire - est essentielle ».
- « Les meilleures architectures, spécifications et conceptions sont issues d'équipes qui s'auto-organisent ».
- « À intervalle régulier, l'équipe réfléchit aux moyens de devenir plus efficace, puis accorde et ajuste son comportement dans ce sens ».

2.2 Les méthodes agiles

On trouve aujourd'hui une dizaine de méthodes "agiles" documentées, en particulier :

- Rapid Application Development (RAD)
- Dynamic Systems Development Method (DSDM)
- Extreme Programming (XP)
- Adaptive Software Development (ASD)
- Feature Driven Development(FDD)
- Scrum
- Crystal clear
- Rational Unified Process (RUP)
- Agile Unified Process (AUP)

Il faut noter que certaines de ces méthodes ne respectent pas totalement les valeurs fondamentales et principes généraux du manifeste agile, et que leur classement en méthode agile est discutable. Par exemple le RUP est considéré comme agile par certains auteurs et non agile par d'autres...

3 - Caractéristiques communes des méthodes agiles

Les principaux aspects communs aux méthodes agiles sont :

- participation active des utilisateurs (avec pouvoir de décision) à la définition et au contrôle des résultats (groupes de travail formalisés), d'où une approche collaborative au lieu d'un contrat figé,
- réalisation par une méthode itérative avec des itérations courtes,
- réalisation par une équipe autonome, auto organisée et motivée,
- planification non figée, réexaminée en permanence en fonction des demandes des utilisateurs, des priorités, de l'avancement...
- réalisation d'une documentation juste suffisante et autant que possible intégrée au code,
- conception en vue de l'évolutivité (conception objet...),
- recherche de solutions simples, de normes et techniques raisonnables...

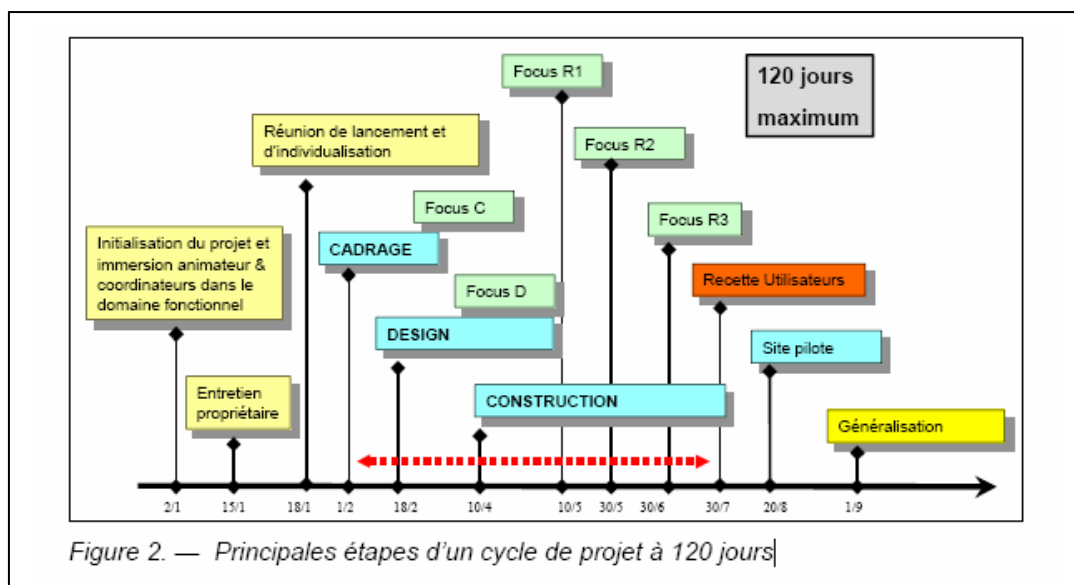
4 - Caractères particuliers de méthodes agiles

4.1 Rapid Application Development (RAD)

La méthode RAD structure le cycle de vie du projet en 5 phases :

- l'**Initialisation** définit l'organisation, le périmètre et le plan de communication,
- le **Cadrage** définit un espace d'objectifs, de solutions et de moyens,
- le **Design** modélise la solution et valide sa cohérence systémique,
- la **Construction** réalise en prototypage actif (validation permanente),
- la **Finalisation** est un contrôle final de qualité en site pilote.

Vous pouvez noter que les deux premières phases sont la préparation du projet, et ensuite les autres phases sont un développement itératif.



La construction est réalisée en utilisant la technique itérative.

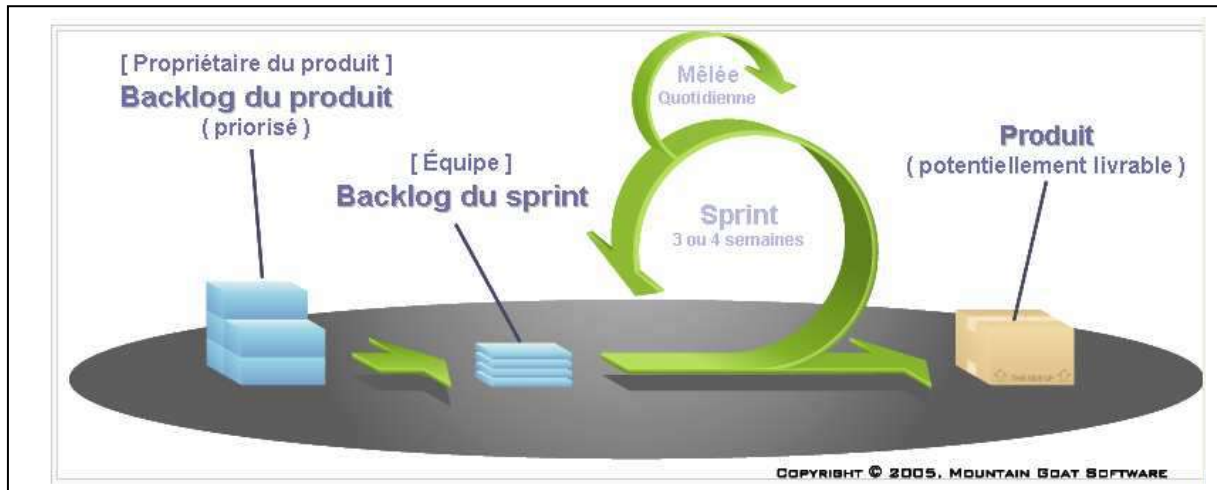
La méthode RAD met l'accent sur la réalisation de cycles courts (90 à 120 jours au maximum). D'autre part la participation du client doit être quasi permanente et est formalisée en particulier avec un groupe d'animation et de rapport (GAR) et des groupes de validation.

4.2 SCRUM

Scrum vient de la mêlée du rugby, et une des particularités est de prévoir chaque matin une courte "**mêlée**" où l'équipe fait le point de l'avancement et décide des travaux à mener.

Scrum utilise une approche fonctionnelle pour récolter les besoins des utilisateurs. L'objectif est d'établir une liste de fonctionnalités à réaliser, que l'on appelle **backlog de produit**.

Scrum est un processus *itératif* : les itérations sont appelées des **Sprints** et durent de 2 et 4 semaines. Chaque **Sprint** possède un **but**, et on lui associe une liste d'*items de backlog de produit* (fonctionnalités) à réaliser.



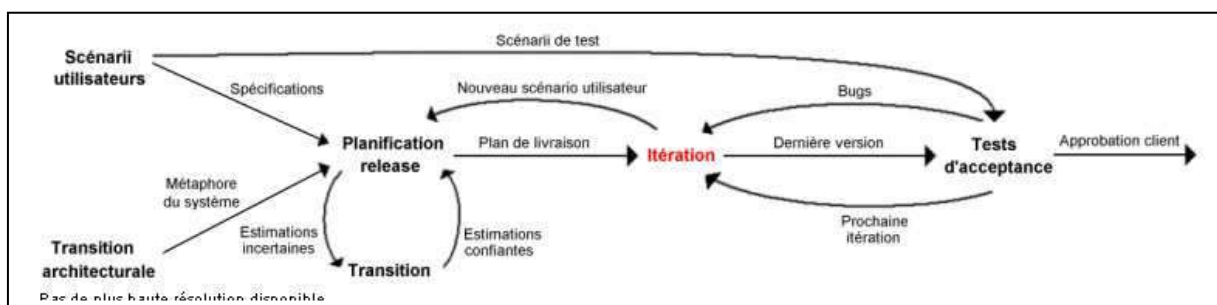
4.3 Extreme Programming (XP)

L'Extreme Programming demande un représentant du client présent en permanence sur site pendant la durée du projet. Il fournira les "scenarios client" qui définissent le besoin.

L'Extreme Programming repose sur des itérations de quelques semaines dont les étapes sont les suivantes :

- une phase d'exploration détermine les scénarios clients qui seront fournis pendant cette itération ;
- l'équipe transforme les scénarios en tâches à réaliser et en tests fonctionnels ;
- chaque développeur s'attribue des tâches et les réalise avec un binôme ;
- lorsque tous les tests fonctionnels passent, le produit est livré.

Le cycle se répète tant que le client peut fournir des scénarios à livrer. Généralement le cycle de la première livraison se caractérise par sa durée et le volume important de fonctionnalités embarquées. Après la première mise en production, les itérations peuvent devenir plus courtes (une semaine par exemple).



5 - Intérêt des méthodes agiles

Un premier intérêt est la réduction de risques qu'apporte l'utilisation des techniques itératives : réduction des risques d'échec du projet, réduction des risques de dépassement du délai du projet, réduction des risques de réaliser un produit mal adapté aux besoins...

Les méthodes agiles apportent de plus de meilleures garanties d'obtenir un produit bien adapté aux besoins ou au moins aux besoins prioritaires, d'une part en faisant participer activement les utilisateurs au développement, d'autre part en facilitant la prise en compte de changements au cours du projet.

6 - Les limites et les risques des méthodes agiles

Les méthodes agiles demandent que la réalisation puisse être faite dans un cadre collaboratif et non contractuel, ce qui limite en pratique son utilisation à certains types de développements tels que les développements d'outils en interne à une société.

De plus il faut disposer d'utilisateurs, qui soient suffisamment représentatifs de l'ensemble des utilisateurs futurs, et qui acceptent de collaborer à la réalisation du projet en y consacrant une charge de travail significative. Cela peut arriver pour des développements internes d'outils de gestion, mais il y a beaucoup de cas où ce n'est pas réaliste.

Ces méthodes ont été conçues pour de petits projets (équipe de 4 à 6 personnes sur quelques mois). L'utilisation sur de plus gros projets est en théorie prévue en décomposant et parallélisant, mais la faisabilité n'est pas évidente.

Le problème majeur me semble être l'absence de maîtrise des coûts. On peut s'engager à fournir un produit fonctionnel et bien adapté aux besoins les plus prioritaires pour un coût et un délai fixé, mais on ne peut ni s'engager à ce que ce produit couvre l'ensemble des besoins, ni s'engager sur le coût et le délai pour réaliser un produit qui couvre l'ensemble des besoins.

7 - Planification du projet

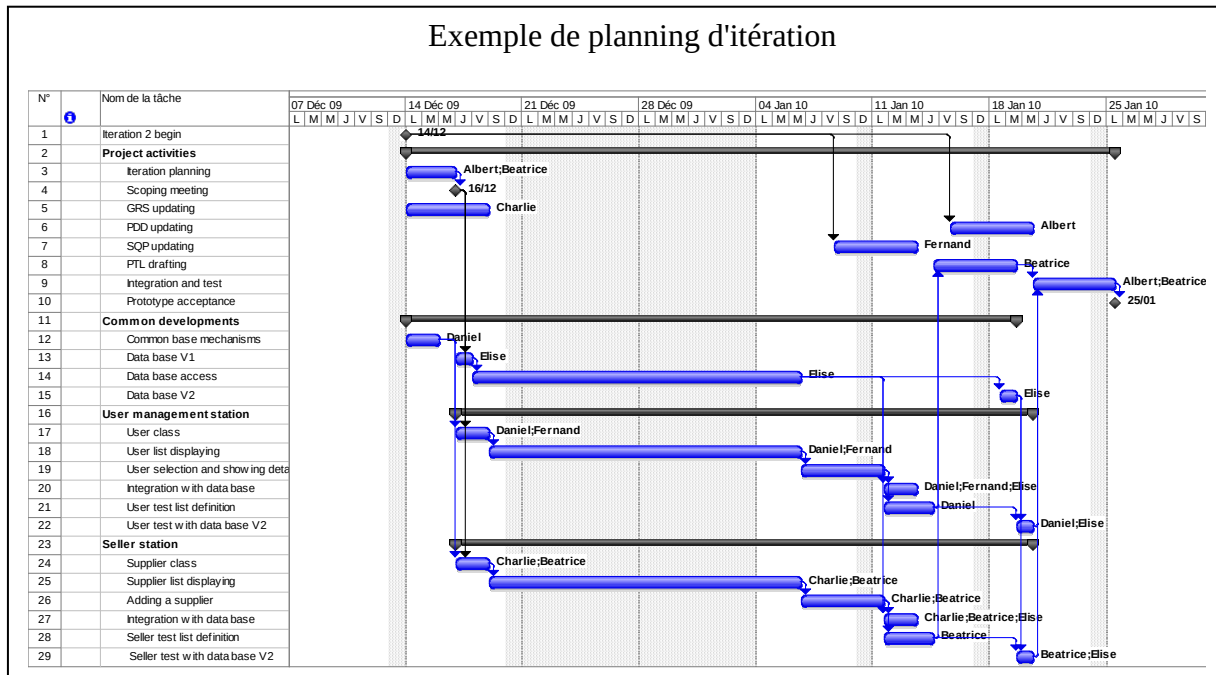
Les méthodes agiles utilisent toutes une technique de réalisation itérative, et supposent que le contenu de chaque itération sera défini au début de l'itération en fonction de l'état d'avancement et des priorités du client.

Cela permet de s'adapter facilement aux demandes de modification du client.

En contrepartie il n'est pas possible au début du projet d'identifier toutes les tâches du projet et de faire un planning complet du projet.

On pourra procéder par étapes :

- au début du projet on se limite à identifier les niveaux supérieurs de l'arborescence des tâches, et on fait un planning général avec ces tâches de niveau supérieur et il se limitera en général à montrer la suite des itérations,
- au début de chaque itération, on définit de façon détaillée les tâches de l'itération et on fait le planning détaillé de l'itération.



Bien que le contenu précis de chaque itération soit défini au début de chaque itération, il faut prévoir au départ un contenu prévisionnel pour chaque itération. Cela permet de définir un ordre d'implémentation des fonctions et un délai d'aboutissement s'il n'y a pas trop de demandes de modification. Ensuite, le contenu réel des itérations pourra être différent, mais en mettant à jour ces contenus prévisionnels on pourra suivre l'avancement du produit et le délai de réalisation.

Pour ces contenus prévisionnels et contenus d'itération on s'efforcera de parler en termes de fonctions ou fonctionnalités du produit. Cela facilitera le suivi et le dialogue avec le client.

Il ne semble pas utile de rédiger un plan de développement logiciel (PDL) formalisé tel que présenté dans le chapitre préparation et suivi de projet. On pourra se limiter à faire et tenir à jour le planning général et un tableau de contenu des itérations.

8 - Conduite des activités

8.1 Analyse du besoin

Il est nécessaire de faire une analyse du besoin au début du développement pour savoir quelles seront les fonctionnalités à implémenter.

Dans les méthodes classiques où on recommande d'analyser le besoin dans le détail au début du développement, et de décrire ce détail dans une Spécification Technique de Besoin.

A l'inverse, dans les méthodes agiles, on recommande de ne pas faire d'analyse détaillée au début du développement, mais uniquement au moment d'implémenter la fonctionnalité.

Le réalisateur qui a une fonctionnalité à coder approfondira le besoin à travers le codage ou la définition des tests, présentera le résultat au client, et modifiera en fonction des remarques du client. Cela s'approche d'une démarche par approximations successives.

Dans ces conditions il n'y a pas besoin de rédiger une Spécification Technique de Besoins avec tout le détail des fonctions ou cas d'utilisation.

Il est cependant utile d'avoir un document de besoin pour présenter l'utilisation du produit, les contraintes à respecter et lister les fonctionnalités à implémenter.

Ce document pourrait être appelé Spécification Générale de Besoins (SGB) pour le distinguer d'une Spécification Technique de Besoin classique et contenir :

- la présentation de la mission du système,
- une liste des fonctionnalités,
- les règles métier à respecter,
- des exigences complémentaires.

Un guide de rédaction est fourni.

8.2 Conception de l'architecture

Le principe dans les méthodes agiles est de ne surtout pas chercher à optimiser et limiter les travaux d'architecture pour ne pas anticiper sur ce qui sera réalisé ou non réalisé (inutile de faire du travail pour ce qui ne sera peut être jamais réalisé...).

De même lorsqu'on réalise une fonction, on ne prévoit pas les futures fonctions à implémenter, c'est lorsqu'on implémentera ces fonctions qu'on modifiera si besoin les fonctions déjà implémentées.

Cependant il faut avoir une architecture qui permette l'ajout progressif des fonctions et permette de modifier facilement les fonctions.

Pour cela il est fortement recommandé d'utiliser une conception objet et des schémas d'architecture type Modèle Vue Contrôleur (MVC).

Il est fortement souhaitable avant de commencer à coder des fonctions de faire une étude générale d'architecture au niveau des principaux objets et du déploiement.

Par contre on ne regardera pas le détail du contenu des classes et des interfaces comme recommandé dans les méthodes classiques. Ce détail sera examiné lors de la réalisation et commenté dans le code.

Il n'y aura pas lieu de rédiger un Document d'Architecture Logiciel (DAL) détaillé, mais il sera tout de même utile de consigner les résultats de l'étude générale d'architecture (souvent appelée conception préliminaire).

Ce Document de Conception Préliminaire (DCP) pourrait contenir :

- des tableaux des classes correspondants aux objets principaux,
- les diagrammes de classe,
- des diagrammes de séquence pour quelques scénarios d'utilisation,

- un diagramme de déploiement,
- le schéma de la base de données.

Un guide de rédaction est fourni.

8.3 Tests

Les méthodes agiles conduisent à reprendre un grand nombre de fois les tests de chaque fonction.

En effet à chaque itération on reprend les fonctions déjà codées, on les modifie souvent et on ajoute de nouvelles fonctions. Il faut tester les modifications, les nouvelles fonctions, et s'assurer qu'il n'y a pas de régressions.

On reprendra donc les tests des fonctions à chaque itération.

Il faut donc faire un effort sur la définition des tests et rechercher l'automatisation de leur passage.

Il est recommandé, avant de coder une fonction, d'établir la liste des tests qui permettra de vérifier le fonctionnement de la fonction.

9 - Conclusion

Les méthodes agiles contiennent des concepts intéressants.

Cependant une approche totalement agile ne peut être utilisée que si plusieurs conditions sont remplies : client disponible pour coopérer dans le projet, conduite du projet dans un cadre non contractuel, pas d'exigence de maîtrise des coûts...

Cependant si toutes ces conditions ne sont pas remplies, on peut tout de même s'inspirer des concepts des méthodes agiles pour définir une méthode raisonnablement agile répondant aux contraintes.