# Scrum

# Introduction

**Scrum** est une méthode agile pour la gestion de projets. Elle a été conçue pour améliorer grandement la productivité dans les équipes auparavant paralysées par des méthodologies plus lourdes. Les racines de Scrum se retrouvent dans la publication de Takeuchi et Nonaka dans "The New New Product Development Game"[1] pour l'aspect métaphore du Rugby ainsi que pour les notions d'ingénierie concourante et dans la rupture méthodologique qualifiée d'itérative, incrémentale et adaptative (par rapport au modèle cascade) dont la première version opérationnelle a été publiée par James Martin en 1991 sous le nom de RAD, en ce qui concerne sa structure fondamentale.

La méthode Scrum a été conçue pour la gestion de projets de développement de logiciels. Elle peut aussi être utilisée par des équipes de maintenance. Dans le cas de très grands projets, les équipes se multiplient et on parle alors de *Scrum de Scrums*. La méthode Scrum peut théoriquement s'appliquer à n'importe quel contexte où un groupe de personnes qui ont besoin de travailler ensemble pour atteindre un but commun - comme gérer une petite école, des projets de recherche scientifique ou planifier un mariage.

Par contre, la méthode Scrum ne couvre aucune technique d'ingénierie du logiciel. Aussi, son utilisation dans le contexte du développement d'une application informatique, nécessite de lui adjoindre une méthode complémentaire (comme l'Extreme Programming, par exemple).

**Scrum** est le fruit de l'effort entre Ken Schwaber et Jeff Sutherland, qui en 1995 pour la première communication, lors de ACM conférence (Object-Oriented Programming, Systems, Languages & Applications) ont mis au point les grands principes de Scrum. Mais ce n'est qu'en 1996, que Scrum est né officiellement avec la publication du premier article[2] définissant cette méthodologie.

Le terme *Scrum* est emprunté au rugby et signifie *mêlée*. Ce processus s'articule en effet autour d'une équipe soudée, qui cherche à atteindre un but, comme c'est le cas en rugby pour avancer avec le ballon pendant une mêlée.

Le principe de base de Scrum est de focaliser l'équipe de façon itérative sur un ensemble de fonctionnalités à réaliser, dans des itérations de durée fixe de une à quatre semaines, appelées **Sprints**. Chaque Sprint possède un **but** à atteindre, défini par le *Directeur de produit*, à partir duquel sont choisies les fonctionnalités à implémenter dans ce sprint. Un sprint aboutit toujours sur la livraison d'un produit partiel, mais fonctionnel. Pendant ce temps, le *ScrumMaster* a la charge de réduire au maximum les perturbations extérieures et de résoudre les problèmes non techniques de l'équipe.

Un principe fort en Scrum est la participation active du client pour définir les priorités dans les fonctionnalités du logiciel, et pour choisir celles qui seront réalisées dans chaque sprint. Il peut à tout moment compléter ou modifier la liste des fonctionnalités à réaliser, mais jamais celles qui sont en cours de réalisation pendant un sprint.

## Idées clé

- Le client au cœur du projet
- Esprit d'équipe
- La communication est la clé
- Simplicité, Efficacité, et Qualité
- Flexibilité aux changements
- Avancement basé sur le concret

## Rôles


Scrum définit trois rôles principaux : le **Directeur de produit**, le **ScrumMaster**, et l'**Équipe**. Des **Intervenants** peuvent s'intégrer également au projet de façon plus ponctuelle.

### *Directeur de produit*

Le **Directeur de produit** (*Product Owner*) est le représentant des clients et utilisateurs. C'est lui qui définit l'**ordre** dans lequel les fonctionnalités seront développées, et qui prend les décisions importantes concernant l'orientation du projet. Le terme *Directeur* n'est d'ailleurs pas à prendre au sens hiérarchique du terme, mais dans le sens de l'*orientation*.

Dans l'idéal le Directeur de produit travaille dans la même pièce que l'équipe. Il est important qu'il reste très disponible pour répondre aux questions de l'équipe et pour lui donner son avis sur divers aspects du logiciel (interface par exemple).

### *Équipe*

L'**Équipe** ne comporte pas de rôles prédéfinis, elle est **auto-gérée**. Il n'y a pas non plus de notion de hiérarchie interne : toutes les décisions sont prises ensemble, et personne ne donne d'ordre à l'équipe sur sa façon de procéder. Contrairement à ce que l'on pourrait croire, les équipes auto-gérées sont celles qui sont les plus efficaces et qui produisent le meilleur niveau de qualité de façon spontanée.

L'équipe s'adresse directement au Directeur de produit. Il est conseillé qu'elle lui montre le plus souvent possible le logiciel développé pour qu'il puisse ajuster les détails d'ergonomie et d'interface par exemple.

### *ScrumMaster*

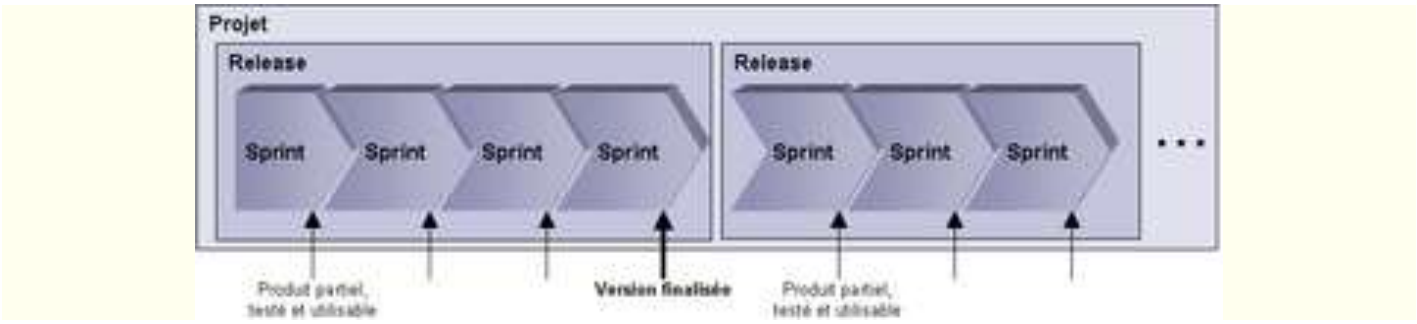
Le **ScrumMaster** joue un rôle capital : c'est lui qui est chargé de protéger l'équipe de tous les éléments perturbateurs extérieurs à l'équipe et de résoudre ses problèmes non techniques (administratifs par exemple). Il doit aussi veiller à ce que les valeurs de Scrum soient appliquées, mais il n'est pas un chef de projet ni un intermédiaire de communication avec les clients.

On parle parfois d'**Équipe étendue**, qui intègre en plus le ScrumMaster et le directeur de produit. Ce concept renforce l'idée que client et fournisseur travaillent d'un commun effort vers le succès du projet.

### *Intervenants*

Les **Intervenants** (*Stakeholders*) sont les personnes qui souhaitent avoir une vue sur le projet sans réellement s'investir dedans. Il peut s'agir par exemple d'experts techniques ou d'agents de direction qui souhaitent avoir une vue très éloignée de l'avancement du projet.

## Planification




Exemple de planification en Scrum

Scrum utilise une planification à trois niveaux : release/projet, sprint, et quotidien.

### *Sprints*

Scrum est un processus *itératif* : les itérations sont appelées des **Sprints** et durent en théorie 30 jours calendaires. En pratique, les itérations durent généralement entre 2 et 4 semaines. Chaque sprint possède un **but**, et on lui associe une liste d'*items de backlog de produit* (fonctionnalités) à réaliser. Ces items sont décomposés par l'équipe en tâches élémentaires de quelques heures, les *items de backlog de sprint*.

Pendant un sprint, les *items de backlog de sprint* à réaliser ne peuvent pas être changés. Les changements éventuels sont pris en compte dans le backlog de produit, et seront éventuellement réalisés dans les sprints suivants.

Il y a une exception à cela : il se peut que l'équipe se rende compte au cours du sprint qu'elle n'aura pas le temps de terminer un item du backlog de sprint, ou au contraire qu'elle aura fini en avance. Dans ce cas, et seulement d'un commun accord entre l'équipe et le directeur du produit, on peut enlever ou ajouter un item à ce qui a été prévu.

### *Releases*

Pour améliorer la lisibilité du projet, on regroupe généralement des itérations en **releases**. Bien que ce concept ne fasse pas explicitement partie de Scrum, il est utilisé pour mieux identifier les versions. En effet, comme chaque sprint doit aboutir à la livraison d'un produit partiel mais fonctionnel, une release permet de marquer la livraison d'une version aboutie, susceptible d'être mise en exploitation.

Il est intéressant de planifier à l'échelle d'une release, en répartissant les items du backlog de produit sur les sprints, en respectant leur priorité. Bien entendu, ce qui est planifié au-delà du sprint courant peut changer à tout moment, rien n'est figé à l'avance.

### *Quotidien*

Au quotidien, une réunion, le **Daily Scrum Meeting**, permet à l'équipe et au ScrumMaster de faire un point d'avancement sur les tâches et sur les difficultés rencontrées.

## Gestion des besoins

### *Backlog de produit*

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** (NDT : Le terme *backlog* peut être traduit par *cahier*, *liste* ou *carnet de commandes*, qui ne collent pas bien avec l'esprit du terme anglais qui évoque aussi une *réserve*, un *retard accumulé* ; aussi ce terme a été gardé tel quel).

A chaque item de backlog sont associées deux attributs : une estimation en **points arbitraires** (voir Estimation), et une valeur *client*, qui est définie par le Directeur de produit (retour sur investissement par exemple). Ce dernier définit dans quel ordre devront être

réalisés ces items. Il peut changer cet ordre en cours de projet, et même ajouter, modifier, ou supprimer des items dans le backlog.

La somme des points des items du backlog de produit constitue le *reste à faire* total du projet. Cela permet de produire un **release burndown chart** (un graphique), qui montre les points restant à réaliser au fur et à mesure des sprints.

**Remarque** : il arrive souvent qu'on utilise dans Scrum les *User Stories* de la méthode *Extreme Programming*, qui proposent des pratiques et des techniques intéressantes (le *Planning poker* pour les estimer par exemple).

### *Backlog de sprint*

Lorsqu'on démarre un sprint, on choisit quels items du backlog de produit seront réalisés dans ce sprint. L'équipe décompose ensuite chaque item en liste de tâches élémentaires (techniques ou non), chaque tâche étant estimée en heures et ne devant pas durer plus de 2 jours. On constitue ainsi le **backlog de sprint**.

Pendant le déroulement du sprint, chaque équipier s'affecte des tâches du backlog de sprint et les réalise. Il met à jour régulièrement dans le backlog du sprint le reste à faire de chaque tâche. Les tâches ne sont pas réparties initialement entre tous les équipiers, elles sont prises au fur et à mesure que les précédentes sont terminées.

La somme des heures des items du backlog de sprint constitue le *reste à faire* total du sprint. Cela permet de produire un **sprint burndown chart** qui montre les heures restantes à réaliser au fur et à mesure du sprint.

## Estimations

Scrum ne définit pas spécialement d'unités pour les items des backlogs. Néanmoins, certaines techniques se sont imposées de fait.

### *Items de backlog de produit*

Les items de backlog de produit sont souvent des *User Stories* empruntées à Extreme Programming. Ces User Stories sont estimées en points relatifs, sans unité. L'équipe prend un item représentatif, et lui affecte un nombre de points arbitraire. Cela devient un référentiel pour estimer les autres items. Par exemple, un item qui vaut 2 points représente deux fois plus de travail qu'un item qui en vaut 1. Pour les valeurs, on utilise souvent les premières valeurs de la suite de Fibonacci (1,2,3,5,8,13), qui évitent les difficultés entre valeurs proches (8 et 9 par exemple).

L'intérêt de cette démarche est d'avoir une idée du travail requis pour réaliser chaque fonctionnalité sans pour autant lui donner une valeur en jours que le directeur de produit serait tenté de considérer comme définitivement acquise. En revanche, on utilise la *vélocité* pour planifier le projet à l'échelle macroscopique de façon fiable et précise.

### *Calcul de vélocité*

Une fois que tous les items de backlog de produit ont été estimés, on attribue un certain nombre d'items à réaliser aux sprints successifs. Ainsi, une fois un sprint terminé, on sait combien de points ont été réalisés, et on définit alors la **vélocité** de l'équipe, c'est-à-dire le nombre de points qu'elle peut réaliser en 1 sprint.

En partant de cette vélocité et du total de points à réaliser, on peut déterminer le nombre de sprints qui seront nécessaires pour terminer le projet (ou la release en cours). L'intérêt, c'est qu'on a une vision de plus en plus fiable (retours d'expérience de sprint en sprint) de la date d'aboutissement du projet, tout en permettant d'aménager les items de backlog du produit en cours de route.

### *Items de backlog de sprint*

Les items de backlog de sprint sont généralement exprimés en heures, et ne doivent pas dépasser 2 journées de travail, auquel cas il convient de les décomposer en plusieurs items. Par abus de langage, on emploie le terme de *tâches*, les concepts étant très proches.

## Déroulement d'un sprint

### *Réunion de planification*

Tout le monde est présent à cette réunion, qui ne doit pas durer plus de 4 heures. La **réunion de planification** (*Sprint Planning*) consiste à définir d'abord un but pour le sprint, puis à choisir les items de backlog du produit qui seront réalisés dans ce sprint. Cette première partie du *sprint planning* représente l'engagement de l'équipe. Compte tenu des conditions de succès énoncées par le directeur de produit et de ses connaissances techniques, l'équipe s'engage à réaliser un ensemble d'items du backlog du produit.

Dans un second temps, l'équipe décompose chaque item du backlog de produit en liste de tâches (items du backlog du sprint), puis estime chaque tâche en heures. Il est important que le directeur de produit soit présent dans cette étape, il est possible qu'il y ait des tâches le concernant (comme la rédaction des règles métier que le logiciel devra respecter et la définition des tests fonctionnels).

### *Au quotidien*

Chaque journée de travail commence par une réunion de 15 minutes maximum appelée **mêlée quotidienne** (*Daily Scrum*). Seuls l'équipe, le Directeur de produit et le ScrumMaster peuvent parler, tous les autres peuvent écouter mais pas intervenir (leur présence n'est pas obligatoire). A tour de rôle, chaque membre répond à 3 questions :

- Qu'est-ce que j'ai fait hier ?
- Qu'est-ce que je compte faire aujourd'hui ?
- Quelles difficultés est-ce que je rencontre ?

Le tour de parole doit être scrupuleusement respecté pour éviter que le Scrum dérive sur des discussions techniques et déborde des 15 minutes. Si le besoin s'en fait sentir, des discussions sont alors menées librement après le Scrum.

Cette réunion a un but de synchronisation pour l'équipe et ne doit pas être vécue comme un *reporting* d'activité. C'est le niveau quotidien du principe *inspect and adapt* de Scrum.

L'équipe se met ensuite au travail. Elle travaille dans une même pièce, dont le ScrumMaster a la responsabilité de maintenir la qualité de son environnement. Les activités se déroulent éventuellement en parallèle : analyse, conception, codage, intégration, tests, etc. Scrum **ne définit volontairement pas** de démarche technique pour le développement du logiciel : l'équipe s'auto-gère et décide en toute autonomie de la façon dont elle va travailler.

**Remarque** : Il est assez fréquent que les équipes utilisent la démarche de *développement guidé par les tests*. Cela consiste à coder en premier lieu les modules de test vérifiant les contraintes métier, puis à coder ensuite le logiciel à proprement parler, en exécutant les tests régulièrement. Cela permet de s'assurer entre autres de la non-régression du logiciel au fil des sprints.

### *Revue de sprint*

À la fin du sprint, tout le monde se réunit pour effectuer la **revue de sprint**, qui dure au maximum 4 heures. L'objectif de la revue de sprint est de valider le logiciel qui a été produit pendant le sprint. L'équipe commence par énoncer les items du backlog de produit qu'elle a réalisés. Elle effectue ensuite une démonstration du logiciel produit. C'est sur la base de cette démonstration que le directeur de produit valide chaque fonctionnalité planifiée pour ce sprint.

Une fois le bilan du sprint réalisé, l'équipe et le directeur de produit proposent des aménagements sur le backlog du produit, et sur la planification provisoire de la release. Il est probable qu'à ce moment des items soient ajoutés, modifiés, ou ré-estimés, en conséquence de ce qui a été découvert

### *Rétrospective du sprint*

La **Rétrospective du sprint** est faite en interne à l'équipe (incluant le ScrumMaster). L'objectif est de comprendre ce qui n'a pas bien marché dans le sprint, les erreurs commises, et de prendre des décisions pour s'améliorer. Il est tout à fait possible d'apporter des aménagements à la méthode Scrum dans le but de s'améliorer. Il faut être très vigilant à ne pas retomber dans des pratiques rigides des méthodologies plus classiques.

## Compléments

### *Lancement du projet*

Scrum présuppose que le backlog de produit est déjà défini au début du projet. Une approche possible pour constituer ce backlog est de réaliser une **phase de lancement**.

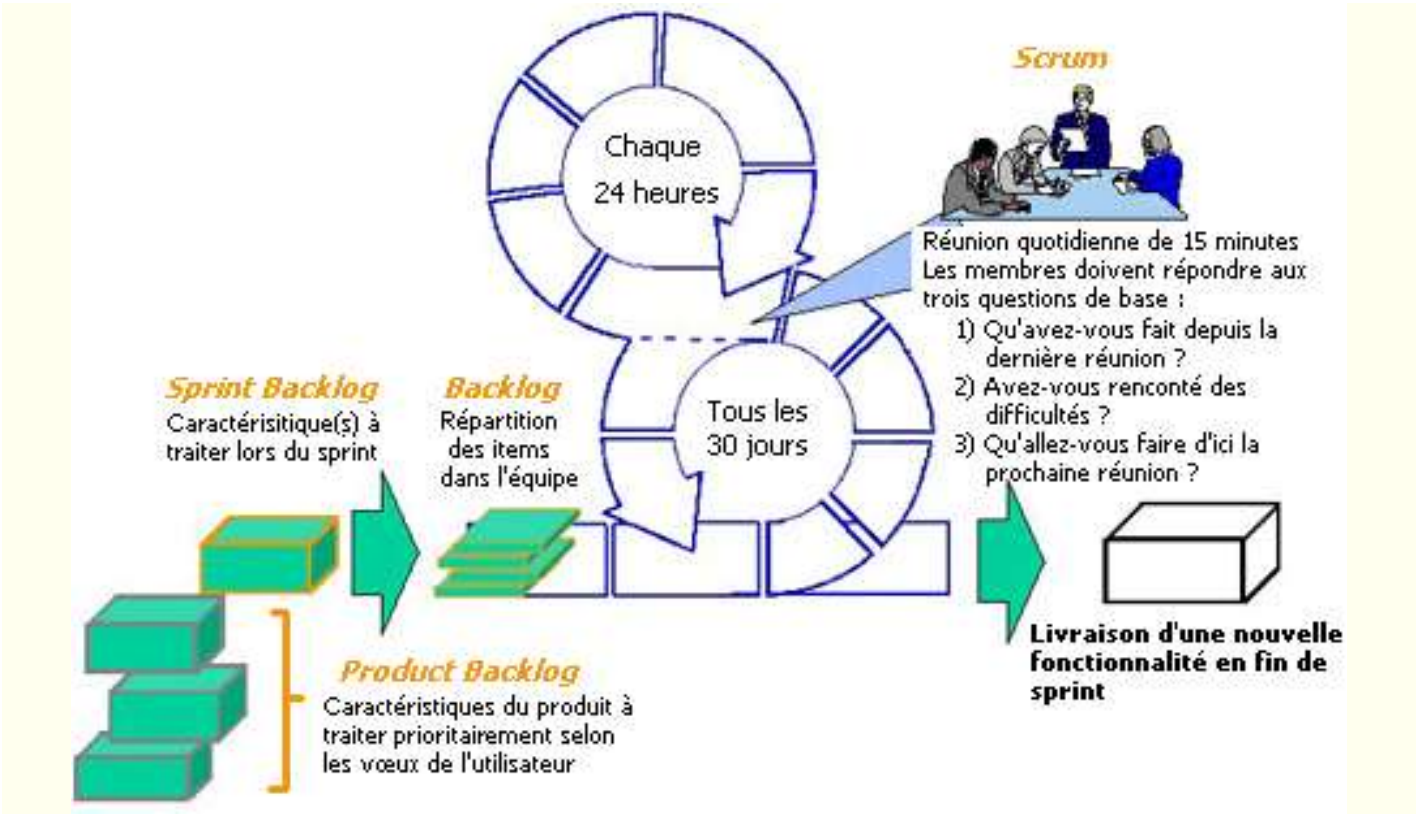
Cette phase de lancement s'articule autour de deux axes de réflexion : l'étude d'opportunité et l'expression initiale des besoins.

L'étude d'opportunité est très variable d'un projet à l'autre, tout dépend du contexte de l'entreprise, de la nature du projet (sous-traitance ou interne), etc. Chaque entreprise a sa propre politique sur cette activité.

L'expression initiale des besoins n'est pas un élément défini dans Scrum et n'est **surtout pas** une activité de contractualisation d'un cahier des charges. L'esprit d'une telle activité dans les méthodes Agiles est de définir d'une part le **cadre du projet** (pour que l'équipe s'imprègne du contexte métier), et d'autre part une **première analyse globale** des besoins. L'objectif est d'identifier un maximum de fonctionnalités que le logiciel devra implémenter, en se limitant à un niveau de précision assez grossier. On peut par exemple utiliser une approche par raffinages successifs, en partant des secteurs métiers concernés par l'application, puis en identifiant les activités métier, qu'on décompose en tâches métier qui correspondent à des fonctionnalités que l'on doit/peut ou non implémenter dans le logiciel final.

L'objectif pour Scrum est de produire la première version du backlog de produit.

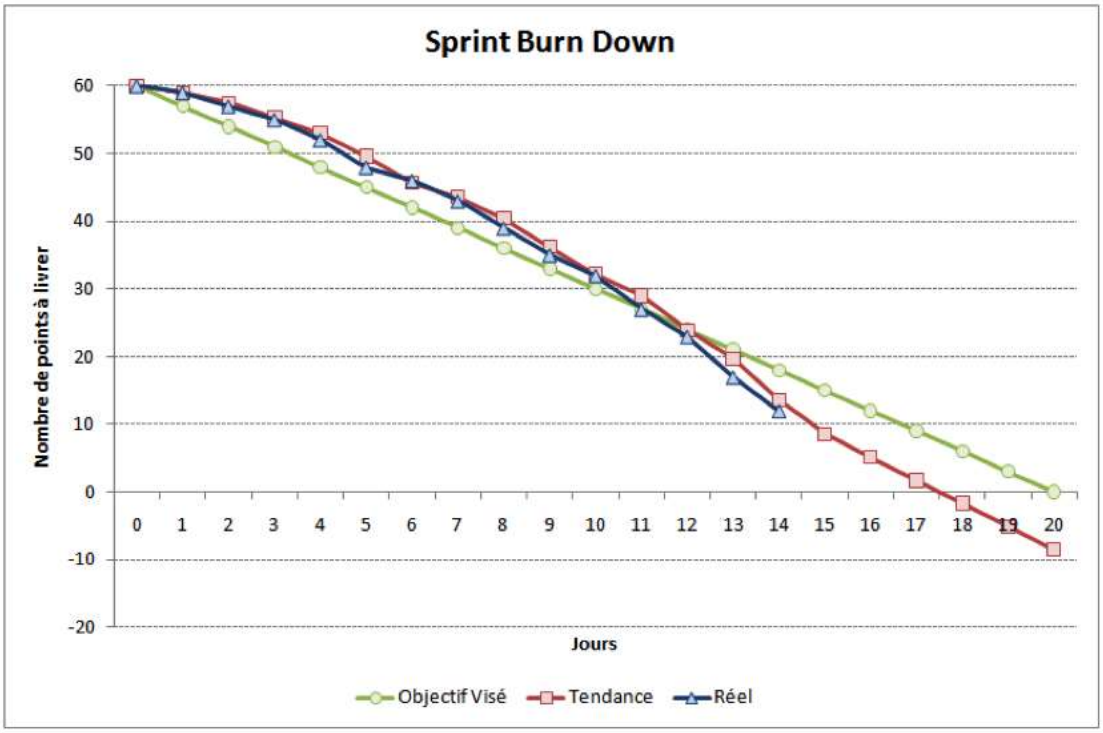
## *Vue globale*




Vue synthétique du processus Scrum

## *Burndown Charts*

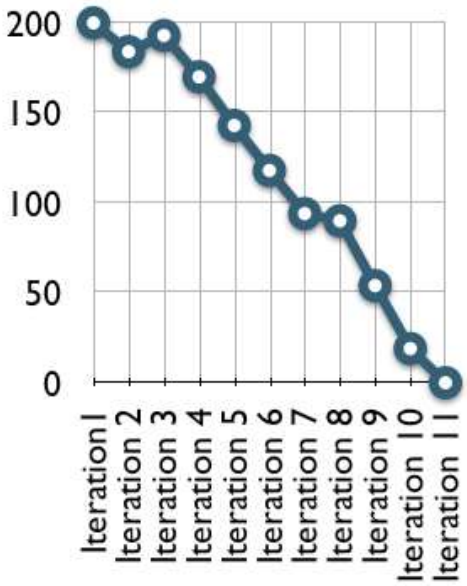
Les **burndown charts** (*graphiques d'avancement*) permettent de visualiser graphiquement l'avancement du travail. Une interprétation simple permet d'avoir une idée sur les échéances futures.

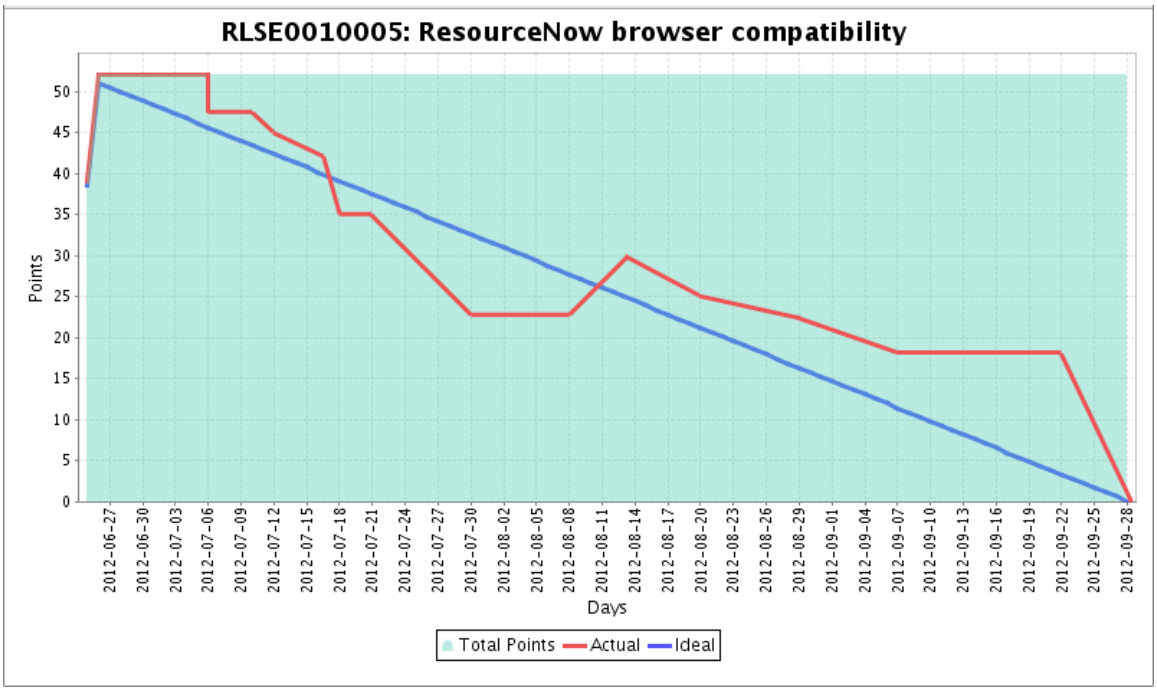



Exemple de Sprint Burndown Chart

Ce graphique représente la quantité totale d'heures restantes à faire dans le sprint, au fil des jours. Il permet d'avoir une vue sur l'avancement du sprint.

## *Release Burndown Chart*




Exemples de Release Burndown Chart

Ce graphique représente la quantité totale de points restant à faire dans la release, au fil des sprints. Il permet d'avoir une vue sur l'avancement de la release.

## *Interprétation*

Ces graphiques sont très intéressants à analyser et interpréter. Outre le fait de montrer l'avancement concret du travail, ils permettent d'anticiper de façon relativement fiable les échéances futures en cours du sprint ou de la release.

On peut observer de légères augmentations du reste à faire sur le burndown chart du sprint. Cela correspond généralement à une « réestimation » à la hausse, suite à la prise en compte de contraintes techniques que l'on n'avait pas vues lors de l'estimation initiale. Si c'est le cas, il est indispensable de comprendre la cause de ces augmentations. Le même phénomène peut s'observer sur des légères diminutions, pour les mêmes raisons.

On peut utiliser en cours du sprint la tendance de la courbe pour avoir une idée approximative de la fin du sprint. Cela consiste à prendre un segment de droite dont la pente est représentative des valeurs déjà recensées, et de le prolonger jusqu'à son point d'intersection avec l'axe des abscisses. Cela nous donne alors la date *a priori* de la fin du sprint, et nous permet alors de prendre une décision sur la suppression (ou l'ajout) d'un item de backlog du produit à réaliser dans ce sprint.

C'est exactement la même chose pour les burndown charts de release. Si la date de publication de la release est clairement au-delà de ce que l'on espérait, on peut aviser en enlevant des items de backlog du produit ou en changeant leur ordre, de sorte que les fonctionnalités les moins importantes soient celles qui risquent de ne pas être développées à temps.

Cette approche, bien que basée sur des tendances approximatives, permet d'identifier très tôt les risques de défaillance et d'agir en conséquence, en conservant à l'esprit le caractère critique des fonctionnalités à développer. Ces décisions importantes relèvent complètement du directeur de produit.

Une dernière chose importante : la fiabilité de la vélocité de l'équipe et des estimations qu'elle a faites augmente au fil des sprints. On élimine de cette façon les risques majeurs au plus tôt dans le projet.

## *Qualité de l'environnement de travail*

Un concept fort de Scrum est la qualité de l'environnement de travail de l'équipe. Cela inclut :

- Pas de changements imposés pendant un sprint
- Toute l'équipe dans une même pièce
- Un tableau blanc et/ou en liège
- Un bon outil de suivi du projet
- Prévenir des interventions extérieures (téléphone, irruption dans la pièce, etc)
- Tout ce qui peut rendre l'équipe plus sereine et efficace

## *Documentation de projet*

Scrum n'impose aucune documentation particulière pour les projets. Des documents sont implicitement produits (backlogs, burndown charts), mais ils ont une vocation avant tout utilitaire.

Produire de la documentation, c'est souvent utile, mais c'est aussi souvent inutile. En plus, il faut la maintenir à jour, et c'est quelque chose qui est rarement fait sur le terrain. Pour

savoir s'il faut rédiger un document, on peut se poser une question très simple : **Est-ce que ce document va m'être** *vraiment utile*, **et** *tout de suite* **?**.

Voici quelques exemples de documents utiles, et dans quels cas :

- Diagrammes métiers (processus, objets...), associé au backlog de produit : uniquement si la logique métier du client qui concerne l'application est vraiment complexe. Dans ce cas, l'équipe devrait produire ce document avec lui.
- Diagramme de séquence, associé à un item du backlog du produit : uniquement si la fonctionnalité aura une utilisation complexe, tant au niveau métier qu'applicatif.
- Diagrammes d'architecture du logiciel (classes, modules, composants, etc), pour le projet : indispensable pour avoir toujours sous les yeux une vue de l'architecture, et s'assurer ainsi qu'elle est de qualité.
- Les manuels utilisateur, à chaque sprint : les manuels sont produits à chaque sprint, et pas en fin de projet. Utiliser des vidéos de démonstrations commentées est une solution efficace.
- Les FAQ pour la hotline : des cas classiques où les utilisateurs ne vont pas comprendre un comportement métier. Cela permet de traiter un maximum de problèmes au niveau de la hotline, avant que cela n'arrive aux équipes de développement/maintenance.

Bref, un document ne doit être produit que si **son utilité est réelle et immédiate**.

# Outils pour Scrum

Un bon artisan n'est rien sans un bon outil. Il n'y a pas aujourd'hui d'outil *ultime* pour Scrum. Voici un petit panel des possibilités.

## *Papier et Crayon / Tableur*

Scrum peut être mis en pratique avec trois fois rien : deux listes suffisent. La liste des items du backlog de produit, et la liste des items du backlog de sprint. La saisie et la mise à jour des données est simplement un peu moins agréable.

# Conclusion

## *Scrum*

Scrum est un processus de développement de logiciels qui s'intéresse plutôt à l'organisation du projet qu'aux aspects techniques. Son cadre de travail est sa force, si bien que cette méthode pourrait être appliquée à d'autres domaines. Son approche itérative et basée sur les besoins priorisés du client lui confèrent une flexibilité extrême. Elle incarne bien par-là l'état d'esprit de la mêlée de rugby : avancer tous ensemble vers un but commun, la réussite du projet.

Scrum est un processus intéressant comme premier pas vers les méthodes Agiles : il est facile à comprendre et à pratiquer. La seule difficulté relève plutôt de l'environnement organisationnel (voir la mise en garde ci-dessous).

## *Mise en garde*

On entend de plus en plus de sociétés clamer qu'elles sont Agiles, comme argument commercial à la mode, parce qu'elles utilisent quelques pratiques des méthodes Agiles. Mais **être Agile**, c'est bien plus que de mettre en pratique une méthode. L'Agilité requiert des dispositions particulières qui sont loin d'être faciles à mettre en place :

- Une **organisation adaptée** : il est très difficile de faire admettre aux organes décideurs d'une entreprise de travailler de façon Agile. En effet, cela signifie de ne pas disposer dès le départ d'une date et d'un budget très précis, mais plutôt d'un ordre de grandeur.
- Un **état d'esprit** : être Agile, c'est privilégier l'esprit d'équipe, et pas seulement dans la réalisation technique. Le client doit comprendre que l'on attend de lui un investissement personnel bien supérieur à celui des méthodes plus classiques, en testant le logiciel souvent et en participant à toutes les réunions.
- Un **environnement favorable** : éliminer les contraintes dans une entreprise n'est pas toujours évident. Comment limiter les appels téléphoniques trop fréquents, les irruptions dans la pièce de l'équipe, les opérations "urgentes" demandées aléatoirement par la direction, etc ?

Bref, utiliser une méthode comme Scrum au niveau de l'équipe technique ne suffit pas. L'Agilité est une façon de travailler différente de ce dont on a l'habitude, c'est l'attitude du changement, de la flexibilité, de l'adaptation, et de l'amélioration continue. Ce n'est pas aussi simple qu'une méthode...

# Glossaire

**Directeur de produit (*Product Owner*)**

Personne responsable de produire et maintenir à jour le backlog de produit. C'est lui qui en détermine les priorités et qui prend les décisions concernant l'orientation du projet.

**ScrumMaster**

Membre de l'équipe dont l'objectif principal est de la protéger des perturbation extérieures. Il est complètement transparent pour la communication entre l'équipe et les clients, et n'a aucun pouvoir hiérarchique sur l'équipe. C'est en revanche un facilitateur pour les problèmes non techniques de l'équipe.

**Backlog de produit (*Product Backlog*)**

Liste des fonctionnalités qui devront être réalisées par le logiciel.

**Backlog de sprint (*Sprint Backlog*)**

Liste des tâches à accomplir pendant un sprint. Elles correspondent à la réalisation des items de backlog du produit affectés au sprint.

**Mêlée quotidienne (*Daily Scrum*)**

Réunion quotidienne de 15 minutes, qui a pour but de faire le point sur ce qui a été fait depuis la dernière mêlée, ce qui est prévu de faire jusqu'à la prochaine et quelles sont les embûches rencontrées durant le travail.

**Sprint**

Nom d'une itération dans Scrum. Cette itération dure 30 jours calendaires en théorie, mais en pratique on utilise plutôt entre 2 et 4 semaines. Pendant une itération, l'équipe doit développer une liste d'items du backlog de produit qui a été définie au début de ce sprint.

**Graphique d'avancement (*Burndown Chart*)**

Graphique qui montre la tendance du reste à faire total de jour en jour (pour les sprints) ou de sprint en sprint (pour les releases).

http://www.techno-science.net/?onglet=glossaire&definition=797