To be Agile or not to be Agile? That is the Question - Part 1: Les Valeurs de l'Agilité

Winael

2020-02-25T10:37:24+01:00

ATTENTION : Cet article est un billet d'humeur. Le discours ne sera donc pas lissé et risque peut-être de choquer dans un monde où il ne faut surtout pas se faire remarquer. Et c'est tant mieux !

Parfois au détour d'une veille on peut tomber sur ce genre d'article qui vous donne envie de réagir avec force. Dans cet article publié sur ZD.net, il est fait état d'une situation qui ne peut, à mon sens, plus durer. L'un des co-auteurs du Manifeste du Développement Agile de Logiciels, publié en 2001, Ron Jeffries, déplore que l'Agilité est aujourd'hui à ce point travestie qu'elle ne sert plus qu'à mettre la pression sur les équipes de développement pour produire plus vite, dans des environnements toujours plus stressants. L'Agilité est devenu un business dans lequel de véritables vampires se sont engouffrés, sur-usant du terme Agile pour vendre tout et n'importe quoi.

Mais alors l'Agilité qu'est-ce que c'est ? A en croire certains faux prophètes, l'Agilité est un ensemble de méthodes pour magiquement rendre vos équipes plus performantes (Les fameuses "Méthodes à Gilles", vendues par une bonne parties de charlatans du "Digitale").

Coupons cours à la discussion. L'Agilité ce n'est pas des méthodes, c'est un changement d'état d'esprit. Bref, on change d'abord la tête, ensuite on change la façon de travailler. Pas l'inverse, sinon c'est l'échec assuré, et de l'incompréhension à tout les niveaux.

L'Agilité regroupe quatre valeurs fondamentales, qui reposent en grande partie sur du bon sens :

1) On préfère et favorise les interactions et les gens plutôt que les outils et les processus :

Alors qu'est-ce que cela veut dire ? Qu'on oublie les outils de ticketing et les processus internes de la société ?

Et bien non.

Juste qu'il ne faut pas se planquer derrière. Il n'y a rien de plus rageant quand il y a un soucis de production, que celui qui puisse agir, qui travaille à côté de toi, doive attendre un **ticket incident qui est en train de faire le tour de la Terre deux ou trois fois** (parce que le **service desk outsourcé on-ne-sait-où**, est constitué d'une équipe de personnes mal formées et payées au lance pierre, qui **escaladent les tickets au petit bonheur la chance**). Du coup pour palier à cette problématique, on a inventé les Tasks Force. Une équipe de spécialistes qui se réunissent en cas de crises pour agir après avoir passé plus d'une heure pour définir une stratégie, alors que **donner juste la possibilité au collègue d'outrepasser le process de façon exceptionnelle pour corriger le tir, c'est quand même plus simple que de devoir réparer toute la prod partie au tas à cause de l'inertie des process.**

Favoriser les interactions et les gens, c'est aussi **ne pas devoir attendre 3 semaines une ouverture de flux, dont une parce que la demande a pourri dans la file de la personne chargée de dispatcher la demande pendant qu'il était en congé** (Bus Factor = 1), alors que **le collègue en charge de l'opération est assis à 10 mètres...**

Donc les outils et les process, c'est pas mal, mais **la bêtise bureaucratique des porteurs de parapluies professionnels, c'est à oublier quand on veut être efficace.**

2) On préfère et favorise de faire des logiciels opérationnels plus qu'une documentation exhaustive :

Là c'est pareil, souvent dans les **modèles en Cycle en V** on nous apprends à **produire un nombre incalculable de documents en tant que livrables**, notamment sur la phase d'architecture.

Sauf qu'une fois arrivée à l'implémentation, souvent la **mise en œuvre dérive de l'architecture prévue dans la documentation sans que celle-ci ne soit forcément revue** (car oui on a accumulé du retard, on a plus le temps !!!).

Figure 1: “Plein de docs”

Quant à l’écriture de documents d’exploitation… ma foi… pas le temps, on devait livrer il y a deux mois déjà. Là encore, c’est du vécu. En tant qu’ingénieur d’exploitation, on se retrouve à **exploiter des applications, avec une documentation datée, sans procédures dans le DEX (Dossier d’exploitation), et du code non commenté qu’on comprend pas**.

Et là, c’est le drame… On passe par une phase de rétro-ingénierie.

Du coup, on comprend vite l’intérêt de **créer de la documentation au fur-et-à-mesure que l’on itère sur l’application développée**. **La documentation doit faire partie intégrante du livrable de fin d’itération** (sprint, release, etc…), d’autant qu’aujourd’hui, il est possible de générer une partie de la doc à partir des commentaires du code et qu’un code bien commenté, est plus facile à maintenir. **Code et Documentation, l’un ne va pas sans l’autre**.

3) On préfère et favorise la collaboration avec les clients plutôt que les négociations commerciales :

Toujours du vécu, le commercial qui te contacte pour que tu développes LA fonctionnalité qu’il a vendu au client, sans savoir si c’était faisable, ta bande passante, la charge de ton équipe.

Et bien évidement, il a **négocié en accordant au client des pénalités sur la qualité et sur le retard, et c’est aux développeurs de se débrouiller pour qu’il y ait le moins de pénalités de retard possibles**. Il en résulte souvent un **produit qui ne correspond pas à ce qu’attend le client**

(car au passage les besoins ont été très mal définis), **livré souvent très en retard, bâclé, et difficilement maintenable** (puisque la documentation d'exploitation est peu exploitable), ce qui cause en sus des difficultés pour l'équipe d'exploitation, sans compter le délai interminable du processus de remonté d'incident.

Résultat : on paie beaucoup de nouvelles pénalités sur la qualité de services (SLA), tout en ayant un **client très mécontent** qui menacera de partir au plus tôt. Il est vrai que je n'ai pas fait d'études de commerce, mais à mon avis, c'est plutôt très **mauvais pour le business**.

Alors comment faire une collaboration avec le client ?

Il suffit de regarder ce qu'il se fait sur Steam avec les early access. Une première version du jeu non fini est vendue à prix réduit. (On appelle cela un MVP). Les développeurs vont ensuite **être à l'écoute de la communauté** de joueurs qui commence à s'organiser autour du jeu et **donner des retours (feedback)**. Ils vont corriger les bugs remontés par les joueurs, et implémenter les idées les plus intéressantes, et **tenir régulièrement informée leur communauté de l'avancement du développement**. Bref, de la **gestion de communauté d'utilisateurs**, de la **transparence et de la confiance**, ce qui va permettre de faire la promotion de leur jeu par le bouche à oreille, déclencher de nouvelles ventes.

L'un des leaders du marché en matière de **collaborations avec le clients**, c'est **Amazon Web Service**. A tel point que ses utilisateurs savent que si une fonctionnalité leur manque, leur désir sera comblé dans les six mois. AWS c'est une croissance extraordinaire. Bref, **la collaboration, ça paie**. Il ne vous reste plus qu'à formaliser cette collaboration dans un contrat en y spécifiant les SLA, mais **si le travail est bien fait, ce sont des pénalités en moins à payer** et une **meilleure réputation** qui se traduira par de nouveaux contrats.

Meilleur pour le business non ?

4) On préfère et favorise l'adaptation au changement plutôt que de suivre un plan :

En règle général, les personnes qui parlent d'Agilité sans s'y connaitre ne retienne peut-être que ça. ***"Si tu te prétends Agile, alors tu dois adopter l'adaptation au changement, c'tout, fin de la discussion."***" Oui… mais non.

OK **le monde est Volatile, Incertain, Complexe et Ambigu** et un projet peut changer pour de nombreuses raisons.

C'est pourquoi, **plutôt que d'avoir un plan ultra structuré, on préférera une vision qui nous donnera une direction**. Avoir une direction permet d'avancer.

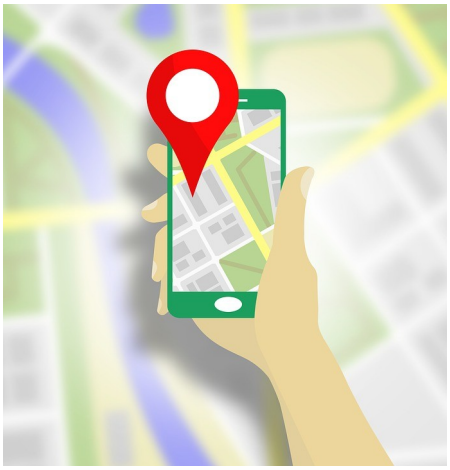

Figure 2: Planifier sa vision

Il faut prendre ça comme des indications d'une application GPS. Ex: Le temps est mitigé, et il est décidé de sortir faire une activité. S'il n'a pas été décidé quoi faire avant de partir, on risque de faire du sur place, ne pas se décider et finalement ne rien faire. Du coup, il est décidé de partir sur une base nautique. On imprime le parcours étape par étape pour s'y rendre. Manque de pot, accident sur le parcours, et 3 heures d'embouteillages plus tard, on arrive et… la base nautique est fermée. Dommage !

Avec une application GPS, un autre itinéraire est proposé, ce qui permet d'arriver au plus tôt à la base de loisir. Malheureusement celle-ci est fermée pour cause d'intempérie. Fort heureusement, grâce aux points d'intérêts, il est possible de trouver un parc d'attractions à proximité et finalement tout le monde s'amuse bien (même la personne qui a besoin d'un contrôle permanent). On voit bien qu'ici une **vision** a entraîné le groupe sur le **chemin**, et que malgré les **obstacles**, le fait d'avoir un **plan qui évolue** (le GPS) a permis au groupe d'**atteindre une destination acceptable pour tous**.

Voilà, dans cet article nous avons revu les 4 valeurs fondamentales sur lesquelles reposent l'**état d'esprit Agile**. On voit bien que **ce n'est pas un ensemble de méthodes, mais une façon d'appréhender sa manière de travailler pour être à la fois plus efficace et travailler dans des conditions plus sereines**.

Dans la prochaine partie de cet article nous verrons comment se traduit les 12 principes de l'Agilité qui découlent des valeurs sus-mentionnées.

N'hésitez pas à faire part de vos remarques.

A bientôt

Retrouvez l'article en epub ou pdf