Description de la méthode SCRUM à travers deux expériences en entreprise

Gérard Memmi

▶ **To cite this version:**

Gérard Memmi. Description de la méthode SCRUM à travers deux expériences en entreprise. [Rapport de recherche] LTCI - Laboratoire Traitement et Communication de l'Information [Paris]. 2020. hal-03089815v3

HAL Id: hal-03089815
https://hal.science/hal-03089815v3

Submitted on 15 Dec 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Description de la méthode SCRUM à travers deux expériences en entreprise

Gérard Memmi
LTCI, Telecom Paris, Institut Polytechnique de Paris

Résumé : Deux expériences conduites dans deux jeunes pousses américaines servent de base pour décrire et discuter sur la méthode scrum. Le contexte de jeune pousse rend encore plus critique l'importance des questions relatives à l'affectation de ressources limitées et au respect rigoureux des délais. Après avoir introduit le manifeste Agile, le processus scrum est décrit. Les concepts désormais classiques de backlogs accompagnés des autres artefacts de la méthode Scrum sont intégrés à des artefacts de développement produit plus traditionnels afin de passer à l'échelle. Le concept de sprint est légèrement modifié pour mieux s'adapter à un environnement où les ressources sont rares. La gestion des livraisons produit est ensuite discutée en particulier celles correspondant à des versions majeures et critiques du produit. Ces dernières impliquent d'organiser et de coordonner plusieurs équipes scrum. Cet aspect de la méthode, tout du moins pour les deux expériences considérées, est essentiel. Un ensemble de questions fondamentales doivent être résolues pour que la méthode scrum soit adoptées par les équipes de développement et réussisse à passer à l'échelle. Ce rapport se termine par plusieurs recommandations pour améliorer la productivité du développement produit dans un contexte similaire à nos deux jeunes pousses.

Mots clés : Manifeste Agile, méthode scrum, backlog, sprint, jeune pousse, génie logiciel

Abstract: Two experiments conducted in two American seedlings serve as a basis for describing and discussing the scrum method. The context of a start-up makes the importance of issues related to the allocation of limited resources and strict adherence to deadlines even more critical. After introducing the Agile Manifesto, the scrum process is described. The now classic concepts of backlogs along with the other artifacts of the Scrum method are integrated with more traditional product development artifacts in order to scale up. The concept of sprint is slightly modified to better adapt to an environment where resources are scarce. The management of product deliveries is then discussed, in particular those corresponding to major and critical product versions. The latter involve organizing and coordinating several scrum teams. This aspect of the method, at least for the two experiments considered, is essential. A set of fundamental issues must be resolved for the scrum method to be adopted by the development teams and succeed to scale up. This report concludes with several recommendations to improve the productivity of product development in a context similar to our two start-ups.

Keywords: Agile Manifesto, Scrum method, Backlog, Sprint, Start-up, Software Engineering

Note : Une version préliminaire de ces notes beaucoup moins détaillée a été publiée dans [Memmi 11]. Ces notes contiennent de nombreux ajouts, plus un état de l'art plus complet et mis à jour.

1. INTRODUCTION

La méthode de développement logiciel appelée *scrum* (qui signifie mêlée en termes de rugby) est ici décrite à travers deux retours d'expériences dans le cadre de deux jeunes pousses ayant un écosystème différent. Nous étudions la réalisation d'un produit nécessitant une équipe de taille trop importante pour être contenue dans un seul processus scrum ; il est d'usage d'avoir recours à la notion de « scrum de scrums ». Outre qu'elle introduit un niveau de hiérarchie contraire aux principes de la méthode Agile, et sans prétendre introduire de principes méthodologiques vraiment nouveaux, il nous est vite apparu qu'il nous fallait modifier cette notion de scrum de scrums, en particulier du fait que nos projets se développaient au sein de jeunes pousses, une première fois dans une jeune pousse n'ayant pas encore de base client développée, une seconde fois dans une jeune pousse ayant déjà de nombreux utilisateurs. Nous tenterons d'expliquer comment ce passage à l'échelle en termes de force de développement fut réussi.

Le mot « scrum » est évocateur : la mêlée, en rugby, est le moment où une partie de l'équipe a un objectif commun et va se rassembler, s'auto-organiser, se souder et rapidement décider d'un plan d'actions prenant en compte divers éléments de la rencontre. C'est un effort intense pendant un temps assez court qui est demandé à chaque fois que l'arbitre le décidera durant la rencontre. Le taux de succès de prise de balle à la mêlée est considéré comme essentiel pour remporter la partie. Cette notion d'équipe où chacun a une fonction précise, où chacun doit savoir s'adapter rapidement aux évènements est au cœur de la méthode Scrum. On commence à comprendre dès lors quelques principes fondamentaux de la méthode à savoir d'établir de petites équipes très

soudées, dans un effort court et bien défini, concentrées sur un objectif précis et circonscrit. Au bout de cette période, il sera impératif de livrer un code fonctionnel de qualité dont les clients perçoivent la valeur. Avant de décrire plus en détails les aspects essentiels de la méthode, donnons quelques éléments du contexte dans lequel nos deux projets ont été développés.

2. DEUX EXPÉRIENCES UTILISANT LA MÉTHODE SCRUM

La méthode scrum est décrite en exploitant deux exemples où elle a été déployée et utilisée dans les années 2000. Ces deux expériences ont été menées au sein deux jeunes pousses (ou start-ups). Le fait même qu'il s'agisse de jeunes pousses n'est pas anodin : dans toute jeune pousse les ressources sont rares et chères ; les processus de remontée d'information atteignent fréquemment le président directeur général ; chaque erreur est vite visible dans la mesure où elle peut avoir un impact important sur l'entreprise. Chaque nouvelle opportunité d'affaire peut secouer l'entreprise entière en quelques jours et peut potentiellement imposer de changer les plans de développement produit de façon considérable et parfois brutale.

La première entreprise développait un produit de CAO (un environnement de tests de très grands circuits électroniques intégrés (ASIC)); il s'agissait donc de développer des algorithmes originaux, de concevoir des outils de compilation et stockage de données hautement complexes. Autre élément important : la jeune pousse partait d'une table rase sans aucun client préalable. Deux conséquences immédiates : pas de pression de la part d'une base de clients (cependant bien sûr, une forte pression sur l'emploi du Conseil d'Administration composé en partie des financiers de la jeune pousse) à livrer le produit et des cas d'usage validés seulement lors de trop rares réunions avec des clients potentiels et très discutés (voire trop simplifiés) en interne par l'équipe de développement.

La seconde entreprise se trouvait dans une situation très différente. Elle développait un produit dans le domaine de la santé (traitement collaboratif du dossier médical). Sa base de clients et d'utilisateurs était importante au moment où la décision fut prise d'appliquer la méthode scrum. Les besoins en développement étaient donc exprimés de manière beaucoup plus fiable et précise que dans le premier cas. La pression des clients était très importante quant à la livraison du produit dans les délais impartis. Les utilisateurs exigeaient une interface très soignée et très proche de leur habitudes ce qui était en fort contraste avec la première expérience où le produit développé était destiné à des ingénieurs intéressés par une interface programmable. L'équipe de tests prenait plusieurs jours pour tester manuellement toutes les possibilités offerte par l'interface sur des scénarios définis avec précision. Des rapports de défauts pouvaient arriver avec plusieurs jours de délai ce qui n'était pas sans poser des problèmes importants d'intégration et de convergence de livraison produit. Autre différence notable : la première expérience se situait dans le cadre d'un développement d'un produit nouveau dont les fonctions et contours étaient à définir alors que pour la seconde expérience le produit était déjà défini et il s'agissait de le faire évoluer et de passer à l'échelle afin de satisfaire une base d'utilisateurs plus vaste et plus variée [Memmi 09].

Les sections suivantes décrivent la méthode scrum telle qu'elle a été rédigée dans [Schwaber… 02] ou [Schwaber 04] en soulignant de nombreux points très positifs mais également telle qu'elle a été interprétée, intégrée aux méthodes et aux données existantes des deux jeunes pousses et surtout adaptée pour tenir compte de l'organisation et de la culture de chaque entreprise ainsi que de la situation et de la nature particulières de chaque produit. Ce mode d'introduction d'une méthode de travail est complètement consistant avec le principe d'adaptabilité exprimé dans le Guide Scrum [Sutherland…17] ainsi que dans la méthode Agile décrite dans la section suivante. Nous nous attacherons à analyser et expliquer les éléments de la méthode qui nous ont paru les plus intéressants.

3. MANIFESTE de la MÉTHODE AGILE et NOTIONS de BASE de la MÉTHODE SCRUM

La méthode Scrum a été décrite pour la première fois dans [Takeuchi…86] où les auteurs mettent l'accent sur la vitesse d'exécution et l'agilité avec l'intention d'augmenter de façon significative la productivité d'équipes de développement de produits industriels. De manière similaire, Ken Schwaber, qui a adapté la méthode au développement logiciel vers les années 90-96, fait état de gains de productivité de plusieurs ordres de grandeur [Schwaber… 02]. D'autres méthodes agiles respectant le manifeste Agile [Agile 01] sont souvent rapprochées de la méthode scrum : la programmation extrême [XP…99] ou encore les méthodes DSDM (Dynamic System Development Method) [DSDM…95] [DSDM…14], SAFe [Safe 17] [Safe 19], FDD (Feature-driven Development) [Palmer…02], ou Kanban [Anderson…16] pour n'en citer que quelques-unes.

3.1. Manifeste de la méthode Agile

Le manifeste de la méthode Agile [Agile 01] énoncée dans ses douze principes dans la Figure 1: le Manifeste Agile tiré de [Agile 01] ci-dessous a été signé en février 2001 par 17 professionnels de la programmation dont K. Schwaber, J. Sutherland et M. Beedle qui sont les principaux auteurs de la méthode Scrum ([Schwaber… 02] et [Schwaber…17]). La méthode Agile est aujourd'hui très répandue et repose sur quelques idées simples telles l'attention et le dialogue constant avec le client ou l'utilisateur. Ce dialogue permet à la fois de mieux comprendre et de suivre au plus près l'évolution du besoin client. Elle ouvre la porte à l'acceptation de changements pouvant être tardif durant le développement du produit ce qui se démarque nettement des méthodes plus classiques comprenant une spécification qui une fois figée sera suivie d'un développement puis de tests telles que le cycle de vie du logiciel en V ou en cascade. Deux autres points importants du manifeste de la méthode Agile sont d'une part l'intention de délivrer régulièrement de fréquentes versions incrémentales du produit et d'autre part la volonté forte de maintenir et constamment améliorer les performances de l'équipe de développement à travers des revues et des retours d'expérience permettant également d'adapter les différents outils de production et processus aux besoins d'évolution du produit.

	Énoncé des douze principes de la méthode Agile
1	*Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.*
2	*Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.*
3	*Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.*
4	*Business people and developers must work together daily throughout the project.*
5	*Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.*
6	The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7	*Working software is the primary measure of progress.*
8	*Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.*
9	Continuous attention to technical excellence and good design enhances agility.
10	Simplicity—the art of maximizing the amount of work not done—is essential.
11	The best architectures, requirements, and designs emerge from self-organizing teams.
12	At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Figure 1: le Manifeste Agile tiré de [Agile 01]

Dans ce qui suit, nous ferons parfois référence à l'un ou l'autre de ces principes au moment de son application.

3.2. Composants principaux du processus scrum

La méthode scrum opère sur un ensemble de composants que l'on peut séparer en trois (comme pour de très nombreuses méthodes au demeurant) : les artefacts, les étapes principales du processus, les différents rôles que les participants remplissent.

1. *Les artéfacts* :

le « *backlog produit* » contient les diverses fonctions et caractéristiques non fonctionnelles qu'il est désirable de développer. Ces fonctions peuvent être décrites dans des « *story cards* » qui peuvent être disposées sur un tableau de type kanban afin d'obtenir une vue synthétique du statut des travaux en cours. Le « *backlog de sprint* » est un sous-ensemble du backlog produit. Le tableur de charge de travail (graphique d'avancement ou *burndown chart*) est utile afin de mesurer la charge de travail restant à effectuer, de surveiller et constater la convergence ou la divergence du sprint. Le *kanban* permet à tous de visualiser essentiellement le travail en cours.

2. *Les étapes du processus* :

le « *sprint* » est le niveau d'organisation le plus petit ayant pour but le développement d'un quantum du livrable ; la réalisation de livraison produit (ou version produit) (release en anglais). Les *réunions journalières* (daily

meetings) obligatoires des équipes de développement sont conçues pour rendre compte de l'évolution des travaux en cours et de pouvoir être aussi prédictible que possible.

3. *Les équipes et les différents rôles* :

Le « *gardien du produit* » maintient le backlog produit. Le « *maître de scrum* » anime un sprint, les membres de l'équipe de développement ou « *équipe scrum* » vont participer aux différents sprints qui constituent le développement d'une version du produit.

Dans la suite, nous décrirons ces éléments de la méthode scrum. En effet, ils nous ont semblé saillants dans nos deux expériences. Nous essayerons de préciser quand et comment nous avons parfois divergé afin d'une part prendre en compte le fait d'être une jeune pousse aux ressources limitées et d'autre part intégrer la méthode avec des artéfacts existant ainsi qu'avec tous les participants à la réalisation du produit. Souvenons-nous qu'une méthode n'a pas à être suivie à la lettre mais plutôt dans son esprit. Elle doit toujours être acceptée et pour cela adaptée, progressivement intégrée au milieu dans lequel elle va être utilisée.

3.3. « *Pigs* » et « chickens »

Les « *pigs* » et les « *chickens* » (termes qu'on se refusera à traduire par cochons et poulets tant différentes sont les connotations attachées à ces mots entre l'anglais et le français) sont deux attributs séparant en deux camps l'ensemble des participants à la réalisation du produit. Les désignations proviennent d'une blague américaine [Schwaber 04] dont la chute insinue que les « *pigs* » vont en quelque sorte donner leur sang pour le succès d'un projet, ils sont donc directement engagés ; alors que ce ne sera nettement pas le cas des « chickens » qui s'impliquent et contribuent ou a minima observent et encouragent les « pigs ».

Dans nos deux expériences, les ingénieurs ont vraiment tout donné pour livrer leurs développements dans les temps avec la plus haute qualité possible ; sans conteste possible ils faisaient partie des « *pigs* » tout comme le gardien du produit. Dans le cadre de jeunes pousses les enjeux sont tellement élevés sur le fait de livrer le plus tôt possible que tous les membres de l'entreprise s'intéressent de près ou de loin à la réalisation du produit (surtout si l'entreprise n'a qu'un seul produit comme ce fut le cas dans nos deux expériences!); à ce titre, tous les membres de l'entreprise sont au moins des « chickens ». Ainsi, le management (tout comme les clients ou prospects) font partie des « chickens ». Qu'ils soient « pigs » ou « chickens », tous les acteurs de l'entreprise surtout si celle-ci est une jeune pousse sont soumis à de hautes doses de stress pour délivrer dans les temps le meilleur produit possible. Les « chickens » ne sont pas simplement en train d'observer calmement la réalisation du produit. Certains « chickens » essayeront d'orienter le développement ou de changer certaines fonctions ; ce sera le rôle du gardien du produit de documenter et classer les bonnes idées ou du maître de scrum de protéger les équipes de ces pressions. D'autres « chickens » loin d'être inutiles peuvent donner de précieuses idées pour le backlog produit ; ils se doivent par ailleurs d'être présents quand les choses tournent mal, dans des situations de crise et de s'impliquer afin de trouver des solutions.

4. « BACKLOG PRODUIT » ET AUTRES ARTEFACTS DE LA MÉTHODE SCRUM

L'objet central de la méthode est certainement le backlog produit. De nombreux auteurs ont adaptés et enrichi l'ensemble des artefacts utiles à l'application efficace de la méthode scrum. Ces éléments sont par essence très simple et chacun est libre de les adapter à son écosystème. Ainsi, le graphe d'avancement ou burndown chart se retrouve chez Google ([Striebeck 06], ou le kanban chez SAFe ([Safe 19]).

4.1 Backlog produit, story card et backlog de sprint

Le backlog produit et sa décomposition en backlogs de sprint est un artefact central de la méthode scrum. Le backlog produit contient la liste ordonnée de tous les éléments du produit qui nécessitent un développement. Les éléments du backlog produit peuvent être très variés et peuvent être fonctionnels ou non ; de simples défauts venant de la base d'utilisateurs via l'équipe de support ou des composants nécessitant des développements importants.

Ces éléments sont identifiés et évoluent dynamiquement soit au cours des développements et de la qualification des versions précédentes, soit le long des travaux de la direction technique, mais surtout lors des nombreuses rencontres avec les clients ou utilisateurs, en particulier lors des rencontres avec les groupes utilisateurs (User Groups) qui, bien sûr restant en ligne avec un des principes essentiels de la méthode Agile, ont un poids important sur l'évolution du produit.

Dans ce dernier cas, ces éléments ou « histoires d'utilisateurs» (user story) peuvent être décrits de manière concise dans des « story cards ». Ces cartes auront un format bien défini. Comme le montre la Figure 2, elles ont une identité, un titre et une description des actions composant « l'histoire ». Ces actions mettent en jeu un ou plusieurs participants qui chacun tiennent un rôle supporté par le produit. Enfin, la carte doit également décrire les conditions que doit satisfaire la réalisation de l'histoire. Ces conditions pourront servir plus tard à engendrer des tests unitaires puis à valider le produit. Ces histoires d'utilisateurs peuvent de plus avoir une première estimation de la charge de travail nécessaire pour les développer ; elles sont alors au cœur de la méthode de programmation extrême [XP 99]. On pourra également utiliser des outils simples pour les éditer, les trier ou les manipuler ([Rees 02], [Dwitam…20]).

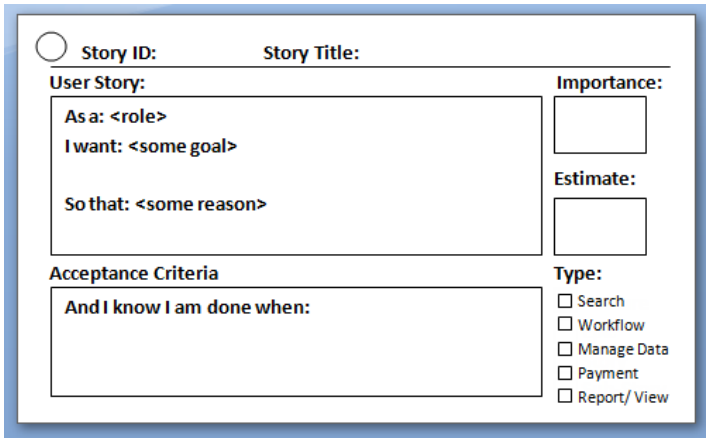

Figure 2: Exemple de Story Card, hormis l'histoire et les conditions d'acceptation, on indiquera l'importance et une première estimation de la charge de travail des fonctions décrites dans l'histoire (tirée de https://mazoea.wordpress.com/agile/

Chaque élément du backlog produit (voir Figure 3) a une courte description, une priorité vue de la base client, une liste de clients importants demandant le changement ou impactés par lui, un effort estimé de développement (EE), une valeur perçue (VP) auprès des clients (ici, haute, moyenne, ou faible). Les éléments du backlog sont ordonnés par priorités commerciales déterminées par le gardien du produit avec l'aide de l'équipe marketing, et plus tard, parfois seulement dans le backlog de sprint, par niveaux de sévérité techniques déterminées par les équipes de développement. Les dépendances techniques entre l'élément à développer et différentes parties du code sont également enregistrées et seront analysées car elles sont sources de difficultés de développement (surtout quand les interfaces ne sont pas conçues pour cette nouvelle intégration) et également sources de régressions. Nous voyons bien que plus VP sera élevé et EE bas plus haute sera la priorité et la probabilité d'être sélectionné dans un backlog de sprint. Dans l'exemple de la Figure 3, le premier élément du tableau est clairement moins prioritaire que le second puisqu'à la fois il coute plus cher en développement (EE : une semaine contre 2 jours) et rapporte moins en valeur perçue (VP : low contre medium) et présente le même impact sur la base client que le second élément.

Priority	Product Area	Customer Perceived Value	Level of Effort	Release	Bugzilla ID	Item Name	Requested By	Customers Affected	Status	Entry	Feature Description
P2	LETTERS	LOW	1W	3.4.2	1243	Print Preview Repository	EOS	All	OPEN	5/12/08	Is there any way in which the nurses can have a repository of the letters (in Read only format) where they can go to view the letters prior to printing them?
P1	LETTERS	MEDIUM	2D	3.4.2	1162	Signatures	EOS	All	OPEN	6/6/08	The ability to include a digital signature in a letter

Figure 3: Deux éléments du backlog produit provenant d'une de nos jeunes pousses. Il est stocké dans une base de données spécialisée (ici Bugzilla). De nombreux champs sont identiques à la base de données contenant les bugs du produit.

Le backlog de sprint est un extrait du backlog produit. Dans nos expériences, il a été défini lors de réunion entre l'équipe de développement et le gardien du produit. Chaque élément du backlog de sprint est cette fois assigné à un développeur de l'équipe. L'estimé de temps de développement de chaque élément est affiné et défini en termes de jours de travail. La somme de ces estimés va permettre de calibrer le sprint et d'initialiser le tableur de calcul de la charge de travail. La méthode recommande de calibrer le sprint à 30 jours de travail ce que nous n'avons pas trouvé toujours possible.

Le gel du backlog de sprint, c'est-à-dire sa stabilisation une fois qu'il est défini, a toujours été un point important de la méthode, particulièrement difficile à respecter dans une jeune pousse. D'une part, les « chickens » vont constamment chercher à influencer et changer la définition du plan de développement. Enfin, (le mieux étant parfois l'ennemi du bien) tout développeur se rend compte régulièrement que son code a des faiblesses et est tenté de s'arrêter et de recommencer pour faire nettement mieux, plus élégant et plus maintenable. Chacun a rencontré ce double phénomène dangereux et source de divergences pour lequel nous n'avons pas de solution simple si ce n'est de revenir au strict respect des délais. À moment donné, il faut se décider à arrêter toute fluctuation dans la définition de la livraison produit et remettre toute nouvelle proposition à la version qui suivra. C'est un point délicat : il faut se souvenir du point 2 de la méthode Agile demandant de savoir accueillir tout changement même tard dans le développement.

4.2 Intégrations des backlogs dans un environnement de livraison produit

Il y a un backlog produit par produit dans l'entreprise, mais il peut y avoir plusieurs backlogs de sprint pour chaque nouvelle version produit. Ces deux catégories de backlog ne doivent pas rester isolées et sont en relation avec d'autres artefacts de la ligne produit (voir le schema synthétique Figure 4). L'un des plus importants est la vision technique du produit venant du bureau du directeur technique. Elle pourra contenir l'architecture du produit et ses évolutions ou encore des évaluations de composants logiciels qui sont candidats à une intégration future, ou encore des nouveaux algorithmes en préparation. Surtout le backlog produit et backlog de sprint ont une structure de données similaire à celle de la base de défauts ouverte à tous les membres de l'entreprise tout particulièrement les équipes support et les équipes de tests favorisant des mouvements de l'un vers l'autre, par exemple lors du triage (voir un extrait de backlog produit Figure 3). Dans nos expériences la vision marketing est directement entrée dans le backlog produit par le gardien du produit ; elle n'a donc pas d'artefact spécifique sur la description technique du produit à partager.

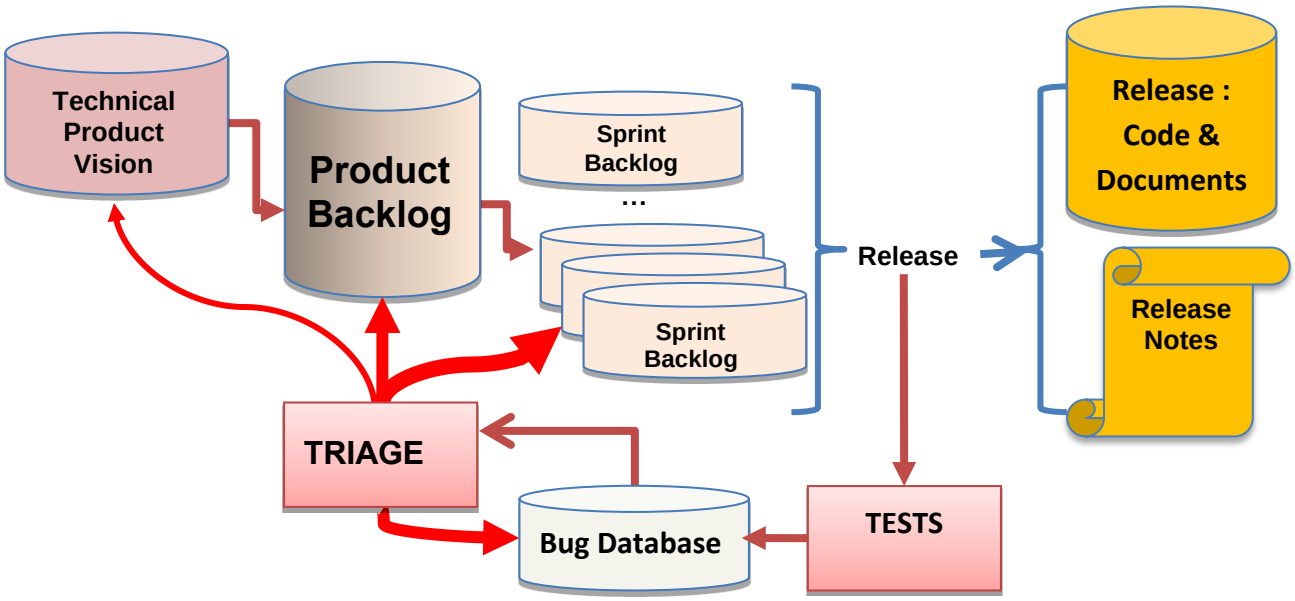

Figure 4: Le backlog produit et les backlogs de sprint doivent s'intégrer aux autres artéfacts produit : base de données de défauts, vision technologique et architecture du produit, code, tests et documentations (technique et utilisateur).

Un premier point d'intégration est de mettre en relation le backlog produit avec les travaux de l'équipe d'assurance qualité. Dans nos deux expériences cette équipe était chargée de construire une suite de tests (incluant des tests provenant des utilisateurs pour une des expériences) et de tester en continu le développement produit courant. Les tests unitaires quant à eux relevaient de la responsabilité des développeurs, ils étaient obligatoires et certains étaient engendrés automatiquement. L'équipe d'assurance qualité reporte et décrit soigneusement tout problème rencontré lors des tests dans la base de données de défauts de manière à ce que le problème soit aussi aisément que possible reproductible. Les réunions de triage peuvent après tout être considérées comme des réunions remplaçant celles de scrum de scrums puisque c'est le lieu où se rencontrent en particulier le gardien du produit, tous les maîtres de scrum, un représentant de l'architecte produit ou du directeur technique pour discuter, donner des priorités, ranger les problèmes dans le backlog produit ou la base de défauts (avec une priorité basse), dans un des backlogs de sprint (avec une priorité haute), ou encore plus rarement vers le bureau du directeur technique, car le problème pose une question technologique qui pourra donner naissance à un projet spécial et changer la vision et la feuille de route du produit. Le triage est également le lieu où les maîtres de scrum ou leurs représentants vont pouvoir échanger de façon transparente à tous de petites tâches d'un sprint vers l'autre.

Un second point d'intégration important est d'interfacer le backlog produit avec la vision produit qui appartient à l'architecte produit ou au directeur technique. Dans notre première expérience, il existait un bureau de direction technique avec deux architectes. La vision produit était constituée d'une base de données en termes de technologies à évaluer ou d'algorithmes à concevoir alors que le backlog produit était exprimé majoritairement en termes de fonctionnalités. Il en résultait une intégration parfois difficile consistant à constamment identifier les dépendances et correspondances entre les éléments technologiques et les fonctionnalités produit.

4.3 Graphique d'avancement (burndown chart)

On utilisera un graphique d'avancement ou un graphe de temps restant (burndown chart) qui sera géré par le maître de scrum (voir Figure 5 et 6). Cet outil est important pour contrôler la convergence (ou la divergence) du sprint ; il est décrit dans [Schwaber 04] ou [Meyer 14].

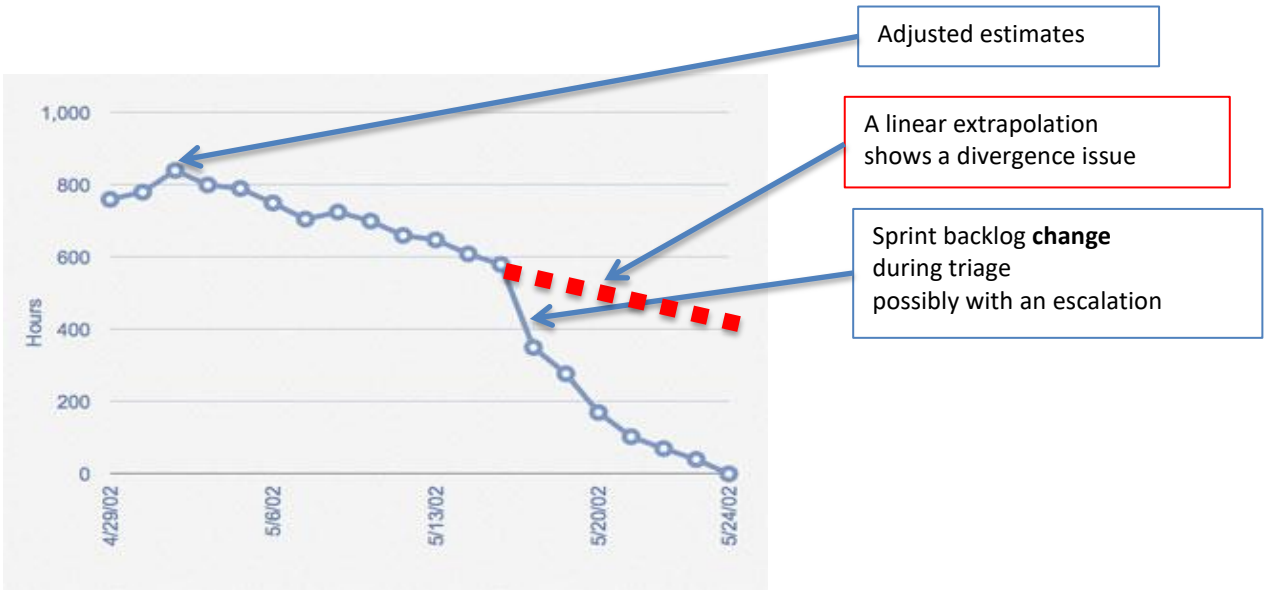

Figure 5: Graphique d'avancement ou graphe de temps (Burndown chart) : une divergence importante vers le dernier tiers du sprint est bien indiquée par une extrapolation en pointillés rouges. Cette fois il est trop tard pour ajouter un nouveau membre à l'équipe scrum : la divergence est corrigée par un changement à la baisse du backlog de sprint négocié par le maître de scrum probablement durant une séance de triage.

Dès les premiers jours du sprint, on constatera souvent une légère augmentation ou ajustement dans l'estimation du temps de développement. Ceci est essentiellement due à une analyse plus poussée des éléments du backlog de sprint et doit rester modérée comme on peut le voir Figure 5. Le maître de scrum est la seule personne qui édite ce graphique pour un sprint donné. Il pourra également procéder à une extrapolation de la courbe de charge et

analyser la vitesse de résolution des tâches en cours pour détecter une potentielle divergence au plus tôt. Les figures 5 et 6 montrent bien cette divergence et dans le premier cas montre une réaction du maître de scrum qui amène à une convergence du sprint alors que dans le second cas, le maître de scrum n'a pas agi de façon visible sur le graphique et son sprint a clairement divergé. Il y aura une revue avec le gardien du produit pour prendre une décision sur le résultat du sprint.

La Figure 5 montre un graphique basique où l'abscisse représente le temps qui passe en termes de journées de travail et l'ordonnée la charge de travail restante en termes d'heures œuvrées. Ce type de graphique a été utilisé avec succès dans une de nos deux jeunes pousses. D'autres types de graphiques d'avancement peuvent être un peu plus évolués comme celui de la Figure 6 montrant non seulement la charge de travail restante mais le statut (sous forme de surface) des diverses tâches du backlog de sprint : celles qui ne sont pas encore en développement, celles en cours de développement et celles qui sont terminées.

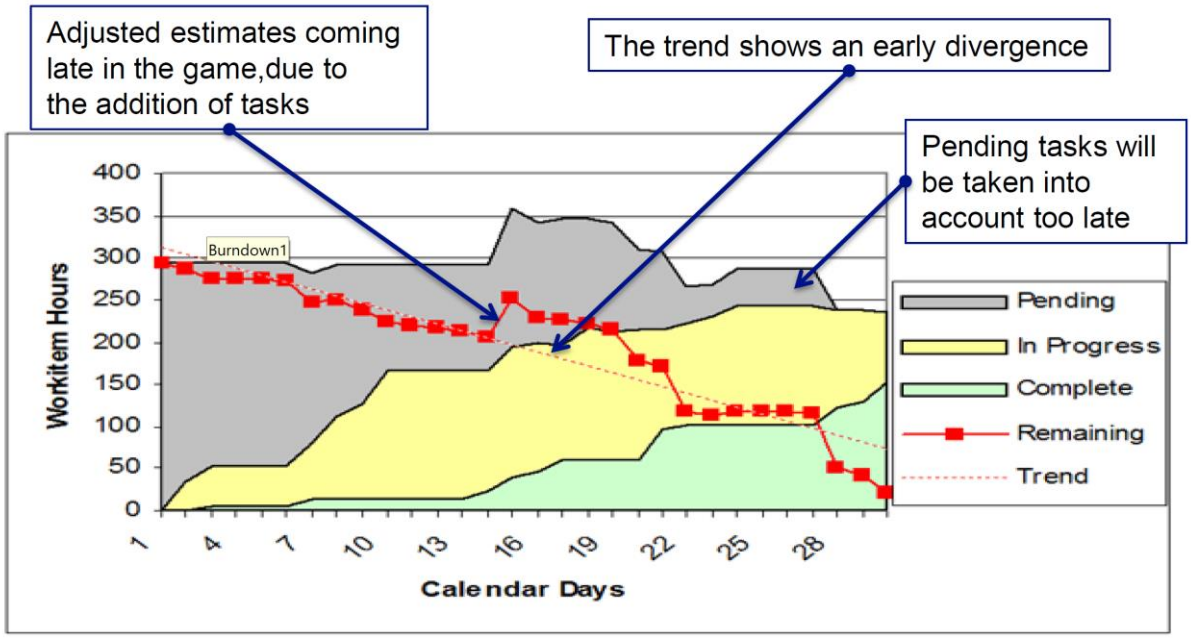

Figure 6 Par rapport au burndown chart de la e burndown chart est enrichi de l'évolution respective des tâches non commencées, en progrès ou terminées ce qui annonce la structure du tableau kanban décrit dans [Anderson…16]. Sur ce burndown chart (pris dans https://www.mitchlacey.com/intro-to-agile/scrum/burndown-charts), on remarque l'ajout de tâches arrivant bien tard dans le sprint. Cet ajout se détecte par une augmentation brutale de l'estimation d'heures de travail. Egalement, la quantité de tâches en progrès augmente beaucoup trop tard ; ainsi le sprint s'achève sans que toutes les tâches soient terminées. Le maître de scrum aurait pu voir dès la moitié du sprint qu'il serait difficile de le finir dans les temps avec toutes les fonctions achevées. Il y aura logiquement une revue avec le gardien du produit pour décider du sort de ce sprint qui n'a pas convergé et comprendre comment l'équipe peut améliorer ses performances dans le futur. Il pourra décider soit d'accorder quelques jours supplémentaires, soit de n'intégrer qu'une partie des résultats du sprint, soit de refuser le sprint entier ce qui peut déclencher une réunion de crise impliquant le management.

4.3 Kanban

Une façon de visualiser publiquement l'état d'un sprint est de présenter sur un tableau divisé en colonnes les différentes tâches du sprint, chaque colonne correspondant à une étape du processus d'accomplissement d'une tâche (voir Figure 7). Ce tableau s'appelle souvent « tableau *kanban* » kanban voulant dire « grand tableau visuel » en japonais. Une caractéristique importante est qu'il constitue souvent un mural visible par tous dans l'entreprise. Chacun peut ainsi se rendre compte de l'avancement du travail, de ce qui est planifié, puis assigné, puis en cours d'exécution, puis testé, terminé et enfin disponible.

Dans le cadre de nos jeunes pousses, nous utilisions une forme de kanban seulement dans les dernières semaines précédant la livraison en se restreignant à la zone 'travail en cours' regroupant les derniers éléments à développer sur tous les sprints non encore terminés pour la version à livrer. Le fait que le tableau soit public constitua une motivation supplémentaire pour chaque membre d'équipe scrum de terminer la tâche qui lui était assignée et

pour les chickens de se rendre compte de l'état d'avancement des travaux sans avoir à déranger les pigs au travail.

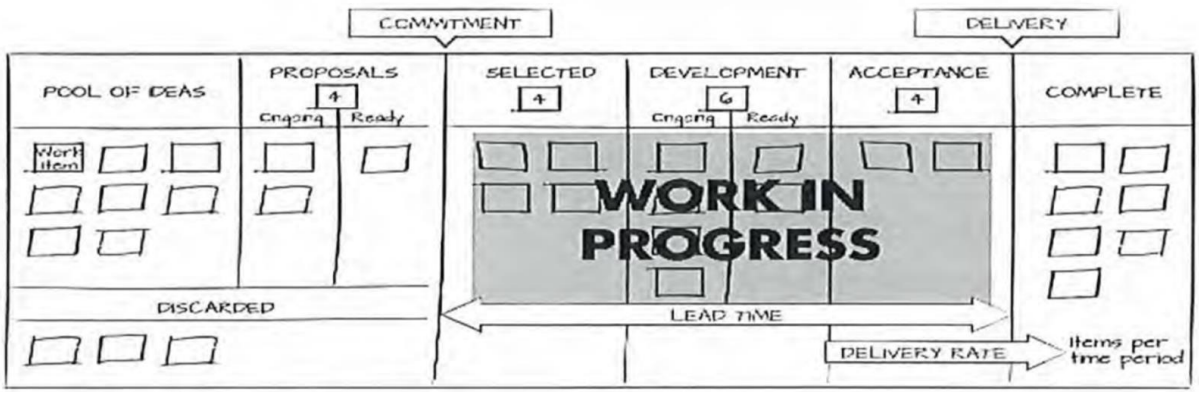

Figure 7 Tiré de « Essential Kanban Condensed » par D. J. Anderson et A. Carmichael. Un aspect important sur ce schéma est d'avoir deux lignes de séparation claire délimitant le travail en cours (work in progress) du travail non encore commencé (ou encore retiré) et du travail déjà accompli. Cette zone de 'travail en cours' est sans doute la plus importante et celle qui évolue le plus vite.

5. DEUX ÉTAPES DE LA MÉTHODE SCRUM

5.1 Notion de « Sprint »

Un sprint vise à réaliser un incrément dans le développement du produit. Il a pour but de développer en un temps très limité un ou plusieurs éléments du produit décrits dans le backlog de sprint avec une qualité satisfaisante aux yeux d'un utilisateur. L'équipe de développement affectée à un sprint se dénomme équipe scrum et sera animée par un maître de scrum.

La règle que de nombreux développeurs tiennent à respecter est la durée de 30 jours fixée pour l'exécution d'un sprint. Cette limite de temps n'a pas été appliquée comme une règle stricte mais seulement comme un ordre de grandeur; l'essentiel étant pour l'équipe d'analyser leur backlog de sprint et d'annoncer la période de temps dont elle aura besoin afin que la planification de la version produit puisse devenir maîtrisable avec une charge de travail aussi optimisée que possible. Dans nos deux expériences certains sprints ont pu durer jusqu'à 60 jours. Il y eut également des sprints courts de 15 ou 20 jours. En outre, ce qui a paru essentiel fut de pouvoir être prédictible. A ce titre, l'usage de graphiques d'avancement et leurs mises à jour régulières furent des éléments très positifs.

Un autre changement important que nous avons déjà évoqué plusieurs fois est le processus de triage. Il fut convenu que l'idée de scrum de scrums décrite dans [Schwaber 04] restait relativement vague ; par conséquent, le processus de triage, qui était un point fort du processus précédant l'introduction de la méthode scrum, fut préservé et intégré à la méthode scrum. Le triage est la seule réunion pouvant altérer le contenu d'un backlog de sprint à partir des résultats obtenus par les tests menés par l'équipe d'assurance qualité et avec le consentement des participants à la réunion. C'est donc une réunion en dehors du sprint où des décisions de développement sont prises ; c'est par principe la seule.

Les sprints vont avoir un degré de criticité différent suivant la nature de leur backlog, cette criticité s'entendant d'un point de vue commercial. Ils auront également un degré de complexité d'un point de vue technique et, bien sûr, si ces deux notions ne se recouvrent pas, elles doivent avoir une relation de dépendance : une tâche complexe techniquement si elle n'est pas elle-même critique, devrait au moins pouvoir faciliter le développement de plusieurs tâches critiques du point de vue commercial. La formation des équipes scrum va prendre en compte ces éléments, les meilleures équipes allant naturellement sur les sprints les plus complexes. D'autres tâches totalement maitrisées d'un point de vue technique, pourront être assignées à des équipes externalisées.

Les sprints sont vécus de façon fort différente d'un sprint à l'autre : plus on se rapproche de la livraison du produit, plus la tension monte moins tout retard est supporté. Les premiers sprints ayant une échéance éloignée de la livraison seront développés avec soin et décontraction alors que les sprints dont l'échéance est proche de la date de livraison ressentent une pression grandissante. Les journées de travail s'allongent considérablement sans

que la qualité augmente nécessairement. C'est une des raisons pour lesquelles on choisira de développer tout d'abord les sprints dont les backlogs sont les plus difficiles. On essayera également de ne pas s'attaquer à une fonction sensible dans un sprint proche de la date de livraison. Ainsi, il sera important de considérer les sprint dans leur ensemble, d'analyser leur interdépendances (de les minimiser si possible), et de les ordonner en minimisant les risques.

5.2 Sprints et livraisons ou versions produit (product releases)

La méthode scrum recommande que les sprints soient composés d'équipe scrum d'environ cinq à sept personnes. Cette équipe scrum va rester stable tout au long du sprint. Le besoin d'optimiser les ressources de développement d'une jeune pousse ne permet pas de respecter cette recommandation. En particulier, quelques ingénieurs peuvent être les seuls à avoir une spécialité technique donnée. Ils constituent une ressource rare qu'il faut savoir partager, car il s'est avéré fréquent qu'ils soient requis pour développer différents éléments de différents backlogs de sprint. On a donc introduit le rôle de spécialiste qui peut aller d'un sprint à un autre en accomplissant les tâches relevant de sa spécialité. Par exemple, une jeune pousse n'aura qu'un seul ingénieur de release, un seul administrateur de base de données ou un seul concepteur d'interface utilisateur ou encore un seul rédacteur technique.

L'équipe assurance qualité elle aussi est en quelque sorte spécialisée et opère sur tous les sprints, Elle est constamment responsable de l'exécution de la base de tests et reporte les résultats durant le triage. Elle valide (ou invalide quand le code testé présente une régression) dans la base de bugs le fait qu'un bug est résolu dans un champ qui lui est réservé.

Une fois la version produit définie en termes d'éléments du backlog produit par l'organisation marketing, c'est à l'équipe de développement d'analyser, d'ajouter des éléments à développer de nature technologique qui vont permettre d'implanter efficacement les fonctionnalités demandées, d'analyser les dépendances et de donner les premiers estimés relativement précis. Reste ensuite le découpage en sprints, c'est un exercice réservé à l'équipe scrum qui va découper de façon à minimiser les dépendances entre sprints, en définissant des interfaces claires, en équilibrant les éléments critiques. Puis, les sprints seront ordonnés entre eux en considérant les dépendances résiduelles et en minimisant le chemin critique. Il peut arriver qu'un sprint soit développé en anticipation d'une future version. Ce sprint aura bien évidemment une priorité basse et pourra être dérouté en cas de crise et de besoin aigu de forces de développement. Le gardien du backlog produit doit s'abstenir d'intervenir à ce niveau de l'organisation des différents sprints.

5.3 Réunions Journalières

Durant les réunions journalières d'un sprint, chaque membre de l'équipe fait le point sur son travail durant une à deux minutes pas plus. Des réunions techniques sur les difficultés rencontrées peuvent y être décidées, cependant ces discussions techniques doivent absolument être évitées afin de tenir le rythme et la durée des réunions journalières. Les 'chickens' peuvent assiter à ces réunions à condition de rester silencieux et de ne pas intervenir. Le maître de scrum consigne l'évolution de la charge résiduelle de travail sur son graphique d'avancement (burndown chart) le plus souvent implémenté à l'aide d'un tableur spécialisé dont on peut trouver différents exemplaires sur le web. La méthode scrum impose de tenir ces réunions de façon régulière et de maintenir leur durée fixe. Afin d'assurer qu'elles restent concises, ces réunions se tiennent parfois debout dans le but de maintenir son rythme et sa durée.

Dans un premier temps, ces réunions n'ont pas été suivies régulièrement, de plus leur durée pouvait varier de façon importante. Puis on s'est aperçu que les contraintes de régularité, de rythme et de durée étaient vraiment importantes et constituaient un élément clé de la méthode. Cela n'est pas venu très facilement : les ingénieurs trouvant souvent ces réunions où l'on ne doit pas discuter technique assez frustrantes Puis, tout le monde s'y est mis. Elles commençaient à 9 heures du matin de façon régulière. Petit à petit, elles contribuèrent à souder l'équipe, les développeurs découvraient ce que chacun d'entre eux faisait et comment ils pouvaient s'entre-aider. Chacun comprit qu'il s'agissait d'identifier et non de résoudre les difficultés de compréhension des spécifications ou les difficultés techniques qui pouvaient survenir. Dans chaque expérience la même évolution est survenue et les réunions ont toujours fini par être suivies assidument.
Il est paru très important de commencer les réunions à heure fixe et de maintenir leur courte durée également de façon fixe. Sauf la première réunion de sprint qui durait toujours assez longtemps, car elle avait pour but d'introduire les participants et le backlog de sprint, les autres réunions de sprint ont toujours été maintenues entre cinq à dix minutes selon le nombre de membres du sprint. Le maître de scrum est bien sûr critique pour terminer les réunions dans les temps. L'emploi de développeurs sous-traitants et délocalisés (en Irlande puis en Inde) a posé un problème important de synchronisation : le fait de se connecter par téléconférence faisait vraiment

perdre trop de temps sur une réunion de durée si courte. Le problème fut résolu par un rapport en différé de leur activité.

6. LES DIFFERENTS ROLES

6.1 Gardien du produit

Généralement, le « gardien du produit » n'est pas un ingénieur de développement ni un manager ; typiquement, il appartiendra à l'organisation marketing produit de l'entreprise. Sa tâche consiste essentiellement à organiser et maintenir le backlog produit : aucun élément du backlog n'est ajouté ou retiré sans son accord. Il est responsable de l'établissement des priorités des différents éléments du backlog. Il participe à la définition de la version produit à livrer avec le directeur du développement, qui lui, va en particulier évaluer son coût de développement et de qualification, le directeur du marketing produit, qui représente la base clients, et le directeur technique pour défendre les aspects technologiques. Le rôle du gardien du produit dans la définition de la version produit à livrer s'arrête là : il doit laisser l'équipe de développement définir les différents sprints qui doivent être techniquement cohérents et dont l'ensemble formera la version produit. Il participera ensuite aux séances de triage, aux revues de sprint, et donnera son avis pour livrer ou non la version produit. Il organise et participe également aux revues finales de sprint. Il joue un rôle important grâce à un dialogue constant avec l'équipe d'assurance qualité dans la qualité finale de la livraison produit comprenant la documentation utilisateur mise à jour incluant les notes de version produit. Il est bien placé dans l'organisation de l'entreprise pour définir et mener les tests de recette qui seront effectués par les utilisateurs.

6.2 Maître de Scrum

Le maître de scrum a un rôle prépondérant pour le succès du sprint dont il est responsable. Avec son équipe, il va définir le backlog de sprint qui est le plus souvent, mais pas systématiquement, un sous-ensemble de la version produit. Durant tout le sprint il aura la charge de maintenir le backlog de sprint et de contrôler la convergence du processus de développement.

De la même façon que le gardien du produit ne doit pas être un développeur, le maître de scrum ne devrait pas être un manager. D'une part, son rôle est de ne surtout pas évaluer les membres de l'équipe, ce qu'un manager ne pourrait pas s'empêcher de faire, d'autre part, il doit avoir des connaissances techniques qui lui permettent de comprendre rapidement les situations décrites par ses collègues durant la réunion journalière qu'il anime. Il est de sa responsabilité que ces réunions commencent et finissent à l'heure. Il est arrivé de nommer des développeurs juniors à cette place et avec succès. Une bonne connaissance de la méthode, un haut degré de motivation et de légitimité du maître de scrum aux yeux du reste de l'équipe se sont révélés les ingrédients véritablement nécessaires pour qu'un maitre de scrum réussisse son sprint.

Un bon maître de scrum saura protéger son équipe et son backlog de sprint de toute demande de changement qu'il jugera trop riquée pour la convergence du sprint. La pression est intense dans une jeune pousse : il y a toujours quelqu'un (un « chicken » bien sûr) dans l'entreprise qui aura une meilleure idée et voudra l'imposer. Pour aider le maître de scrum à résister à toute tentation, une notion de « gel de spécification » fut introduite dans le processus du sprint : après quelques jours passés en analyse et spécification du sprint, il ne sera plus possible de la changer et toute « meilleure » idée devra être entrée dans le backlog produit. Les très bons développeurs sont connus et toujours très sollicités au sein même de l'entreprise, leur avis peut être demandé dans d'autres sprints ; là également le bon maître de scrum devra veiller au grain et ne pas permettre que ses ressources passent trop de temps sur des tâches qui ne font pas partie du sprint.

Le maître de scrum participe au triage où il représente l'équipe scrum de son sprint et doit être capable de comprendre si un défaut signalé par l'équipe d'assurance qualité sera résolu à temps par son équipe pour l'accepter au programme de son sprint. Il pourra également faire état des problèmes découverts par son équipe et devant être résolus par d'autres équipes scrum quand ils relèvent de la définition d'autres backlogs de sprint.

Un bon maître de scrum, évaluant la convergence de son sprint et la reportant dans son graphique d'avancement, saura anticiper toute divergence importante et soit résoudre la question au sein de l'équipe, soit le cas échéant, faire remonter le problème suffisamment tôt pour ne pas provoquer un retard dans la livraison produit. Il pourra proposer exceptionnellement d'arrêter ou de suspendre son sprint avec l'accord de la hiérarchie s'il constate qu'il diverge trop, ce qui pourrait entraîner un changement important dans la livraison de la version produit. C'est arrivé de très rares fois, particulièrement dans notre première expérience, en rencontrant un obstacle de nature algorithmique qui se révélait beaucoup plus complexe qu'anticipé au départ de la release. L'algorithme en

question fut alors remonté au bureau de la direction technique. A la place, un algorithme plus simple, mieux connu mais sous-optimal a alors été rapidement développé et intégré.

6.3 Équipe Scrum

L'équipe scrum est une petite équipe de l'ordre de 4 à 6 développeurs animée par le maître de scrum. Elle va contribuer à définir le backlog de sprint qu'elle va s'engager à délivrer dans un temps qu'elle va elle-même estimer en s'appuyant sur les données disponibles du backlog et de sa propre connaissance du code du produit. Elle se doit d'être et surtout rester très soudée lors du déroulement du sprint tout spécialement vers la fin lorsque les tensions peuvent augmenter dû à des obstacles imprévus. Elle doit également rester focalisée sur les tâches décrites dans le backlog de sprint. Cependant, elle continue d'interagir avec le management, les « chickens » ou certains « pigs ».

6.3.1 Équipes scrum et management

La méthode scrum comme d'autres méthodes agiles semble éliminer les fonctions de management. Un des principes fondamentaux de la méthode Agile n'est-il pas l'auto-organisation des équipes de développement (principes 5 et 11), ce qui est certainement à la source de cette question ?
En fait, dans nos expériences, le rôle du management n'a pas été éliminé même s'il a su laisser un bon degré d'indépendance aux différentes équipes scrum. Par exemple, le management s'est s'impliqué à la demande d'un maître de scrum à chaque constat (sur le graphique d'avancement) d'une forte divergence d'un sprint (il n'y a pas que des sprints qui réussissent), ou pire encore si le sprint affrontait une remise en cause du sprint entier ou de l'équipe scrum en situation de mésentente.
Le management s'est également impliqué dans le montage des équipes scrum, par exemple en équilibrant le nombre d'ingénieurs débutants et d'ingénieurs confirmés. Mélanger des ingénieurs confirmés et des ingénieurs débutants au sein d'un sprint a permis pour ces derniers une acquisition de connaissances accélérée. Dans nos expériences, il était toujours bon de changer et reformer les équipes scrum d'un sprint à l'autre dans la mesure du possible.

Le maître de scrum va négocier avec le management l'obtention de spécialistes pour accomplir un sprint. Ces spécialistes passent en général un temps compté dans chaque sprint de la version produit où leur spécialité est nécessaire et demandée. Dans une jeune pousse, ils constituent une ressource critique et il est très important de bien maîtriser leur charge de travail.

Le management va également gérer et anticiper la formation des équipes scrum d'un sprint au suivant. Il est important d'anticiper, plus d'une semaine avant la fin d'un sprint, les négociations qui impliquent les développeurs sur la définition du sprint suivant (qui ne fera pas nécessairement partie de la version produit courante). Nous avons trouvé qu'il est important que les développeurs sachent à tout moment quelle charge de travail est attendue d'eux à un horizon d'au moins un sprint et de s'y préparer. Signalons qu'il est tout à fait possible que quelques ingénieurs ne soient pas impliqués dans un sprint pour participer à un projet spécial en dehors de la livraison du produit (par exemple avec le directeur technique ou l'architecte de l'entreprise ou encore pour développer une preuve de concept ou cas d'usage spécifique pour un nouveau client) ou pour se former ou encore pour participer à une conférence ou une exposition et démonstration du produit.

Il faut également évoquer le cas où des membres de l'équipe scrum travaillent en mode de sous-traitance délocalisée. Nous avons vu que cela pose des problèmes pour tenir les réunions journalières. Cela pose également des problèmes de management, car il peut se faire que l'on ne contrôle plus les personnels impliqués dans les sprints : l'entreprise de sous-traitance peut à tout moment les retirer et les remplacer surtout dans des cas de prolongation de contrat. Il est important de négocier ce point au moment des discussions du contrat de sous-traitance. Ajoutons qu'il est assez rare de voir un sous-traitant avec une véritable mentalité et attitude de « *pig* » (sauf peut-être pour des sous-traitants de longue durée). Enfin, il faut réfléchir à ce qui sera sous-traité et bien définir la technologie qui sera définie comme « cœur de métier » qui ne sera pas elle, sous-traitée.

6.3.2 Interactions entre l'équipe scrum et les « chickens »

Que peuvent bien faire les « chickens » alors que les « *pigs* » s'auto-organisent, développent le produit et travaillent intensément ?
Rappelons que les « chickens » sont tous ceux qui ne sont pas directement engagés dans le développement du produit ; ceci comprend non seulement le management dont nous avons déjà amplement parlé et le haut management allant jusqu'au PDG, mais aussi les branches marketing et ventes de l'entreprise, et la direction

technique sans oublier les services après-vente. Dans nos deux cas d'étude, tout le personnel fut au moins « chicken ».
Enfin, les utilisateurs pourront selon les auteurs mais surtout selon leur degré d'implication être considérés soit comme des chickens soit comme des « *pigs* ». Il est bon ici de rappeler les principes 2 et 8 du Manifeste Agile et de signaler que certains utilisateurs se rapprocheront des équipes de développement remontant des besoins en fonctionnalités nouvelles et des cas d'usages bien argumentés.

Dans une jeune pousse, il n'est pas rare de voir le directeur des ventes ou le haut management revenir enthousiaste d'une visite chez un prospect en annonçant une affaire importante à la condition impérative que le produit soit livré dans sa prochaine version avec une nouvelle fonction qui, parfois, n'aura jamais été identifiée auparavant dans le backlog produit ni même dans la vision produit. Cette information va mettre très peu de temps à se propager dans la jeune pousse avec un risque important de défocaliser certains ingénieurs impliqués dans le développement de sprint importants pour la version produit. C'est alors au directeur de développement cette fois de savoir-faire écran et d'exercer son jugement professionnel pour dire non (ou oui dans certains cas qui se doivent de rester limités). C'est souvent une décision extrêmement difficile surtout dans une jeune pousse où toute vente (ou même promesse de vente) est si critique à la vie de l'entreprise, mettant en balance un principe central d'agilité avec un risque de ne pas livrer dans les délais. Un aspect important des responsabilités du directeur de développement est donc de protéger les développeurs, de garder son sang-froid et de leur faire confiance pour la bonne réalisation du produit (selon le 5ème principe de la méthode Agile). Une solution de compromis sera pour le directeur de développement de définir quand cela est possible et à la condition que la date de livraison produit ne soit pas déroutée, une version à livraison limitée à quelques clients avertis (parfois appelée version alpha puis beta). Il a d'autre part une vision globale des différents sprints et jouera un rôle important dans la constitution et l'évolution des équipes scrum en décidant des profils des nouveaux collaborateurs à embaucher ou en décidant de sous-traiter (en particulier en off-shore).

Un autre « chicken » important est l'architecte produit ou le directeur technique (voir [Falesi…10], [Madison 10], et. [Abrahamson…10] sur la relation entre architecture et agilité). L'architecte produit maintient la vision produit qu'il construit en dialoguant avec les équipes scrum d'une part ou directement avec les clients d'autre part. Il va valider du point de vue technique le contenu des backlogs de sprint et de leur complexité. Même lorsqu'il n'est pas impliqué dans une équipe scrum, son rôle est important, car il aura la fonction de dialoguer avec le gardien du produit et, plus largement, avec l'organisation marketing. A ce titre, il se doit de savoir traduire les besoins exprimés par le marketing en fonctions implémentables par l'équipe de développement et inversement par exemple, les limitations techniques en limitations fonctionnelles. Il est responsable de l'évaluation de la faisabilité, de l'estimation de la charge de travail nécessaire au développement d'une fonction et de la décision quant aux briques technologiques nécessaires à la réalisation du produit. Il est une force de proposition et d'approbation ; sur ces chapitres, nous trouvons une limitation importante du degré d'indépendance des équipes scrum. De la même façon que backlogs produit et vision doivent être mis en relation, il doit régner une bonne compréhension ainsi qu'une bonne adhérence entre l'architecte et les équipes scrum : l'architecte devant être à l'écoute des problèmes techniques rencontrés par les développeurs et l'équipe scrum devant respecter les choix technologiques décidés sous la responsabilité de l'architecte et les intégrer dans leur environnement.

6.3.3 Interactions avec différents personnels agissant directement sur le produit

Un sprint ne représente après tout qu'un élément de la réalisation du produit. Il faut parler d'autres personnels qui sans faire partie de l'équipe scrum ne sont pas moins engagés (« pigs ») : de l'équipe d'assurance qualité, du ou des rédacteurs de documentation, sans oublier les ingénieurs spécialisés comme les administrateurs de base de données ou le release ingénieur en charge des constructions de versions et de configurations dont le travail sera extrêmement rarement abordé dans un backlog de sprint.

Dans nos deux expériences, l'équipe d'assurance qualité était en charge de la branche produit du gestionnaire de version. Chaque sprint se développait sur sa propre branche qui était bien sur différente de la branche produit. L'intégration dans la branche produit ne se faisait qu'après une séance de triage où la qualité du code était évaluée par l'équipe d'assurance qualité et discutée. L'avis de cette équipe était prépondérant. Le principe, qui n'est pas décrit dans la méthode scrum, était de préserver la branche produit de toute régression (voir [Memmi 05] pour plus de détails).

Quant au rédacteur de documentation technique (une jeune pousse n'a que très rarement plus d'un rédacteur de documentation technique), il fut traité comme un spécialiste et sa tâche fut ainsi parfaitement intégrée au planning des différents sprints.

7. CONCLUSION

La méthode scrum n'est pas une méthode miraculeuse, nous l'avons trouvé d'aucun secours quand les éléments à développer sont mal compris, incomplets ou conservent un aspect technique complexe non résolu. Sur ce dernier point [Wasserman 11] décrit une situation analogue à notre expérience : aux USA, seul le développement de logiciels complexes n'aurait pas basculé majoritairement vers l'usage de méthodes agiles.

Cette réserve liminaire à notre conclusion mise à part, nous avons reconnu de nombreux points très positifs au cours de sa mise en place et de son application dans le respect de la culture des jeunes pousses où la méthode scrum a été déployée. Nous énumérons ci-dessous ceux qui nous ont semblés amener un clair gain de productivité et que nous recommandons.

Donner des responsabilités aux ingénieurs qui sont usuellement des responsabilités de management, faire confiance aux équipe de développement, les laisser accomplir leurs tâches, ne faire intervenir les « chickens » essentiellement que dans l'enrichissement du backlog produit ou encore dans un processus clair et transparent d'escalade managériale sont des facteurs importants de succès dans le déploiement de toue méthode agile telle que scrum.

Maîtres de scrum et gardiens du produit doivent avoir de fortes personnalités, savoir détecter au plus tôt quand les choses ne vont pas et savoir prendre les mesures adaptées. Ils doivent aussi se sentir investis de responsabilités critiques pour le succès de la version produit (ce sont des « *pigs* », ne l'oublions pas). Ils doivent savoir à la fois défendre et ne pas gaspiller les ressources de l'entreprise, bien comprendre et satisfaire les véritables besoins des utilisateurs. Ils doivent constamment inspecter, documenter, nettoyer et enrichir le backlog produit qui reste sous la seule responsabilité du gardien du produit.

Un autre point important dans nos expériences fut la prise de conscience du rôle du backlog produit et de son intégration avec la base de défauts pour pouvoir former avec la vision produit l'historique du produit dès lors que ces éléments sont horodatés. En effet, les deux premières bases rassemblent ce que l'entreprise a appris sur ce qui fonctionne, les faiblesses et les défauts du produit. Alors que le document de vision technique du produit prévoit par définition les futurs traits d'architecture ou de nouvelles fonctionnalités. La définition de ce qu'est un produit logiciel devrait inclure ces éléments.

Les courtes réunions journalières se sont révélées extrêmement utiles contre toute attente initiale. Dès les premières phases de déploiement de la méthode. Elles ont contribué à unir le corps des ingénieurs en intégrant plus rapidement les nouveaux arrivants ou favorisant la formation des jeunes qui trouveront aisément un tutorat efficace à l'intérieur de leur équipe scrum. Après plusieurs essais, il fut possible de mener ces réunions de façon régulière et ponctuelle, vivante et dans une durée bien contrôlée.

Si l'utilisation d'un kanban s'est avérée impraticable en début de projet car il aurait contenu trop de détails ou bien n'aurait pas représenté efficacement le travail de chacun, elle a au contraire été très bien accueillie et s'est révélée très utile en fin de projet alors qu'il ne restait qu'entre dix à vingt tâches à terminer. On aurait pu craindre que le fait que le tableau soit public joue un rôle d'accusateur pointant vers les développeurs qui n'ont pas terminé leur travail ; il n'en fut rien, au contraire un esprit d'entre aide s'était installé pour pouvoir livrer dans les meilleurs délais.

Progresser tout en respectant la culture de l'entreprise, son organisation et son écosystème produit, en ayant une très bonne connaissance des équipes de développement et de leurs capacités a été un facteur clé lors de la mise en place de la méthode scrum. Les réunions *post mortem*, qui lorsqu'elles sont bien préparées, ont toujours été très utiles voire indispensables la période de mise en place et de premiers déploiements de la méthode. C'est à travers ces réunions rassemblant tout le corps des ingénieurs que la méthode scrum fut discutée et adaptée aux besoins spécifiques des équipes et de l'entreprise. Par exemple, la durée de 30 jours d'un sprint a été changée ; ainsi le rôle de spécialiste (par exemple, pour les interfaces graphiques, l'administration des bases de données, ou pour la documentation technique et la documentation utilisateur) fut introduit. Remarquons que ces réunions répondent bien au 12ème principe du manifeste de la méthode Agile (Figure 1).

L'enchaînement rapide des sprints et leurs contributions aux diverses versions produit, la multiplication des produits et leur constitution en ligne produit, la spécialisation de certains développeurs crée un vaste problème d'optimisation d'affectation des ressources de développement. Il nous a semblé au vu de nos deux expériences que la méthode scrum ne passe pas à l'échelle de façon très évidente c'est-à-dire ne s'adapte pas simplement au développement de grand projet par de très grandes équipes d'ingénieurs (en particulier lorsqu'elles sont

géographiquement dispersées sur plusieurs sites) nécessitant de nombreux sprints dont certains seraient conduits en parallèle (voir [Safe 19] pour une proposition de solution de cette question de passage à l'échelle). D'autres auteurs pointent également vers cette question d'affectation de ressources. Il faudra à l'avenir proposer un modèle de description et un moyen d'optimiser cette affectation pour obtenir un nouveau gain en productivité.

8. RÉFÉRENCES

[Abrahamson…10] P. Abrahamson, M. Ali Babar, and P. Kruchten *"Agility and Architecture: can they coexist?*" IEEE Software, March-April 2010.

[Agile 01] *"Manifesto for Agile Software Development"* www.agilemanifesto.org , 2001.

[Anderson…16] D. Anderson and A. Carmichael "*Essential Kanban condensed*" http://leankanban.com/wp-content/uploads/2016/06/Essential-Kanban-Condensed.pdf , Lean Kanban U. Press, 2016.

[DSDM…95] "*DSDM Consortium"* www.dsdm.org , 1995.

[DSDM…14] "*DSDM Agile Project Framework Handbook*" https://www.agilebusiness.org/page/TheDSDMAgileProjectFramework January 2014.

[Dwitam…20] "*User stories collection via interactive chatbot to support requirements gathering*" F. Dwitam, A. Rusli, TELKOMNIKA Telecommunication, Computing, Electronics and Control Vol.18, No.2, pp. 890-898, April 2020.

[Falesi…10] D. Falessi, G. Cantone, S. A. Sarcia, G. Calavaro, P. Subiaco, and C. D'Amore "*Peaceful Coexistence: Agile Developer Perspectives on Software Architecture*" IEEE Software, March-April 2010.

[Fernandez…13] V. Fernandez, T. Houy, and C. Khalil "*Les méthodes agiles de développement informatique"* Presse des Mines, 2013.

[Madison 10] J. Madison "*Agile Architecture Interactions"* IEEE Software, March-April 2010.

[Memmi 05] G. Memmi "*Conduite du Test à Chrysalis: un Retour d'Expérience"* présentation invitée à *ICSSEA'05* Paris également dans Génie Logiciel 72 pp 53-58, Mars 2005.

[Memmi 09] G. Memmi "*Mettre un logiciel à l'échelle n'est souvent possible qu'en ajoutant un supplément de fonctions"* Génie Logiciel 88 pp 2-9, Mars 2009.

[Memmi 11] G. Memmi *"Deux retours d'expérience avec la méthodologie SCRUM"* Génie Logiciel 98 pp 52-58, Septembre 2011.

[Meyer 14] B. Meyer "*Agile! The Good, the Hype and the Ugly"* Springer, 2014.

[Palmer…02] S. R. Palmer, J. M. Felsing "*A Practical Guide to Feature-driven Development*" Prentice Hall, 2002.

[Rees 02] "*A Feasible User Story Tool for Agile Software Development?*" Proceedings of the 9th Asia-Pacific Software Engineering Conference (APSEC'02), IEEE, 2002.

[Safe 17] "*Safe 4.5 Introduction, Overview of the Scaled Agile Framework for lean Enterprises*", 2017. http://editor.scaledagile.com/wp-content/uploads/delightful-downloads/2018/01/White_Paper_SAFe-4.5.pdf

[Safe 19] "*Achieving Business Agility with SAFe® 5.0*" A Scaled Agile, Inc. White Paper, December 2019.

[Schwaber… 02] K. Schwaber and M. Beedle**:** "*Agile Software Development with Scrum"* Pearson Int. Ed. 2002.

[Schwaber 04] K. Schwaber**:** "*Agile Project Management with Scrum"* Microsoft Best Practices 2004.

[Sutherland…11] J. Sutherland and K. Schwaber: "*The Scrum Papers: Nut, Bolts, and Origins of an Agile Framework"* Draft January 2011 http://jeffsutherland.com/ScrumPapers.pdf

[Schwaber…17] K. Schwaber and J. Sutherland "The Scrum Guide, The definitive Guide to Scrum: The rule of the Game" https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf November 2017.

[Striebeck 06] M. Striebeck "Ssh We are adding a process…" Proc; of Agile'06, IEEE, Minneapolis, MN, USA, July 2006.

[Takeuchi…86] H. Takeuchi and I. Nonaka**:** "*The new new product development game"* Harvard Business Review, January 1986.

[Wasserman 11] A. Wasserman: *"Software Engineering Issues for Mobile Application Development"* Mobicase'11, October 2011. http://www.mobileseworkshop.org/papers/Wasserman_foser2010.pdf

[XP…99] "*Extreme Programming: A gentle introduction"* www.extremeprogramming.org