

Une réflexion sur les méthodes de développement logiciel plaçant au cœur de la démarche la relation client-développeur.

La conduite de projet du développement logiciel : une utopie ?	1
Quelques explications de l'échec	1
Vaudeville Client-développeur	2
Le développement Agile	3
Vers une démarche de partenariat	4
Les douze principes (extrait du livre blanc de « Business Interactif »).....	4
4 ^{ème} Dimension un outil agile	5
Extraits - Bibliographie - Liens	5

La conduite de projet du développement logiciel : une utopie ?

Obtenir une solution informatique qui corresponde réellement à ses besoins dans le délai prévu et dans l'enveloppe budgétée. Ce souhait, en apparence raisonnable, combien de décideurs engageant un projet informatique l'ont vu se réaliser ?

Nombre d'études montrent une proportion effarante de projets n'aboutissant jamais, environ 1/3 selon certaines sources ! (cf. annexe). Et sur les 2/3 restant combien de projets menant au conflit, à l'insatisfaction ?

Face à ce constat, les sociétés de développement logiciel ont tenté de trouver des solutions dans l'élaboration et le suivi de processus de développement formalisés : la méthode MERISE en étant le fleuron en France.

Mais force est de constater qu'on ne planifie toujours pas la réalisation d'un logiciel comme celle d'un pont, et que les échecs, retards, ratages abondent toujours.

Quelques explications de l'échec

Dans la plupart des projets, les échecs relèvent rarement de la technique à elle seule.

Les méthodes classiques ont privilégié la **prédictivité**. L'analyse des fonctionnalités souhaitées lors de la phase d'étude rendait possible une prévision des ressources, des délais et du coût. Tout le problème consistait alors à formaliser le recueil de ces fonctionnalités et à les modéliser dans un langage standard : MERISE, UML,...

Dans la vraie vie, quelques complications surviennent :

- postuler que le client (ou l'utilisateur final) est capable d'exprimer la totalité de ses besoins avant d'avoir aperçu le moindre prototype relève généralement du doux rêve ;
- poursuivre en supposant qu'une fois les besoins exprimés il ne changera pas d'avis, de priorité, (simplement parce que le contexte de son métier a changé), relève de l'entêtement ;
- cette phase d'étude (recueil des besoins) puis de conception (modélisation) est longue, coûteuse (fait appel à des intervenants chers) et sans retour réellement exploitable par le client ;
- une fois cette phase passée, introduire le moindre changement dans les spécifications initiales devient problématique d'un double point de vue d'architecture et de coût ;

- le client qui est seul capable de valider le résultat de l'étude ne maîtrise souvent pas le langage utilisé pour la modélisation et se retrouve en conséquence exclu du processus dès la fin des interviews. Rendez-vous lui est donné à la livraison où il aura la surprise de découvrir un logiciel qui aura peu de chance de répondre réellement à des besoins qui n'auront généralement pas tous été explicités.
- supposer que l'industrialisation du processus informatique consiste à générer automatiquement du code à partir de spécifications formalisées est un fantasme de directeur des ressources humaines. En pratique, l'industrialisation de l'informatique réside dans la génération automatique de code binaire par le compilateur à partir du code source écrit par le développeur. Le développement demeure une activité de création humaine.

Ce type de processus prédictif peut certes s'appliquer à la réalisation de grands chantiers où le client (maître d'ouvrage) est représenté par des personnes formées à la démarche. Cependant dans des projets fluctuants d'informatisation de clients néophytes ou non secondés par un service informatique, il se révélera inadapté et générateur de conflits.

Vaudeville Client-développeur

La première scène du vaudeville client-développeur se déroule dans le bureau du client :

- Client : monsieur le développeur, je n'y connais rien en informatique, j'ai besoin d'un logiciel pour informatiser ma société (ou mon service). Combien ça coûte ? Ah, bien sûr, nous sommes pressés, il me faut impérativement quelque chose pour la fin du mois.
- Développeur : j'entends bien votre demande monsieur le client, mais je ne peux évaluer faisabilité, coût, délai qu'à partir d'un cahier des charges précis et détaillé.
- Client : pas de problème, j'ai tout dans la tête. Voilà c'est très simple ...
et il présente les grandes lignes de son attente.

Que va-t-il se passer à partir de là ?

Soit le développeur est un peu téméraire et il va proposer directement une réalisation au forfait basée sur ce recueil très sommaire d'informations, soit il va essayer d'obtenir une étude plus détaillée.

Le premier conflit va se nouer ici : le client soucieux de faire jouer la concurrence a contacté plusieurs développeurs ou sociétés de services. Il ne va pas financer une étude par chaque postulant. S'il retient une société pour une étude complémentaire, le résultat de cette étude sera-t-il exploitable par les autres sociétés ? Oui, en théorie, si la rédaction est formalisée dans un langage pour informaticien, mais dans ce cas c'est le client qui devient incapable de savoir de quoi parlent les spécialistes.

Si à l'inverse une société accepte de s'engager dans une étude complémentaire gratuite, elle va avoir tout intérêt à ne pas y passer trop de temps car si 5 sociétés ont été retenues, au moins 4 auront perdu leur mise ! Le résultat ne sera donc pas au niveau des exigences de réalisation.

Cela se concrétisera par une contractualisation (donc fermeture) sur des bases extrêmement floues entre un développeur qui promet de réaliser dans un certain temps pour un certain montant des fonctionnalités qui ne sont pas clairement détaillées à un client qui ne sait pas encore vraiment ce qu'il veut !

Tous les ingrédients d'un marché de dupes sont présents.

La scène deux se déroule dans le même décor, le bureau du client. Le développeur effectue la livraison du logiciel. Il est en retard car il a rencontré des problèmes techniques non prévus (et en partie imprévisibles sinon dans la certitude de leur existence à un moment ou l'autre du projet). Il est fatigué,

stressé, car il a compilé la dernière version au milieu de la nuit. Le client est inquiet, sa confiance est ébranlée à cause du retard.

- Client : Ah ! voilà enfin le logiciel. Montrez-moi ça.

Démonstration par le développeur, ça se passe bien, le client reprend confiance, le développeur se détend.

- Client : Bon, c'est bien. Juste un petit détail, vous avez bien pensé que tout ce qu'on imprime il faut pouvoir l'exporter, bien sûr ?
- Développeur : *(un temps)* pardon ?
- Client : ben oui, il faut bien qu'on puisse le réutiliser pour *(ceci-cela)*
- Développeur : mais vous ne me l'aviez pas demandé !
- Client : oui, mais c'était évident. Si on ne peut pas exporter votre logiciel ne sert à rien !

...

Au premier désaccord sur le périmètre fonctionnel résultant d'une différence d'interprétation, le client aura le sentiment que le développeur essaye de le gruger en ne remplissant pas ses engagements. Le développeur lui ne pourra pas inclure sans y perdre son bénéfice une demande qui n'était pas prévue. Les deux protagonistes sont, en général, de bonne foi. Le client estime sincèrement que la fonctionnalité en question était implicite. Le développeur expose rationnellement que la fonctionnalité n'était pas explicitement demandée.

Et c'est le conflit...Les portes claquent.

Si les deux acteurs sont intelligents, un compromis sera trouvé, mais outre le temps perdu, la dynamique cassée, il laissera un goût d'amertume de chaque côté, chacune des parties s'estimant perdante. Le climat de confiance ne sera pas restauré et la qualité du logiciel ira en décroissant. Le client n'étant pas satisfait de son produit, le développeur n'ayant plus aucun désir de satisfaire un client qui ne reconnaît pas la valeur de son travail.

Le développement Agile

En réaction aux échecs rencontrés dans l'application des processus de développement classiques et en opposition à la lourdeur de leur aspect procédurier, plusieurs sociétés ou intervenants du monde du développement logiciel se sont mis d'accord sur un manifeste commun de promotion de méthodes dites **Agiles**.

Ces méthodes valorisent :

- **Les interactions et les individus** par rapport aux processus et aux outils
- **La livraison d'un logiciel fonctionnel** par rapport à une documentation exhaustive
- **La communication avec le client** par rapport à la contractualisation
- **La prise en compte du changement** par rapport au respect d'une planification figée

Tout en reconnaissant de la valeur aux éléments situés à droite des énoncés précédents, elles en accordent d'avantage à ceux situés à gauche.

Partant du constat que « trop de méthode tue la méthode », la réflexion commune s'appuie sur deux caractéristiques principales :

- Les méthodes agiles sont « **adaptives** » plutôt que prédictives : elles reconnaissent le changement comme une composante incontournable du processus de développement ;
- Les méthodes agiles, d'inspiration humaniste plus que cartésienne, sont orientées vers les personnes plutôt que vers les processus.

Vers une démarche de partenariat

Les relations entre maîtrise d'ouvrage et maîtrise d'œuvre conditionnent la réussite d'un projet. Les deux parties doivent **travailler ensemble** pour bâtir un logiciel de qualité à leur satisfaction mutuelle. Le développement logiciel doit constituer une activité plaisante où chacun se voit confier une part de responsabilité dans **une stratégie gagnant-gagnant**.

Cela impose une révision des pratiques contractuelles, en particulier le forfait, qui présuppose un accord préalable sur ce qui doit être réalisé, entraînant les intervenants dans une attitude de défense du périmètre figé. ASD (*Adaptive Software Development*), l'une des méthodes agiles, propose ainsi de se focaliser plutôt sur la définition d'**une mission** qui servira de référentiel dans l'exploration itérative des spécifications puis dans leur réalisation.

Les méthodes agiles reconnaissent des droits au client et au développeur, c'est dans la reconnaissance et le respect réciproque de ces droits que réside le **contrat de collaboration**.

Les douze principes (extrait du livre blanc de « Business Interactif »)

1. Notre priorité est de satisfaire le client en lui livrant très tôt et régulièrement des versions fonctionnelles de l'application source de valeur
2. Accueillir le changement à bras ouverts, même tard dans le processus de développement. Les méthodologies agiles exploitent les changements pour apporter au client un avantage concurrentiel
3. Livrer le plus souvent possible des versions opérationnelles de l'application, avec une fréquence comprise entre deux semaines et deux mois
4. Clients et développeurs doivent coopérer quotidiennement tout au long du projet
5. Construire des projets autour d'individus motivés. Leur donner l'environnement et le support dont ils ont besoin et leur faire confiance pour remplir leur mission
6. La méthode la plus efficace de communiquer des informations à une équipe et à l'intérieur de celle-ci reste la conversation en face à face.
7. Le fonctionnement de l'application est le premier indicateur d'avancement du projet
8. Les méthodes agiles recommandent que le projet avance à un rythme soutenable : développeurs et utilisateurs devraient pouvoir maintenir un rythme constant indéfiniment
9. Porter une attention continue à l'excellence technique et à la conception améliore l'agilité
10. La simplicité — art de maximiser la quantité de travail à ne pas faire — est essentielle
11. Les meilleures architectures, spécifications et conceptions sont le fruit d'équipes qui s'auto-organisent
12. A intervalles de temps réguliers, l'ensemble de l'équipe s'interroge sur la manière de devenir encore plus efficace, puis ajuste son comportement en conséquence.

4^{ème} Dimension un outil agile

Quoi qu'inspiré par la Programmation Orientée Objet (POO), le mouvement Agile promeut des concepts dont lesquels nous nous retrouvons totalement. À la fois par conviction et à la lumière de notre expérience de développement dans le monde micro-informatique depuis la fin des années 80.

Bien avant la publication du manifeste, nos meilleures expériences de développement l'ont été dans le cadre de projet mettant en œuvre une réelle collaboration dans un climat de confiance.

Notre outil de développement privilégié, 4^{ème} Dimension de la société française 4D S.A., de par sa souplesse et son intégration, a toujours été le support de notre agilité.

En 2003, son architecture intégrée nous permet ainsi d'offrir, sans rupture de développement, un déploiement monoposte ou client/serveur, multi plateforme MacOS et Windows, un serveur Web intégré, une interopérabilité ODBC, Oracle, XML, et une ouverture aux composants distribués à travers les services Webs via SOAP.

Extraits - Bibliographie - Liens

- L'eXtreme Programming, ed. Eyrolles
P2 chap. 3 « Selon les résultats d'une enquête menée en 1994 sur un échantillon de 8 000 projet, seuls 16% d'entre eux ont été finalisés dans le temps et le budget initialement prévus. Pire, 32 % de ces projets ont été abandonnés en cours de route »
- Jean-Pierre Vickoff : www.RAD.fr
- Agile Alliance : <http://agilealliance.org>
- Livre blanc de business Interactif : www.businessinteractif.fr
- Dossier de O1 net : www.O1net.com/article/151119.html
- Livre blanc de DesignUp sur l'eXtreme programming : www.design-up.com