Techniques de l'Ingénieur

Réf. : **H3230 V1**

Méthode Agile SCRUM - Comment l'utiliser de manière opérationnelle ?

Date de publication :
10 janvier 2019

Date de dernière validation :
29 janvier 2021

Cet article est issu de : **Technologies de l'information | Technologies logicielles Architectures des systèmes**

par **Agusti CANALS**

Mots-clés
Méthode | Sprint | Release | SCRUM

Résumé Cet article, après une description rapide de l'historique concernant les méthodes agiles, présente l'approche SCRUM, illustrée par l'exemple, à travers les concepts essentiels nécessaires à la bonne utilisation opérationnelle.

Keywords
Method | Sprint | Release | SCRUM

Abstract This Paper, after a quick description of the history of agile methods, presents the SCRUM approach, illustrated by the example, through the essential concepts necessary for the proper operational use.

Pour toute question :
Service Relation clientèle
Techniques de l'Ingénieur
Immeuble Pleyad 1
39, boulevard Ornano
93288 Saint-Denis Cedex

Par mail :
infos.clients@teching.com
Par téléphone :
00 33 (0)1 53 35 20 20

Méthode Agile SCRUM

Comment l'utiliser de manière opérationnelle ?

par **Agusti CANALS**
Directeur d'Unité Fonctionnelle (Technique)
CS Communication & Systèmes Toulouse, France

L'Agilité, et plus particulièrement les prémices de la méthode SCRUM, sont apparues dans les années 1990 ; cependant leur utilisation opérationnelle dans le domaine de l'informatique ne s'est généralisée qu'au début des années 2000, et a commencé à être utilisée de manière courante, au début des années 2010. À noter que Jeff Sutherland et Ken Schwaber décrivent les principes de la méthode dans le « SCRUM Guide » en 2011, puis en 2016. [2]

Cet article montre, dans un premier temps, l'évolution de SCRUM depuis 2010, puis il fournit une description, illustrée par l'exemple, des techniques essentielles et suffisantes de la méthode SCRUM ; enfin il montre quand et comment il faut l'utiliser pour en retirer les meilleurs résultats, tout en mettant en avant les conditions nécessaires pour réussir son application sur un projet.

1. Méthodes agiles, où en est-on ?

L'article [H 3 202] « Méthodes Agiles » publié en février 2010 par Jacques PRINTZ, Professeur au Conservatoire National des Arts et Métiers (CNAM), fait le tour de la question en détaillant les fondamentaux de ces méthodes. Sa conclusion, et notamment sur SCRUM, est plutôt mitigée.

La citation suivante illustre parfaitement son point de vue :

« Imaginer qu'une solution puisse émerger comme par miracle, comme le laisse penser SCRUM, est de la pensée magique qui n'a pas sa place dans la science de l'ingénieur », J. PRINTZ

Ce qui était peut-être vrai hier, cela reste sujet à débat tout de même, ne l'est plus aujourd'hui. En effet SCRUM ne préconise pas des solutions spontanées trouvées par miracle ! SCRUM préconise une organisation de projet qui permet de trouver des solutions en s'appuyant sur les approches proposées par le génie logiciel, et notamment : la gestion des exigences, de configuration, des tests… la conception, le respect des critères qualité (Plan qualité…). **Ce qui change**, c'est l'organisation, le vocabulaire, et la souplesse qui en découle. L'ingénierie se fait autrement, mais elle se fait.

2. Évolution depuis 2010, qu'en est-il aujourd'hui ?

Aujourd'hui quand nous utilisons SCRUM : la qualité, la méthodologie, les normes… sont **présentes**. Cela fait partie de ce que nous pouvons appeler le « *normal work* ». En effet, SCRUM nous permet une organisation plus souple, et sans doute beaucoup plus efficace tout en gardant une bonne rigueur. Ce point n'était pas suffisamment clair par le passé.

Cette rigueur est d'autant plus importante lorsque nous nous intéressons aux tests. Pour les traiter, SCRUM préconise l'**Intégration Continue** (IC), non obligatoire mais fortement conseillée. Cette technique permet de préparer et de gérer les tests (aussi bien fonctionnels qu'unitaires) tout en automatisant leur exécution. À noter que nous pourrions également parler de DEvOPS, préconisé aussi par SCRUM. Cette technique inclut l'IC et pousse l'automatisation jusqu'au déploiement opérationnel en continu ! Nous n'en dirons pas plus ici, il faudrait en effet plusieurs articles pour détailler ces techniques.

Enfin, comme nous le verrons par la suite, une équipe SCRUM est limitée en nombre de personnes. Au-delà de dix, il faudrait faire du « SCRUM de SCRUM » et utiliser des Framework et des méthodes plus globales, par exemple « **Scaled Agile Framework (SAFe)** » qui intègre entre autres des équipes SCRUM. À noter que SAFe est en cours de mise en place, et semble-t-il avec succès, dans la société Airbus.

Aujourd'hui nous pouvons pratiquer SCRUM avec des petites équipes. Ces dernières années de nombreux projets (chez Airbus, à l'ESA, au CNES, chez CS…) ont prouvé l'efficacité de l'approche, par exemple « Orfeo tool box » https://www.orfeo-toolbox.org/ outil « *open source* » diffusé par le CNES, ou encore « IKATS » https://ikats.org/ outil également « *open source* » diffusé par CS.

Au-delà, pour les gros systèmes, nous devons nous organiser avec des approches de type SAFe.

3. Présentation détaillée de SCRUM

SCRUM présente un ensemble de techniques permettant de s'organiser au sein d'un projet dit « Agile ». Nous allons nous concentrer dans cet article sur un nombre limité de techniques permettant de pratiquer l'agilité SCRUM de manière opérationnelle et efficace. Pour en savoir plus, vous pouvez consulter la 5e édition du livre publié par Claude AUBRY, une référence sur SCRUM en France [1]

SCRUM est une méthode (considérée également comme un cadre de travail) « générique » multidomaines (exemple : Systèmes embarqués, systèmes d'information…), dédiée à la « Gestion de Projet ». Elle a pour objectif d'améliorer la qualité du produit réalisé, la productivité, ainsi que le niveau de satisfaction de tous les intervenants (ce qui inclut l'équipe de réalisation). En quatre mots, l'objectif principal est de : « maximiser la valeur ajoutée », pour le client, mais aussi pour l'équipe de réalisation, ainsi que pour toutes les parties prenantes.

3.1 Préambule

Les quatre grands principes des méthodes agiles à retenir sans modération, sont :

(1) Individus et interactions contre processus et outils
(2) Logiciel qui fonctionne contre documentation exhaustive
(3) Collaboration du client contre négociation de contrat
(4) Réponse au changement contre suivi d'un plan prédéfini.

SCRUM est plus qu'une méthode, une approche… c'est UNE véritable PHILOSOPHIE mettant en avant :

L'esprit d'équipe ; l'engagement de l'équipe ; l'auto-organisation de l'équipe avec l'aide d'un « coach » (le SCRUM Master) ; la responsabilisation et l'amélioration de l'équipe (échanges nombreux et encouragés, management intégré aux pratiques) ; l'amélioration continue (cycles courts, performances mesurables) et la visibilité client (releases régulières, implication du client dans l'équipe en tant que Product Owner).

3.2 Cycle de vie

Le cycle de vie SCRUM est une organisation basée sur une décomposition simple et unique.

3.2.1 Le projet, les releases et les Sprints

Un « **Projet** » est toujours décomposé en « **Releases** » et en « **Sprints** » (figure **1**). Un « **Sprint** » est une période de temps fixe permettant d'avoir un livrable à la fin de celle-ci. Une « **Release** » est un ensemble de « **Sprints** » produisant une version utilisable du produit.

À noter que le premier « Sprint » d'une « Release » que l'on nomme communément « **Sprint 0** » peut avoir une durée différente, en général plus longue que les « Sprints » classiques. Ce « Sprint 0 » est particulier et unique. Il ne faut surtout pas le négliger sous peine de générer une « catastrophe ».

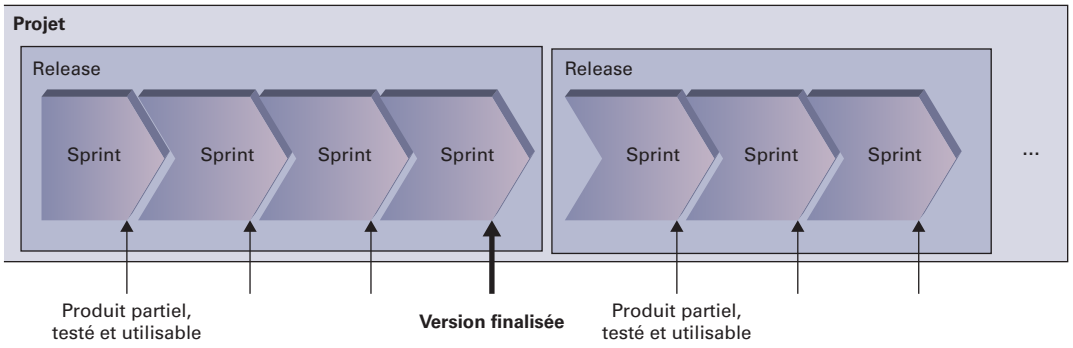

Figure 1 – Cycle de vie d'un projet SCRUM

3.2.2 Sprint 0

Le premier « Sprint », ou « Sprint 0 » d'une « Release », a une importance capitale pour la réussite du projet. En effet, durant cette période, nous allons organiser le projet (infrastructure, outils…), préciser les définitions (notamment de prêt et de fini) et réaliser les tâches générales d'architecture.

3.3 Équipe

Une équipe SCRUM est bien évidemment multidisciplinaire et surtout « autogérée ». La meilleure efficacité est obtenue lorsque l'ensemble de l'équipe est localisé au même endroit, idéalement dans la même pièce. Cependant nous pouvons également constituer des équipes distribuées sur plusieurs lieux (bâtiments, villes, pays…), il suffit alors de mettre en place les moyens nécessaires pour garder le contact permanent, comme si l'on était dans la même pièce.

L'équipe doit être mixte, client et fournisseur. En effet, l'implication du client ou « *Product Owner* » est essentielle dans ce type d'organisation.

Une équipe projet ne doit pas dépasser dix membres actifs, (le *Product Owner*, le SCRUM *Master* et 8 équipiers). Si tel n'était pas le cas, il faudrait s'organiser autrement, à savoir, par équipes de dix et ainsi faire du SCRUM de SCRUM, mais ce n'est pas le but de cet article de rentrer dans ces considérations, nous resterons donc basés sur l'hypothèse.

Une équipe projet ne doit pas dépasser dix membres actifs : le Product Owner, le SCRUM Master et huit équipiers (figure **2**).

3.3.1 Product Owner (PO)

Le PO est le client, il est intégré dans l'équipe projet, il voit l'équipe travailler et surtout il prend sa part dans les phases de définition du travail à réaliser et de validation. À noter que, pour la validation, il peut s'appuyer sur une équipe « client », son équipe, ou sous-traiter s'il le souhaite au fournisseur qui réalise, ou pourquoi pas à un autre fournisseur spécialement chargé de la validation.

Ce mode collaboratif Client/Fournisseur(s), n'est pas du tout habituel dans un projet dit classique. Il est nécessaire pour que cela fonctionne, que les différentes parties aient une confiance absolue et soient totalement transparentes. Dans le cas contraire, nous allons droit vers des conflits et un échec certain.

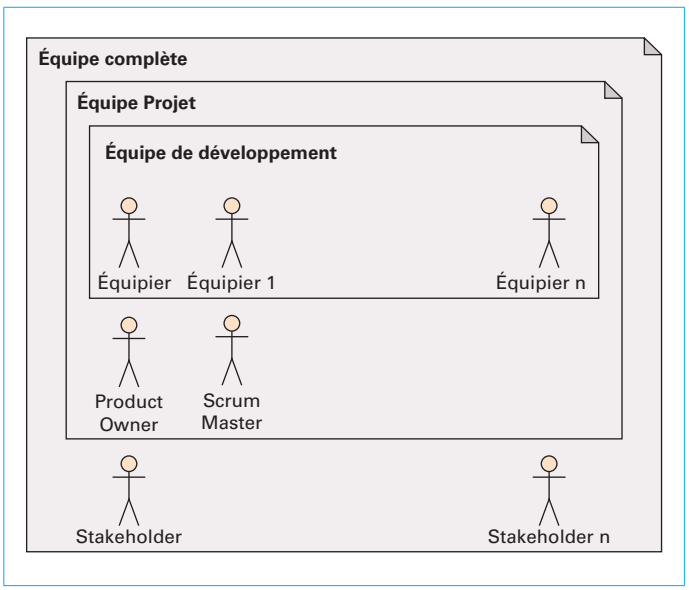

Figure 2 – L'équipe SCRUM

Le PO définit le produit et organise le travail en classant les fonctionnalités par ordre décroissant, en fonction de la valeur ajoutée qu'elles lui rapportent.

À noter que, sur certains projets, le client n'a pas le temps de s'impliquer !!! Dans ce cas, nous avons deux possibilités. Soit nous laissons « tomber » SCRUM et nous partons sur une organisation traditionnelle. Soit nous utilisons la notion de « PO Proxy ». Ce nouveau concept inventé par des « pragmatiques » pour remédier à la « non-disponibilité » du client, est très intéressant, mais reste dangereux.

Le « PO Proxy » est un membre de l'équipe de la société sous-traitante, ou, pourquoi pas d'une autre société. Dans tous les cas, le « PO Proxy » doit maîtriser le métier du client et connaître parfaitement ses besoins. C'est lui qui fera office de PO, prendra des décisions, organisera le travail, réalisera les validations… et qui se chargera de la relation avec le client, et notamment en cas de conflit (client en désaccord avec les décisions prises) il devra gérer la situation.

3.3.2 SCRUM Master (SM)

Le SM (un coach, un facilitateur) est le garant du respect de la méthodologie, il s'assure du bon déroulé du cérémonial, il protège l'équipe (qui doit être dans un cocon durant le « Sprint ») et il est l'interlocuteur privilégié du PO.

Le SM (un coach, un facilitateur) est le garant du respect de la méthodologie.

3.3.3 Équipier

L'équipier est un membre de l'équipe directement responsable du développement et de la production des livrables. L'équipier peut bien sûr être spécialisé (ex : codeur, testeur…). Il est également possible de faire intervenir des experts externes au projet en cas de besoin et de manière ponctuelle.

3.3.4 Stakeholder (SH)

Le SH représente toute personne impliquée dans le projet, même de manière épisodique, (ex : le Commerce, la Direction, l'utilisateur final du produit…). Il est invité aux démonstrations de fin de « Sprint », et de fin de « Release » ; à cette occasion il peut exprimer son avis, faire des suggestions…

3.4 Concepts

Parmi les nombreux concepts véhiculés par la méthode SCRUM, nous allons nous concentrer sur ceux que nous considérons essentiels pour le bon déroulement du projet.

3.4.1 Story

Il existe deux types de « *Stories* » : les « *User Stories* » fournies par le PO et les « *Technical Stories* » déduites par l'équipe projet en fonction des points techniques à résoudre.

Une « *User Story* » décrit un cas d'utilisation du produit, à noter que cela peut correspondre à une ou plusieurs fonctionnalités.

Une « *Technical Story* » décrit un point technique à résoudre, par exemple une étude de faisabilité, une évaluation de performances…

Pour la description d'une « *Story* », il est recommandé de suivre le squelette suivant :

En tant que…
Je veux…
Parce que j'en ai besoin pour…
Et je le validerai comme suit…

Par **exemple** :

« **En tant que**

Superviseur du système,

Je veux

Pouvoir visualiser les différentes courbes de température du boîtier contenant le régulateur**,**

Parce que j'en ai besoin pour

Calculer le débit optimal du liquide de refroidissement, afin de maintenir une température du boîtier comprise entre 14 et 18 degrés,

Et je le validerai comme suit

Via des capteurs de température connectés, émettant la valeur acquise toutes les deux secondes vers le serveur de supervision. »

3.4.2 Feature

Une « *Feature* » est un regroupement de « *Stories* » autour d'un même sujet.

Par exemple, nous devons développer une application qui réalise des acquisitions de paramètres puis des prétraitements et des traitements de ces derniers, et enfin une visualisation des résultats obtenus. Nous pourrions dans ce cas créer quatre « *Feature(s)* » : Acquisition, Prétraitement, Traitement et Visualisation. En gros, les quatre grosses fonctionnalités de l'application. Ensuite, en fonction des contraintes opérationnelles, ces « *Feature(s)* » vont se peupler de « *Stories* », ces dernières seront ensuite reprises et traitées dans les « Sprint ».

3.4.3 Backlog

Le backlog est destiné à recueillir tous les besoins du client que l'équipe projet doit réaliser. Il contient donc la liste des fonctionnalités constituant un produit et les éléments nécessitant l'intervention de l'équipe projet.

Il peut y avoir trois types de « Backlog » dans un projet, le « Backlog du projet » (ou produit), le « Backlog de la release » et le « Backlog de Sprint ».

3.4.3.1 « Backlog du projet »

Il regroupe l'ensemble des fonctionnalités (sous forme de « *Features* » et de « *Stories* ») nécessaires à la réalisation du projet. Le « Backlog du projet » doit être ordonné suivant une règle simple, à savoir, le rapport « Valeur (ou Valeur Ajoutée)/Complexité » soit V/C. Plus la valeur ajoutée pour l'utilisateur final est importante et plus la complexité de réalisation est faible, plus la « *Story* » est en haut du classement. Bien noter que ce n'est pas un regroupement de « *Stories* » par priorité, mais un classement ordonné qui sera traité suivant l'ordre établi.

3.4.3.2 « Backlog de la release »

Il regroupe un sous-ensemble du « Backlog projet » (sous forme de « *Features* » et de « *Stories* ») nécessaire à la réalisation de la « Release ». Le « Backlog de la release » doit être également ordonné suivant la même règle que pour le projet.

À noter que ce « Backlog » n'est pas obligatoire, nous pouvons très bien passer du « Backlog projet » directement au « Backlog de Sprint ». Là encore ce choix est laissé au projet.

3.4.3.3 « Backlog de Sprint »

Il regroupe un sous-ensemble du « Backlog projet » ou du « Backlog release » (suivant le cas) sous forme de « *Stories* » nécessaires à la réalisation du « Sprint ». Le « Backlog de Sprint » n'est pas forcément ordonné, en effet tout doit être fait et validé en fin de « Sprint ».

3.4.4 Dashboard

Le Dashboard est un outil de travail, il se présente sous forme de tableau, physique (collé au mur) ou virtuel (via l'utilisation d'un outil), il permet de réaliser le suivi du projet au jour le jour. La structure de ce tableau peut varier suivant les projets. La figure **3** présente un exemple opérationnel utilisé de manière courante.

Chaque ligne du tableau correspond à une « *Story* », cette dernière se décompose en une ou plusieurs tâches. Les tâches (sous forme de post-it) vont ensuite circuler sur la ligne du tableau correspondant à la « *Story* », d'abord vers la colonne « En cours » (« À faire » et « À vérifier ») puis vers la colonne « Terminé ».

Backlog	En cours		Terminé	Sprint #
	À faire	À vérifier		Fin le : / /
				Tâches récurrentes
				Bac à sable
				Gourmandises

Figure 3 – Un exemple de Dashboard

Noter qu'une tâche peut bien sûr revenir en arrière si elle ne passe pas l'étape de vérification.

La dernière colonne en plus du nom (ou objectif) et de la date de fin du « Sprint » propose plusieurs zones, à savoir : « Tâches récurrentes » (par exemple compléter le document de conception...), le « Bac à sable » permettant à chaque membre de l'équipe de proposer de nouvelles « *Stories* »... (ces dernières seront analysées lors de la « réunion de "revue du Backlog" ») et les « **Gourmandises** » !!!! Quèsaco ??

En fait en début de projet l'équipe établit des règles, par exemple : « *tout membre de l'équipe arrivant en retard à une réunion aura une X, et au bout de trois X, il devra apporter des croissants pour le petit-déjeuner de l'équipe !!!* » Bien sûr les règles et les « punitions » associées, sont laissées à la décision de l'équipe, il peut d'ailleurs ne pas y avoir de règles.

3.4.5 Point de valeur

Les points de valeur sont attribués par le PO, ils représentent la valeur ajoutée de la *story* par rapport à ses attentes. Ces points suivent une suite de Fibonacci modifiée : 1, 2, 3, 5, 8, 20, 40, 100.

3.4.6 Point de complexité

Les points de complexité sont attribués par l'équipe lors de la réunion de planification, ils représentent la complexité de la *story* par rapport aux *stories* de référence. Ces points suivent également une suite de Fibonacci modifiée : 1, 2, 3, 5, 8, 20, 40, 100.

3.5 Quelques définitions essentielles

Les définitions données ci-après **sont essentielles** pour la bonne conduite et la réussite du projet. Il est essentiel de définir ce que veut dire « Prêt » et ce que veut dire « Fini ». Ces définitions doivent être précises et sans ambiguïté. Par exemple elles doivent permettre de décider de manière factuelle si une « Release », un « Sprint », une « *Story* » ou une « Tâche » est finie.

Ces définitions sont rédigées au plus tard, au début du « Sprint 0 » de la première « Release » par le PO en accord avec l'équipe projet.

3.5.1 Définition de prêt

La définition de « Prêt » se présente sous la forme d'une série d'exigences qui doivent être respectées pour qu'un élément (« Release », « Sprint », « *Story* », « Tâche ») soit considéré comme prêt.

Exemple, définition de prêt pour une « *user story* » :

- l'objectif de la « *story* » est clairement défini (le Pourquoi) ;
- la description des éléments à réaliser est clairement définie (le Quoi). Elle ne présente pas d'ambiguïté d'interprétation, pas de « non-dit » ;
- la « *story* » peut être implémentée indépendamment des autres ***stories*** identifiées dans le « backlog » de « Sprint » (quitte à devoir mettre des bouchons) ;
- la description est suffisamment décomposée pour être estimable par l'équipe ;
- la « *story* » apporte de la valeur pour l'utilisateur final ;
- aucun élément n'est noté AC (À Confirmer) et encore moins AD (À Définir) ;
- l'attribution d'un ou plusieurs identificateur(s) d'exigence a été faite afin d'assurer la traçabilité ;
- des Tests d'acceptation sont associés à la « *story* » (et donc aux exigences) ;
- le niveau de détail atteint est suffisant pour la « *story* » et l'équipe accepte sa définition ;
- la « *story* » est suffisamment petite, elle doit représenter moins d'1/3 de la capacité du « Sprint ». Sinon il faut la décomposer.

Bien sûr, dans un cas réel, la définition doit être plus détaillée et doit être validée par l'ensemble des intervenants.

Par **exemple** définir ce que doit être la « capacité d'un Sprint ».

3.5.2 Définition de fini

La définition de « Fini » se présente sous la forme d'une série d'exigences qui doivent être respectées pour qu'un élément (« Release », « Sprint », « *Story* », « Tâche ») soit considéré comme fini.

Exemple, définition de fini pour une *story*/tâche :

- le codage est terminé et le code est géré en configuration ;
- TOUS les tests associés (TU, Tests Fonctionnels, tests de performances, tests de robustesse) sont terminés, gérés en configuration et s'exécutent correctement ;
- la couverture du code par les tests est conforme aux taux attendus par le niveau de criticité applicable ;
- la documentation associée est mise à jour ;
- la documentation du code (via les tags) est à jour ;
- les activités qualité sont terminées.

Bien sûr, dans un cas réel, la définition doit être plus détaillée et doit être validée par l'ensemble des intervenants.

Par **exemple** détailler « les activités qualité ».

3.6 Cérémonial

À la figure **4**, nous pouvons voir un exemple d'organisation, concernant un « Sprint », dont la durée récurrente sera de dix jours ouvrés. Nous pouvons y voir différents types de réunions faisant partie du cérémonial préconisé par SCRUM :

– une pratique quotidienne, le « Daily SCRUM » ;
– la planification du « Sprint » réalisée en début de « Sprint » ;
– la revue du « Backlog » toujours réalisée quelques jours avant la fin du « Sprint » ;
– la rétrospective de « Sprint » réalisée le dernier jour du « Sprint ».

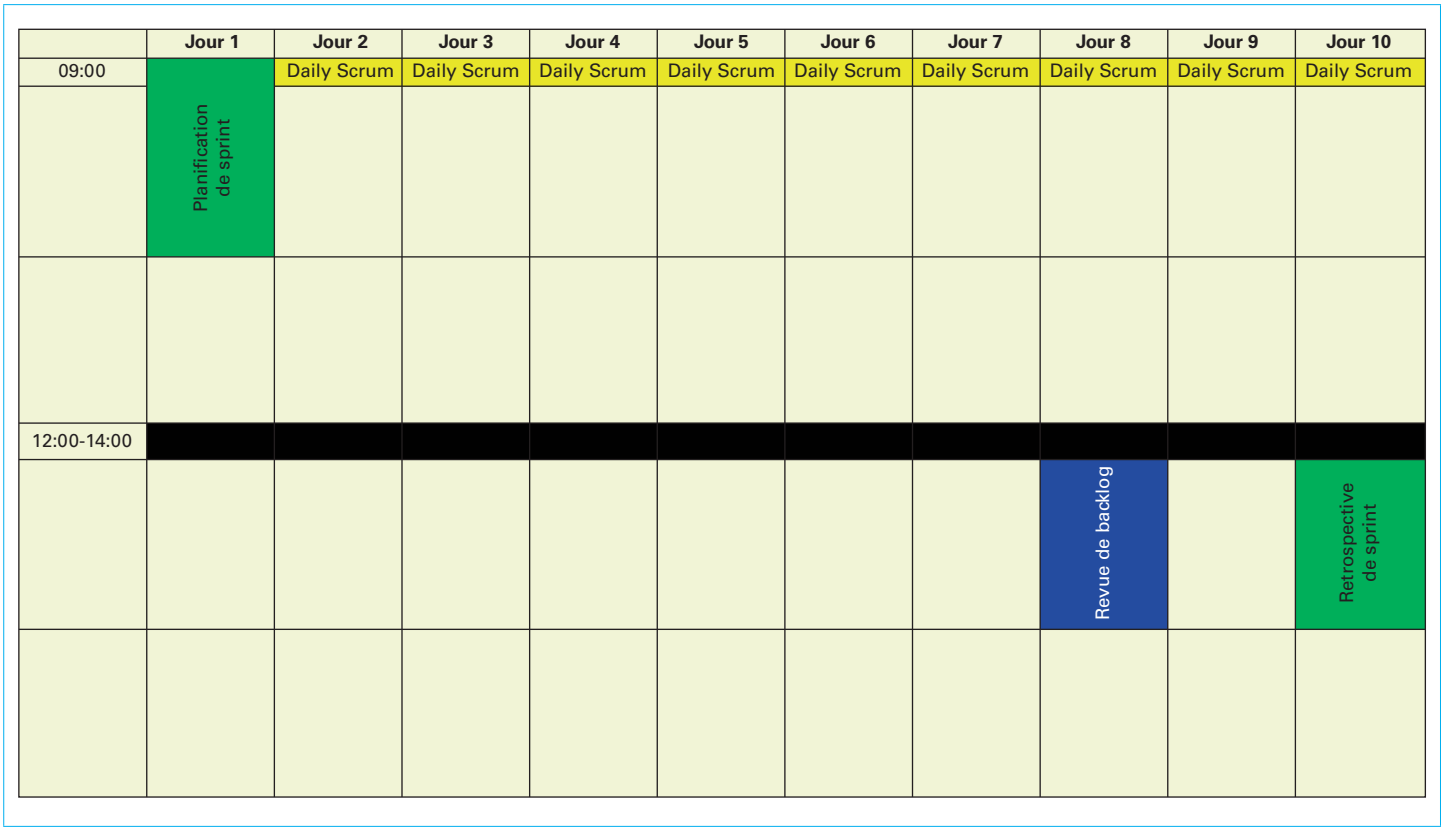

Figure 4 – Un exemple d'organisation pour un « Sprint »

3.6.1 Pratiques quotidiennes

3.6.1.1 Daily SCRUM

Le Daily SCRUM est une réunion d'information qui se déroule chaque jour à une heure bien précise (choisie par l'équipe en début de projet), en général en tout début de journée, ou juste avant la pause-café.

Si nous restons puristes, la manière de faire est la suivante : l'équipe s'installe debout et forme un cercle (le café est le bienvenu !). Chaque membre énonce les trois points suivants : « ce que j'ai fait hier », « ce que je fais aujourd'hui », « les problèmes (points durs) que je dois résoudre ». ATTENTION, pas de débat, on énonce juste de manière factuelle. Bien sûr le « SCRUM Master » va noter les points qui nécessitent d'être traités, et organisera par la suite des réunions de travail dédiées. À noter que de manière spontanée, un membre de l'équipe aidera, après la réunion, un autre membre ayant énoncé un point dur.

POINT ESSENTIEL, nous avançons ensemble, en équipe.

Note : La durée de cette réunion est en général de 15/20 minutes, pas plus.

3.6.1.2 « Normal WORK »

Il existe une rumeur très répandue, qui énonce la contre-vérité suivante :

« *Lorsque nous appliquons une méthode Agile, et plus particulièrement SCRUM, nous n'appliquons plus l'état de l'art préconisé par le Génie Logiciel, ni les règles qualité préconisées par les différentes normes* ».

Cette rumeur est TOTALEMENT fausse, en effet, le génie logiciel et la qualité font partie intégrantes du « Normal Work ». Ainsi la gestion des exigences, de la traçabilité, de la configuration… la spécification, la conception, la vérification des règles de codage, l'application des techniques de modélisation (par exemple UML) et de vérification (par exemple les méthodes formelles), les tests de non-régression… **sont utilisés** de manière aussi efficace que dans une approche classique.

Ce qui change, et effectivement cela peut perturber, si l'équipe ne s'est pas préparée à ce changement, **c'est l'organisation et la manière de faire**. Par exemple dans chaque « Sprint » il peut y avoir une tâche récurrente du type : « Compléter la documentation de conception » ce qui aboutit en fin de projet à une documentation complète décrivant les architectures logique et physique du système réalisé. Donc, le même résultat que pour une approche classique en termes de documentation finale.

Par contre, dans une approche classique, la documentation de conception complète est produite en phase de conception bien avant les phases de codage, alors qu'en SCRUM cette même documentation est produite au fil de l'eau et ne sera finalisée qu'en fin de projet. Eh oui !! Ça change !!!

3.6.1.3 Tenue à jour du tableau de bord ou « Dashboard »

Afin de bien suivre (et visualiser rapidement) l'avancement du projet d'un seul coup d'œil au tableau de bord, les post-it (les n « Tâches » d'une « *Story* ») sont déplacés au fil de l'eau suivant l'avancement des « *stories* ».

À noter qu'il peut y avoir des propositions de nouvelles « Tâches », qui seront bien sûr ajoutées au tableau de bord, et pourquoi pas de nouvelles « *Stories* » qui, elles, seront ajoutées dans le « Bac à Sable » et analysées ultérieurement.

3.6.2 Réunion de planification

La réunion de planification permet à l'équipe d'analyser, d'évaluer et de planifier la réalisation des « *stories* » choisies pour le « Sprint ».

Il est important de noter qu'avant de réaliser une évaluation et de planifier un « Sprint », nous devons disposer d'une référence. Cette dernière est établie par l'équipe lors du « Sprint 0 » de la première « Release » et sera ajustée, révisée, si nécessaire, à chaque « Sprint 0 ».

La référence minimale est de disposer de trois « *User stories* » déjà opérationnelles, spécifiées, codées, testées, validées et documentées. Ces trois « *user stories* » sont évaluées : facile, moyenne et difficile, par exemple, facile correspond à 1, moyenne à 5 et difficile à 20. Mais, bien sûr, on peut choisir n'importe quelle échelle, l'essentiel c'est d'avoir un consensus de l'équipe.

3.6.2.1 Planning Poker

Le fonctionnement est simple, nous pouvons le résumer à travers les huit points suivants :
1/ Chaque participant choisit un jeu de cartes d'une couleur (par exemple, orange comme dans la figure **5**),
2/ Le PO présente la « *user story* » à évaluer,
3/ Chaque participant pose des questions, il y a débat sur le contenu, et donc un échange avec le PO,
4/ Chaque participant choisit une carte (sans la montrer aux autres, bien sûr),
5/ Tout le monde retourne sa carte,
6/ Si la valeur est la même pour tous, par exemple « 5 » l'estimation est validée,
7/ Si les valeurs sont différentes, chacun s'exprime pour expliquer son point de vue, il y a débat,
8/ Le processus se répète tant que l'équipe n'aboutit pas à un consensus.

Noter que si le consensus n'est pas possible la « *User story* » retourne dans le « Backlog », **elle n'est pas prête à être évaluée** !

Note : La durée de cette réunion est en général de deux heures, pas plus.

3.6.3 Réunion de revue de Backlog

La réunion de « Revue de backlog » réalisée quelques jours avant la fin du « Sprint » permet de mettre à jour le « Backlog produit (projet) » (ex : ajouter des « *Stories* », en enlever, en modifier le contenu…), de changer l'ordonnancement, si nécessaire. En effet, il est important de comprendre qu'un « Backlog » est un ensemble de « *Stories* » ordonnées qui vont être traités dans l'ordre.

Figure 5 – Les cartes utilisées pour le planning poker

Une bonne pratique, *qui devrait même être une obligation,* consiste à maintenir un ensemble de « *Stories* » **prêtes**, permettant d'avoir 1,5 à 2 « Sprint(s) » d'avance. Ce principe est important, même essentiel, pour éviter les périodes dangereuses dites de *« famine » ! quèsaco ??*

> Une période dite de « **famine** » est une période pendant laquelle l'équipe projet n'a rien à faire, car rien n'est prêt (se reporter à la définition de « prêt »)!!!

Que faire dans ce cas ?

– ne rien faire en attendant des « *Stories* » prêtes, mais dans ce cas qui finance cette période ? le client ? nous ? **Ceci doit être défini et écrit dans le contrat** signé par les deux parties ; dans ce cas aucun problème. Si tel n'est pas le cas, le risque de conflit est TRÈS grand ! D'ailleurs, lorsqu'un projet Agile a des problèmes, c'est souvent, pour ne pas dire toujours, lié à cette période de « famine » ;

– affecter l'équipe sur un autre projet ? Oui, possible, mais délicat, car il faudra négocier le retour et gérer un planning sur deux projets, des conflits en perspective !

– partir sur des « *Stories* » non prêtes, oui, possible, mais SUPER dangereux !!! Pour toutes les raisons que vous pourriez imaginer : résultat non conforme aux attentes, risque de travailler pour rien…

Note : La durée de cette réunion est en général de deux heures, pas plus.

3.6.4 Réunion de démonstration de fin de « Sprint »

La réunion de démonstration de fin de « Sprint » permet à l'équipe projet de présenter le travail réalisé à l'ensemble des invités, à savoir toutes les parties prenantes. À noter que ces dernières sont invitées systématiquement, mais qu'elles ne se déplacent pas toujours pour assister à la démonstration. Or, elles devraient !! En effet, c'est pendant cette réunion qu'elles peuvent exprimer leurs points de vue. Si elles ne le font pas, personne ne le fera à leur place et, dans ce cas, le produit final ne correspondra peut-être pas tout à fait à leurs attentes.

Cette réunion peut suivre la démarche suivante :

– préparer l'environnement de démonstration : salle, slides, infrastructure… ;

– inviter les participants (équipe et parties prenantes) ;

– en début de réunion, rappeler les objectifs du « Sprint », présenter les « *Stories* » réalisées et l'organisation de la démonstration ;

– dérouler la démonstration ;

– capitaliser les échanges (remarques, commentaires, compléments…) ;

– calculer la vélocité réelle de l'équipe.

Note : La durée de cette réunion est en général comprise entre deux et quatre heures.

3.6.5 Réunion de rétrospective

La réunion de rétrospective est organisée à la fin de chaque « Sprint », à la fin de chaque « Release » et à la fin du projet.

L'objectif de cette réunion est de faire ressortir de manière factuelle les éléments positifs et négatifs concernant la période analysée. Afin de rendre ce moment ludique nous pouvons utiliser un « *Serious game* ». Un bon exemple est le « *Speed Boat* » (figure **6**). Il s'agit d'un jeu conçu pour réfléchir ensemble et identifier les forces, les faiblesses…, par exemple d'un processus de développement.

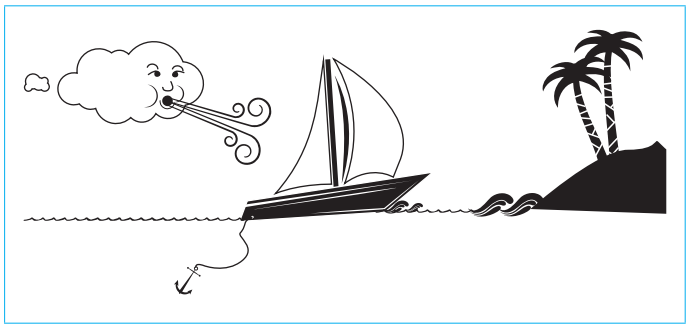

Figure 6 – Un exemple de « Speed Boat »

Les règles du jeu sont les suivantes.

Phase 1

1/ Un membre de l'équipe dessine sur le tableau : un bateau à voile avec une ancre et une île.
2/ Chaque membre de l'équipe dispose de 5 minutes pour écrire sur n « post-it » de couleur (par exemple : vert, rouge et bleu) ce qui s'est bien passé, ce qui s'est mal passé et ce qui est indéfini, mais qui pose question.
3/ Quand les 5 minutes se sont écoulées, chaque membre de l'équipe colle ses post-it verts sur la voile, ses post-it rouges sur l'ancre et ses post-it bleus sur l'île.

Suite à l'étape numéro 3 (collage des post-it), nous pouvons visualiser l'état du projet instantanément. Par exemple, la voile est couverte de post-it et il n'y a pratiquement rien sur l'ancre et sur l'île (figure **7**) ! Dans ce cas, le projet a le vent en poupe et avance rapidement. Nous pouvons avoir également une situation inverse, à savoir, tout sur l'ancre et rien sur la voile ! Dans ce cas, la situation est critique, le bateau n'avance pas ! Et bien sûr nous pouvons également avoir une situation moyenne à savoir une voile bien remplie, une ancre et une île moyennement remplis ! Dans ce cas, le bateau avance, mais difficilement.

Phase 2

1/ Un membre de l'équipe analyse l'ensemble des post-it pour éliminer tous les doublons en accord avec l'équipe.
2/ Ensuite, nous avons le choix, soit nous chercherons à éliminer les points qui nous ralentissent, soit nous chercherons à développer nos points forts pour devenir encore plus performants. À noter que ce qui est indéfini sera systématiquement analysé et sera, soit reclassé sur l'ancre ou la voile, soit abandonné. À l'issue de ce travail, nous aurons défini un plan d'actions, mais attention, il faut savoir, que trop d'actions tuent l'action ! Notre recommandation est de traiter un ou deux points, ceux qui paraissent essentiels, et donc de créer une, deux ou trois actions, pas plus. Ces actions devront être traitées lors d'un prochain « Sprint ».

Note : La durée de cette réunion est en général d'une heure, pas plus.

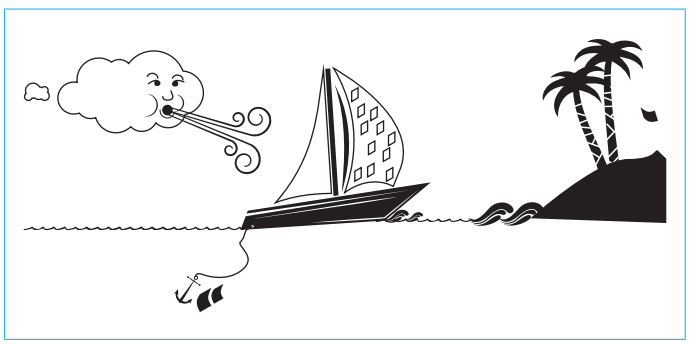

Figure 7 – Un exemple de « Speed Boat » qui avance (tout sur la voile)

3.7 Indicateurs

Parmi les nombreux indicateurs proposés par SCRUM, nous en avons retenu trois qui nous paraissent essentiels, la « Vélocité » le « Burn-down chart » et le « Burn-up chart ».

3.7.1 Vélocité

La « Vélocité » correspond au nombre de points de complexité que l'équipe est capable de réaliser pendant un « Sprint ». La « Vélocité » est propre à une équipe, elle permet de mesurer la progression de l'efficacité de l'équipe. Il ne faut surtout pas l'utiliser pour comparer des équipes, ce serait une erreur monumentale. Nous insistons sur ce point car les managers ont tendance à penser que cette métrique est un indicateur de comparaison d'équipes, alors qu'en réalité c'est un indicateur de progression d'une équipe.

À noter qu'au début du projet, la « Vélocité » de l'équipe est inconnue, il faudra donc se baser sur des hypothèses et l'ajuster au fur et à mesure de l'avancement. Il faut considérer au moins trois « Sprints » pour trouver la bonne valeur. À noter également que si l'équipe change en cours de projet, par exemple, suite au remplacement d'un équipier (chevronné ou non) au début d'un « Sprint », la « vélocité » sera également à recalibrer.

3.7.2 Burn-down chart

Ce graphique permet de suivre le « Reste à faire » pendant le « Sprint ». La courbe indique au jour le jour, le nombre de points de complexité réalisés. Elle est mise à jour au fil de l'eau, l'idéal est qu'elle atteigne zéro le dernier jour du « Sprint ».

3.7.3 Burn-up chart

Ce graphique permet de suivre la progression du produit (projet) « Sprint » après « Sprint ». La courbe indique le nombre de points de valeur acquis, l'idéal étant qu'elle atteigne la valeur du « Backlog » de produit (projet) à la fin du projet.

4. Illustration par un exemple

Il s'agit de réaliser une application client/serveur permettant de gérer toutes les activités nécessaires au bon fonctionnement d'une école de danse, son nom « Salsa World » : les pré-inscriptions, les inscriptions, le suivi comptable (entrées, sorties – paiement des professeurs…), le suivi des plannings, l'organisation et le suivi des événements, la communication (site web, Twitter, Facebook…), la logistique…

Voici quelques exemples de « *Stories* » permettant d'initialiser le travail :

« *Story* ED0001 »

En tant que

Responsable des inscriptions

Je veux

Pouvoir saisir et visualiser les informations concernant l'adhérent depuis mon portable, tablette ou *smartphone*, à savoir : Nom, Prénom, Adresse postale, Adresse Email, Mobile, Type d'adhésion (une danse, multidanse, cours particulier, en couple), Mode de paiement (liquide, chèque, carte bleue), annuel ou mensuel, Suivi du paiement.

Parce que j'en ai besoin pour

Communiquer ainsi que pour réaliser le suivi des encaissements,

Et je le validerai comme suit

En demandant une visualisation des informations (préalablement saisies) concernant une liste d'adhérents choisie pour les tests.

« *Story* ED0002 »

En tant que

Responsable de la communication

Je veux

Pouvoir informer les adhérents des faits du jour, des modifications de planning, des événements à venir, des stages ponctuels..., depuis mon portable, tablette ou *smartphone*, par Email et Facebook

Parce que j'en ai besoin pour

Maintenir l'intérêt et ainsi assurer une animation active de l'école

Et je le validerai comme suit

D'une part, en utilisant la fonction d'envoi d'Email à une liste d'adhérents choisis pour les phases de test, et d'autre part, en me connectant à la page Facebook de « Salsa World » à partir de notre application.

« *Story* ED0003 »

En tant que

Responsable du suivi des profs

Je veux

Pouvoir saisir les heures réalisées semaine par semaine (depuis mon portable, tablette ou *smartphone*), disposer d'une fiche « prof », (préalablement saisie, du même type que celle des adhérents) avec les informations suivantes : Nom, Prénom, Type de cours (ex : Salsa Niveau 1, 2...), Adresse postale, Coût brut de l'heure, Email, Mobile, RIB

Parce que j'en ai besoin pour

Assurer le suivi des paiements (sur réception de facture) à partir des heures effectuées. À noter que l'application doit impérativement notifier le responsable du suivi des profs par SMS, toutes les semaines, pour lui demander la saisie des heures et tous les mois, pour lui demander de réaliser les paiements.

Et je le validerai comme suit

D'une part, en demandant une visualisation des informations (préalablement saisies) concernant une liste de professeurs choisie pour les tests, et d'autre part, en réalisant plusieurs paiements « test » à partir de factures « test ». Enfin, les notifications attendues doivent arriver sous forme de SMS sur le mobile du responsable du suivi des profs.

Après avoir écrit « toutes » les « *Stories* » (pas forcément, mais un nombre important tout de même, ne pas oublier que nous devons avoir au moins deux « Sprints » d'avance) et éventuellement, après les avoir classées par « Feature(s) », par exemple, ici : « Communication », « Comptabilité »... nous pouvons élaborer un planning et définir l'objectif de la première « Release » et du premier « Sprint » suivant le « Sprint 0 ». Nous allons partir de l'hypothèse, issue de notre expérience concernant ce type de réalisation, que nous allons effectuer deux « Releases » de trois mois et que le « Sprint » de base est de dix jours ouvrés. Nous allons donc réaliser un projet d'une durée de six mois comprenant deux « Releases » et six « Sprint(s) ».

Nous allons mettre en place l'équipe suivante :

– un PO, le responsable de l'école qui jouera tous les rôles dans un premier temps, à savoir, responsable des inscriptions, responsable de la communication...

– un SM, le responsable du bon déroulement du projet délégué par l'entreprise en charge de réaliser l'application ;

– deux équipiers, également délégués par la société en charge de réaliser l'application.

Le SM ainsi que les deux équipiers doivent maîtriser les techniques informatiques suivantes : JAVA (pour la partie interface homme machine côté client), JAVA/JEE (pour la partie serveur) et PosgreSQL (pour la partie stockage).

L'objectif choisi de la première « Release » est de réaliser la partie « Communication ». Cependant, comme préconisé, nous allons débuter le travail par un « Sprint » particulier, le « Sprint 0 ».

Lors de ce dernier, nous avons d'une part figé les « *stories* » de référence, qui nous permettront de réaliser les évaluations, défini le « prêt » et le « fini » qui nous permettront de ne pas tomber dans les problèmes décrits dans cet article, réalisé un premier schéma d'architecture physique, initialisé les documentations (spécifications, conception, manuel d'utilisation...), mis en place l'intégration continue et validé les technologies. Nous ne détaillerons pas plus ce travail dans cet article, la figure **8** présente un schéma de la première version de l'architecture.

Le « Sprint 1 » va donc consister à implémenter les *stories* « prêtes » concernant la « communication ». Dans notre liste incomplète (ce n'est pas l'objectif de cet article de réaliser un exemple complet) de « *stories* », cela concerne « *Story* ED0002 ».

Nous sommes partis de l'hypothèse que la « Vélocité » de notre équipe est de « 16 », nous affinerons cette vélocité au fil des « Sprint(s) ».

Suite au « Planning poker », notre « *Story* » a été évaluée à « 5 » points, nous pourrons donc ajouter d'autres « *Stories* » pour atteindre « 16 », par exemple une de « 5 » et deux autres de « 3 ». Ne pas oublier qu'une « *Story* » ne peut pas représenter plus du tiers du volume total du « Sprint », dans notre cas « 5 ».

Nous allons donc maintenant découper « ED0002 » en tâches :

– Tâche 0 (PO, SM, Équipier 1 et Équipier 2) :

- Compléter les documents de test, de conception et d'utilisation ;

– Tâche 1 (SM) :

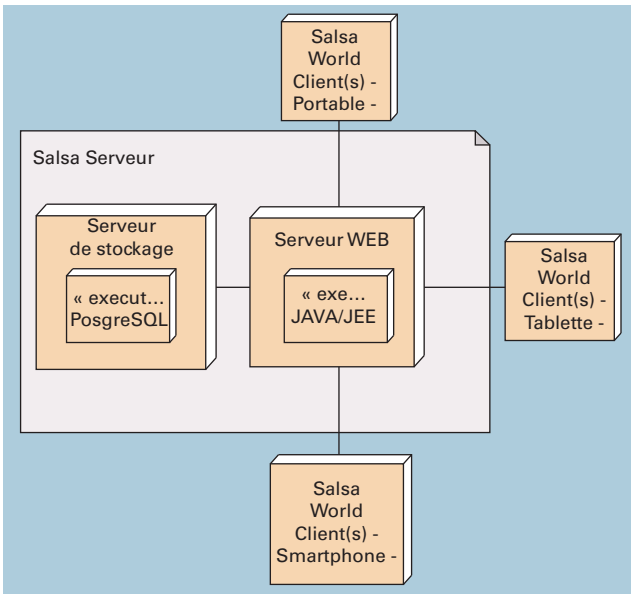

Figure 8 – Première version de l'architecture (non sauvegardée et non redondée)

- Mise en place de la base de données PostgreSQL et de l'architecture physique (différentes composantes du serveur) ;

– Tâche 2 (Équipier 1) :

- Créer dans l'interface homme machine client, un bouton « Envoyer mail d'information »,
- Sur activation de ce bouton, afficher une zone de saisie libre proposant un ensemble simple d'options de traitement de texte (Chapitre, gras, italique,...),
- Sur sauvegarde de cette zone, l'outil demande la liste des destinataires (on pourra choisir tous les adhérents, un groupe d'adhérents particulier, par exemple juste les adhérents Salsa, ou saisie libre des mails). Le choix étant validé, le mail est envoyé et sauvegardé dans la base interne de l'application, à des fins d'historisation. Bien sûr, cet historique devra pouvoir être consulté depuis notre application. Par exemple via un bouton « Consulter historique des mails » ;

– Tâche 3 (Équipier 2) :

- Créer la page Facebook, contenu à définir et à débattre pendant le « Sprint » courant ;

– Tâche 4 (Équipier 2) :

- Configurer notre application pour qu'elle puisse s'interfacer avec Facebook ;

– Tâche 5 (SM) :

- Réalisation des contrôles qualité ;

– Tâche 6 (PO) :

- Réalisation des tests depuis un portable, une tablette et un *smartphone* sur les Os : Android, IOS, Linux et Microsoft !

Nous avons affecté chaque « tâche » à un membre de l'équipe, juste pour l'exemple. Dans la réalité, les « tâches » sont distribuées sur le « Dashboard » et chaque équipier s'en approprie une en fonction de ses compétences et des tâches restantes. À l'exception du PO à qui nous réservons les « tâches » de spécification et de validation.

Le « Sprint » va ainsi se dérouler jusqu'à la « revue de Sprint ». Bien sûr, nous aurons fait au préalable la « revue du backlog » et la « rétrospective ».

Puis nous passerons au « Sprint » suivant, puis à la « Release » suivante jusqu'à la présentation finale du résultat.

5. Conclusion

Dans cet article, nous avons vu les concepts essentiels du SCRUM permettant de réaliser un projet opérationnel en suivant cette approche. Après une lecture détaillée, vous serez en mesure de pratiquer le SCRUM et de réussir des projets de petite taille. Essayez, le résultat vous surprendra. Ne pas oublier que dès que l'on dépasse dix équipiers nous devons mettre en place une organisation de plus haut niveau et ainsi réaliser du SCRUM de SCRUM.

Les conditions de succès lorsque nous appliquons SCRUM sur un projet sont liées à plusieurs points essentiels, d'abord le type de projet.

En effet, pour un projet de R&D, pour un projet opérationnel « pas clair, mal spécifié... » ou pour une réponse à un appel d'offres (gérée comme un projet), le SCRUM est idéal, pourquoi ? Car dans ces cas, nous devons obtenir la meilleure solution/proposition possible en respectant une durée fixe, à partir de spécifications légères, floues et/ou incomplètes.

Si le projet est très bien spécifié, et que le client ne souhaite pas s'impliquer, qu'il est réticent à changer les spécifications, à les remplacer par d'autres, à les supprimer, à en créer des nouvelles... en gros qu'il ne veut rien changer, parce qu'il veut ce qu'il a commandé, même si à la fin des fins le produit ne correspond pas aux besoins du moment, on est dans une situation délicate ! Dans ce cas on peut cependant pratiquer des techniques agiles choisies, toujours utiles (Ex : rétrospective...) mais il est prudent de gérer le projet de manière traditionnelle. Par contre si le client est ouvert, qu'il accepte les concepts Agile et qu'il est prêt à s'impliquer, là il faut foncer.

L'autre cas où l'utilisation de l'Agilité pose problème, c'est quand l'une des deux parties (voire n parties si nous sommes dans un projet multipartenaires) refuse la transparence. Sans transparence pas d'agilité, ceci est une règle d'or, on avance en équipe et en confiance. Retenez donc que sans confiance et sans acceptation de la transparence, l'agilité sera une source de conflits sans limite !

Enfin au niveau de l'équipe, le SCRUM remet l'humain au centre, ce qui est une très bonne chose, c'est même génial. Une équipe SCRUM avance ensemble, nous ne parlons plus d'individualités mais d'équipe. En effet, dans une équipe il y a toujours des « très bons », des « moyens », des « moins bons »... Oui, certes ! Mais quelle que soit la compétence d'un équipier, ce qu'il fait, il le fait bien, il mérite donc le respect de tous. Qu'il résolve un verrou technique ou qu'il modifie un document de base, le travail est, et doit être fait. Donc pas de condescendance pas de conflits, on avance ensemble et on gagne.

Une bonne équipe SCRUM peut soulever des montagnes, essayez et vous verrez très rapidement ce qui change. Elle est unie, travaille en synergie et avance pour réussir.

6. Sigles, notations et symboles

Symbole	Description
SAFe	Scaled Agile Framework
PO	Product Owner
SM	SCRUM Master
SH	Stakeholder

Méthode Agile SCRUM

« Comment l'utiliser de manière opérationnelle ? »

par **Agusti CANALS**
Directeur d'Unité Fonctionnelle (Technique)
CS Communication & Systèmes Toulouse, France

Sources bibliographiques

[1] AUBRY (C.). – *Scrum – 5e éd. – Pour une pratique vivante de l'agilité* – Dunod Éditeur, 16 mai 2018.

[2] SUTHERLAND et SCHWABER (K.). – *Le Guide définitif de Scrum : les règles du jeu* – Juillet 2016. https://scrumguides.org

À lire également dans nos bases

PRINTZ (J.). – Méthodes Agiles *Technologies de l'information, Génie logiciel.* Technologies de l'information, technologies logicielles, architecture des systèmes [H 3 202], 10 février 2010.

Outils logiciels

ICESCRUM : La nouvelle génération d'outil de gestion de projet agile :
https://www.icescrum.com/fr/

TULEAP : Développement et gestion de projet agile :
https://www.tuleap.org/

Événements

Agile TOUR :
http://at2018.agiletour.org/
et
https://www.agiletoulouse.fr/
(25 & 26 Octobre 2018)

Annuaire

Constructeurs – Fournisseurs – Distributeurs (liste non exhaustive)

CS Communication & Systèmes : préconise l'utilisation de SCRUM et fournit des prestations d'accompagnement à la demande
http://www.c-s.fr

Organismes – Fédérations – Associations (liste non exhaustive)

Meetup :
https://www.meetup.com/fr-FR/Agile-Toulouse/

Techniques de l'Ingénieur

Appuyez-vous sur l'expertise technique et scientifique de référence.

Gagnez du temps et sécurisez vos projets en utilisant une source actualisée et fiable.
+ de 10 000 articles de référence, articles interactifs, fiches pratiques et parcours pratiques, rédigés par les meilleurs experts et validés par des comités scientifiques.

15 domaines d'expertise

- ✓ **Automatique - Robotique**
- ✓ **Biomédical - Pharma**
- ✓ **Construction et travaux publics**
- ✓ **Électronique - Photonique**
- ✓ **Énergies**
- ✓ **Environnement - Sécurité**
- ✓ **Génie industriel**
- ✓ **Ingénierie des transports**
- ✓ **Innovation**
- ✓ **Matériaux**
- ✓ **Mécanique**
- ✓ **Mesures - Analyses**
- ✓ **Procédés chimie - bio - agro**
- ✓ **Sciences fondamentales**
- ✓ **Technologies de l'information**

NOS ÉQUIPES SONT À VOTRE DISPOSITION

Par téléphone
☎ 33 (0)1 53 35 20 20

Par email
infos.clients@teching.com

www.techniques-ingenieur.fr