Véronique **Messager Rota**
Préface de Jean **Tabaka**

Gestion de projet

Vers les méthodes agiles

ISBN : 978-2-212-12165-0

EYROLLES

2

Méthodes traditionnelles ou méthodes agiles ?

L'entreprise doit avoir l'agilité du banc de poisson qui agit et réagit avec vitesse, simultanéité et précision.

Jean-Pierre Chardon, P-DG Schneider Electric France, préface in Jérôme Barrand, *Le Manager agile*, Dunod, 2006.

Depuis des décennies, les projets sont gérés avec une approche classique, le plus fréquemment « en cascade » ou son adaptation « en V », basée sur des activités séquentielles : on recueille les besoins, on définit le produit, on le développe puis on le teste avant de le livrer au client.

Ces méthodologies se caractérisent par un attachement farouche à tout planifier, « tout doit être prévisible », en tout début de projet. Voilà pourquoi on les qualifie d'approches « prédictives ». Un plan de management du projet décrit comment et quand le travail sera réalisé, les modalités de planification, d'exécution, de suivi et de clôture du projet.

Cette volonté persistante de vouloir piloter le projet par les plans (*plan-driven development*) a conduit les acteurs d'un projet à redouter, voire à s'opposer systématiquement à tout changement : changement dans le contenu ou le périmètre du projet, dans le processus de développement, au sein de l'équipe, bref à toute modification des plans initiaux, auxquels on doit rester conforme.

« Quand on trouve une recette qui marche bien, on a du mal à la quitter même si l'on constate que son efficacité semble diminuer ; il existe une inertie due à la peur du

changement, à la recherche de facilité ou à l'ivresse du succès (ce qui marchait hier doit marcher demain…). Eh bien non !!![1] »

Fort du constat que les plans initiaux sont finalement toujours modifiés et que les besoins évoluent en permanence pour répondre aux changements du marché, ces approches prédictives se sont révélées trop « rigides » parfois, exposant les organisations à trop peu de réactivité dans le contexte de nouveaux projets stratégiques.

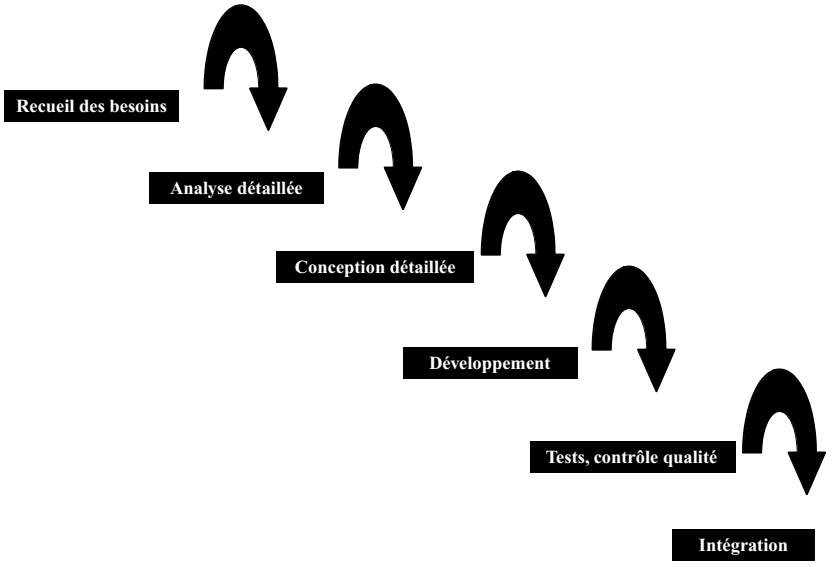
Sont alors apparues, dans les années 1990, des méthodes moins prédictives, plus souples face aux besoins d'adaptation, facilitant ainsi l'agilité des organisations face aux contraintes du marché.

Ce sont les méthodes dites « agiles».

Limites des approches classiques

Caractéristiques d'une approche « en cascade »

Figure 2-1
Les phases du cycle « en cascade »



Le cycle en « cascade » se caractérise par des phases séquentielles, qui se succèdent après la validation des livrables produits lors de la phase précédente :

- Tous les besoins sont exprimés et recueillis lors de la première phase, puisque l'analyse détaillée de ces besoins, puis la conception du système qui répondra à ces besoins, en dépendront.

1. Jérôme Barrand, in *Le Manager agile*, *op. cit*.

- La conception du système, bien que textuelle ou représentée sous forme de diagrammes, doit être validée avant le démarrage des développements.
- Les développements doivent être achevés pour permettre à l'équipe de testeurs de lancer ses campagnes de tests fonctionnels et techniques.
- Enfin, une fois, et seulement une fois, que les anomalies ont été corrigées, on peut procéder à l'intégration globale finale et à la mise en production du système.

Dans ce contexte, et sur la base du périmètre défini, on demande au chef de projet de s'engager sur un planning détaillé de réalisation, prévoyant les jalons de début et fin de phases, et les activités à mener.

On devine très rapidement, si l'on ne les a déjà expérimentées, les failles de cette approche.

Les failles d'une approche « en cascade »

Métaphore du chalet

Prenons l'exemple du projet de construction d'un chalet à la montagne : le client souhaite un chalet différent de ceux qui peuplent déjà la station, original, construit avec des matériaux écologiques et respectant les nouvelles normes environnementales. Il a confié la réalisation des plans à un architecte qui a pris en compte ces considérations ; mais le client sait que certains choix et certaines décisions ne pourront être définitifs qu'après avoir vu réellement le chalet prendre forme. Il sait que le coût du chalet ainsi que le délai de réalisation pourront évoluer en fonction de ces choix définitifs, même s'il dispose d'une enveloppe globale qu'il ne souhaite pas dépasser et d'une date de livraison approximative.

Cet exemple paraît tout à fait courant pour celui qui a déjà été confronté à un tel projet et personne ne contesterait la légitimité des hésitations de ce client devant un engagement important.

Comment, dans ces conditions, fournir un planning détaillé de tous les travaux, de toutes les ressources matérielles et humaines, de toutes les fournitures… en sachant qu'au gré des choix du client, la nature même des travaux et des matériaux pourrait évoluer ? Comment fournir un budget détaillé définitif, alors que le client, en fonction de certains choix, devra peut-être renoncer à tel ou tel équipement initialement prévu pour rester dans l'enveloppe budgétaire qu'il s'est fixée ?

Le développement d'un nouveau produit ou d'un nouveau logiciel est confronté aux mêmes conditions : le chef de produit marketing aura sondé ses consommateurs pour définir son nouveau produit, mais lors de la réalisation, certaines attentes ne pourront être satisfaites en raison d'un surcoût imprévu ou d'une nouvelle norme incompatible apparue en cours de développement ; le responsable des ressources humaines d'une grande entreprise aura associé ses collaborateurs pour définir les fonctionnalités d'un nouveau logiciel de recrutement en ligne ; mais la complexité technique sous-estimée au départ ou la difficulté d'intégration avec le système central en place contraindront peut-être l'équipe à revoir à la baisse ses attentes ou à accepter un délai supplémentaire ; ou bien encore, le site d'un concurrent faisant apparaître une nouvelle fonctionnalité attrayante amènera le responsable à faire évoluer son cahier des charges en ajoutant cette nouvelle demande en cours de projet.

La rigidité de l'approche

On déplore que la nouveauté, la marge de manœuvre laissée, à juste titre, au client pour préciser ou faire évoluer ses attentes, la non-prévisibilité de tous les événements soient difficilement compatibles avec une approche prédictive comme celle du cycle en cascade.

> À la différence, lors des projets industriels de développement de produits sur une chaîne d'assemblage, tout est (presque) prévisible et le degré de nouveauté (presque) néant : les spécifications peuvent alors être précises dès le début et le budget ainsi que le délai sûrement établis.

En fait, une fois le plan de management du projet validé, il constitue la référence de base. La préoccupation majeure du chef de projet devient alors de coller au plus près au plan, quels que soient les événements ; tout écart constaté, concernant la durée des activités, la productivité ou la disponibilité des ressources ou encore les risques imprévus, est perçu comme un échec, vécu par certains comme une incompétence ou une incapacité à anticiper.

L'approche « en cascade » est par conséquent trop rigide pour permettre des retours en arrière ; elle suppose que l'on fasse bien du premier coup. Une décision ou une anomalie détectée dans une phase aval de la cascade peuvent remettre en cause partiellement ou totalement des travaux validés précédemment et considérés comme définitifs.

> Comment revenir sur une conception validée il y a deux mois lorsque l'on constate, à la fin des développements, que l'architecture développée ne permet pas de respecter les exigences de performance ? D'autant que l'approche en cascade n'encourage pas, explicitement, le prototypage qui aurait pu éviter cette mauvaise surprise.

L'effet tunnel

Figure 2-2
La « boîte noire »

L'effet tunnel est une autre des caractéristiques de l'approche « en cascade » : un projet dure un an, la phase de recueil des besoins dure deux mois et le client ne voit le résultat que neuf mois plus tard !

Que s'est-il passé entre-temps ? « On ne sait pas trop ce qu'ils font ces informaticiens ! », « Que va-t-il sortir de la « boîte » ? », « Mais, ce n'est pas ce que l'on attendait ! » ou bien « C'est ce que nous voulions mais notre besoin a un peu évolué depuis ! »

D'une part, la non-transparence des équipes de développement suscite des sarcasmes sur leur capacité à coopérer, d'autre part, la longueur des phases techniques auxquelles le client n'est pas associé rend celui-ci dubitatif sur le résultat à venir. Ce qui ne favorise pas la collaboration efficace entre informaticiens et utilisateurs !

D'autant plus si le résultat livré n'est pas conforme à ce qui est attendu.

Une mauvaise communication

L'absence de jalons intermédiaires prohibe la validation de ce que sera la version finale du produit. Il faut attendre que la phase de développement soit bien avancée pour découvrir les premiers écrans. Les mauvaises surprises en fin de cycle de vie et le refus du changement par les équipes de développement pénalisent la qualité des relations avec les utilisateurs. Elles en deviennent même parfois conflictuelles ; les uns s'attachent fermement à leurs plans initiaux pour livrer ce qui était convenu à l'échéance prévue, même si le résultat ne correspond plus ou pas totalement à ce qui est réellement attendu ; les autres ressentent cette rigidité comme un désintérêt pour la valeur ajoutée du produit final.

La succession d'intervenants, au travers des différents corps de métier, nuit également à la fluidité de l'information, crée même une déperdition d'information et d'énergie ainsi que de nombreuses ruptures de charge.

La levée tardive des facteurs à risques

Dans un cycle de vie « en cascade », les facteurs à risques sont levés tardivement, puisque les tests de performance ou d'intégration, par exemple, sont reportés après les développements, tout comme l'appréciation des IHM (Interface homme-machines), qui – on le sait – sont souvent sujettes à d'interminables débats très subjectifs.

L'impact des risques augmente avec l'avancement du projet, puisque plus une anomalie est détectée tardivement, plus le retour arrière est complexe, plus sa correction coûtera cher et plus les effets de bords seront menaçants.

Une documentation pléthorique

Afin de se prémunir contre ces risques, l'approche « en cascade » s'attache fortement à la production d'une documentation importante.

La documentation permet de repousser le moment où il va falloir aborder la phase de codage, phase irréversible.

Elle rassure et, s'il en était nécessaire, elle apporte la preuve que la réalisation progresse ; elle matérialise l'avancement et engage les parties prenantes. En effet, il est plus aisé de s'opposer au changement en brandissant un document contractuel validé précédemment !

Malheureusement, cette documentation, souvent trop pléthorique, ne reflète pas la réalité des développements : on a beau valider un document d'architecture, celle-ci reste théorique et conceptuelle tant qu'elle n'est pas implémentée et testée dans des conditions réelles ; on a beau présenter des maquettes papier au client, celui-ci est plus sensible à ce qu'il voit concrètement sur un écran (*IKIWISI, I'll Know It When I See It* !).

Au final, on s'interroge sur l'utilité de cette documentation, qui n'est, en outre, pas toujours mise à jour tout au long du projet et devient donc vite inexploitable.

Dans ce contexte de méthodes trop rigides, comment augmenter le niveau de satisfaction des clients tout en facilitant la gestion des projets et en améliorant la qualité des développements ?

C'est précisément avec les méthodes dites « agiles » que l'on va pouvoir adopter une approche plus souple, plus « adaptative » aux aléas du projet.

Une alternative : les méthodes agiles

Qu'est-ce qu'une méthode agile ?

Qu'est-ce qu'une méthode agile ?

Une méthode agile est une approche itérative et incrémentale, qui est menée dans un esprit collaboratif, avec juste ce qu'il faut de formalisme. Elle génère un produit de haute qualité tout en prenant en compte l'évolution des besoins des clients.

Une approche itérative et incrémentale

Le principe du développement itératif consiste à découper le projet en plusieurs étapes d'une durée de quelques semaines ; ce sont les *itérations*. Au cours d'une itération, une version minimale du produit attendu est développée puis soumise, dans sa version intermédiaire, au client pour validation. Les fonctionnalités sont ainsi intégrées au fur et à mesure du cycle de vie sur un *mode incrémental*, le système s'enrichissant progressivement pour atteindre les niveaux de satisfaction et de qualité requis.

Chaque itération est un mini-projet en soi qui comporte toutes les activités de développement, menées en parallèle : analyse, conception, codage et test, sans oublier les activités de gestion de projet. L'objectif est d'obtenir, au terme de chaque itération, un sous-ensemble opérationnel du système cible et, au terme de la dernière itération, la version finale du produit.

Attention

Le résultat d'une itération n'est pas un prototype ou une « proof of concept », mais bien une version intermédiaire du produit final.

Les itérations se succèdent et ne peuvent être parallélisées ; elles correspondent à des « tranches de temps » ou des « boîtes de temps » dont la date de fin est fixe. Elles sont d'ailleurs nommées, dans la littérature dédiée, *sashimi*. Ce terme japonais décrit l'assiette où sont magnifiquement rassemblées toutes les lamelles du poisson.

- L'idée maîtresse est de reconnaître que l'on ne peut pas tout connaître ni tout anticiper, aussi longue soit notre expérience ; il est, par conséquent, plus sage d'avancer prudemment, pas à pas et de s'adapter au fur et à mesure, en tenant compte des spécificités du projet, plutôt que de tout prévoir et tout planifier à l'excès, en sachant qu'inévitablement des changements plus ou moins prévisibles surviendront en cours de projet.
- On ne dispose plus d'un plan de management du projet unique, établi au début du projet, qui planifie une liste d'activités plus ou moins détaillée ; mais on dresse une liste de besoins macroscopiques et on élabore un macroplanning initial qui fixe les grandes échéances et les jalons principaux du projet. À chaque itération, on sélectionne, avec le client, les fonctionnalités qui seront détaillées puis développées, en fonction de leur priorité et on établit le microplanning correspondant aux activités nécessaires pour le développement de ces fonctionnalités.

- Le principe du *timeboxing* – une date d'échéance fixe immuable pour l'itération – permet de mobiliser les efforts sur des objectifs clairs à court terme. Si les objectifs ne sont pas atteints, les enseignements seront tirés lors du bilan de l'itération afin de corriger les conditions de l'itération suivante, si nécessaire.
- Les avantages de l'approche itérative et incrémentale sont présentés dans le tableau 2-1.

Tableau 2-1 Avantages du développement itératif et incrémental

Avantage	Les +
La communication est de meilleure qualité.	Les malentendus, incompréhensions, incohérences sont mis en évidence tôt dans le projet ; il est donc encore possible de les corriger. L'utilisateur a la possibilité de clarifier ses exigences au fur et à mesure. Le client reçoit des « preuves » tangibles de l'avancement du projet.
La visibilité est meilleure.	Le client peut ainsi visualiser les travaux plus régulièrement, au fil de l'eau, sans attendre la fin du projet, puisqu'à la fin de chaque itération, les fonctionnalités retenues sont développées, testées, documentées et validées, prêtes pour l'exploitation.
La qualité est évaluée en continu.	Les tests sont effectués à chaque itération, les anomalies détectées sont corrigées au fur et à mesure.
Les risques sont détectés très tôt.	Grâce aux activités de développement précoces, les risques sont détectés tôt et résolus rapidement.
L'équipe prend confiance.	L'itération donne une occasion d'apprendre, donc de capitaliser ou d'adapter les pratiques pour la suite du projet. Les premières itérations fiabilisent les prévisions. Le changement n'est plus une menace, mais au contraire, l'opportunité de mieux faire et de mieux satisfaire le client.
Les coûts sont contrôlés.	Les coûts sont limités, en termes de risques, au périmètre de l'itération ; s'il faut reprendre une itération, on ne perd que les efforts de cette itération et non la valeur du produit dans sa globalité. On peut aussi arrêter le projet à l'issue de quelques itérations si l'on n'a plus de budget.

Freddy Mallet[a] nous précise, avec humour, les avantages du développement itératif

Cette notion de timebox (ou *time frame*) est omniprésente dans les méthodologies agiles, mais que signifie évoluer dans une boîte de temps ? C'est tout simplement se donner une contrainte de réalisation « au mieux » d'un ou plusieurs objectifs dans un cadre de temps immuable et de préférence le plus court possible.

Prenons une métaphore qui parle directement ou indirectement au plus grand nombre pour comprendre l'intérêt de la démarche. Vous vous dites qu'il serait souhaitable que vous perdiez 10 kilos en quelques mois pour vous sentir bien dans vos baskets. Quelle que soit la recette diététique utilisée : le régime dissocié, le sport, slim-fast, la prière, etc., pourquoi est-il préférable de découper, voire d'oublier cet objectif général de perte de 10 kilos au profit de boîtes de temps de 2 semaines par exemple avec des objectifs précis en début de chaque boîte de temps ?

– **Appréhension de la réalité** : si je vous dis que vous devez perdre 10 kilos en 4 mois, est-ce que ça vous parle ? Si je vous dis que vous devez perdre 600 g par semaine, est-ce que ça vous parle mieux ? À titre personnel oui, car en découpant l'objectif initial, on s'approche d'une

☞

Freddy Mallet[a] nous précise, avec humour, les avantages du développement itératif *(suite)*

réalité plus palpable qui nous contraint à un regard également plus objectif. Après ce premier découpage, vous allez peut être vous dire que 400 g, c'est déjà pas mal et allez donc partir sur une base plus viable.

– **Maîtrise du risque** : si jamais vos objectifs sont trop élevés, le fait de raisonner en boîte de temps permet de vous remettre en question et éventuellement de rectifier le tir dès la première ou deuxième boîte de temps. À l'inverse, si vous vous étiez fixé un objectif de perte de 10 kilos sur 6 mois sans étape intermédiaire, votre faculté de vous mentir à vous-même l'aurait peut être emporté sur votre objectivité. En gros, les boîtes de temps vous contraignent à un feedback honnête sur vos avancées. Dans un cas, vous allez donc éventuellement revoir votre objectif initial à 8 mois et au final vivre l'expérience comme un succès, et dans l'autre cas potentiellement échouer là où vous auriez pu réussir.

– Entretien d'une pression quotidienne positive ou plutôt **régulation de la pression** : que se passerait-il si, à l'école, l'évaluation des connaissances ne se faisait qu'une seule fois à la fin de l'année ? Grossièrement les étudiants travailleraient en sur-régime le dernier trimestre et « glanderaient » plus ou moins fort les deux premiers trimestres. L'être humain est ainsi fait. Donc il faut trouver un moyen d'étaler dans le temps cette pression bénéfique quand elle est bien gérée.

– **Motivation** : plus un objectif est à portée de main, plus la motivation pour atteindre cet objectif sera naturelle et importante. Perdre 10 kilos en 4 mois ou 600 g par semaine revient au même. Seulement dans le deuxième cas vous avez « la rage », le *fighting spirit*, et vous allez même peut-être dépasser vos espérances.

– **Satisfaction** : en reprenant l'exemple précédent, dès la première semaine si vous atteignez votre objectif de perte de 600 g, vous ressentez une unité de satisfaction (notion équivalente à un *stroke* en analyse transactionnelle) que vous allez vouloir ressentir à nouveau dans les prochaines boîtes de temps. Ces unités de satisfaction sont l'énergie qui va permettre de soutenir la dynamique de progression.

– **Augmentation des degrés de liberté** : cette dernière notion peut paraître étrange puisqu'une boîte de temps est avant tout une contrainte, donc comment l'ajout d'une contrainte peut-il augmenter mes degrés de liberté ? En raison d'un aspect psychologique assez facile à comprendre et qui s'applique à bon nombre d'entre nous : plus la montagne qu'on nous demande de gravir est haute, moins nous allons nous autoriser d'expérimentation, figés par l'enjeu du défit. À l'inverse si nous évoluons dans des boîtes de temps, que les premières se passent bien, nous nous sentons maître de la situation et gagnons donc en liberté d'esprit et d'action.

Réaliser un projet de développement d'une charge de 1 000 jours/homme est aussi simple et compliqué que de perdre 10 kilos en 4 mois !

a. Voir http://www.freddymallet.com/

Un esprit collaboratif

L'une des valeurs essentielles des méthodes agiles est de placer les individus et leurs interactions au centre du dispositif, plutôt que de mettre au point et de « sur-outiller » des processus lourds.

Elles privilégient en effet la communication entre les différents acteurs d'un projet, au sein de l'équipe mais également entre l'équipe et ses différents interlocuteurs comme le

client et les utilisateurs. On entend par communication le partage d'information, l'échange de points de vue différents ou complémentaires, l'entraide et non la concurrence, les relations « partenariales » avec le client…

Cet esprit d'équipe peut s'exprimer au travers des qualités suivantes :

- le respect des opinions des autres ;
- la capacité à exprimer des opinions différentes de façon non agressive ;
- l'aptitude à rechercher et atteindre le consensus sans frustration ;
- une prédisposition à l'autodiscipline, voire à l'autogestion.

On mesurera l'importance de ces qualités dans l'organisation de l'équipe, dans la prise de décision, dans la prévention ou la résolution des conflits, dans le dialogue avec le client.

La compétence des collaborateurs, leur motivation et la possibilité, pour chacun, d'exprimer son individualité (au service du groupe) favoriseront la créativité et la performance de l'équipe et garantiront les meilleures chances de succès au projet.

Le rôle du chef de projet s'en trouve modifié : au lieu de « commander » et contrôler son équipe, il devient le manager qui sait créer les conditions optimales pour permettre à chacun de contribuer efficacement au résultat de l'équipe en vue d'une meilleure satisfaction du client. Ce rôle de manager sera largement décrit au chapitre 6, dédié au management des hommes.

Un formalisme léger

On qualifie souvent les méthodes agiles de méthodes « légères », en comparaison avec les méthodologies classiques qui exigent un formalisme et un outillage « lourds ».

Seulement quelques livrables à produire, en plus de l'essentiel (les versions intermédiaires du produit), quelques rôles définis, quelques étapes, quelques réunions… et la démarche est formalisée.

> Une différence entre les deux approches est essentielle : seuls les éléments clés sont « prescriptifs », il y en a peu mais ils doivent être suivis avec rigueur ; cela entre en opposition avec les méthodes classiques pourvues de nombreux points dont aucun n'est réellement suivi sérieusement.

Des outils, oui, mais efficaces, à bon escient et réduits au strict nécessaire pour l'automatisation des tâches récurrentes, en particulier les tests et l'intégration continue. La compétence des ressources et la communication entre elles sont, on vient de le voir, privilégiées ; par conséquent, on ne doit pas inutilement doter une équipe d'outils complexes auxquels elle devra se former et s'adapter ; il faut des outils qui s'adaptent à la façon de travailler[1], pour supporter la démarche, mais qui ne sont pas une fin en soi.

1. Voir *L'agilitateur*, *http://agilitateur.azeau.com/*

Cette légèreté offre l'avantage de faire évoluer l'organisation, les processus et les outils, si nécessaire ; on est bien sur une approche adaptative, on parle même d'approche *empirique* : on observe, on ajuste, on expérimente, on apprend, on corrige… Le processus agile de départ est défini au démarrage du projet ; au fur et à mesure, l'équipe découvre ce qui fonctionne dans le contexte du projet, soumet à des discussions ce qui ne fonctionne pas, améliore le dispositif, en fonction des spécificités du projet… et ce précisément grâce à sa simplicité.

Un produit de haute qualité

Les méthodes agiles sont parfois qualifiées, par leurs détracteurs, de méthodes artisanales ou de « bricolage », ce qui revient à dire que la qualité n'y est pas une préoccupation essentielle.

Si l'on considère que le niveau de qualité minimal d'un produit est sa capacité à satisfaire le client, tant sur le plan fonctionnel que sur celui des exigences de performance, de facilité d'utilisation ou d'évolutivité, c'est précisément là une autre des idées fondamentales de l'approche agile : satisfaire le client et lui apporter de la valeur.

- Grâce, tout d'abord, à la sélection des fonctionnalités à implémenter en priorité, basée sur la livraison de valeur en continu ; en effet, on s'attachera à développer et livrer rapidement celles qui ont une importance capitale pour le client. On évitera ainsi de vouloir satisfaire l'exhaustivité des besoins exprimés initialement qui ne sont pas toujours utiles ni porteurs de valeur à l'arrivée.
- Grâce notamment au feedback permanent recueilli auprès du client en lui montrant une version intermédiaire aboutie du produit – celui-ci est aligné en permanence sur les attentes qui peuvent évoluer. Le résultat est visible et non décrit théoriquement dans une documentation.
- Grâce à des campagnes de tests et au contrôle qualité au cours de chaque itération, tout défaut peut être détecté et corrigé immédiatement.
- Grâce au *refactoring* – micro-évolutions ou « nettoyages » quotidiens du code, intégrés dans les activités de développement –, on évite toute dégradation progressive du code en améliorant sa lisibilité et en améliorant sa maintenabilité. En effet, le refactoring, en éliminant les duplications anarchiques dans le code, assure que le code fait une seule chose à un seul endroit. C'est là le signe d'un code bien conçu.
- Grâce à l'adoption d'une approche adaptative, la qualité du processus – qui conditionne la qualité du produit – est également régulièrement mesurée au cours de revues : tout écart constaté fait l'objet d'une discussion et d'une modification éventuelle.
- Grâce, également, au respect de normes de codage partagées par tous les membres de l'équipe, l'évolutivité de l'application est garantie.

Au final, on a du mal à penser que les méthodes agiles ne placent pas au centre de leur démarche la qualité et la satisfaction du client ! Dans la pratique, elles se révèlent plus disciplinées et offrent un meilleur pilotage.

Peut-on vraiment montrer et livrer quelque chose au client à une telle fréquence ?

1) La réponse du coach **Freddy Mallet**, consultant spécialisé sur les méthodologies agiles.

Il n'existe pas de fréquence idéale mais disons que dans la pratique cette fréquence est souvent égale ou inférieure à 1 mois. Avant de se poser la question de la faisabilité, posons-nous la question du pourquoi montrer et livrer quelque chose au client à une telle fréquence ?

À vrai dire, comment faire autrement pour mesurer objectivement l'avancement d'un projet de développement ?

La question peut paraître simpliste mais si, sur un projet de développement informatique, vous avez 500 fonctionnalités (*use cases*, *user stories*, etc.) à implémenter et qu'au bout de 2 mois, vous en avez toujours 500 à implémenter, comment faites-vous à la fin de ces deux mois pour ré-évaluer objectivement votre charge et votre planning ? On peut toujours se rassurer en se disant qu'on a fait un énorme travail d'architecture, de documentation et de formation mais la vérité c'est qu'au bout de ces deux mois, on n'en sait pas plus sur notre capacité à délivrer les fonctionnalités requises.

À l'inverse, si vous vous efforcez de délivrer des fonctionnalités à chaque fin d'itération, vous allez très rapidement pouvoir évaluer objectivement votre niveau de productivité et affiner itération après itération la charge et la date de fin sans grande possibilité de vous mentir à vous-même.

C'est également un bon moyen pour l'équipe de développement de rester focalisée sur son objectif principal : délivrer des fonctionnalités ayant une valeur pour le client. Là encore, la remarque peut paraître presque simpliste mais prenez un développeur au hasard dans votre entreprise et posez-lui la question : « Quel est ton objectif principal ? » Les réponses seront-elles toutes orientées client ?

Enfin, c'est offrir au client la possibilité de confronter la représentation abstraite qu'il s'était faite du produit avec la réalité. C'est mettre en œuvre un principe de rétroaction sans lequel il ne peut y avoir d'agilité business possible. Il existe une chimère assez répandue au sein des services de développement informatique qui consiste à considérer que le client doit pouvoir parfaitement exprimer et spécifier au pixel près son besoin pour les 8 à 12 mois à venir, et ce de manière totalement abstraite. L'exercice est comparable en difficulté à imaginer et spécifier dans un document Word toute la décoration d'une maison virtuelle sans assistance informatique. Cette chimère donne lieu à des batailles ou à des tensions régulières entre maîtrise d'ouvrage et maîtrise d'œuvre. Je joue moi-même le rôle du client dans le cadre du développement d'un outil de contrôle de qualité logicielle (Sonar). Autant je sais exactement où je veux aller, autant il m'est très difficile de spécifier mon besoin. Chaque période de test offerte à la fin d'une itération me permet d'ajuster ma vision du produit. C'est aussi cela l'agilité.

Une fois la question du pourquoi levée, les problématiques réelles de mise en œuvre qui peuvent se poser portent essentiellement sur le niveau d'implication du client. Cependant, si on arrive à expliciter auprès de ce même client les bénéfices qu'il peut retirer de cette démarche itérative et incrémentale, il y a de grandes chances qu'il soit prêt à jouer le jeu.

2) La réponse du coach **David Gageot**, directeur technique chez Valtech Technology Consulting, cabinet de conseil en technologies Agiles.

J'aime renverser cette question : comment penser que l'on peut être capable de livrer en une seule fois une application ? Développer seulement quelques fonctionnalités à la fois permet un engagement concret des utilisateurs. Le fait de leur montrer régulièrement l'application en cours de croissance facilite leur feedback et leur permet de mieux creuser leurs propres besoins. ☞

Peut-on vraiment montrer et livrer quelque chose au client à une telle fréquence ? *(suite)*

Le découpage facilite également le travail des développeurs. Plutôt que de gérer à la fin tous les problèmes tels que les bogues, la montée en charge, l'ergonomie, l'installation ou l'adéquation avec le métier, ils ont l'opportunité de trouver des solutions petit à petit et de les valider au plus tôt.

Cependant, il est vrai que c'est parfois un challenge technique que de parvenir à livrer des modules indépendants fonctionnellement et techniquement. Il faut alors penser nos applications comme un ensemble de services indépendants, capables de collaborer (façon SOA), soit comme un ensemble de plug-ins (façon Eclipse). Cette dernière solution est une approche facilitée par les dernières avancées techniques.

3) La réponse de **Dominic Williams**, coach XP.

Oui, les équipes avec lesquelles j'ai pratiqué XP étaient réellement capables de livrer aux utilisateurs une nouvelle version chaque semaine, et il ne s'agit pas de cas isolés.

Pour y arriver, le premier ingrédient est d'éliminer du processus toutes les files d'attentes et passages de relais. Il faut une équipe polyvalente et autonome. Chaque membre de l'équipe doit être capable de tout faire et l'équipe ne doit pas dépendre d'une autre équipe, par exemple pour faire les tests.

Le deuxième ingrédient est l'automatisation. Tout ce qui est répétitif doit être automatisé : compilation, installation, exécution des tests, mesure du temps passé par tâche, etc.

Enfin, il faut avoir des gens à qui livrer et prêts à tenir le même rythme. Un maître d'ouvrage disponible et consciencieux peut faire l'affaire, mais la véritable agilité consiste à court-circuiter cet intermédiaire et livrer directement aux utilisateurs. Certains sont impatients d'avoir les dernières nouveautés et prêts à changer leurs habitudes de travail. Dans d'autres cas, il faut prévoir, par exemple, d'introduire les nouveautés de façon optionnelle.

L'acceptation du changement

Embrace change, dit Kent Beck, l'un des « pères » du mouvement agile… « Accueillez le changement à bras ouverts » plutôt que de le craindre et de le combattre.

On sait que nombre de paramètres sont imprévisibles lors d'un projet ; il s'agit alors de mieux contrôler cette imprévisibilité sans la nier en voulant être systématiquement conforme aux plans initiaux rapidement obsolètes.

On échappera, de fait, aux gaspillages de temps et d'énergie et aux frustrations qui en résultent, constatés sur les projets qui ne peuvent admettre le changement : temps (souvent conséquent) consacré à l'élaboration du planning, temps dédié à l'analyse des écarts, efforts fournis pour rattraper le retard, temps accordé à la négociation et au refus des changements, temps affecté à remobiliser l'équipe…

Une équipe agile se dote de pratiques et d'outils lui facilitant l'accueil du changement.

Origine et valeurs des méthodes agiles

Le mouvement des méthodes agiles est né en 2001 aux États-Unis. Devant l'observation faite du taux important d'échec des projets, notamment dans les années 1990, dix-sept experts en développement logiciel, qui avaient chacun déjà mis au point et expérimenté de nouvelles méthodes, se sont réunis afin d'échanger et de trouver un socle commun de valeurs et de bonnes pratiques.

Le résultat de cette réflexion a abouti au *Manifeste pour le développement logiciel agile*[1] et la création de *l'Agile Alliance*[2], chargée de promouvoir l'agilité dans les organisations et d'apporter du soutien aux équipes qui veulent démarrer un projet agile.

Le *Manifeste* décline quatre valeurs en treize principes applicables dans toute démarche agile. Chaque méthode adopte ensuite sa propre terminologie et préconise un certain nombre de pratiques.

Quelles sont les quatre valeurs du Manifeste ?

– Les individus et leurs interactions avant les processus et les outils.
– Des fonctionnalités opérationnelles avant la documentation.
– Collaboration avec le client plutôt que contractualisation des relations.
– Acceptation du changement plutôt que conformité aux plans.

Cela ne signifie évidemment pas qu'aucun processus n'est défini, qu'on ne se dote d'aucun outil ; bien entendu, des plans et une documentation, certes plus réduits, sont produits.

Les valeurs mises en avant à gauche n'excluent pas les valeurs à droite.

Plus récemment, en 2005, a été publiée la « *Déclaration d'interdépendance* »[3] avec, parmi ses signataires, des acteurs majeurs de l'Agile Alliance. Cette communauté a formalisé les valeurs mises en pratique qui les amenaient à rencontrer tant de succès et de bons résultats dans leurs projets.

Elles se résument en six concepts, qui doivent être considérés de façon interdépendante : valeur, incertitude, client, individus, équipe et contexte ; vous ne pouvez créer de la valeur si vous ne collaborez pas avec un client qui, précisément, définit ce qu'est sa valeur ; vous ne pouvez attendre d'une équipe qu'elle soit performante si vous ne reconnaissez pas la contribution des individus qui la composent ; vous ne pouvez maîtriser l'incertitude si vous ne prenez pas en considération la spécificité du contexte.

1. *http://www.agilemanifesto.org/*
2. *http://www.agilealliance.org/*
3. *http://www.pmdoi.com/*

Principes des méthodes agiles

Les treize principes décrits dans le Manifeste sont déclinés des quatre valeurs citées précédemment ; ils sont présentés dans le tableau 2-2.

Tableau 2-2 Les treize principes du Manifeste

Principe	Description
Notre priorité est de satisfaire le client en lui livrant très tôt et régulièrement des versions opérationnelles de l'application, source de valeur.	Grâce au développement itératif, chaque livraison intermédiaire donne lieu à une validation par le client ; son feedback est essentiel pour garantir la conformité de la livraison avec ses attentes, pour prendre en compte ses remarques et (re)prioriser.
Accepter le changement dans les exigences, même tard dans le cycle de vie, pour garantir la compétitivité du client.	Cet état d'esprit caractérise une équipe agile qui démontre ainsi sa capacité à comprendre et apprendre comment satisfaire encore mieux la demande.
Livrer le plus souvent possible des versions opérationnelles de l'application, à une fréquence allant de deux semaines à deux mois.	Une version intermédiaire du produit final, visible et testable, satisfait davantage le client qu'une documentation à valider. Il a, de ce fait, la preuve tangible que le projet avance et peut exprimer son point de vue sur le résultat présenté.
Client et développeurs doivent coopérer quotidiennement tout au long du projet.	Les relations conflictuelles ne font pas partie de l'esprit agile ; on préférera des relations collaboratives et de partenariat basées sur la confiance et le consensus. Le client (ou son représentant) est accessible et disponible, totalement impliqué dans toutes les phases du projet.
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.	Le facteur clé du succès d'un projet est l'équipe. Tout obstacle à son bon fonctionnement devra être levé ; un changement, s'il s'avère nécessaire, sera apporté aux processus, aux outils, à l'environnement, à la composition de l'équipe
La méthode la plus efficace de communiquer des informations à une équipe et au sein de celle-ci reste la conversation en face à face.	Par défaut, on privilégie l'échange oral à l'écrit, pour lever toute ambiguïté et favoriser la rapidité de la compréhension. Tout ne peut être formalisé par écrit, notamment la « connaissance tacite », c'est-à-dire l'information « informelle », la culture du projet, détenues par chacun au sein d'une équipe.
Le fonctionnement de l'application est le premier indicateur d'avancement du projet.	Il n'existe pas d'autre indicateur plus pertinent que le pourcentage ou le nombre d'exigences satisfaites ; on ne mesure pas un projet à la quantité de documents produits ou au nombre de lignes de code, non significatifs pour le client.
Les méthodes agiles recommandent que le projet avance à un rythme soutenable.	La qualité du travail fourni dépend du rythme de travail qui doit être adapté en fonction des spécificités du projet. Le rythme doit être soutenu et soutenable sur la durée du projet.
Sponsors, développeurs et utilisateurs devraient pouvoir maintenir un rythme constant indéfiniment.	Ce rythme de travail est à déterminer par l'ensemble des membres de l'équipe et par le client, en fonction de la productivité de l'équipe et des priorités du client. Mais travailler en heures supplémentaires, surtout pour corriger des bogues ou des régressions, n'apporte aucune valeur ajoutée.

Tableau 2-2 Les treize principes du Manifeste *(suite)*

Principe	Description
Porter une attention continue à l'excellence technique et à la conception améliore l'agilité.	Maintenir un code propre, évolutif et performant est un objectif permanent de l'équipe ; il ne s'agit pas de produire rapidement du code non exploitable par les autres, ni du « jetable ». De plus, cela évite surtout d'enliser les développements ultérieurs, avec des modifications cassant un développement fragile, nécessitant des interventions à des endroits variés du code.
La simplicité – art de maximiser la quantité de travail non fait – est essentielle.	La simplicité garantit l'évolutivité du système. La complexité, au contraire, coûte davantage et rend plus difficiles les évolutions inhérentes au développement incrémental. La conception ne doit comporter que des éléments utiles.
Les meilleures architectures, spécifications et conceptions sont le fruit d'équipes qui s'auto-organisent	Le chef de projet agile n'est plus celui qui attribue des tâches. L'équipe, elle-même, se responsabilise et définit ses travaux à réaliser, le partage des tâches s'effectuant sur la base du volontariat.
À intervalles réguliers, l'ensemble de l'équipe s'interroge sur la manière de devenir encore plus efficace, puis ajuste son comportement en conséquence.	L'environnement d'un projet n'est pas constant ; l'équipe agile, qui en a conscience, s'interroge en permanence sur la façon d'améliorer son fonctionnement afin de s'adapter aux nouvelles conditions. C'est aussi l'acceptation du changement !

Principales méthodes agiles

Les principales méthodes agiles se nourrissent toutes des valeurs et principes du Manifeste ; cependant, si elles ont un tronc commun de pratiques, elles se différencient par leur degré de formalisme – le poids de la méthodologie dans la documentation produite, les étapes formelles, les revues, le rythme du projet ou le nombre et la longueur des itérations.

Quelles sont leurs spécificités ?

Les principales méthodes agiles sont présentées ci-après par ordre alphabétique.

ASD (Adaptative Software Development)

En 2000, Jim Highsmith (signataire du Manifeste) publie un ouvrage sur la méthode ASD, *Adaptative Software Development, a collaborative approach to managing complex systems*.

Le cycle de vie d'un projet ASD se déroule autour d'une série de cycles en trois volets :

- **Spéculation**
  - initier le projet (mission, contraintes, collaborateurs, expression des exigences, identification des risques…) ;
  - déterminer la durée du projet, le nombre d'itérations et les dates associées (4 à 8 semaines par itération) ;

  - affecter un objectif (*mission*) à chaque itération ;
  - dresser une liste de tâches à réaliser.
- **Collaboration**
  - livraison des composants ;
  - communication forte et assez informelle.
- **Apprentissage**
  - contrôle qualité ;
  - suivi et bilan d'avancement ;
  - communication forte et assez informelle.

Ses caractéristiques principales sont :

- focaliser sur l'objectif (*mission focused*) ;
- se baser sur des composants (*component-based*) ;
- itérer ;
- découper le temps et fixer des deadlines (*timeboxing*) ;
- piloter le projet par les risques (*risk-driven development*) ;
- accepter le changement.

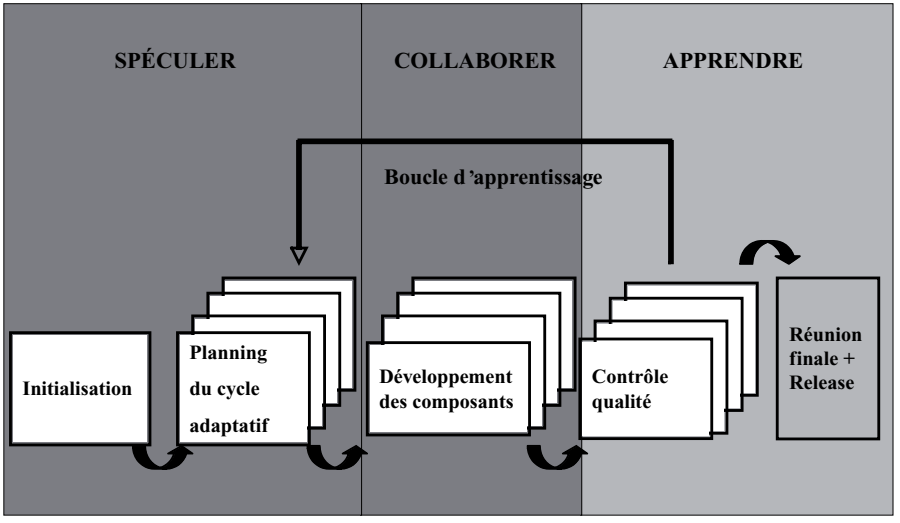


Figure 2-3
Le cycle de vie ASD

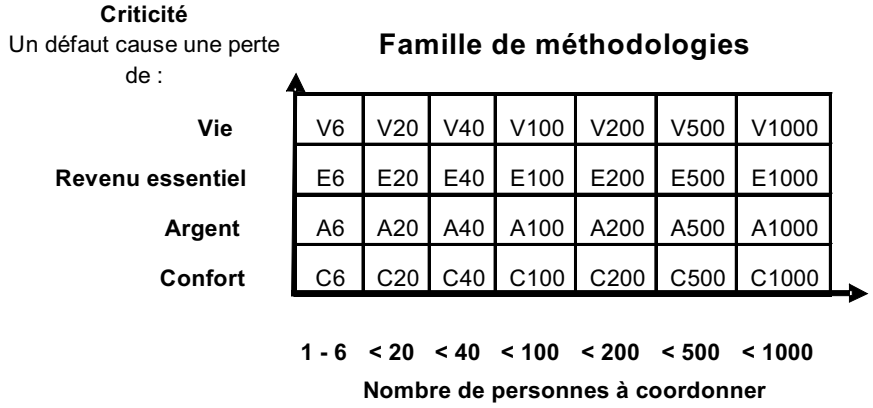
Ce cycle de vie permet un apprentissage et une adaptation permanents aux évolutions du projet.

Crystal

Crystal, ou plus exactement la famille de méthodologies Crystal (figure 2-4), a été mise au point par Alistair Cockburn (signataire du Manifeste).

Le principe est de sélectionner la méthode applicable en fonction de la criticité du projet et du nombre de personnes à coordonner. En effet, on ne gère pas de la même façon un projet de dix personnes et un projet de cent personnes, ou un projet de développement d'un intranet documentaire ou d'un système de sécurité.

Figure 2-4
La famille de méthodologies Crystal



Crystal Clear désigne la famille de méthodologies dédiées aux équipes inférieures à 8 personnes.

Quelles sont les propriétés de Crystal ?

– Des livraisons fréquentes.
– Des aménagements permanents.
– Une communication osmotique.
– Confiance, liberté d'expression et sécurité personnelle.
– Focus sur l'objectif et disponibilité.
– Un contact permanent avec les utilisateurs.
– Un environnement de travail approprié pour l'automatisation des tests, la gestion de configuration et les intégrations fréquentes.
– Une collaboration étroite entre toutes les parties prenantes, y compris en dehors de l'équipe.
– Une réflexion constante sur ces propriétés.

DSDM (Dynamic Software Development Method)

DSDM est le fruit du travail d'un consortium de sociétés désirant utiliser RAD (voir ci-après), de façon structurée et indépendante, en Grande-Bretagne.

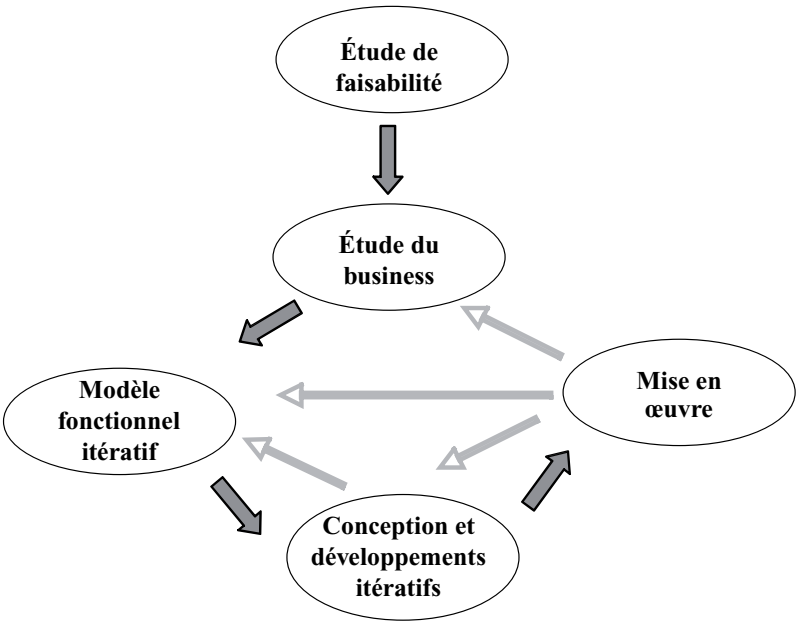
DSDM adhère aux valeurs et principes du Manifeste, puisque l'un de ses représentants, Arie Van Bennekum, est l'un de ses signataires. DSDM a cependant mis neuf autres principes complémentaires en avant.

Quels sont les neuf principes de DSDM ?

– Implication active des utilisateurs.
– Autonomie et pouvoir de décision des équipes.
– Livraisons fréquentes.
– Adéquation aux besoins des clients comme seul critère d'acceptation du produit.
– Développement itératif et incrémental.
– Modifications réversibles.
– Définition globale macroscopique des besoins.
– Intégration des tests dans tout le cycle de vie.
– Collaboration et coopération entre toutes les parties prenantes.

Le cycle de vie proposé par DSDM (figure 2-5) est un cadre général qui doit être adapté à chaque projet. Il présente cinq étapes :

Figure 2-5
Le cycle de vie DSDM



- **L'étude de faisabilité** : comme son nom l'indique, cette phase permet de bien définir le problème à résoudre et d'en étudier la faisabilité, sur les plans technique, méthodologique et budgétaire.

- **L'étude du « business »** : cette phase vise à déterminer et à analyser les processus métier qui doivent être automatisés et les besoins en informations ; grâce à des *ateliers facilités*, les utilisateurs hiérarchisent leurs besoins. Une définition de l'architecture globale ainsi qu'un plan global de prototypage sont produits pour préparer les phases suivantes.
- **Le modèle fonctionnel itératif** : ce modèle produit décrit les besoins en détail et permet de définir quand et comment ils seront satisfaits. Le résultat est une série de modules logiciels constituant un prototype fonctionnel.
- **La conception et les développements itératifs** : il s'agit, dans cette phase, de fournir un système intégrant toutes les fonctionnalités, conforme aux besoins définis.
- **La mise en œuvre** : cette dernière phase est la phase de livraison et de prise en main de l'application par les utilisateurs qui doivent la tester (après avoir été formés, le cas échéant) et contrôler la qualité de la documentation avant la mise en production. Enfin, un bilan est dressé afin de capitaliser sur les bonnes pratiques mises en œuvre.

DSDM se caractérise également par une spécialisation des acteurs du projet, dont le rôle est précisément décrit.

Quels sont les différents rôles décrits par DSDM ?

Côté utilisateur

– Le sponsor exécutif : sa position hiérarchique, son engagement et sa disponibilité dans le projet garantissent que les ressources humaines et matérielles seront mises à disposition pour le bon déroulement du projet.

– Le visionnaire : il est « l'expert métier » ; il porte la « vision » et s'assure de la cohérence des besoins exprimés ; c'est lui qui participe activement aux ateliers facilités et à la conception du système.

– L'ambassadeur : il assure l'interface entre l'organisation et l'équipe de développement ; il est le représentant des autres utilisateurs et doit fournir les informations nécessaires au développement ; il organise les tests utilisateurs et la formation de ces derniers.

– Le chef de projet : ce rôle peut être confié à un représentant des utilisateurs ou à un membre de l'équipe informatique. Il est responsable de l'organisation et de l'avancement du projet et doit assurer un reporting régulier.

Côté informatique

– Le coordinateur technique : il est le responsable technique du projet.

– Le chef d'équipe : il est en charge de la gestion opérationnelle de l'équipe informatique.

– Indépendant de l'équipe de projet, le facilitateur est chargé d'organiser les ateliers.

– Le rapporteur : il est responsable de la rédaction des comptes rendus de toutes les réunions (ateliers, sessions de prototypage…) et de l'enregistrement des décisions prises.

Il faut souligner que cette méthode est particulièrement bien documentée, depuis 1995, en plusieurs langues, avec des mises à jour régulières.