Scrum, ou les principes du rugby appliqués au développement applicatif

Par Christophe Legrenzi, chercheur et consultant international,
expert associé de *Best Practices Systèmes d'information*

Les besoins des clients sont rarement figés. Lorsqu'il faut développer une application dans un contexte où les changements sont fréquents, les méthodes de gestion de projet classiques s'avèrent souvent mal adaptées. En effet, celles-ci se basent sur un cahier des charges détaillé mais peu évolutif. Quant à l'organisation, elle est définie une fois pour toutes en début de projet : analyse, conception, développement et tests s'enchaînent sur un mode séquentiel, tandis que les membres de l'équipe restent chacun dans leur rôle.

De nombreux chercheurs et experts du développement ont pointé la rigidité d'approches qui aboutissaient parfois à livrer des applications très éloignées des besoins réels des clients, ceux-ci ayant évolué en cours de route. L'effet « tunnel » était souvent incriminé. Au cours des années 1990, la recherche d'autres approches a conduit à l'apparition des méthodes agiles, caractérisées par des cycles courts et des processus incrémentaux.

Scrum fait partie de ces méthodes agiles, aux côtés de XP pour eXtreme Programming, RAD, Lean IT ou encore Crystal Clear.

1. PRÉSENTATION DE LA BEST PRACTICE

L'histoire de Scrum débute en 1986, dans un article écrit par Hirotaka Takeuchi et Ikujiro Nonaka, deux chercheurs en management japonais. Cet article, intitulé The New New Product Development Game (*www.cs.utexas.edu/users/downing/papers/SCRUM.pdf*), décrit deux méthodes de développement de produits. Les auteurs s'appuient sur des métaphores sportives pour illustrer leur propos.

La première méthode, qui repose sur une approche séquentielle, est comparée au relais quatre fois cent mètres. La seconde consiste en une approche holistique, dans laquelle une seule et même équipe, polyvalente, travaille de manière itérative. Cette deuxième méthode est comparée au rugby, d'où le terme « Scrum », la mêlée en anglais, tire son origine.

En 1991, Peter DeGrace et Leslie Hulet Stahl, dans l'ouvrage *Wicked Problems, Righteous Solutions*, reprennent le terme Scrum pour décrire cette approche. Au début des années 1990, des méthodologies basées sur les principes décrits par les deux chercheurs japonais sont mises en pratique dans différentes entreprises. Ken Schwaber d'un côté, Jeff Sutherland, John Scumniotales et Jeff McKenna de l'autre, travaillent ainsi sur ce type d'approche. En 1995, Ken Schwaber et Jeff Sutherland présentent conjointement un article décrivant la méthode Scrum, lors de l'atelier *Business Object Design and Implementation* de la conférence OOPSLA, au Texas. Au cours des années suivantes, tous deux travailleront ensemble à enrichir la méthode. Enfin, en 2001, Ken Schwaber co-écrit avec Mike Beedle un livre présentant les fruits de ces travaux : *Agile Software Development with Scrum*.

La méthode Scrum repose sur une répartition des rôles bien définie. Les rôles principaux sont les suivants :

- Le *Scrum Master* n'est pas un manager au sens classique du terme. En effet, dans Scrum l'équipe est autogérée. Le rôle de *Scrum Master* diffère donc de celui d'un chef de projet traditionnel : il s'agit de protéger l'équipe de développement en lui assurant un rôle de rempart. Le *Scrum Master* intercepte tout événement perturbateur et gère les aspects administratifs afin de ne pas entraver le travail des développeurs. Il vérifie aussi que le processus de Scrum est bien respecté.

- Le *Product Owner*, directeur du produit, représente le client et doit idéalement être dans les mêmes locaux que l'équipe de développement.

- L'équipe de développement (*Team*) est un groupe plurifonctionnel de quatre à dix personnes, qui effectue aussi bien l'analyse que le design, l'implémentation et les tests et fonctionne sur le principe de l'auto-organisation.

- Et, enfin, les *Stakeholders* désignent les autres personnes impliquées dans le projet mais sans être membres de l'équipe.

Une fois les rôles établis, l'équipe entame alors le développement sous la forme d'une série d'itérations. Celles-ci, les sprints, durent entre deux et quatre semaines. L'objectif est d'aboutir, à l'issue de chaque sprint, à une application en état de marche, et donc potentiellement livrable, même si elle ne répond encore que partiellement au besoin. Les différents sprints vont permettre d'étendre l'application au fur et à mesure, en fonction des priorités définies par le client. Un ensemble de

sprints permettant d'aboutir à une version finalisée constitue une *release*.

Au début de chaque sprint, le *Product Owner* sélectionne les fonctionnalités qui vont être développées et les présesnte à l'équipe lors du *Sprint Planning Meeting*, une réunion de lancement qui ouvre chaque itération. Cette présentation doit tenir en quatre heures. Pour choisir les fonctionnalités, le représentant du client se base sur le *backlog* du produit, un ensemble d'exigences en fonction de la valeur métier et du temps estimé nécessaire pour le développement : entre deux fonctionnalités au coefficient métier identique, celles nécessitant moins de temps seront prioritaires, le retour sur investissement étant plus rapide.

L'équipe évalue ensuite, parmi les fonctionnalités retenues, celles qui peuvent être développées lors du prochain sprint et fixe alors la liste des tâches à effectuer. Cette seconde partie ne doit pas non plus dépasser quatre heures. Une fois définie, la liste des fonctionnalités à développer est gelée pour toute la durée de l'itération. Les tâches de cette liste ne sont pas assignées, ce sont les membres de l'équipe qui choisissent eux-mêmes celles qu'ils vont effectuer, toujours dans un principe d'auto-organisation. Nul besoin d'outils compliqués, un simple tableau de liège couvert de Post-it suffit donc pour gérer la répartition des tâches.

L'équipe est au cœur de l'approche Scrum. Les membres n'ont pas de rôles préétablis et toutes les décisions sont prises de manière collective. Lors du sprint, une réunion quotidienne, le *Daily Scrum Meeting*, permet à l'équipe de faire un point sur l'avancement du projet. Cette réunion ne doit pas excéder quinze minutes. Lors de cette réunion, chaque membre de l'équipe doit répondre à trois questions :

- Qu'avez-vous fait hier ?
- Que comptez-vous faire aujourd'hui ?
- Avez-vous rencontré des difficultés ?

Des membres externes à l'équipe peuvent assister à ces réunions, mais seule l'équipe peut s'y exprimer.

A la fin du sprint, si certaines fonctionnalités n'ont pu être développées, elles retournent dans le *backlog*. Deux réunions viennent clore chaque sprint : la première, le *Sprint Review Meeting*, dure quatre heures et permet de faire un bilan du travail effectué. La seconde, le *Sprint Retrospective*, permet à l'équipe de faire le point sur ce qui a bien et mal fonctionné dans le processus.

Enfin, Scrum s'appuie sur des graphiques pour suivre l'avancement des projets. Le *Sprint Burndown Chart* permet d'afficher les tâches restant à faire dans un sprint, tandis que le *Release Burndown Char*t fait de même au niveau d'une *release*. Ces graphiques sont importants pour ajuster les *backlogs* en fonction des échéances souhaitées par les clients. Ils permettent de visualiser à tout moment les délais nécessaires pour le développement, et si besoin est, à revoir les exigences pour tenir compte des contraintes de planning.

2. REGARD CRITIQUE

Scrum est l'une des méthodes agiles les plus populaires. Néanmoins, cette popularité ne doit pas faire oublier que la méthode n'est pas adaptée à toutes les situations, et qu'elle a ses points forts comme ses points faibles.

Comme pour l'*Extreme Programming*, la petite taille des équipes en Scrum s'avère ainsi peu adaptée aux gros projets et au développement d'applications d'entreprise. Rappelons que l'XP, dont les créateurs sont Kent Beck, Ward Cunningham et Ron Jeffries, a été élaborée chez Chrysler pour une application annexe au système de paie. La méthode a été affinée de 1996 à 1999 et publiée par Kent Beck dans le livre *Extreme Programming Explained*.

Comme avec d'autres méthodes agiles, le risque avec Scrum est de laisser la porte trop grande ouverte aux changements, en se retrouvant avec une application qui n'est jamais totalement finalisée. Il est donc important de bien respecter les limites prévues par Scrum pour débattre et figer les différents *backlogs*, mais aussi d'avoir en tête un planning global, avec des échéances précises pour chaque version. Il s'agit bien d'une méthode dont le but est le développement d'une application, et donc à terme, il s'agit d'avoir un produit fini à livrer, non un simple prototype.

La précision des estimations peut aussi poser problème, celles-ci étant bien souvent en dessous du temps réellement nécessaire au final pour effectuer une tâche. Les réunions de fin de sprint sont essentielles pour réajuster les estimations et s'approcher à chaque fois un peu plus d'une vision réaliste des choses.

Par ailleurs, Scrum est axée sur la gestion de projet, contrairement à d'autres méthodes agiles comme XP. Si l'approche insiste sur la planification, les réunions ou la répartition des rôles, en revanche, Scrum ne couvre pas les aspects purement liés à l'ingénierie du développement, comme les bonnes pratiques de programmation ou la planification des tests.

À ce titre, Scrum et XP sont donc assez complémentaires,

chacune couvrant les aspects peu ou pas évoqués par l'autre. Bien entendu, les équipes Scrum peuvent également adopter en interne un ensemble de règles et de bonnes pratiques pour le développement, sans pour autant se baser strictement sur XP. Le développement piloté par les tests est notamment devenue une pratique courante dans de telles équipes.

Scrum n'impose, en outre, aucune règle en matière de documentation, ce qui peut peut poser des problèmes pour les projets stratégiques. L'absence de documentation introduit en effet une dépendance envers l'équipe et son savoir-faire, l'avenir de l'application pouvant alors être compromis si certains membres quittent l'entreprise.

Enfin, dans Scrum, le projet est étroitement lié à la cohésion de l'équipe. Pour que l'auto-organisation fonctionne, il faut que les membres comprennent et acceptent la démarche et qu'ils soient prêts à s'impliquer dans la gestion du projet. Il faut également une certaine polyvalence au niveau des compétences et un bon degré d'adaptabilité. De fait, les approches agiles ne conviennent pas à tous les profils, il est important d'en être conscient.

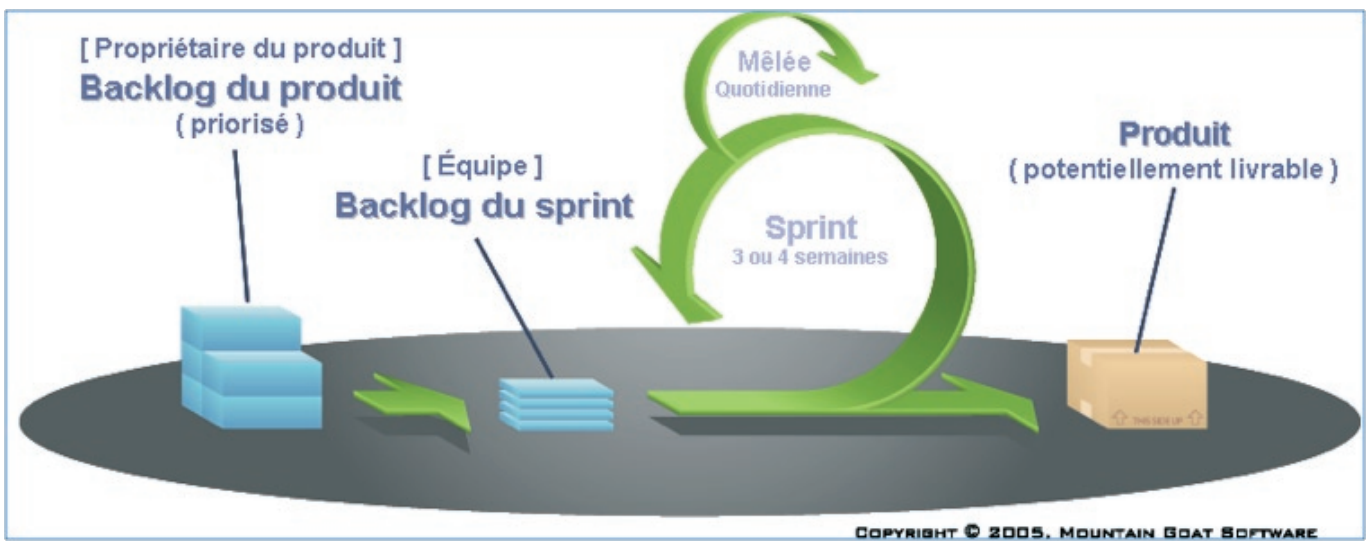
3. QUE FAIRE ? QUELQUES PISTES DE SOLUTIONS

Des méthodes agiles comme Scrum ont apporté un bol d'air frais dans le monde du développement, ce qui explique en partie leur succès. Elles mettent en avant des qualités comme l'autonomie et le sens des responsabilités, tout en favorisant la proximité avec le client et la réflexion collective. Si l'agilité permet de répondre à certains besoins, elle n'est pas non plus la panacée. Il ne s'agit donc pas d'abandonner les approches traditionnelles en imposant une méthode agile de manière dogmatique, mais de savoir reconnaître quand une méthode comme Scrum est mieux appropriée.

Pour Scrum, comme pour toute méthode agile, il faut donc garder en mémoire les sept limites potentielles suivantes que nous résumons ci-dessous :

- risque pour des applications au cœur du métier qui ont une durée de vie d'au moins huit à douze ans, et dont la performance opérationnelle d'entreprise dépend fortement ;
- tendance à une fuite en avant en étant constamment en train de développer et en ne gelant pas des versions ;
- risque opérationnel quand la séparation des fonctions de développement et de la mise en production n'est pas clairement assurée ;
- tendance à développer des fonctionnalités à faible valeur ajoutée pour l'entreprise en laissant trop la main aux utilisateurs sans filtrage organisé ;
- risque de fragilisation du code et construction « d'usines à gaz » ;
- documentation insuffisante induisant une dépendance forte vis-à-vis des individus ayant développé l'application ;

Le processus global de Scrum



- les technologies utilisées ne sont pas toujours en phase avec les enjeux de performance requis par les métiers, par exemple pour les temps de réponse, la simultanéité des accès, etc.

Nous recommandons une approche « hybride » : l'utilisation des méthodes classiques mais robustes pour les projets à enjeux au cœur du métier, et l'utilisation de méthodes agiles pour tous les projets qui induisent durée de vie plus courte et moins de formalisation.

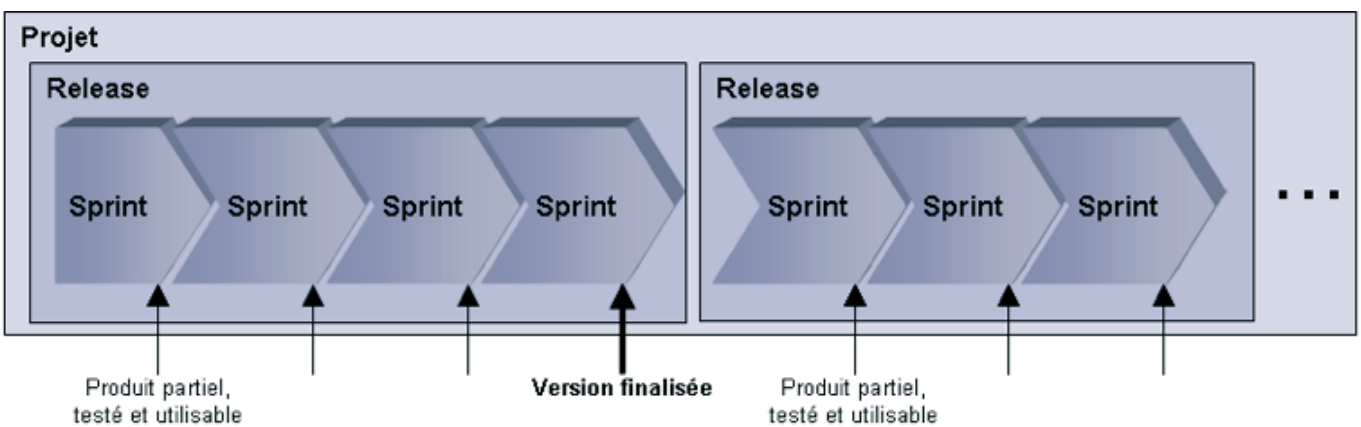
En outre, dans les entreprises habituées aux modèles de développement classiques, la transition vers Scrum n'est pas forcément évidente. La cohabitation non plus. Aussi, des équipes bien distinctes peuvent être constituées. Mieux vaut donc commencer par des projets peu sensibles, pour permettre aux équipes de se faire la main et d'appréhender ce nouveau mode de fonctionnement. En effet, la méthode doit être bien comprise et appliquée de manière rigoureuse, faute de quoi elle risque d'agir simplement comme un révélateur des dysfonctionnements au sein des équipes et de faire rapidement dégénérer le projet. Comme toute méthode agile, Scrum nécessite un respect de la méthode et un professionnalisme encore plus poussé. ◆

L'histoire du cochon et de la poule, ou la philosophie de Scrum en une fable

Un cochon et une poule se promènent dans la rue. La poule regarde le cochon et lui dit : *« Hé, que dirais-tu d'ouvrir un restaurant ensemble ? »* Le cochon regarde la poule et lui répond : *« Bonne idée, et comment l'appellerions-nous ? »* Le poulet réfléchit et dit : *« Et si on l'appelait "Les Œufs et le Jambon ?" » « Je ne pense pas, dit le cochon, je serais totalement engagé alors que tu ne serais qu'impliqué. »*

Cette petite histoire illustre la distinction entre les « cochons », ceux dont la responsabilité est directement engagée par le projet de développement, et les « poules », toutes les autres personnes concernées par le projet mais qui ne sont pas directement responsables en cas d'échec. Les « cochons » sont l'équipe de développement, tandis que les « poulets » sont les clients, utilisateurs et intervenants externes. ◆

Exemple de planification en Scrum



Articles à paraître

- OGC P3O (gestion de portefeuille de projets)
- Prince 2 (gestion de projet)
- OPM3 (gestion de projets)
- Togaf (architecture d'entreprise)
- Iso 20000 (gestion des services)
- Cisq (qualité logicielle)
- TCO (coûts des actifs informatiques)
- ABC (coût des activités informatiques)
- Lean six Sigma (maîtrise des processus)
- Informatique *vs* système d'information

Articles déjà publiés

- La gestion de projet selon PMBOK , numéro 65 de *Best Practices Systèmes d'Information*, 11 avril 2011
- La gestion de projet selon PMBOK , numéro 65 de *Best Practices Systèmes d'Information*, 11 avril 2011
- Analyse de la valeur appliquée au système d'information, numéro 62 de *Best Practices Systèmes d'Information,* 28 février 2011
- Les points de fonction, numéro 64 de *Best Practices Systèmes d'Information*, 28 mars 2011
- La gestion de projet selon PMBOK, numéro 65 de *Best Practices Systèmes d'Information*, 11 avril 2011