SCRUM

Scrum est une méthode de gestion de projet, et non d'ingénierie informatique!

Table des matières

I. INTRODUCTION

Les utilisateurs sont de plus en plus nombreux et réclament des sites web toujours plus agréables à consulter, avec des interfaces intuitives, des temps de réponse courts… Peu se rendent compte qu'ils utilisent un outil qui a demandé la mise en œuvre de nombreuses ressources au travers de développement de logiciels complexes.

Des entreprises réussissent mieux que d'autres à mettre à la disposition des utilisateurs des outils performants et agréables à utiliser. Ce sont celles-là qui s'accaparent de nombreuses parts de marché et deviennent des acteurs économiques majeurs et incontournables. Google, par exemple, est fréquemment rangée dans la catégorie des entreprises à succès. Or, gérer un projet n'est pas chose aisée. Il suffit de consulter les chiffres d'échecs des projets informatiques pour s'en rendre compte.

Quelques chiffres…Etude sur les projets [Standish Group - CHAOS]

- **31.1% des projets ne sont jamais terminés**
- **16.2% terminés dans les délais et les budgets**
- **42% dans le respect des éléments du début du projet**
- **145% de moyenne de dépassement des coûts**
- **163% de moyenne de dépassement des délais**

Aux Etats-Unis, le coût des projets abandonnés ou en échec représenterait 75 milliard de $ (2001)

Quand le projet n'échoue pas, on constate que les dépassements de budget et de délais sont aussi très fréquents. On peut alors se demander si **les méthodes traditionnelles de gestion de projets sont encore bien adaptées pour mener à bien le développement d'un logiciel dans un environnement mondialisé très concurrentiel.**

Il existe un nouveau courant de pensée en matière de gestion de projet de développement logiciel qui propose de traiter les problèmes évoqués précédemment et qui a pour **objectif de livrer les projets en temps et en**

heure, tout en permettant aux clients de changer d'avis fréquemment. Il s'agit des **méthodes agiles**.

Parmi elles, une méthode semble particulièrement appréciée de ceux qui l'utilisent : **Scrum**. Les fondements de Scrum se trouvent dans un article qui synthétise les meilleures pratiques les plus courantes dans **10 entreprises innovantes japonaises** : « The new new product development game » publié en 1986. Cet article introduit en particulier le terme de scrum pour **désigner les pratiques d'itérations et d'équipes autogérées et adaptatives**.

En 1994, Jeff Sutherland introduit des pratiques issues de cet article dans la société dont il est vice-président à l'époque – Easel Corporation. Il est aussi influencé par un article traitant de l'excellente productivité d'un projet chez Borland Corporation qui utilisait efficacement **les réunions quotidiennes**.

En 1995, Ken Schwaber travaille avec Jeff Sutherland chez Easel Corporation sur la formalisation de Scrum. Leurs résultats sont décrits dans un article3 de Ken Schwaber.

En 2001, Ken Schwaber s'associe à Mike Beedle pour affiner Scrum et aboutir à la version décrite dans le livre « Agile software development with Scrum » publié fin 2001, qui marque réellement l'apparition de Scrum en tant que méthode de gestion de projet.

Ce dossier propose de dresser un état de l'art de Scrum. Nous verrons dans un premier temps ce qui caractérise les méthodes agiles, puis nous verrons la méthode Scrum en détail. Nous étudierons ensuite quelques outils permettant de supporter la méthode. Enfin, nous envisagerons quelles sont les perspectives pour cette méthode.

II. RAPPEL SUR LES METHODES AGILES

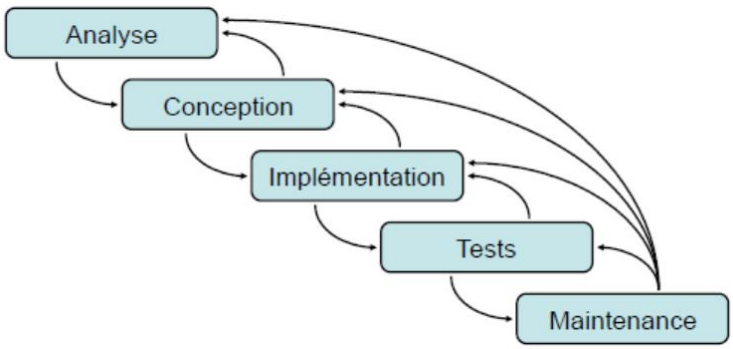
1) Le besoin d'agilité

La plupart des logiciels ne correspondent pas à un processus de développement défini ou à un problème de production en masse qu'on peut trouver en usine. **Le développement logiciel produit un nouveau logiciel à chaque fois**. Il est par conséquent **difficile de prévoir dès le début du projet des spécifications fiables**

D'autre part, des études datant de 1988 et 1997 établissent qu'au cours d'un projet, **les changements dans les besoins exprimés au début du projet sont de l'ordre de 25 à 35%**. Jeff Sutherland suggère même que le **taux de changement** peut atteindre plus de 50% dans les projets actuels.

En outre, la **flexibilité** donne un avantage compétitif certain : pouvoir réagir même tardivement dans le développement d'un projet permet de mettre sur le marché un produit compétitif, au plus près des besoins du consommateur.

Il semble par conséquent indispensable d'inclure une certaine dose d'agilité dans la façon de gérer un projet. Or, la méthode classique en cascade ne le permet pas.



C'est d'ailleurs pour cela que les **cycles de vie itératifs et évolutifs** ont été créés.

Les méthodes agiles sont un sous-ensemble des méthodes itératives et évolutives qui existent depuis les années 70. Il est donc important de constater que **l'abandon du modèle en cascade au profit du modèle itératif et évolutif ne fait pas la spécificité des méthodes agiles**. Le Department of Defense américain a d'ailleurs publié une note dès 1994 demandant l'abandon du

modèle en cascade au profit du modèle itératif et évolutionnaire, donc bien avant le manifeste de l'agilité publié en 2001.

2) Le manifeste de l'agilité

a. Les 4 valeurs des méthodes agiles

Scrum fait partie des méthodes agiles. Toutes ces méthodes partagent des valeurs communes qui ont été reprises dans le **« Manifesto for Agile Software Development »** ou manifeste pour le développement agile de logiciel.

Les méthodes agiles sont basées sur le pragmatisme et le développement itératif. Elles définissent un cadre moins rigide que les méthodes traditionnelles. Les 4 valeurs clés des méthodes agiles sont les suivantes :

1. **« Individuals and interactions over processes and tools »**, c'est-à-dire : privilégier les personnes et les interactions entre elles, plutôt que les procédures et les outils. Cela signifie qu'il est nécessaire d'avoir une équipe soudée dont les membres communiquent ensemble.

2. **« Working software over comprehensive documentation »**, c'est-à-dire : privilégier la mise en œuvre d'un logiciel fonctionnel plutôt que de dépenser trop d'énergie dans la constitution de documentation exhaustive. Le but final d'un projet est après tout de produire un logiciel. Il sera en outre plus facile pour le client de valider ses besoins grâce au logiciel plutôt que de le faire à partir de documentation.

3 **« Customer collaboration over contract negotiation »**, c'est-à-dire : privilégier la collaboration avec le client plutôt que la négociation de contrat. Le client doit prendre une part importante dans le développement du logiciel et peut même faire partie intégrante de l'équipe. Le but est de recueillir un retour régulier qui permettra de coller au mieux aux attentes du client. Pratiquer de la sorte aura plus d'intérêt que de s'en tenir au contenu négocié dans le contrat.

4 **« Responding to change over following a plan »**, c'est-à-dire : être réactif au changement plutôt que suivre un plan établi à l'avance.

3) Les 12 principes des méthodes agiles

Les 4 valeurs précédentes sont déclinées en 12 principes qui sont les suivants :

1. **« Our highest priority is to satisfy the customer through early and continuous delivery of valuable software. »**, c'est-à-dire : notre priorité la plus grande est de satisfaire le client à travers la livraison rapide et continue de logiciel présentant un intérêt pour lui.

2. **« Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage. »**, c'est-à-dire : accueillir les demandes de changements même tardifs dans le développement. Les processus agiles s'attachent au changement pour donner un avantage compétitif au client.

3. **« Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale »**, c'est-à-dire : livrer du logiciel fréquemment, de 2 semaines à 2 mois avec une préférence pour les périodes courtes.

4. **« Business people and developers must work together daily throughout the project »**, c'est-à-dire : les acteurs métier et les développeurs doivent travailler ensemble quotidiennement pendant toute la durée du projet.

5. **« Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done. »**, c'est-à-dire : construisez les projets autour de personnes motivées. Donnez-leur l'environnement nécessaire et répondez à leurs besoins, et faites leur confiance pour mener à bien le travail.

6. **« The most efficient and effective method of conveying information to and within a development team is face-to-face conversation. »**, c'est-à-dire : le moyen le plus efficace de véhiculer l'information vers et à l'intérieur d'une équipe de développement est la conversation en face à face.

7. **« Working software is the primary measure of progress. »**, c'est-à-dire : la métrique principale pour juger de la progression d'un projet est le logiciel fonctionnel.

8. **« Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely. »**, c'est-à-dire : les processus agiles encouragent le développement durable. Les

financeurs, les développeurs, et les utilisateurs doivent pouvoir maintenir un rythme constant indéfiniment.

9. **« Continuous attention to technical excellence and good design enhances agility. »**, c'est à-dire : une attention continue à l'excellence technique et au bon design améliore l'agilité.

10. **« Simplicity--the art of maximizing the amount of work not done--is essential. »**, c'est-à dire : la simplicité --l'art de maximiser le montant de travail non fait --est essentiel. Eliminer le travail superflu et inutile.

11. **« The best architectures, requirements, and designs emerge from self-organizing teams. »**, c'est-à-dire : les meilleures architectures, besoins, et conceptions proviennent des équipes auto organisées.

12. **« At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly. »**, c'est-à-dire : à intervalles réguliers, l'équipe réfléchit aux manières de devenir plus efficace, puis ajuste ses comportements de façon à s'y conformer.

III. SCRUM

4) Qu'est-ce que Scrum ?

La définition du dictionnaire pour scrum est la suivante :

scrum: noun Rugby: an ordered formation of players in which the forwards of each team push against each other with heads down and the ball is thrown in..

C'est-à-dire : en rugby, une formation ordonnée de joueurs dans laquelle les avants de chaque équipe se poussent les uns les autres avec les têtes baissées et la balle est introduite.

Une recherche de la traduction donne la réponse suivante : scrum, SPORT mêlée f.

Scrum est donc un terme utilisé en rugby qui signifie mêlée. A priori, pas grand-chose à voir avec une méthode de gestion de projet agile... Pourtant l'image est assez parlante. Dans Scrum, comme au rugby, l'équipe de développement va constituer une formation soudée qui va faire bloc pour avancer et venir à bout des obstacles pour atteindre son objectif.

Scrum est une méthode agile utilisée dans le développement de logiciels. Elle vise à **satisfaire au mieux les besoins du client tout en maximisant les probabilités de réussite du projet**.

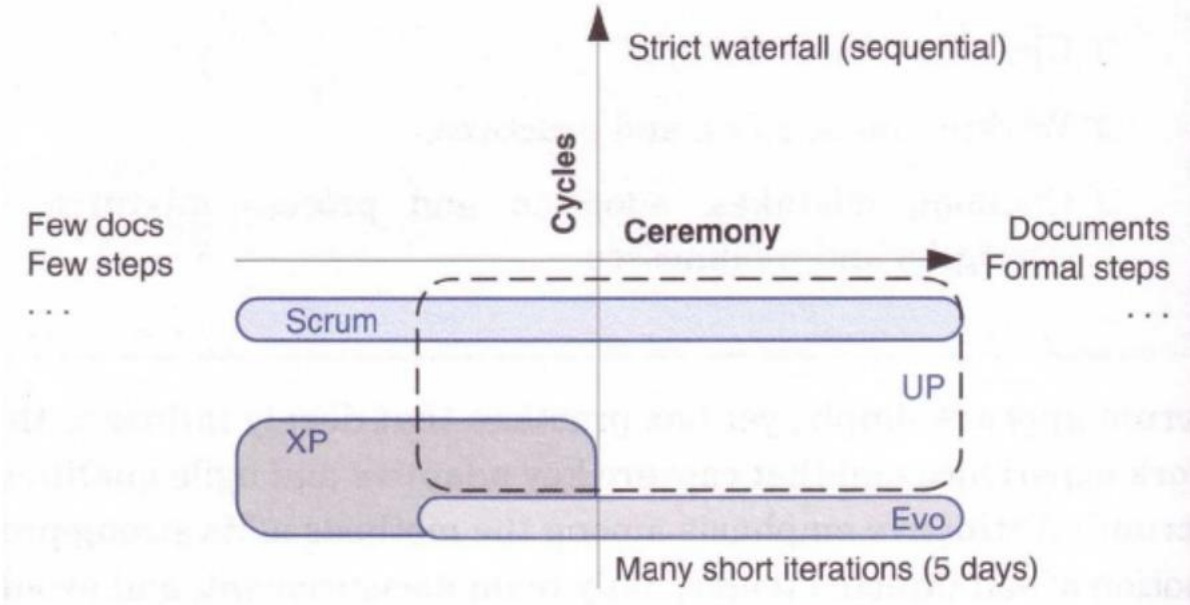
Scrum suppose que **le développement d'un logiciel n'est pas prévisible et ne peut donc pas suivre de processus défini. Le résultat du projet n'est pas connu au départ.** Il dépend des réorientations que lui donne le client en cours de route. Une fois que le financement est terminé, le projet s'arrête. Comme le client classe lui-même les fonctionnalités par ordre d'importance, il est certain d'obtenir un logiciel à forte valeur ajoutée en fin de projet.

Un projet utilisant Scrum est composé d'une suite d'itérations courtes de l'ordre de 3 à 6 semaines appelées sprints. Le projet peut être réorienté par le client à la fin de chaque sprint, et uniquement à ces moments là.

Scrum est une **méthode pragmatique basée sur des pratiques de bons sens** qui proviennent de l'expérience des concepteurs de la méthode. **Elle est particulièrement utile lorsqu'un projet est très complexe ou que les inconnues sont trop nombreuses pour appliquer un processus défini qui rendrait un plan obsolète peu après le début du projet.**

5) Positionnement de Scrum

Situation de Scrum par rapport aux principales méthodes agiles



La figure montre où se positionne Scrum parmi d'autres méthodes agiles. Les critères pris en compte sont : en ordonnée, le nombre de cycles – nombre et durée des itérations -et en abscisse la rigidité de la méthode -documents à produire, étapes à respecter… Il est possible de constater que Scrum laisse une grande latitude quant aux documents à produire.

Une équipe standard est composée de 6 à 8 personnes en Scrum. Il sera bien entendu nécessaire de recourir à des formes étendues de Scrum pour des projets nécessitant de nombreuses personnes. Les projets mettant en jeu la vie des utilisateurs demanderont plus de documentation que ceux de pur confort.

b. Parts de marché

Une enquête menée en décembre 2006 montrait une prédominance de Scrum et 98+ parmi les méthodes agiles, avec **Scrum à environ 60% de parts de marché**.

Cela est confirmé par une enquête menée en été 2008 qui montre que Scrum et XP sont les 2 méthodes agiles les plus utilisées dans les entreprises ayant recours à ces méthodes avec 49% pour Scrum et 30% pour Extreme Programming.

En ce qui concerne l'adoption des méthodes agiles par rapport aux méthodes traditionnelles, une étude de février 2008 aux Etats-Unis montre que 69% des entreprises ayant répondu à ce questionnaire utilisent une méthode agile.

c. Les principes clés

Scrum se conforme en tous points aux valeurs du manifeste de l'agilité évoquées précédemment. Les principes clés qui lui sont propres sont les suivants :

• **Auto-organisation et pouvoir de décision donné à l'équipe** : les membres de l'équipe décident eux-mêmes des solutions à mettre en œuvre pour arriver au résultat souhaité.

• **Sprints en isolement** : pendant un sprint, rien ne doit interférer avec l'équipe, et surtout pas le client qui a tout le loisir d'intervenir entre les sprints pour réorienter le projet.

• **Délais fixes** : le projet se déroule dans des laps de temps fixes qu'il faut respecter. Cela a comme conséquences d'obliger l'équipe à se concentrer sur l'essentiel et à faire ce qui est possible et non à rechercher la perfection.

• **Réunion quotidienne de 15 minutes avec une liste de questions prédéfinies**.

• **Démonstration du résultat du sprint** : à chaque fin de sprint, pour le product owner en particulier, mais aussi pour tous ceux qui sont intéressés par le projet.

• **Planning adaptatif** : à chaque fin de sprint, le planning de développement est adapté en fonction des nouvelles priorités données par le client.

d. Les rôles

1. Les poules et les cochons

En Scrum, on distingue plusieurs rôles répartis dans 2 catégories : les poules et les cochons. Cette appellation vient d'une histoire drôle:

Un jour, une poule et un cochon se promènent ensemble. Soudain, la poule dit au cochon : « Ouvrons un restaurant ensemble. Qu'en penses-tu ? ». Le cochon réfléchit un instant puis lui répond enthousiaste : « C'est une excellente idée. Comment l'appellerais-tu ? ». La poule lui dit « Jambon et oeufs ». Le cochon lui répond alors : « Non merci ! Je serais impliqué alors que toi tu serais seulement concernée. »

2. Les cochons

Les cochons sont donc ceux qui s'impliquent dans le projet. Il s'agit de l'équipe, du scrummaster et du product owner.

i. Le product owner

Il s'agit de la personne représentant le client. Le terme pourrait être traduit par « directeur produit ».

Le product owner :

• Représente toutes les personnes qui ont un intérêt dans le logiciel qui sera produit.

• Finance le projet.

• Est responsable de l’établissement des besoins initiaux du projet qui sont listés dans le backlog produit.

• A des objectifs de retour sur investissement.

• Décide des releases, c'est à dire des versions qui seront mises à disposition des utilisateurs.

• Est responsable de la mise à jour du backlog produit. Il doit notamment classer régulièrement les besoins exprimés dans le backlog produit pour s'assurer que les fonctionnalités qui génèreront le plus de valeur pour le logiciel seront implémentées en priorité. Il doit également mettre à jour les besoins en fonction des incréments de logiciel qu'il teste à chaque fin de sprint.

ii. Le scrummaster

Une traduction littérale pour scrummaster serait « maître de la mêlée ». Ce rôle semble simple mais il est en réalité délicat à appréhender car il diffère énormément de la culture projet traditionnelle. Le scrummaster n'est pas un chef de projet. Il n'a aucune autorité hiérarchique sur l'équipe. Il s’agit plus d’un rôle de coach, et de facilitateur. Un catalyseur de projet en quelque sorte.

Son rôle est de :

• Isoler l'équipe des perturbations extérieures en cours de sprint.

• Apprendre la méthodologie Scrum à l'équipe.

• S'assurer que l'équipe respecte bien la méthode : que chacun fait bien du Scrum et ne dévie pas vers d'anciennes méthodes connues et maîtrisées.

• Répondre aux besoins de l'équipe.

• Faire cadrer scrum dans la culture d'entreprise.

• Travailler avec le product owner pour sélectionner avec lui les besoins apportant le plus de valeur ajoutée au logiciel. Il est responsable de la réussite du projet, au même titre que l'équipe.

iii. L'équipe

Les responsabilités et le pouvoir de décision sont transférés du chef de projet (dans les organisations traditionnelles) vers l'équipe. Ainsi l'équipe :

• Développe les fonctionnalités : elle transforme les besoins en incréments fonctionnels de logiciel.

• Est autogérée et auto organisée : c'est à elle de décider comment arriver au mieux à produire le logiciel à partir des besoins exprimés par le product owner.

• Est multidisciplinaire. Si les spécialités persistent, elles sont toutes présentes au sein de l'équipe. Les méthodes d'estimation supposent que les personnes ne sont pas spécialisées et que chacun est capable de remplir toutes les tâches du backlog de sprint.

• Est responsable collectivement du succès de chaque itération. Toutes les cartes sont entre ses mains pour réussir.

3. Les poules

Les poules sont tous les autres, ceux qui ont un intérêt dans le projet mais qui ne sont pas directement impliqués. Il s'agit par exemple des utilisateurs, du management…

4. Les certifications

La Scrum alliance1 délivre 3 niveaux de certification.

Figure 4 les niveaux de certification Scrum

Il peut sembler étrange dans un premier temps d'avoir des certifications pour le product owner. En y réfléchissant, cela est tout à fait logique, puisque le product owner est un rôle essentiel dans la réussite d'un projet en Scrum.

e. Organisation de la production

5. Vue d'ensemble de Scrum

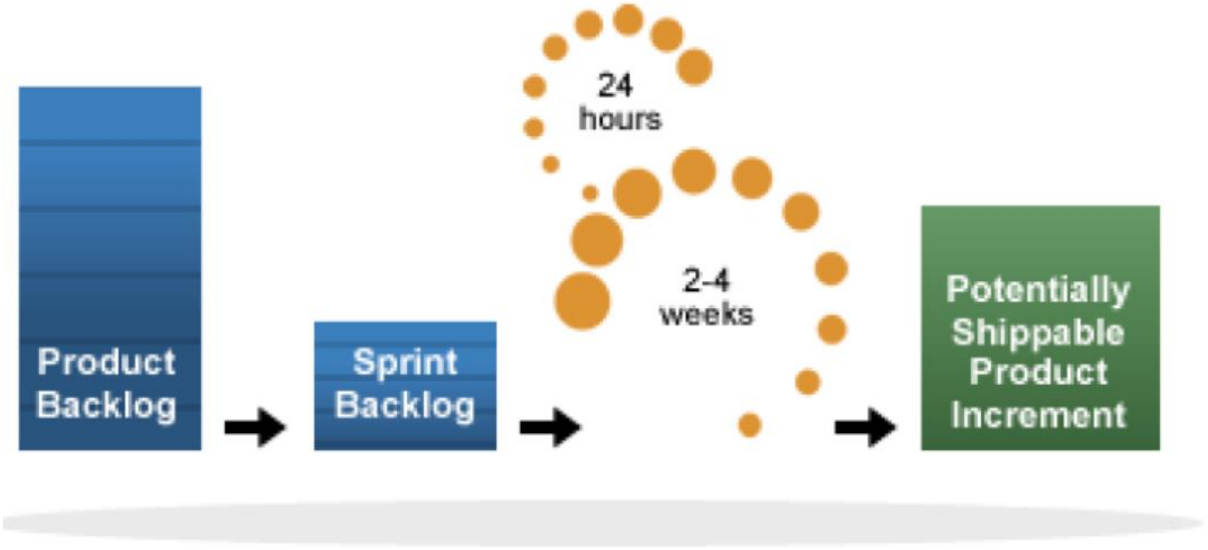


Figure 5 : organisation globale de scrum

1. Le projet est organisé en sprints de 2 à 4 semaines. C'est pendant cette phase que les fonctionnalités choisies sont développées et que le logiciel est créé. Les itérations s'appellent sprints afin de refléter l'augmentation de productivité de l'équipe.
2. Le backlog du produit constitue l'ensemble du travail connu sur le projet à un instant t. Il est régulièrement remis à jour et les besoins qui le constituent sont aussi régulièrement priorisés.
3. Le travail à faire durant un sprint est listé dans le backlog du sprint. Le contenu du sprint est un extrait du backlog produit. Les besoins sont priorisés de la même façon que dans le backlog du produit.
4. Durant le sprint, une réunion quotidienne appelée daily scrum, ou mêlée quotidienne rassemble l'équipe afin de réorienter le sprint si besoin est. Il s'agit du point de contrôle de l'équipe.
5. A la fin d'un sprint, l'équipe livre au client un incrément de logiciel fini potentiellement livrable.

Ce cycle est répété jusqu'à ce que :

- La date de fin de projet est atteinte
- Le client ne peut plus financer le projet.
- Le client considère que le logiciel délivre suffisamment de valeur et décide d'arrêter le développement.

Il apparaît donc que dans le triptyque classique de gestion de projet, un projet Scrum sera adapté en modifiant l'envergure du projet. Comme le client s'assure que les besoins sont classés selon leurs priorités, c'est à dire la valeur qu'ils apportent au projet, ce sont toujours les tâches les plus importantes qui sont faites en premier.

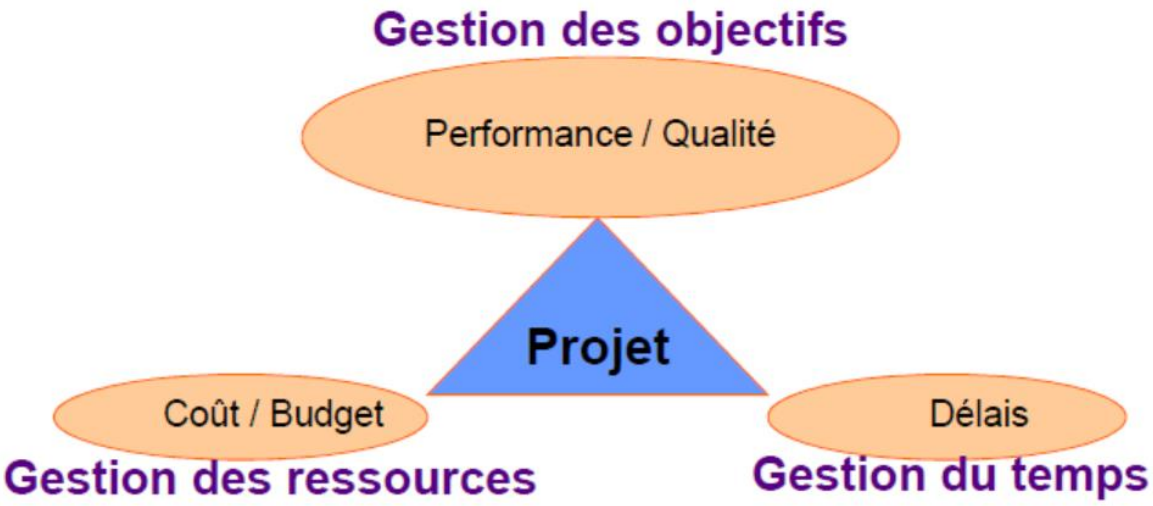


Figure 6 : le triptyque de la gestion de projet.

6. Backlog produit

Le backlog produit est un des éléments principaux de Scrum. Il s'agit d'un catalogue qui recense toutes les fonctionnalités souhaitées sur le projet. Il est créé par le product owner. Les entrées sont décrites dans les termes du client, c'est-à-dire en termes métier.

Definition Backlog : catalogue des tâches à effectuer. Il existe le backlog produit qui liste tous les besoins connus à traiter, et le backlog de sprint, qui est un extrait du backlog produit.

Le product owner priorise le backlog en fonction des éléments qui représentent le plus de valeur pour lui. Il a la possibilité de changer ce backlog a tout moment : ajouter ou enlever des éléments, changer les priorités, modifier les descriptions...

Un backlog produit ressemble à l'extrait de document suivant :

Backlog produit - Site marchand XY					
ID_Item	Titre	Importance	Estimation	Démonstration de la fonctionnalité	Commentaires
1	Ajouter un article dans le panier	30	3	Le client sélectionne un article dans la page parmi une liste de produits, l'ajoute au panier, puis visualise le contenu du panier pour vérifier que le nouvel article a bien été ajouté à sa commande.	Nécessite un diagramme de séquence UML
2	Consulter l'historique des commandes	20	5	Le client se loggue, puis sélectionne l'option "historique de mes commandes". Une page récapitulant l'ensemble des commandes passées s'affiche.	On ne tient pas compte des utilisateurs qui n'ont pas de compte pour le moment.
3	Ouvrir un compte sur le site	10	3	Sur la page d'accueil, l'internaute sélectionne l'option "ouvrir un compte", puis crée son compte en indiquant son mail et un mot de passe. Le mail de confirmation de compte est reçu dans les 5 minutes qui suivent la validation du compte.	

Figure 7 : extrait d'un backlog produit

L'importance est donnée par le product owner. Plus le chiffre est élevé, plus le besoin est important à ses yeux et doit être développé en priorité par l'équipe.

L'estimation est faite par l'équipe. Elle peut être de 2 types : en jours idéaux ou en points d'histoire:

- **Estimation en jours idéaux** : il s'agit dans ce cas de déterminer le temps nécessaire pour terminer une tâche en considérant que chaque journée de travail permet une productivité de 100% : pas de réunion, aucune interruption de collègues…
- **Estimations en points d'histoire** : dans ce cas, les points attribués à chaque besoin sont totalement indépendants du temps nécessaire pour les réaliser. C'est la taille relative entre les différents besoins qui est importante.

Il est conseillé d'estimer en points d'histoire, même s'il semble plus facile d'introduire en premier lieu l'estimation en jours idéaux dans une équipe.

Cette estimation peut ne pas être précise à ce stade de la démarche. Il est par contre important de pouvoir comparer les charges de travail des besoins entre eux.

La colonne « Démonstration de la fonctionnalité » indique comment il faudra démontrer la réalisation du besoin lors de la réunion de revue de sprint. Si l'équipe pratique le test driven development, la description peut aussi servir pour construire les tests unitaires.

D'autres personnes que le product owner peuvent ajouter des éléments dans le product backlog. Ce sera notamment le cas pour les tâches non fonctionnelles et les tâches techniques. Par exemple, mettre en place un serveur de builds, mettre en place un environnement de tests...

Par contre, seule product owner peut assigner une importance à un besoin. De même, seule l'équipe peut assigner une estimation à un besoin. Il faudra par conséquent expliquer au product owner pourquoi les tâches techniques et les besoins non fonctionnels sont importants et ce qu'il va gagner à les mettre en place.

7. Le sprint

Un sprint est un délai fixe durant lequel l'équipe fait tout ce qui est nécessaire pour tenir ses engagements.

C'est pendant le sprint que le logiciel est développé par l'équipe. Scrum ne recommande pas de durée de sprint précise. **La durée est de 2 semaines minimum et de 6 semaines maximum**. C'est à l'équipe de choisir une durée qui lui convient. Une fois qu'une durée est choisie, il est nécessaire que cette durée devienne le délai standard de tous les sprints. Il semble qu'il y ait un consensus sur une période de 30 jours calendaires

Le sprint comporte **plusieurs phases** : il démarre par la **réunion de planification du sprint**, se poursuit par **le sprint en soi** pendant lequel a lieu chaque jour la réunion quotidienne. En fin de sprint ont lieu **la revue de sprint** et enfin **le post-mortem du sprint**.

Il est important de préciser que le mot sprint traduit le gain de productivité de l'équipe induit par la mise en place de la méthode, et non le fait que l'équipe travaille beaucoup plus qu'avant. **Un des principes des méthodes agiles est en effet un rythme soutenable de façon indéfinie**.

iv. La réunion de planification du sprint

Elle se déroule avec le product owner et l'équipe complète.

Cette réunion doit durer au maximum 1 journée. On retrouve ici la notion de délai fixé qui est omniprésente en Scrum. Les temps sont volontairement courts afin de se concentrer sur l'objectif, se mettre au travail immédiatement et produire des résultats.

La réunion est découpée en 2 parties : la présentation du travail à faire et la planification du sprint.

Partie 1 -présentation du travail à faire : cette partie de la réunion occupe une demi-journée. Le **product owner** présente à **l'équipe** les besoins qui ont la plus haute importance dans le backlog produit. L'équipe a tout le loisir de lui poser des questions afin de clarifier les choses et d'affiner les premières estimations réalisées. Ce type de dialogue face à face est une des valeurs clés de scrum.

Une fois que l'équipe pense en savoir suffisamment sur le travail à effectuer, **elle sélectionne les besoins qu'elle pense pouvoir implémenter dans le délai imparti**. Elle s'engage ensuite auprès du product owner à faire de son mieux.

Les besoins sélectionnés constituent le backlog de sprint.

Le contenu du sprint est alors gelé et le product owner ne peut plus intervenir pendant toute la durée du sprint. Il peut par contre continuer à faire évoluer le backlog produit.

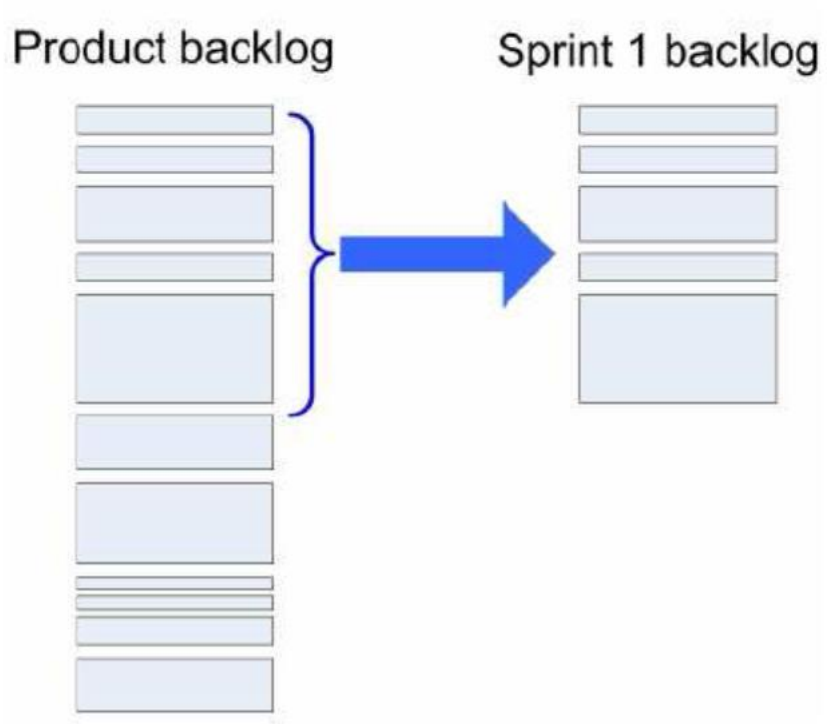


Figure 8 : constitution du backlog de sprint à partir du backlog produit.

Le backlog de sprint est un extrait du backlog produit.

Partie 2 -planification du sprint par l'équipe : Cette deuxième partie de la réunion dure aussi une demi-journée. Le product owner n'est ici pas présent. **L'équipe** est quant à elle présente au complet.

Les besoins sont divisés en tâches et chaque tâche est à son tour estimée afin d'avoir une estimation plus fine du temps nécessaire pour mener à bien le besoin. **La durée des tâches doit être de l'ordre de 0,5 jours à 2 jours maximum**. Des tâches plus longues relèvent plus du domaine d'un budget temps pour des tâches mal définies, donc auxquelles on n'a pas suffisamment pensé. C'est un indicateur qu'il faut se pencher plus en avant sur ces tâches. Des tâches de moins d'une demi-journée sont du domaine des tâches atomiques et deviennent compliquées à suivre, surtout en utilisant un outil tel que le tableau blanc.

8. La réunion quotidienne

Elle a lieu tous les jours à heure fixe le matin. Elle doit se dérouler en 15 minutes maximum.

Cela laisse environ 2 minutes par personne, ce qui est très court. Il faut donc se concentrer sur l'essentiel. Tous les participants sont debout et tournés vers le tableau blanc.

Chaque participant répond à 3 questions :

1. Qu'as-tu fait hier ?

2. Que fais-tu aujourd'hui ?

3. Quels sont les problèmes qui t'empêchent de progresser ?

Les objectifs de cette réunion sont :

- **La communication au sein de l'équipe** : chacun sait qui fait quoi dans l'équipe
- **La synchronisation du travail entre les membres de l'équipe**.
- **La découverte de problèmes à traiter grâce à la troisième question**. Cela permet d'organiser des réunions de travail spécifiques aux problèmes évoqués.
- **Motivation** : le sentiment d'appartenance à une équipe est renforcé et chaque participant subit le regard de ses pairs et ne veut pas les décevoir.

Il est nécessaire que les participants soient très spécifiques quant à leurs tâches. En effet, il ne suffit par exemple pas de dire « je corrige des bugs », il faut préciser de quels bugs il s'agit. L'information ne présente autrement aucun intérêt.

9. Les outils du sprint

- **Le tableau blanc :**

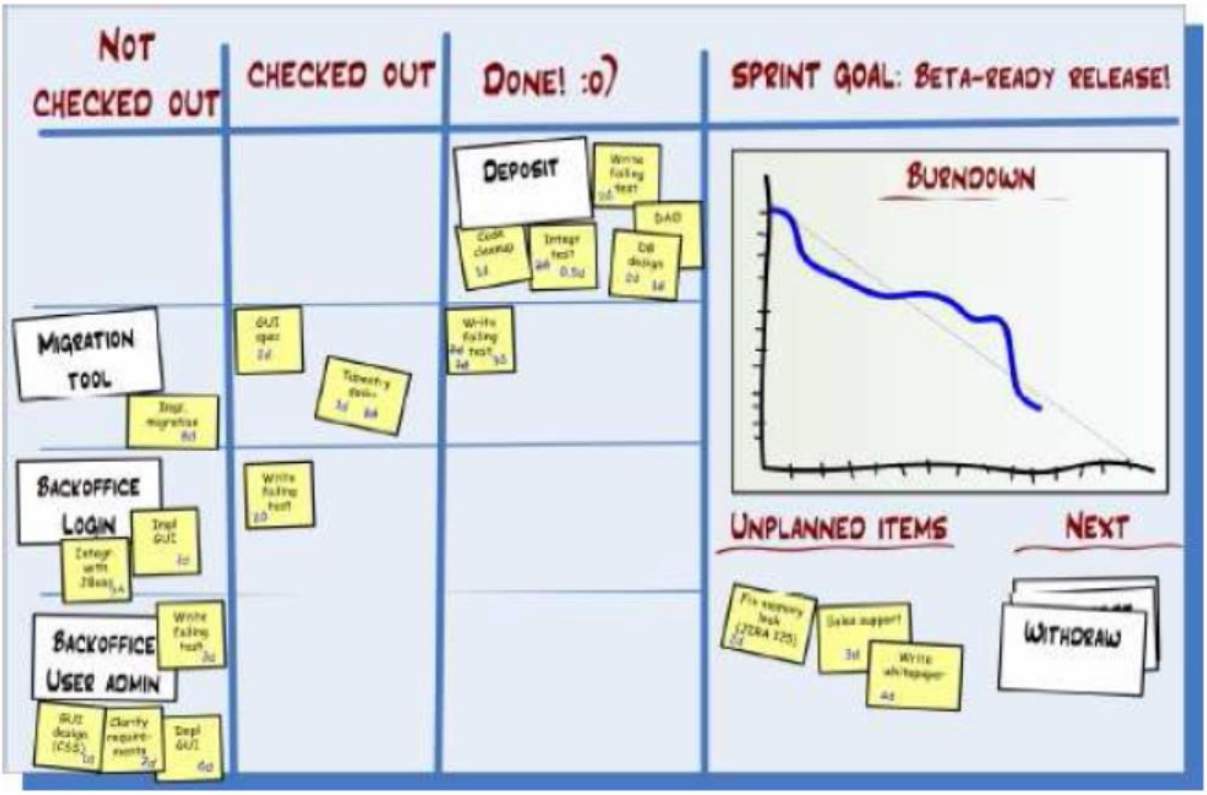


Figure 9 : le tableau blanc en cours de sprint.

Le tableau blanc est accroché dans la salle réservée à l'équipe. Il a l'avantage de servir de support aux réunions quotidiennes d'une part, et de montrer en permanence l'état du sprint d'autre part. Il est impossible d'ignorer ce qui se passe dans l'équipe.

Les post-its blancs représentent les besoins issus du backlog produit. Les post-its jaunes représentent la décomposition des besoins en tâches. Au début du sprint, les besoins découpés en tâches sont à gauche du tableau, classés de haut en bas par importance pour le product owner. Au fur et à mesure du sprint, les tâches se déplacent de la **gauche vers la dro**ite. En plus de l'aspect tactile et ludique de déplacer les post-its, l'équipe se voit progresser avec un indicateur physique. Il est possible de jauger d'un seul coup d'œil l'état du sprint.

A droite du tableau se trouvent :

- **Le burndown chart** sur lequel nous reviendrons un peu plus loin dans ce dossier.

- **La section « Next »**, ou en français « ensuite » : il s'agit des besoins qui seront traités si le sprint se passe mieux que prévu et que l'équipe a la possibilité de traiter plus de besoins.
- **La section « Unplanned items »** ou en français « Tâches non prévues » : il s'agit des tâches qui n'étaient pas prévues à la base et qui viennent s'ajouter à la charge de travail déjà existante. Il ne s'agit pas de nouveaux besoins demandés par le product owner, puisqu'il ne peut pas intervenir en cours de sprint, mais de tâches que l'équipe découvre au fur et à mesure de sa progression, donc en approfondissant sa connaissance du travail à accomplir.

- **Le Burndown chart :**

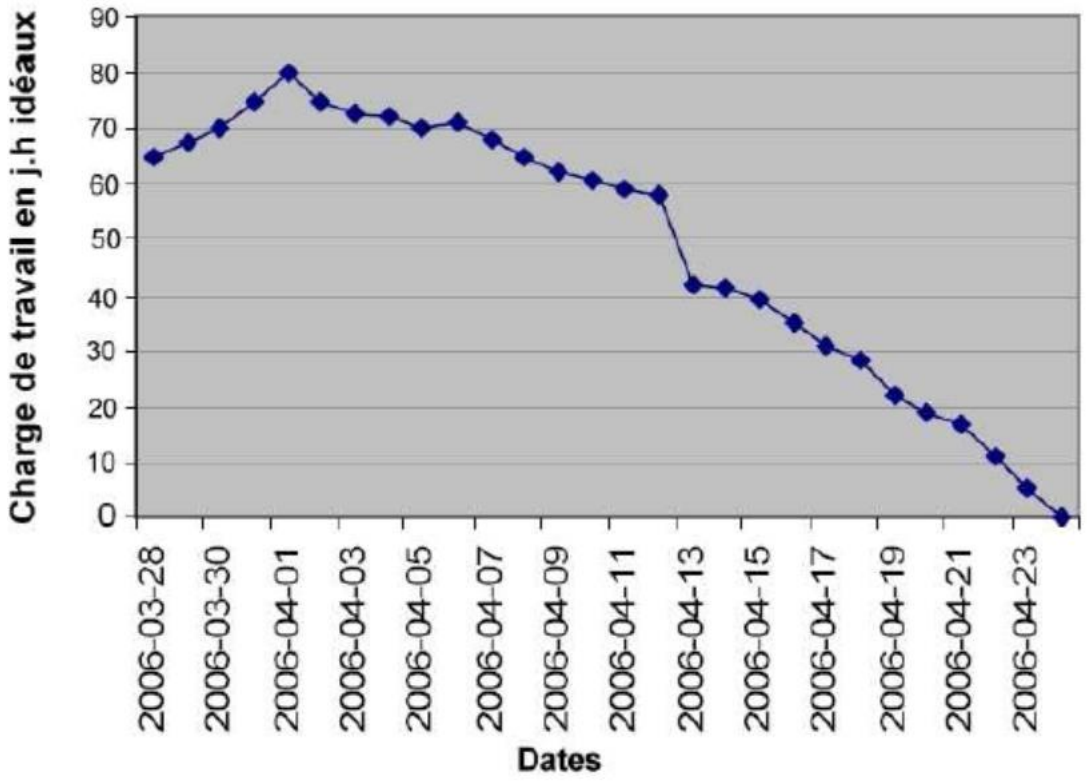


Figure 10 : exemple de Burndown chart.

C'est un graphique qui est mis à jour quotidiennement. **En abscisse se trouvent les dates et en ordonnée la charge de travail restante pour achever le sprint**. Chaque jour, à la fin de la réunion quotidienne, le scrummaster met à jour le graphique en comptant sur les post-its le nombre de jours de travail restant.

Ce graphique est particulièrement intéressant parce qu'il est très simple à appréhender d'une part, et parce que la courbe donne une **tendance sur la santé du sprint** d'autre part. Il est ainsi possible de prendre des **mesures correctives** rapidement : retirer des besoins du sprint si l'on se rend compte que la charge de travail ne permettra pas de finir à temps, ou au contraire d'en ajouter si l'on estime que le sprint sera fini trop tôt. Par extension il est possible de constituer le release burndown chart. Ce graphique sera abordé plus loin dans ce dossier.

10. La revue de sprint

Cette réunion se déroule en 4 heures maximum. Elle réunit **l'équipe**, le **product owner** et les **personnes** ayant un intérêt dans le projet et qui souhaitent y participer. Elle a pour objectif de présenter au product owner l'incrément logiciel produit pendant le sprint.

Les bénéfices tirés de cette réunion sont :

- L'équipe est valorisée pour le travail accompli.
- Communication entre l'équipe et le product owner qui donnera notamment son avis par rapport à ce qu'il voit.
- Communication vers le reste de l'entreprise : tout le monde a le loisir de savoir à quoi l'équipe de développement passe ses journées.
- Une telle réunion motive l'équipe à terminer réellement les tâches entamées. Là encore, le regard des autres compte.

11. Post-mortem du sprint

Cette réunion est tenue directement après la revue de sprint. Elle dure 4 heures au maximum. Les participants sont les membres de l'équipe de développement. L'objectif est d'améliorer le processus de développement pour que l'équipe fonctionne encore mieux dans le sprint suivant. Il est important que les idées viennent de l'équipe afin d'obtenir une adhésion collective.

Lors de cette réunion, chacun s'exprime sur :

• **Ce qui s'est bien passé** : si l'équipe avait la chance de recommencer le sprint depuis le premier jour, que referait-elle exactement de la même manière ?

• **Ce qui aurait pu se passer mieux** : si l'équipe avait la chance de recommencer le sprint depuis le premier jour, que devrait-elle faire de façon différente ? Cela permet à l'équipe d'apprendre de ses erreurs et de trouver des voies d'amélioration.

• **Idées d'amélioration** : il s'agit ici d'évoquer des idées concrètes pour améliorer la suite du développement.

Encore une fois, un tableau blanc peut être utilisé pour rassembler ces idées. Cela permet d'impliquer l'équipe entière. La décision sur les points d'amélioration sera également prise collectivement.

6) Planifier un projet basé sur scrum

f. Répartition en sprints et en releases

Un projet agile ne signifie pas un projet sans organisation... Il s'agit seulement d'un projet beaucoup moins basé sur les documents et qui ne suit **pas un plan traduit sur un réseau PERT ou un diagramme de Gantt**. Le minimum nécessaire pour planifier un projet avec Scrum est:

1. **Connaître la vision du produit** : savoir pourquoi le projet est entrepris et quel est le résultat souhaité.

2. **Avoir constitué le backlog produit**.

L'équipe se réunit ensuite pour procéder à de premières estimations de chaque élément présent dans le backlog produit. Ces estimations seront forcément imprécises mais elles ne tiennent pas lieu d'engagement. Elles permettent de **comparer les poids relatifs des besoins entre eux**.

Le backlog produit est ensuite réparti en sprints et en releases en tenant compte de la **vélocité de l'équipe**.

Backlog produit - Site marchand XY					
ID_Item	Titre	Importance	Estimation	Démonstration de la fonctionnalité	Commentaires
SPRINT 1					
1	Besoin 1	130	12	xxxxxxxx	xxxxxxxx
2	Besoin 2	120	9	xxxxxxxx	xxxxxxxx
3	Besoin 3	115	20	xxxxxxxx	xxxxxxxx
SPRINT 2					
4	Besoin 4	110	8	xxxxxxxx	xxxxxxxx
5	Besoin 5	100	20	xxxxxxxx	xxxxxxxx
6	Besoin 6	95	12	xxxxxxxx	xxxxxxxx
SPRINT 3					
7	Besoin 7	80	10	xxxxxxxx	xxxxxxxx
8	Besoin 8	70	8	xxxxxxxx	xxxxxxxx
9	Besoin 9	60	10	xxxxxxxx	xxxxxxxx
10	Besoin 10	40	14	xxxxxxxx	xxxxxxxx
SPRINT 4					
11	Besoin 11	35	4	xxxxxxxx	xxxxxxxx
12	Besoin 12	25	6	xxxxxxxx	xxxxxxxx
13	Besoin 13	10	7	xxxxxxxx	xxxxxxxx
14	Besoin 14	10	11	xxxxxxxx	xxxxxxxx
15	Besoin 15	10	3	xxxxxxxx	xxxxxxxx

Figure 11 : exemple de backlog produit découpé en sprints.

L'exemple suivant tient compte d'une équipe de 4 personnes, de sprints de 3 semaines et d'un facteur de concentration de 0,7. En 3 semaines, l'équipe peut donc réaliser

3 semaines*5 jours*4 personnes*0,7 concentrations = 42 jours idéaux.

Les sprints sont donc découpés de façon à ce que la somme des besoins n'excède pas 42. Chaque sprint ne dépasse pas 42 jours idéaux de travail. On se rend compte qu'il faudra 4 sprints de 3 semaines pour terminer le produit tel qu'il est connu au moment de l'établissement du backlog produit.

Si le product owner décide que le logiciel peut être mis en exploitation dès que les besoins d'importance supérieure à 60 sont implémentés, alors on en déduit que la première release du produit pourra avoir lieu après le sprint 3, soit 9 semaines après le début du développement.

Même si les estimations sont imprécises, cela donne une bonne première idée de la durée du projet, meilleure en tout cas que dans la plupart des cas où une date est imposée de façon arbitraire.

L'équipe s'engagera d'autant mieux sur ces objectifs que c'est elle qui a estimé les besoins et que l'objectif semble atteignable.

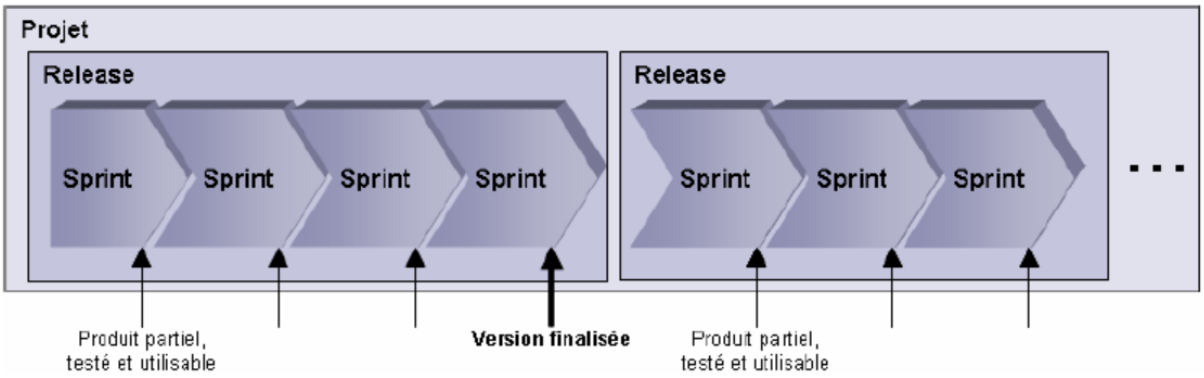


Figure 12 : découpage du projet en sprint et en releases.

Le découpage et le contenu des sprints sera à adapter à chaque fin de sprint, en fonction des résultats du sprint qui s'achève, des modifications apportées au backlog produit par le product owner, et de ce que le logiciel fourni en fin de sprint apprend au product owner et éventuellement aux utilisateurs.

g. Les indicateurs du projet

De la même façon qu'il existe un burdown chart au sein d'un sprint, il existe un graphique utilisé pour une release : **le release burndown chart**.

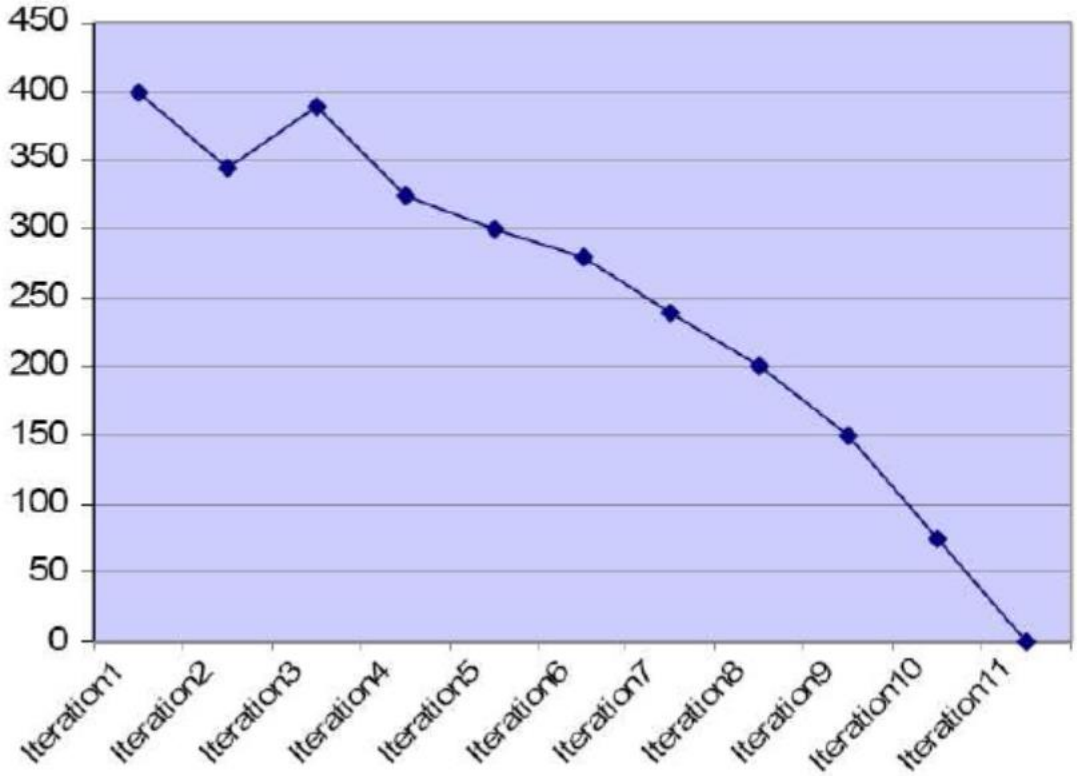


Figure 13 : release burndown chart, format standard

Le release burndown chart permet de se situer sur l'ensemble d'une release. Il peut prendre les deux formats représentés ci-dessus. Dans les deux cas, les **sprints sont en abscisse** et **la charge de travail restante en jours.homme idéaux en ordonnée**.

Remarques sur la figure 13 : on se rend compte que la charge de travail augmente au début de l'itération 3. Cela peut avoir deux causes :

- soit le product owner a ajouté des besoins dans le backlog produit
- soit l'équipe a ré-estimé les besoins listés dans le backlog produit et a revu ses estimations à la hausse.

La date prévisionnelle de fin de la release est donnée par la courbe représentant la tendance de la progression, lorsqu'elle coupe l'axe des abscisses.

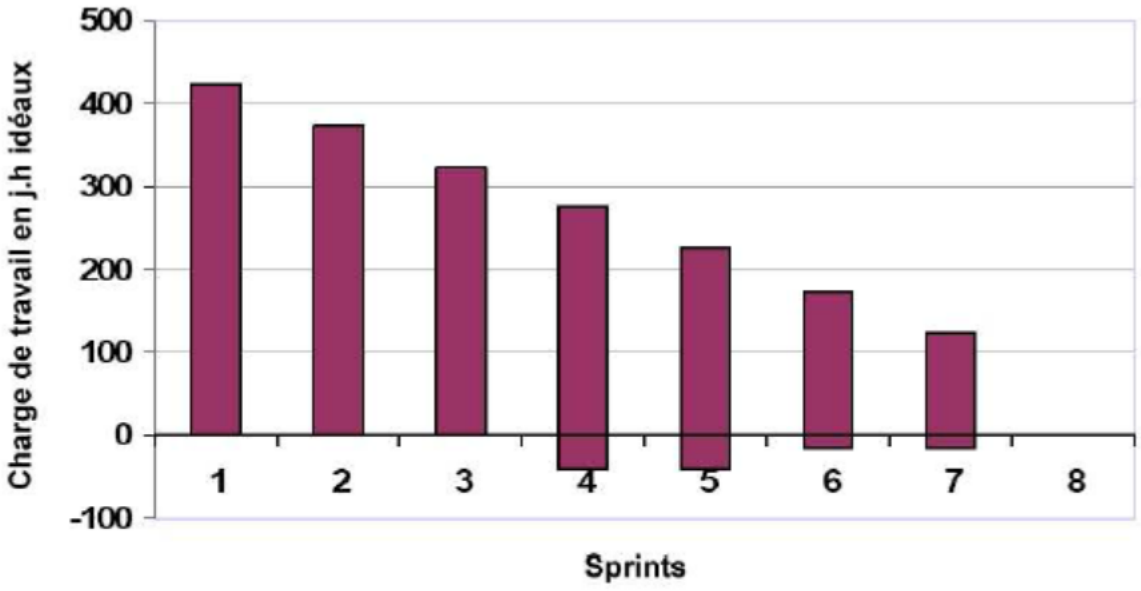


Figure 14 : Release burndown chart, forme alternative

Dans la forme alternative du release burndown chart, la modification dans l'envergure du projet est indiquée en dessous de l'axe des abscisses. Cela permet de distinguer ce qui est du fait de l'équipe -ralentissement ou travail non prévu à la base -et ce qui est du fait d'une modification du backlog produit.

Sur la figure 14, il est ainsi possible de constater que l'envergure du projet a augmenté à la fin du sprint 3 et a été réduite à la fin du sprint 5.

La date prévisionnelle de fin de release est donnée en dessinant 2 lignes de tendances, la première traversant les barres au dessus de l'axe des abscisses, la seconde traversant les barres situées en dessous.

7) Eléments importants à prendre en considération

h. Définition de « fini »

La notion de tâche finie peut varier énormément entre les membres d'une équipe. Aussi Scrum s'attache à ce que les équipes s'accordent sur une définition claire de ce qu'est une tâche finie. Il est conseillé d'avoir une définition standard. Par exemple : le code est relâché sur le logiciel de contrôle de sources et les tests unitaires après intégration sont tous passés.

i. La vélocité

La vélocité est la **mesure du travail** fait dans lequel chaque élément est pondéré en fonction de son estimation initiale.

La vélocité initiale est la somme de travail de tous les besoins sur lesquels l'équipe s'est engagée en début de sprint.

La vélocité constatée est la somme de travail de chaque besoin réellement traité et terminé par l'équipe en fin de sprint. Un besoin entamé mais pas terminé est considéré comme nul, puisque Scrum s'attache à démontrer des fonctionnalités terminées en fin de sprint.et ce qui a été réellement fait en fin de sprint

Figure 15 : comparaison entre ce qui était prévu en début de sprint

Les estimations initiales sont réalisées en **jours idéaux** ou en **points d'histoire**. En cas d'estimation en jours de travail idéaux, on introduira un **facteur de concentration** afin de prendre en considération les perturbations dues à l'environnement. Il est en général de **0,7**. Il s'agit finalement de prendre en considération la différence entre la charge de travail et le délai réel pour accomplir la tâche.

Exemple :

*Ainsi, si le sprint se déroule sur 2 semaines avec une équipe de 5 personnes, nous aurons 10*5 = 50* ***jours.homme*** *de charge de travail à disposition. En introduisant un facteur de concentration de 0,7 on obtient une vélocité estimée de 35 jours.homme. L'équipe peut donc s'engager sur des tâches pour une somme de 35 jours.homme.*

L'estimation en points d'histoire ne nécessite pas l'introduction d'un facteur de concentration.

Après quelques sprints, la vélocité estimée sera affinée en fonction de l'historique de l'équipe.

j. Techniques d'estimation

Les estimations des tâches ne sont pas réalisées de manière traditionnelle, c'est à dire décidées par le chef de projet. **C'est l'équipe qui estime elle-même les tâches qu'elle va devoir traiter**.

Une technique innovante et ludique consiste en **le jeu du poker planning**.

L'équipe se réunit et chacun dispose d'un jeu de 13 cartes. Sur chaque carte figure un nombre représentant des jours idéaux ou des points d'histoire, selon l'unité retenue. On remarquera la non linéarité des chiffres qui correspondent presque à une suite de Fibonacci. Cela est fait afin d'éviter les confusions dues aux estimations trop proches.

L'équipe passe en revue chaque besoin. Pour chacun d'eux, chaque membre sélectionne une carte et au signal du scrummaster, la retourne. Lorsque les estimations varient trop, chacun est amené à expliquer ce qui l'a amené à choisir cette estimation. Le besoin est ainsi clarifié. L'équipe recommence ensuite, jusqu'à ce que les estimations convergent.

Figure 16 : un jeu de carte pour le poker planning.

IV. Scrum et eXtreme Programming

Comme nous avons pu le constater jusqu'à présent, Scrum est une méthode de gestion de projet, et non d'ingénierie informatique. D'ailleurs, Scrum ne recommande pas de méthodes particulières d'ingénierie informatique, et c'est tout à fait normal puisqu'un des principes de Scrum est de donner le pouvoir de décision à l'équipe.

Un autre grand principe de Scrum est la transparence. En effet, afin de pouvoir réorienter la production, l'équipe de développement a besoin de savoir quotidiennement où elle se situe. Or pour arriver à cela, il est nécessaire d'intégrer son code au minimum une fois par jour afin de connaître chaque jour l'état du logiciel.

Force est de constater que ce n'est pas pratique courante et que ce n'est pas chose aisée. Il est par conséquent nécessaire de recourir à des **méthodes d'ingénierie logicielle particulières** pour y arriver.

Les méthodes décrites dans eXtreme Programming peuvent y aider. Scrum et eXtreme Programming sont d'ailleurs souvent associées. Il est à noter que Kent Beck, Ward Cunningham et Ron Jeffries, les créateurs d'eXtreme Programming sont aussi signataires du manifeste de l'agilité. Les 2 méthodes traitent de problématiques différentes et complémentaires et fonctionnent donc particulièrement bien ensemble.

Parmi les bonnes pratiques de l'eXtreme Programming, on retiendra particulièrement le recours à des langages objets permettant la souplesse nécessaire pour le refactoring fréquent du code, les méthodes de TDD1, et d'intégration continue qui permettent d'avoir des builds très fréquents du logiciel en cours de développement.

Def refractoring : Le refactoring consiste à modifier la structure d'un fichier source et si nécessaire à propager ces modifications dans les autres fichiers sources pour maintenir autant que possible l'intégrité du code existant.

8) Scrum pour de grandes équipes

On se rend compte qu'un des points faibles de la méthode décrite jusqu'à présent réside dans le fait qu'elle s'adresse à de petites équipes de 8 personnes

maximum. Il est dès lors possible de penser que cet obstacle est rédhibitoire. Mais il n'en est rien. Des mécanismes ont été mis en place qui permettent de gérer de grandes équipes.

Scrum a d'ailleurs déjà été utilisée maintes fois sur des projets plus importants. On trouve des exemples réussis avec des équipes de 20 personnes, de 40 personnes [et même de l'ordre de 300 personnes Jeff Sutherland fait aussi part de son expérience chez Google pour l'utilisation de Scrum sur le projet adWords avec une équipe de plus de 40 personnes réparties en 5 lieux différents.

k. Principes de base

Il est nécessaire de créer plusieurs équipes de taille standard pour Scrum, soit 8 personnes maximum. Chaque équipe va travailler sur **un backlog de sprint issu du même backlog produit**. Même avec plusieurs équipes Scrum, il y a **un et un seul product owner** et **un seul backlog produit**. Par contre, il y a **un backlog de sprint par équipe**.

On procédera alors de la manière suivante :

1. **Commencer avec une seule équipe de taille standard**.

2. **C'est cette première équipe qui va mettre en place l'infrastructure nécessaire à l'élargissement à plusieurs équipes**. Par exemple :

- un logiciel de contrôle de sources tels que SVN, Perforce ou Source Safe doit être mis en place,
- une architecture technique doit être construite de façon à ce que le travail soit proprement divisé entre les équipes ;
- si l'équipe est dispersée géographiquement, alors une connexion à large bande passante doit être mise en place pour le partage de code et permettre les daily builds, des moyens de communication avec les équipes distantes doivent être installés -messageries instantanées, webcams...

Il faut néanmoins conserver une valeur clé de Scrum : à la fin d'un sprint, l'équipe livre un incrément logiciel possédant de la valeur métier en même temps que la construction de l'infrastructure. Il est en effet indispensable que le product owner sente le projet progresser pour qu'il reste impliqué et ne pas

créer d'inquiétude injustifiée. Pendant cette phase, il faut s'assurer que les tâches non fonctionnelles possèdent la priorité la plus haute, sinon l'élargissement sera inefficace, voire impossible.

3. **Après quelques sprints avec une seule équipe, lorsque tous les pré requis non fonctionnels ont été menés à bien, alors on crée plusieurs équipes qui contiennent chacune un membre de l'équipe d'origine**. Il est bien entendu nécessaire de créer des équipes dont le travail est indépendant de celui des autres équipes.

Il est donc possible d'atteindre « assez facilement » des équipes de 8*8 = 64 personnes. On peut ensuite imaginer reproduire le processus d'essaimage pour agrandir encore l'équipe dans une nouvelle étape.

2. **Adaptation du déroulement d'un sprint**

Tous les jours, chaque équipe procède à une réunion quotidienne. **Elle est suivie d'une réunion appelée Scrum des scrums.** Lors de cette nouvelle réunion, un membre de chaque équipe y participe et représente son équipe pour la synchronisation de toutes les équipes entre elles. Elle prend le même format que la réunion quotidienne et dure 30 minutes.

L'agenda de la réunion est le suivant :

1. **Chacun décrit ce que son équipe a fait la veille, ce qu'elle prévoit d'accomplir ce jour-là et les problèmes qu'elle rencontre.**

2. **Discussion des problèmes inter-équipes, comme les problèmes d'intégration par exemple.**

Il est possible de réorganiser les équipes entre les sprints, bien que ce ne soit pas conseillé de le faire trop fréquemment. Il est en effet important d'obtenir une cohésion d'équipe pour atteindre une productivité optimale. Or, changer la composition des équipes empêche leur agrégation

Il est possible d'introduire un nouveau rôle : celui de **team lead**, qui est en quelque sorte **le scrummaster des scrummasters**. Il n'a aucune équipe en charge et est responsable des problèmes inter équipes : allocation des personnes dans les équipes, organisation du planning des réunions quotidiennes pour toutes les équipes...

3. Travailler avec des équipes géographiquement dispersées

Autant le dire tout de suite, ce n'est pas l'idéal : une grosse partie de Scrum -de même que pour eXtreme Programming -est basée sur la proximité géographique des membres de l'équipe.

On veillera donc à favoriser la communication entre les équipes de façon à ce que l'éloignement soit le moins handicapant possible. Il faudra par exemple mettre en place une connexion avec une bande passante importante qui permet d'utiliser les outils suivants :

• Webcams et casques avec micros à chaque poste.

• Salles de conférences avec matériel de visioconférence, ordinateurs toujours allumés avec logiciels de travail collaboratif...

• Fenêtres virtuelles : grands écrans à chaque emplacement géographique. Ces fenêtres permettent de voir en permanence ce qui se passe dans l'autre endroit. Elle permet de voir qui est là, qui est à son poste, qui parle à qui... Les emplacements seront par exemple la machine à café, pour la convivialité, et la salle de travail.

• Visites régulières des personnes d'un lieu à l'autre : cela permet de faire connaissance et de savoir à qui l'on a affaire. On n'a pas les mêmes relations selon que l'on a déjà rencontré ses interlocuteurs ou non.

• Outil de gestion des besoins et des tâches permettant de voir qui travaille sur quoi.

4. Travailler avec des équipes off-shore

Il est aisé d'imaginer que le problème augmente encore en cas de décalages horaires dus à des équipes situées dans des fuseaux horaires différents. Cela peut arriver en cas de sous-traitance de tout ou partie du projet en off-shore.

Les mêmes outils que ceux évoqués précédemment pourront être utilisés.

On sera confronté en plus à un problème culturel, et particulièrement en Asie. En effet, en Scrum, les équipes sont autogérées et n'obéissent pas à un décideur. Si cette notion peut être difficile à assimiler pour une équipe en

Europe, elle le sera encore plus en Asie où la culture du décideur-exécutant est amplifiée.

Il est par conséquent conseillé de faire appel à des sous-traitants possédant déjà la culture de l'agilité.

Il est à noter qu'en Inde, les sociétés sont sensibles à la demande d'agilité. Cela se traduit par l'existence de conférences sur les méthodologies agiles depuis 2005.

V. LES OUTILS

Les puristes de Scrum recommandent : « **Do the simplest thing possible** », c'est-à-dire d'avoir recours à la simplicité. Il apparaît cependant qu'il n'est pas possible d'utiliser les mêmes outils avec une équipe de 8 personnes situées dans la même pièce et une équipe de 50 personnes disséminées dans plusieurs lieux.

9) Les outils standards

Les outils recommandés sont le tableur, des tableaux blancs et des post-its.

l. Le tableau blanc et les post-its

Il s'agit ici de gérer le backlog produit et le backlog de sprint grâce à un tableur et d'accrocher un grand tableau blanc, de le diviser en colonnes et de coller des post-its représentant les besoins et les tâches du backlog de sprint. Un exemple de tableau blanc peut être consulté figure 9.

12. Avantages

• Outil participatif : chacun peut venir déplacer son post-it et s'arrêter devant le tableau pour réfléchir ou juste voir ce qu'il reste à faire.

• Aide à la communication dans l'équipe lors des réunions quotidiennes. C'est un support pour les questions quotidiennes dont les réponses sont étayées par les tâches affichées.

• L'équipe se voit progresser. Les post-its passent de la gauche à la droite du tableau. C'est un facteur de motivation.

13. Inconvénients

• L'expérience n'est pas capitalisée : les post-its du sprint précédent sont enlevés. Il faut donc consigner manuellement dans un document l'expérience acquise, en terme d'exactitude des évaluations par exemple.

• Les post-its se décollent. Il est fortement conseillé de prendre régulièrement des photos du tableau…

• Les courbes de progression sont à faire à la main.

• Le reporting au client et à sa hiérarchie est à faire manuellement. En cas de confiance totale, alors cela ne pose pas de souci : l'équipe s'engage au début du sprint et réalise une démonstration à la fin. Mais ce n'est souvent pas le cas.

10) Les outils dédiés au management de projets agiles

Les résultats d'enquête montrent que les logiciels dédiés à la gestion de projets agiles ayant la base installée la plus importante sont V1 de VersionOne, Rally de Rally Software, et ScrumWorks de Danube Software pour les logiciels commerciaux, et XPlanner dans le monde open source,IceScrum sera aussi étudié car il est souvent évoqué sur les blogs et dans les discussions sur les forums.

m. Les outils commerciaux

14. V1 de VersionOne

http://www.versionone.com/

Ce logiciel de l'éditeur VersionOne apparaît comme le leader des logiciels spécialisés dans la gestion des projets agiles ou APM – Agile Project Management Il existe en 2 versions : team et enterprise.

La différence réside dans le nombre d'équipes gérées. Alors que la version team ne gère qu'une seule équipe, la version enterprise en gère un nombre illimité. L'intégration avec d'autres outils est possible dès la version team.

Il est possible de faire héberger la solution chez VersionOne ou de l'installer sur site. Les besoins sont alors les suivants : Microsoft Internet Information Server 5 ou 6 avec ASP.net, Miscrosoft SQL Server 2005. L'application se présente sous la forme d'un client web.

Les points forts :

- Intégration avec des outils de développement, d'intégration continue et de tests. Cela permet de rassembler toutes les informations en un seul endroit :
- Outils de développement : Visual Studio et Eclipse. Les tâches de VersionOne apparaissent dans les IDE – Integrated Development

Environment. Les développeurs peuvent mettre à jour l'état de leurs tâches sans quitter leur environnement de développement.

- Outils de contrôle de sources : SVN et Microsoft Team Foundation Server. Permet de mettre en relation des modifications de code et des tâches.
- Outils de bug tracking : Bugzilla et Jira. Permet d'importer des bugs en provenance de ces logiciels et de les mettre à jour à partir de VersionOne.
- Outils d'intégration continue : CruiseControl et Microsoft Team Foundation Build.
- Intégration avec des outils de gestion de test : HP QuickTestPro et FitNesse.
- V1 reprend des vues classiques de la gestion de projet avec Scrum, et notamment la vue du type « tableau blanc ».

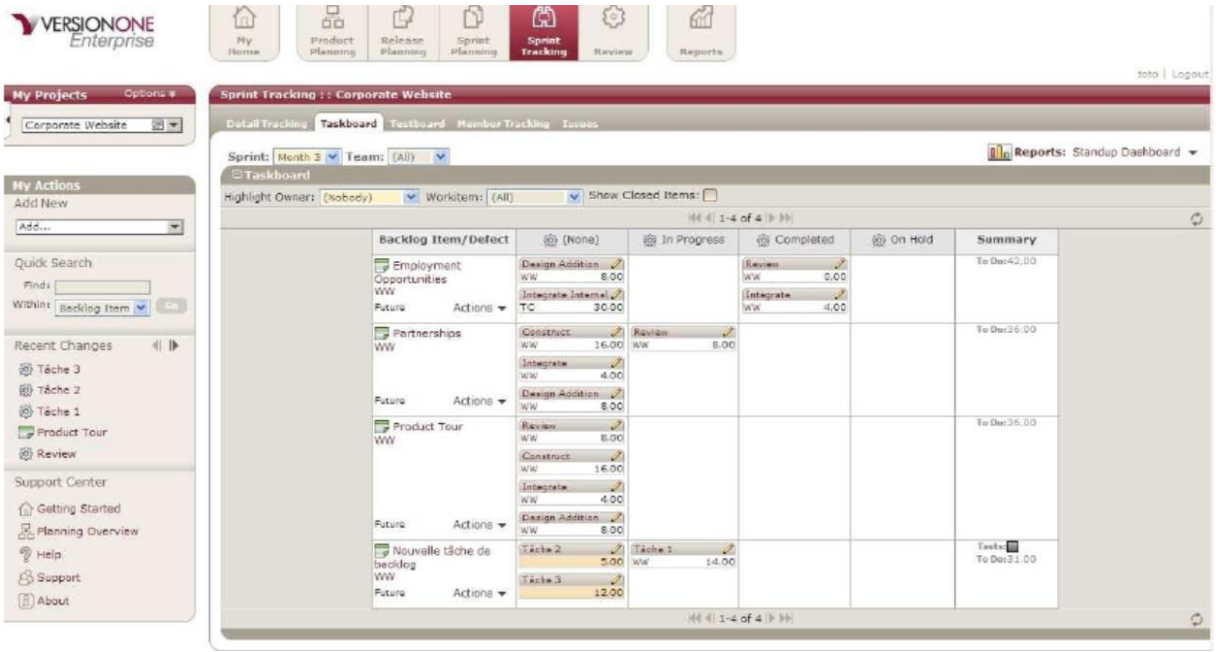


Figure 17 : vue de type « tableau blanc » dans V1

Les points faibles :

• Difficile à appréhender : difficile d'accès de par sa richesse fonctionnelle.

15. Rally de Rally Software

www.rallydev.com

Ce logiciel édité par Rally Software existe en deux versions : Rally Community Edition et Rally

Enterprise Edition.

La première est gratuite pour une équipe avec un maximum de 10 utilisateurs. La seconde est illimitée en terme d'équipes et d'utilisateurs. Elle coûte 35 USD par utilisateur et par mois. En version Enterprise, les coûts sont donc quasiment les mêmes que pour V1. Il est à noter que l'intégration de l'outil avec des environnements de développement et d'autres logiciels n'est possible qu'avec la version payante. Il apparaît clairement que la version gratuite permet d'essayer le logiciel en vue de passer à la version payante.

La solution peut être hébergée chez Rally Software – Rally on demand -ou installée sur site – Rally on premise. Cette dernière solution utilise VMware pour fonctionner. Il est par conséquent nécessaire de l'installer sur une machine avec un serveur VMware. L'environnement virtualisé faisant tourner Rally fonctionne avec : un système linux, JBoss en tant que serveur d'applications, Oracle 10g comme base de données et Oracle TopLink 10g pour le mapping base de données-objets. L'application se présente sous la forme d'un client web.

Les points forts :

- Intégration avec différents outils :
- Développement : Rally offre une intégration avec les IDE Eclipse et Visual Studio.
- Contrôle de source : Rally gère Subversion et Microsoft Team Foundation Server.
- Bug tracking : Bugzilla et FitNesse.
- Intégration continue : Hudson et CruiseControl, ainsi que Ant

L'organisation semble d'emblée plus claire que pour V1. La répartition des besoins en sprints et releases est très bien faite et intuitive : tout se gère grâce au drag and drop des besoins. Les estimations sont alors mises à jour.

• La terminologie Scrum est bien respectée, notamment avec la notion de vélocité et l'expression de la taille des besoins en points d'histoire.

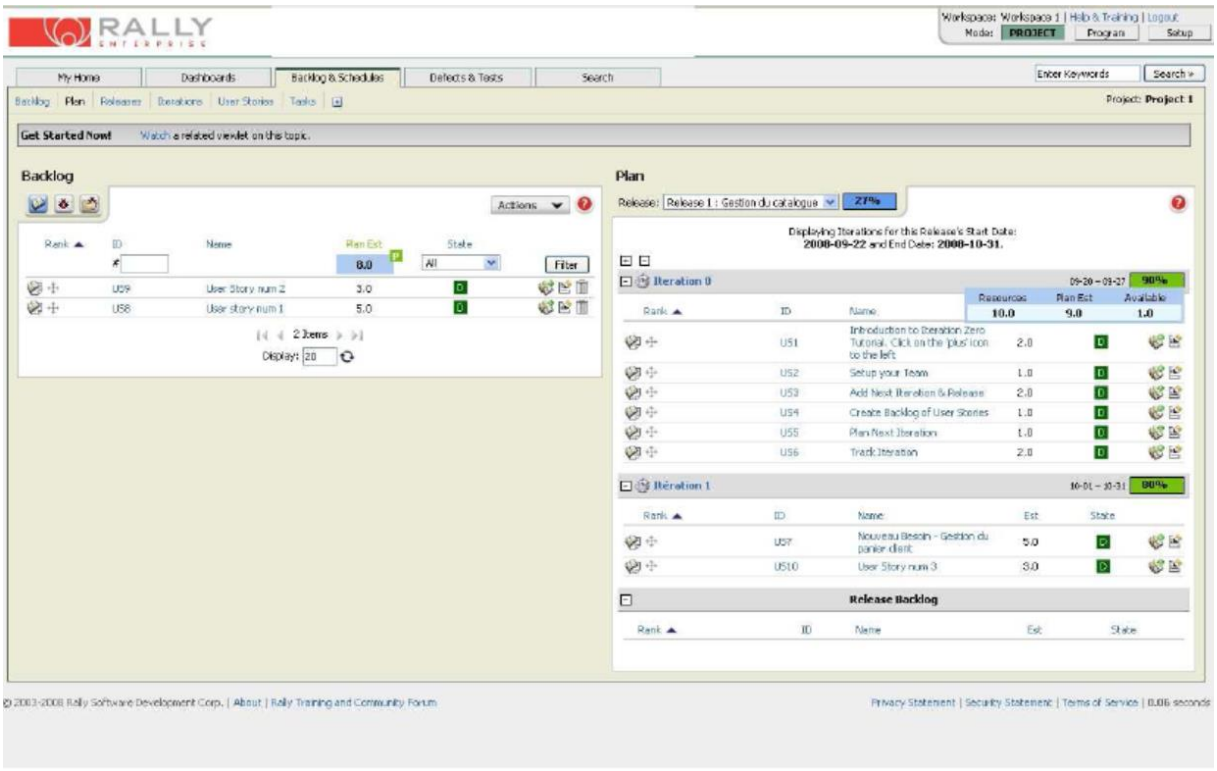


Figure 18 : répartition des besoins du backlog produit (à gauche) dans les différents sprints qui composent une release. La charge de travail estimée est mise à jour.

16. ScrumWorks de Danube Software

http://www.danube.com/

ScrumWorks existe en 2 versions : ScrumWorks basic et ScrumWorks Pro. ScrumWorks basic est gratuit

Les points forts :

• Intégration avec des outils de bug tracking : JIRA et Bugzilla.

• Possède une vue « tableau blanc »

• Donne de l'importance au product owner en lui donnant la possibilité d'indiquer la valeur ajoutée de chaque besoin et en calculant leur ROI1.

Les points faibles :

• Moins de possibilités d'intégrations que V1 et Rally, notamment pour les IDE. ScrumWorks est utilisable à partir d'un client lourd installé et d'un client web.

L'hébergement n'est pas possible. Il est par conséquent nécessaire d'installer un serveur sur site sur un système Windows XP, Vista ou Server 2003, Mac OS ou Linux. ScrumWorks fonctionne par défaut avec la base de données Hypersonic, mais peut être utilisé avec une base de données MySQL.

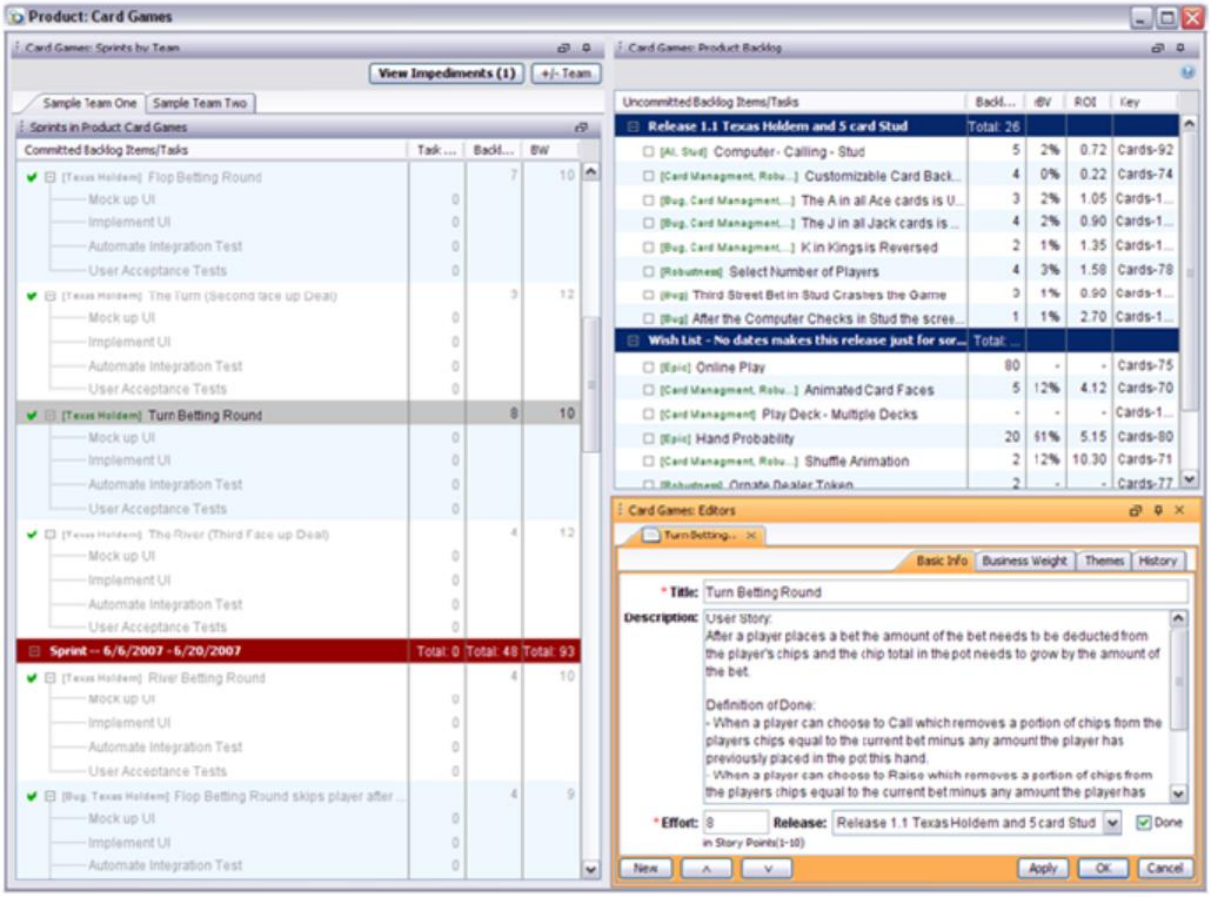


Figure 19 : client lourd de ScrumWorks. Les tâches du backlog réparties en releases sont à droite. Les sprint sont à gauche.

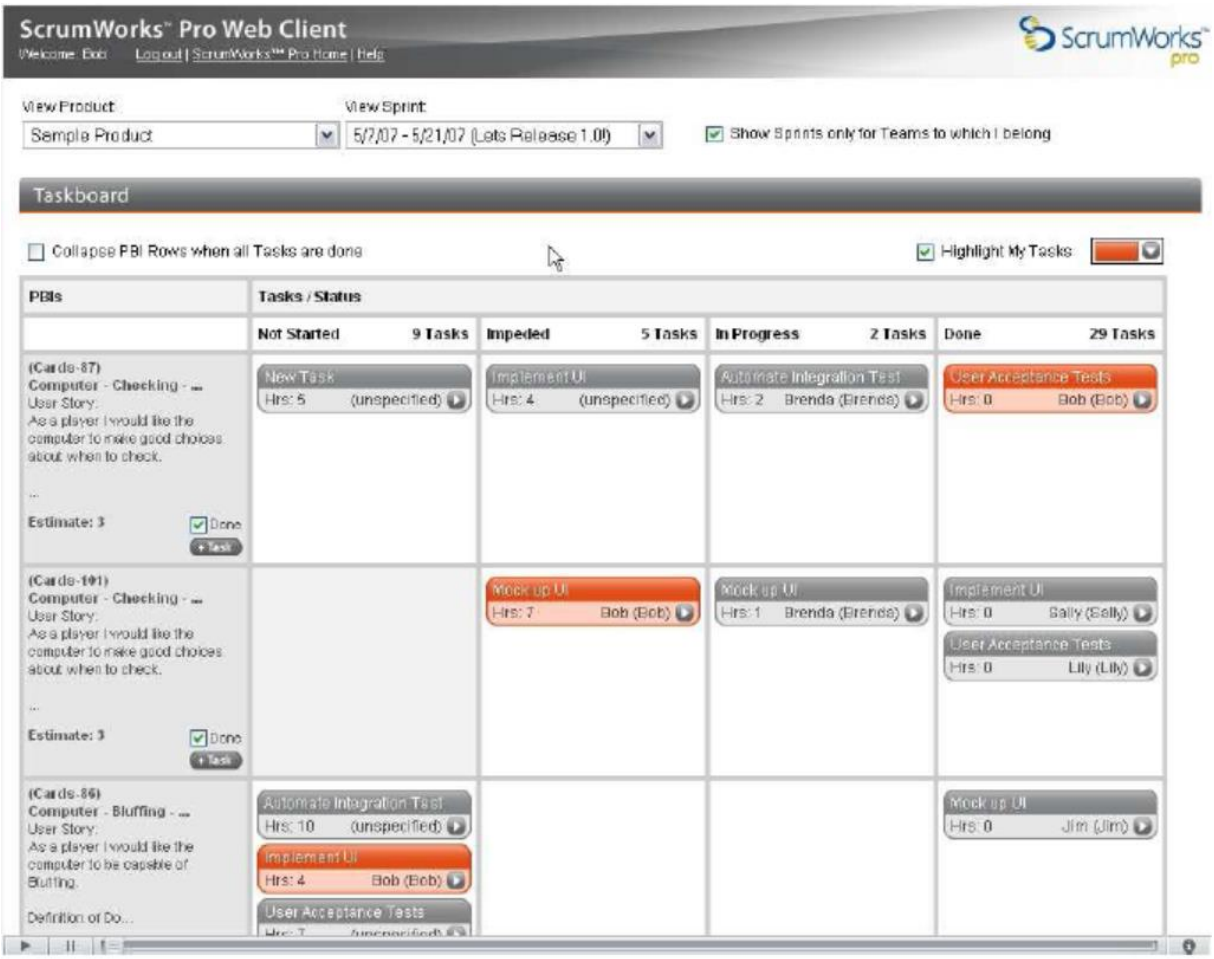


Figure 20 : client web de ScrumWorks. Le tableau des tâches. Source

n. Les outils open source

17. XPlanner

http://xplanner.org/

XPlanner est un outil open source très populaire dédié à la base à eXtreme Programming, et qui s'adapte bien à Scrum. Il permet notamment de générer des burndown charts. XPlanner se présente sous la forme d'un client lourd installé. Il nécessite un serveur Apache avec Tomcat et une base de données MySQL.

Avantages :

- Outil très complet en comparaison à d'autres outils open source très amateurs : reporting de temps, analyses graphiques, notifications par e-mail, API SOAP permettant sa customisation et son extension.

- Des graphiques normalement dédiés à Scrum comme le burndown chart sont disponibles.

Inconvénients :

- Pas de notion de releases, uniquement des itérations.

18. IceScrum

IceScrum est en développement actif. Sa dernière release date de mi-septembre 2008 et la précédente de fin juillet 2008. Il intègre des fonctionnalités intéressantes comme le planning poker, ou les estimations du type Fibonacci.

IceScrum est une application web en java fonctionnant avec un serveur Tomcat, et une base de données supportée par Hibernate, par exemple MySQL.

Avantages :

- Des fonctionnalités originales comme le planning poker.
- Un développement actif.
- Implémente correctement les concepts de projet, releases et sprints.
- Très ergonomique : déplacement des besoins et tâches grâce au drag and drop.

Inconvénients :

- Peu de documentation… Des tutoriaux créés par les utilisateurs sont toutefois
- disponibles.
- Parfois difficile à utiliser : interface non intuitive en dehors de la création d'une
- release.

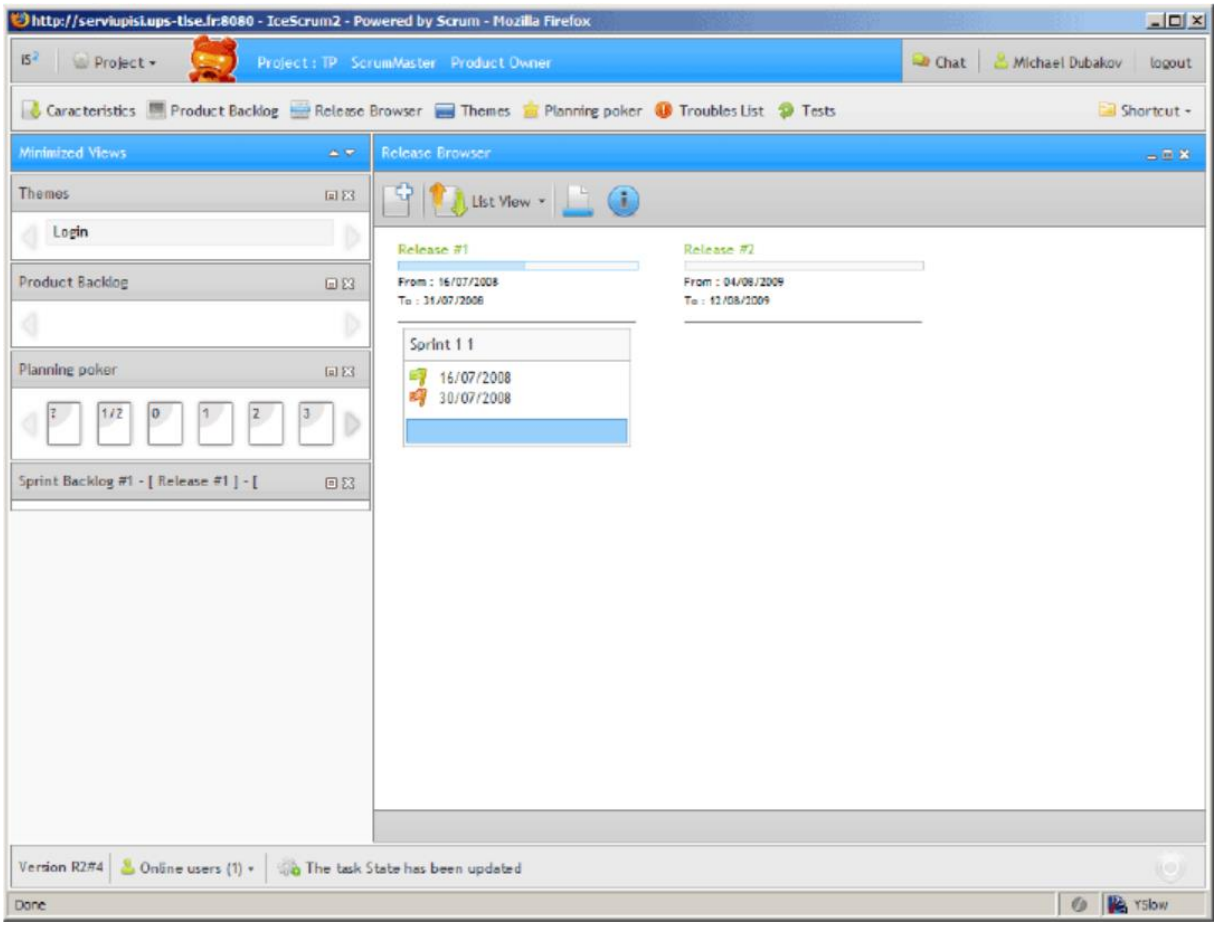
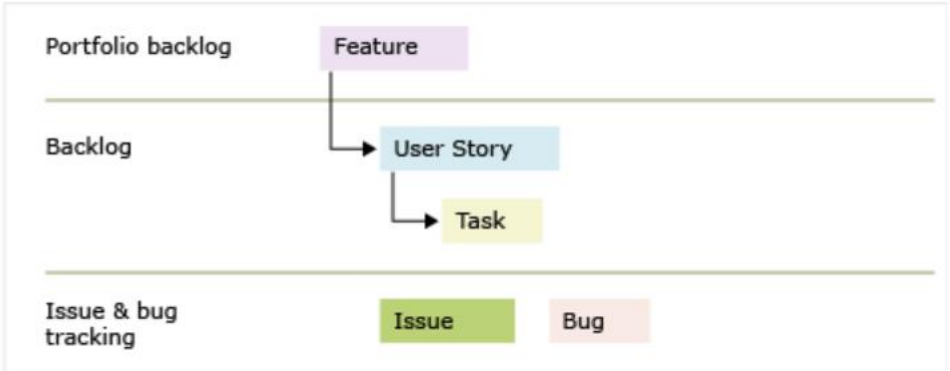


Figure 21 : releases et sprints dans IceScrum.

19. Solutions Framework (MSF) for Agile Software Development

http://msdn.microsoft.com/en-us/library/dd380647.aspx



o. Les autres outils

20. Ingénierie logicielle

Etant donné qu'il est nécessaire d'ajouter à Scrum une méthodologie d'ingénierie logicielle, on pourra utiliser les outils suivants :

- Contrôle de source : Subversion ou Perforce
- Framework de tests unitaires : JUnit sous Eclipse ou un framework de la famille XUnit selon le langage utilisé.
- Un outil d'intégration continue, tel que Maven, CruiseControl ou AntHill.
- Un CASE tool orienté UML avec forward et reverse engineering. Il sera
- principalement utile pour générer des diagrammes à partir du code.

11) Synthèse

Bien que les fondateurs de Scrum insistent sur l'utilisation d'outils simples servant aussi bien à structurer le projet qu'à la communication, il est facile de comprendre les avantages liés à l'utilisation d'outils de gestion de projets dédiés Scrum, d'autant plus qu'ils permettent la communication de plusieurs outils entre eux tels que des IDE, des outils d'intégration continue…

Il ne faudra pas perdre de vue qu'un outil ne remplace pas une méthode et qu'il est possible d'en dévier en se concentrant trop sur l'outil. L'outil doit servir la méthode et non distraire l'équipe.

Quand l'introduction d'un outil se révèlera indispensable, il faudra à mon sens en choisir un permettant l'intégration des autres outils existant dans l'entreprise. La connectivité de V1 et de Rally est en ce sens très intéressante. La reproduction des vues standards de Scrum telles que la vue du tableau des tâches est importante, particulièrement si l'équipe a été habituée à la gestion traditionnelle de Scrum avec un tableau blanc et des post-its.

VI. PERSPECTIVES

12) Evolutions et critiques de la méthode

Il semble que la méthode n'ait subi aucune évolution depuis sa création. D'autre part, il ne semble pas exister de critique de la méthode, au contraire d'eXtreme Programming qui est un peu plus vieille que Scrum débuts officiels fin 1999 – et qui possède bon nombre de détracteurs. Un livre est même sorti qui dénonce spécialement les manques de la méthode1

Quelles en sont les raisons ?

Est-ce parce que la méthode est suffisamment flexible pour s'adapter aux habitudes maisons ? En effet, en regardant la figure 2 page 5, on peut s'apercevoir que finalement seule la fréquence des itérations est clairement indiquée, mais pas les livrables associés2, qui sont laissés à la discrétion de l'équipe.

Il se trouve que Scrum devient un phénomène de mode à son tour. Va-t-on dans ce cas seulement maintenant voir apparaître des critiques de la méthode ?

Il est aussi légitime de se demander si les sociétés qui affirment faire du Scrum en font réellement. Une étude datant de juillet 2008 sur l'implantation des méthodes agiles en entreprise et sur les outils utilisés dans les projets agiles révèlent en effet que Microsoft Project est utilisé dans plus de 24% des cas ce qui semble contradictoire avec la mise en pratique de méthodes agiles et de Scrum en particulier.

A mon sens, les principales critiques que l'on peut formuler sont les suivantes :

- Dépendances entre tâches

Scrum ne prend pas en considération la dépendance entre les tâches. On peut alors arriver à la situation où un besoin est faisable en charge de travail mais pas en délai.

- Polyvalence des programmeurs

Scrum considère que chaque membre de l'équipe est totalement polyvalent et peut faire n'importe quelle tâche du sprint. Scrum recommande d'ailleurs qu'aucun nom ne soit associé aux tâches au début du sprint. La communication dans l'équipe est censée permettre un partage rapide des connaissances et favoriser la polyvalence de chacun. Or les programmeurs dans une équipe sont très rarement polyvalents.

- Productivité

Scrum considère que chacun dans l'équipe a une productivité égale, ce qui n'est jamais le cas.

- Maturité

Les membres de l'équipe doivent être de maturité égale. De trop grandes disparités pourraient provoquer l'apparition spontanée d'une hiérarchie naturelle, ce qui mettrait à mal la méthode.

13) Enjeux de la méthode

Scrum est à la base destinée à des équipes restreintes, de 8 personnes maximum. Les mécanismes nécessaires à son utilisation pour de plus grosses équipes existent. Cependant, d'après des études, il semble que Scrum soit encore utilisée majoritairement sur de petits projets, c'est-à-dire ceux qui ont le plus de chance de succès quelle que soit la méthode employée. La tendance semble cependant être à l'adoption de la méthode pour des projets moyens de 50 personnes, voire de gros projets de plus de 200 personnes

En ce qui concerne les équipes, le pouvoir de décision et d'autogestion qui leur est conféré nécessite une certaine maturité de la part des participants. Il semble en effet nécessaire que les membres aient chacun l'expérience de plusieurs projets.

Quant à l'adoption de la méthode en entreprise, elle remet en cause la hiérarchie traditionnelle et bouleverse les habitudes. Elle s'attaque donc à ce qui est le plus difficile à changer : les mentalités. Elle paraît plus facilement implantable dans de petites structures qui sont obligées d'être réactives pour prendre des parts de marché, voire survivre, que dans des entreprises possédant plus de niveaux hiérarchiques.

14) Scrum et les contrats

p. *Les contrats au forfait et en régie*

Les contrats à coût fixe et à contenu figé tels qu'on les connaît actuellement pour des réponses à des appels d'offre ou pour des projets au forfait ne sont pas adaptés du tout à Scrum. En effet, dans ce type de contrat, le cahier des charges fonctionnel est annexé et sert de référence en cas de litige entre le client et le fournisseur. Or, avec Scrum, le contenu du logiciel final n'est pas connu à l'avance.

L'équipe de développement s'engage seulement à faire de son mieux pour répondre aux besoins du client… Par conséquent, actuellement, seuls les contrats en régie permettent de développer des projets agiles

Des solutions sont pourtant esquissées et des méthodes comme UP recommandent par exemple deux phases de contrat, chacune contenant plusieurs itérations à délais fixes.

La phase 1 du contrat, sans engagement de contenu, a pour objectif, tout en produisant du logiciel, de mieux comprendre la demande du client. Elle permet alors de produire des estimations de bonne qualité

La phase 2 qui peut alors prendre la forme d'un contrat standard au forfait.

q. *Les contrats agiles*

Il est possible d'aller encore plus loin et de mettre en place un autre type de contrat : les contrats agiles.Pour ce type de contrat, il faut bien entendu que le client soit convaincu des bienfaits du développement agile, et souhaite le faire. Néanmoins, il sera confronté à ses direction juridique et financière qui, elles, souhaiteront se couvrir avec des contrats autres que des contrats de régie.

Il peut donc lancer un appel d'offre agile dans lequel il préconise l'utilisation de méthodes agiles. Il donne sa vision du produit et ses souhaits à ce moment. Le prix est fixé par le client et ne sera donc pas un élément de décision. Le périmètre du projet est la variable d'ajustement.

Dans ce cas, l'examen des réponses à l'appel d'offre doit se faire en fonction de :

- La pertinence de la réponse

- La méthode agile préconisée
- La composition de l'équipe qui travaillera sur le projet, en n'oubliant pas une liste détaillée des projets sur lesquels chaque membre a travaillé par le passé.

Les clauses du contrat doivent aussi être adaptées. Le client pourra notamment ajouter les clauses suivantes:

- Le client recevra un logiciel qu'il pourra tester à la fin de chaque itération.
- Chaque itération a une durée fixe de 30 jours calendaires.
- Le client définira les priorités parmi les besoins contenus dans le backlog produit.
- Le client pourra arrêter le contrat à la fin de chaque itération. Cette clause est importante puisqu'il ne pourra plus se servir du cahier des charges fonctionnel annexé au contrat en cas de problème.
- Le client peut changer d'avis sur le contenu fonctionnel entre les itérations.

Le périmètre initial du projet, sous la forme du backlog produit, pourra être réalisé par le client en collaboration avec son fournisseur.

VII. Scrum et CMMi

CMMi -Capability Maturity Model Integration – est un modèle développé par le SEI – Software Engineering Institute -permettant d'évaluer **la maturité des sociétés développant des logiciels** et de les classer en 5 niveaux de maturité, le niveau 1 étant le niveau initial que tout le monde possède par défaut. CMMi vise à **l'industrialisation des processus afin de pouvoir reproduire tous les projets avec le même niveau de qualité**. Il est important pour des entreprises d'être certifiée CMMi puisque certains appels d'offres ne concernent que les entreprises certifiées CMMi niveau 3.

Est-il possible de développer des projets avec Scrum dans une entreprise certifiée CMMi ? Scrum et CMMi sont a priori totalement opposées. CMMi est souvent associé à la bureaucratie et à la rigidité. Or il n'en est rien. En effet, CMMi ne donne qu'un cadre pour arriver à la maturité mais pas les moyens pour y parvenir. CMMi définit le quoi et pas le comment. Par conséquent, les entreprises qui utilisaient la méthode en cascade pour leurs projets avant leur certification CMMi l'ont utilisée pour mettre en place les éléments permettant la certification CMMi.

Sutherland affirme même que Scrum, appliqué dans une société certifiée CMMi niveau 5 permet d'obtenir des résultats encore meilleurs qu'avec Scrum seule ou CMMi seule : productivité doublée, réduction de 40% des défauts…

VIII. Scrum et ITIL

ITIL -Information Technology Infrastructure Library – est un ensemble de bonnes pratiques pour la gestion d'un système d'information. ITIL est actuellement assez répandu. Est-il possible de faire coexister Scrum et ITIL au sein de la même entreprise, c'est à dire de développer des projets en utilisant Scrum dans une DSI mettant en place ITIL ?

La question peut paraître surprenante étant donné qu'ITIL se focalise sur la gestion de la production et non les études. Il existe cependant des liens entre les deux au niveau des processus suivant : gestion des incidents, gestion des problèmes et gestion des changements.

Au niveau de la gestion des incidents, l'équipe peut être perturbée lorsque le centre de service ne sait pas résoudre un incident nouveau et fait appel à elle. Il faudra alors le résoudre toute affaire cessante. Ce type d'interruption peut être anticipé dans la vélocité. C'est en revanche contraire au principe qui spécifie que le scrummaster protège son équipe des perturbations extérieures pour lui permettre de se concentrer.

La gestion des problèmes identique à la gestion des incidents pour l'équipe, avec l'urgence en moins. Les problèmes à résoudre pourront être intégrés dans l'itération suivante s'ils sont considérés comme étant importants par le product owner.

En ce qui concerne la gestion des changements, Scrum facilite les choses étant donné que les itérations sont à durées fixes et que les releases sont planifiées aussi. Ce processus demandera probablement à l'équipe la production de plus de documents qu'autrement. Le product owner et le scrummaster devront faire partie du CAB – Change Advisory Board.

En fait, la finalité de Scrum et d'ITIL est commune : mettre l'utilisateur au centre de leurs préoccupations.

Pour Scrum, le but est de livrer le logiciel le plus en adéquation avec les besoins de l'utilisateur, et pour ITIL de livrer le service le plus en accord avec ce que souhaite l'utilisateur, comme défini dans les SLA – Service

Level Agreement.

Il semble donc que Scrum et ITIL puissent fonctionner ensemble.

IX. Recommandations

Il est important de mettre en oeuvre une stratégie pour introduire Scrum dans une entreprise. Vouloir l'introduire d'emblée suscitera des réactions négatives. En effet, la méthode remet en cause un nombre conséquent d'habitudes et de fonctions. Son introduction fera par conséquent face à de nombreuses réactions liées à la résistance contre le changement.

Une bonne solution est d'introduire la méthode par petits morceaux, sans la citer : on pourra commencer par les réunions quotidiennes de 15 minutes, puis par le tableau blanc. J'ai eu l'occasion d'expérimenter les réunions quotidiennes, et c'est une habitude qui est très vite prise et appréciée par l'équipe.

Cela permettra d'introduire la méthode auprès de sa hiérarchie en étant soutenu par les échos positifs des premières expérimentations, et de suggérer son application plus formelle sur un projet pilote.

Pour ce projet pilote, il faudra prendre soin de choisir un projet de taille restreinte avec une petite équipe de 8 personnes maximum, en sélectionnant des personnes ayant un passé de plusieurs projets derrière eux. Il me semble préférable de faire appel à un coach pour être certain d'appréhender correctement la méthode et de ne pas l'interpréter en fonction du vécu de chacun. La phase de détermination de qui est le product owner me semble essentielle et la personne désignée devra aussi entreprendre une formation sur ce qu'est son rôle.

L'équipe devra être dans une pièce spécialement dédiée au projet. En ce qui concerne les outils, il me semble préférable d'utiliser dans un premier temps des outils standards à disposition : tableau blanc et tableur, afin de pouvoir se concentrer sur les fondements de la méthode.

Il sera bien entendu plus facile de mettre en œuvre la méthode si la demande vient du client, mais c'est encore rare…

Enfin, il est nécessaire de communiquer sur le projet dans l'entreprise : prévenir quand les sprints commencent, donner leurs objectifs, prévenir de la

fin d’un sprint, organiser des réunions de présentation en plus de la réunion de revue de sprint.

Pour les techniques d’estimation, il me semble préférable de commencer par les estimations en jours idéaux qui est plus naturelle que la technique des points d’histoire. Cela sera assimilé plus facilement par l’équipe et peut éviter un rejet précoce de la méthode.

En ce qui concerne les techniques d’ingénierie logicielle, il me semble très important d’adopter au minimum les pratiques de test driven development et d’intégration continue d’eXtreme Programming.

X. CONCLUSION

Nous avons vu que Scrum est une méthode de la famille des méthodes agiles qui se positionne au niveau de la gestion de projet d'un développement logiciel. Nous avons étudié quels sont les rôles qu'elle introduit, et notamment les rôles de scrummaster et de product owner. Nous avons vu les règles que met en place Scrum pour d'une part planifier un projet dans sa globalité et pour le diviser en sprints et en releases, et d'autre part le faire vivre au quotidien et l'adapter selon l'expérience acquise grâce aux incréments logiciels livrés régulièrement.

Nous avons aussi vu que Scrum et les techniques d'ingénierie logicielle d'eXtreme Programming se complètent bien. Cela permet de répondre à l'exigence de connaître quotidiennement l'état d'avancement du projet.

Nous avons vu que si la méthode est au départ destinée à de petites équipes de 8 personnes maximum situées dans une même pièce, les mécanismes permettant de traiter des équipes plus importantes, même disséminées géographiquement, existent. Nous avons aussi étudié les outils dédiés à la gestion de projets agiles les plus populaires, outils propriétaires et open source.

Son adoption n'est pas sans poser des problèmes liés principalement aux changements des mentalités, notamment en ce qui concerne le transfert du pouvoir de décision vers l'équipe.

En regardant de plus près les éléments composant Scrum -livraison d'incréments de logiciel testables par le client à dates fixes, gestion de projet en adaptant son envergure et non les délais, points d'avancement réguliers… - on se rend compte qu'il n'y a finalement rien de totalement innovant dans Scrum. Et certainement pas le fait de développer les logiciels par itérations, puisque ce modèle est connu et décrit depuis longtemps. Ce qui est innovant réside dans le fait de mettre ensemble tous ces éléments. Il s'agit en définitive de bonnes pratiques et de principes de bon sens que chacun a certainement déjà expérimenté sur différents projets. C'est certainement pour cela que personne ne remet en cause Scrum et que la méthode subit aussi peu de critiques. Les techniques de Test Driven Development et d'intégration continue

sont par contre nouvelles mais pas explicitement recommandées par la méthode.

Est-ce que l'innovation la plus importante de Scrum n'est finalement pas de remettre la principale ressource de l'entreprise – l'homme -au centre du projet et de lui donner le pouvoir d'action et de décision ? La motivation qui en résulte est importante, et c'est finalement ce qui est le plus dur à maîtriser dans un projet.

Il est possible de décider de beaucoup de choses, mais pas de la motivation des membres de l'équipe. Or, la différence de productivité peut aller de 1 à 10 selon les personnes et leur motivation, d'où le terme de « sprint » traduisant l'accroissement de la productivité de l'équipe.

Est-ce que la méthode fonctionne ? Les retours d'expériences positives se multiplient, quelque soit la taille du projet. En outre, les techniques décrites sont pragmatiques et s'appuient sur des études scientifiques quant à la productivité et la motivation des équipes . Des enquêtes tendent à prouver que les projets utilisant une méthode agile connaissent un taux de réussite important .En tout état de cause, le besoin d'agilité ne peut pas être remis en cause.

Il est cependant nécessaire de faire attention : les méthodes agiles sont à la mode. Scrum représente un marché intéressant. Un signe indéniable de la maturité du marché est la présence de Microsoft avec

« Visual Studio Team System » et son « Solutions Framework (MSF) for Agile Software Development ».

Beaucoup d'entreprises peuvent donc être tentées par l'agilité, ou penser qu'elles sont déjà agiles, ou se dire agiles pour décrocher des contrats. On peut assez facilement penser faire Scrum, mais en fait-on réellement ? Il en va de même pour les outils dédiés : on peut utiliser V1 de VersionOne et ne pas faire Scrum ! Scrum semble simple à mettre en œuvre, mais c'est probablement plus difficile qu'il n'y paraît puisque la méthode est principalement basée sur l'expérience et demande des équipes ayant une certaine maturité.

Scrum est aussi un état d'esprit et une philosophie de développement. Etre agile ne signifie pas n’avoir aucune rigueur, au contraire. Scrum n’est pas une méthode miracle. Elle demande un réel engagement et du travail pour être appliquée correctement. Le jeu en vaut très certainement la chandelle.

XI. GLOSSAIRE

Burndown chart : les burndown charts se décomposent en sprint burndown et release burndown charts. Le sprint burndown chart est mis à jour quotidiennement lors d'un sprint par le scrummaster. Il représente la charge de travail encore à effectuer pour terminer le sprint. Il est possible d'en dégager une tendance pour voir si le sprint sera fini à temps ou non. Le release burndown chart est un graphique du même type, mais au niveau d'une release.

CMMi : Capability Maturity Model Integration. Modèle développé par le SEI – Software Engineering Institute

-permettant d'évaluer la maturité des sociétés développant des logiciels et de les classer en 5 catégories.

Ce modèle vient à la base d'une demande du Department of Defense américain afin de pouvoir évaluer ses fournisseurs de logiciel. En 1991 apparaît CMM qui sera rapidement décliné en d'autres modèles. En 2001 apparaît CMMi spécifiquement dédié au développement des logiciels. La dernière version, la 1.2, date de 2006. La certification se fait au niveau de l'entreprise.

Daily scrum : il s'agit de la réunion quotidienne de 15 minutes maximum durant laquelle chaque membre de l'équipe dit aux autres ce qu'il a fait la veille, ce qu'il va faire ce jour là, et explique les problèmes qu'il rencontre et qui l'empêchent d'avancer.

Estimation en points d'histoire : une technique d'estimation des besoins. On se focalise sur la comparaison des besoins entre eux plutôt que sur le temps nécessaire pour mettre en œuvre la solution.

Estimation en jours idéaux : une technique d'estimation des besoins. Un jour idéal correspond à une journée de travail pendant laquelle on n'est pas interrompu. Un coefficient de concentration est ensuite appliqué pour déterminer le délai nécessaire pour la réalisation de la tâche.

eXtreme Programing : ou plus simplement XP. Une méthode agile de développement logiciel axée sur l'ingénierie logicielle. Scrum et XP se complètent très bien.

Gantt : diagramme de Gantt. Concept développé par Henry Gantt aux alentours de 1910. Cet outil est souvent utilisé avec un réseau PERT dans la gestion de projet. Il permet de visualiser dans le temps toutes les tâches composant un projet. Il est aussi possible de représenter graphiquement l'avancement du projet.

Intégration continue : technique d'ingénierie logicielle recommandée dans eXtreme Programming. Elle consiste en la création automatique, grâce à un serveur de builds, d'un nouveau build chaque fois qu'un programmeur relâche du code dans le logiciel de contrôle de sources. Le build est alors soumis à une série de tests de non régression qui s'enrichit au fur et à mesure de l'élaboration du logiciel. Les programmeurs sont invités à relâcher leur code au minimum une fois par jour. Il est conseillé de le faire plusieurs fois par jour. Cette technique, bien que non explicitement recommandée par Scrum est très utile pour que l'équipe sache exactement où elle se situe quotidiennement.

ITIL : Information Technology Infrastructure Library. Il s'agit d'un ensemble de bonnes pratiques pour la gestion d'un système d'information. ITIL en est à sa version 3. Elle a 3 niveaux de certifications. Ce sont les personnes et non l'entreprise qui sont certifiées ITIL.

Modèle en cascade : ou waterfall. Modèle de développement dans lequel les différentes phases – spécifications, design, implémentation, tests -se succèdent dès que la précédente est terminée.

Modèle évolutif : modèle de développement itératif qui suppose que les spécifications changent en cours de développement et que par conséquent le planning de développement doit être adapté en conséquence.

Modèle itératif : modèle de développement dans lequel le développement du logiciel est composé de plusieurs itérations. Chaque itération est un mini projet

en soi dans lequel on retrouve les phases de spécifications, design, implémentation et tests. Chaque itération résulte en un logiciel représentant une partie du logiciel final. La plupart des itérations sont internes à l'équipe. Le résultat de la dernière itération est le logiciel final.

PERT : réseau PERT – Program Evaluation and Review Technique, ou technique d'évaluation et d'examen de programmes. Le graphique PERT est une technique de gestion de projet qui permet de visualiser la dépendance des tâches et de procéder à leur ordonnancement. Un graphe de dépendances est utilisé. On indique pour chaque tâche une date de début, de fin au plus tôt et au plus tard. Il est ainsi possible de déterminer le chemin critique.

Poker planning : technique utilisée pour estimer les tâches. Elle a comme avantage qu'elle est ludique, qu'elle implique l'équipe entière et que le dialogue qui s'engage permet de mieux comprendre le périmètre de la tâche et par conséquent d'obtenir une estimation plus juste. Il se joue grâce à des jeu de cartes sur lesquelles des chiffres inspirés de la suite de Fibonacci sont imprimés : 0, 1, 2, 3, 5, 8, 13, 20, 40, 100

Product backlog : ou backlog produit. Catalogue dans lequel sont listés tous les besoins du projet. Le backlog produit est mis à jour régulièrement. Les besoins sont priorisés par le product owner.

Product owner : un rôle clé en Scrum. Il s'agit du représentant du client. C'est lui qui va prioriser les besoins contenus dans le backlog produit et ainsi indiquer l'ordre dans lequel les besoins doivent être développés. C'est lui qui décide des releases.

Refactoring : technique d'ingénierie logicielle préconisée dans eXtreme Programming qui consiste à réécrire du code en fonction des nouvelles

fonctionnalités à développer ou pour le rendre plus simple. Il s'agit de ne pas prévoir tous les cas possibles, mais de faire le code minimum qui répond à un problème donné. Cela oblige bien évidemment à le modifier pour prendre en considération de nouveaux besoins exprimés. Cette technique est étroitement liée aux tests unitaires et au TDD. En effet, les tests unitaires permettent de s'assurer que le refactoring n'a pas entraîné de régressions.

Scrum : se traduit par mêlée en rugby. Désigne une méthode agile dans laquelle l'équipe est soudée et avance ensemble vers un objectif commun.

Scrummaster : un rôle clé en Scrum. Il s'agit plus d'un coach que d'un chef de projet. Il est responsable d'expliquer la méthode et de vérifier qu'elle est bien appliquée au quotidien dans l'équipe. Il s'assure que l'équipe est isolée des perturbations extérieures en cours de sprint.

Scrum des scrums : Lorsque la taille d'un projet requiert plus de 8 personnes, il est alors nécessaire de constituer plusieurs équipes fonctionnant avec Scrum, chacune possédant un scrummaster et procédant à une réunion quotidienne. Le scrum des scrums est la réunion régulière effectuée avec tous les scrummasters des différentes équipes participant à un même projet. Elle peut être quotidienne ou hebdomadaire.

SGVA : Système de Gestion de la Valeur Acquise. Il s'agit de courbes utilisées pour démontrer la progression du projet par rapport à ce qui avait été planifié. Ce type de courbes est demandé par le Department of Defense des Etats-Unis à ses contractants.

Sprint backlog : ou backlog de sprint. Catalogue issu du product backlog qui contient en plus des besoins sélectionnés dans le product backlog, le détail des tâches à faire pour répondre à chaque besoins. L'équipe de développement s'appuie sur le sprint backlog pour développer l'incrément logiciel durant le sprint.

TDD : voir Test Driven Development.

Team lead : Fonction qu'il est possible de créer en cas de plusieurs équipes Scrums travaillant en parallèle sur un même projet. Le team lead fait des arbitrages que les scrummasters des équipes ne peuvent pas faire entre eux. Le rôle de team lead n'est pas explicitement recommandé par Scrum.

Test driven development : une des techniques prônée par eXtreme Programming. Pour chaque tâche, le programmeur écrit les tests unitaires que son code doit passer. Le développement de la tâche est terminée lorsque le code passe avec succès les tests écrits. Il existe des frameworks de tests unitaires comme JUnit pour java par exemple qui permettent de mettre en oeuvre des tests unitaires et des séries de tests unitaires.

Souvent utilisé avec l'intégration continue.

UP : Unified Process. Il s'agit d'une méthode utilisée pour le développement de logiciels utilisant un langage orienté objets. C'est une méthode générique, itérative et incrémentale. Elle est implémentée en diverses méthodes dont RUP et Agile UP par exemple. Pour qu'une méthode réponde aux préceptes de UP, elle doit remplir les conditions suivantes :

- Elle est à base de composant.
- Elle utilise UML.
- Elle est pilotée par les cas d'utilisation.
- Elle est centrée sur l'architecture.
- Elle est itérative et incrémentale.

Vélocité : Notion mettant en relation la charge de travail en jours idéaux et le délai réellement nécessaire pour accomplir la tâche, grâce à un facteur de concentration. La vélocité des sprints précédents est utilisée au début d'un nouveau sprint pour que l'équipe réussisse à calibrer son engagement pour le nouveau sprint.

Waterfall : voir modèle en cascade.