Kanban et Scrum -
tirer le meilleur des deux

Henrik Kniberg et Mattias Skarin
Préfacé par Mary Poppendieck et David Anderson

Traduit par Claude Aubry, Frédéric Faure,
Antoine Vernois et Fabrice Aimetti

C4Media, Editeur de InfoQ.com.

Ce livre fait partie de la collection des livres de la société InfoQ spécialisée dans le développement de logiciel.

Pour toute information ou commande de ce livre ou d'un autre livre de la société InfoQ, prière de contacter books@c4media.com.

Rédacteur en Chef : Diana Plesa
Illustration de la Couverture : Bistrian Iosip
Composition: Accurance

Bibliothèque du Congrès pour la publication de données :

ISBN: 978-0-557-13832-6

Imprimé aux Etats-Unis

Sommaire

Glossaire

Ajouté par les traducteurs pour permettre aux lecteurs de comprendre les choix de traduction de certains termes anglais.

kanban	Mot japonais signifiant étiquette (ou fiche-signal).
Kanban	Pratique (ou outil de processus pour reprendre la définition d'Henrik Kniberg) basée sur l'utilisation de kanban pour matérialiser des informations sur le processus.
Lead time	Temps de cycle.
MMF	Minimum Marketable Feature. Fonctionnalité minimale qui a de la valeur pour l'utilisateur.
scrum	En anglais, la mêlée. Dans ce livre, Scrum est utilisé pour désigner la méthode agile (ou outil de processus pour reprendre la terminologie d'Henrik Kniberg) et scrum pour la réunion quotidienne d'une équipe.
Tableau	endroit où sont placés les kanban (Board en anglais). Avec Scrum, on parle de tableau des tâches.
TAF	Travail à Finir (WIP : work in progress en anglais). Kanban limite le TAF, c'est à dire le nombre de travaux à faire dans une étape du processus. Par raccourci, on dira TAF de 2 pour une limite à 2 kanban dans une colonne du tableau.
Timeboxed	Bloc de temps à durée fixe.
WIP	Work In Progress = Travaux en cours. Traduit par TAF.

Préface de Mary Poppendieck

Henrik Kniberg fait partie des rares personnes qui peuvent extraire l'essence d'une situation compliquée, faire le tri entre les idées importantes et celles qui sont accessoires, et fournir une explication limpide étonnamment facile à comprendre. Dans ce livre, Henrik fait un travail brillant pour expliquer les différences entre Scrum et Kanban. Il précise clairement que ce sont seulement des outils alors que ce qui compte vraiment est d'avoir une boîte à outils complète, de comprendre les points forts et les limites de chaque outil et d'acquérir la manière de les utiliser.

Avec ce livre, vous apprendrez ce qu'est Kanban, ses forces et ses limites, et quand l'utiliser. Vous apprendrez également comment Kanban peut améliorer Scrum, ou tout autre outil que vous utilisez, et à quel moment c'est possible. Henrik montre clairement que le plus important n'est pas l'outil avec lequel on commence, mais la façon dont on améliore constamment son utilisation et comment on développe progressivement son ensemble d'outils

La deuxième partie de ce livre, écrite par Mattias Skarin, rend la lecture encore plus instructive pour le praticien à travers l'application de Scrum et Kanban en situation réelle. Vous y trouverez un exemple montrant la façon dont ces outils ont été mis en place, un par un et ensemble, pour améliorer un processus de développement logiciel. Vous remarquerez qu'il n'y a pas une unique "meilleure" manière de faire les choses ; c'est à vous seul de réfléchir et de découvrir - en fonction de votre contexte - quelle est votre prochaine étape pour améliorer votre processus de développement logiciel.

Mary Poppendieck

Préface de David Anderson

Kanban est basé sur une idée très simple : le nombre de travaux à faire (le TAF) dans un processus doit être limité, donc une nouvelle tâche ne peut être ajoutée que si une autre est, au préalable, terminée ou utilisée à la demande d'un processus aval. Le kanban (littéralement la fiche-signal) est un signal visuel produit pour indiquer qu'un nouveau travail peut être entrepris parce que les travaux en cours n'excèdent pas la limite fixée. Cela ne semble pas très révolutionnaire et on ne s'attend pas à ce que cela affecte fortement la performance, la culture, la capacité et la maturité d'une équipe et de son organisation environnante. Ce qui est surprenant c'est que cela arrive ! Kanban ressemble à un petit changement et pourtant il bouleverse l'entreprise.

En fait, Kanban est une approche de gestion du changement. Ce n'est ni un processus de développement ni un cycle de vie d'un logiciel ni une méthodologie de gestion de projet. Vous pouvez commencer à appliquer le principe de Kanban en partant de la façon dont vous travaillez actuellement. Vous commencez par cartographier votre processus avec sa chaîne de valeur puis vous définissez des limites de TAF pour chaque étape de ce processus. Vous pouvez alors lancer le flux de travail en l'entraînant à travers le système lorsque des signaux kanban sont générés.

Kanban s'avère utile aux équipes qui développent du logiciel avec des méthodes agiles, mais il est aussi de plus en plus appliqué par les équipes qui suivent une approche plus traditionnelle. Kanban fait partie du cadre de l'approche Lean, qui a pour but de favoriser l'amélioration continue, en s'adaptant à la culture des organisations.

Parce que le TAF est limité dans un système Kanban, tout ce qui se bloque, pour une raison quelconque, a tendance à engorger le système. Si trop d'éléments de travail se bloquent, tout le processus finit par s'arrêter. Alors toute l'équipe (et l'ensemble de l'organisation) se concentre sur la

résolution du problème, le déblocage de l'élément concerné et le rétablissement du flux de travail.

Kanban emploie un mécanisme de contrôle visuel pour suivre le flux de travail qui circule à travers les différentes étapes de la chaîne de valeur. On utilise typiquement un tableau blanc avec des post-its ou un système électronique mural. La meilleure pratique est probablement de combiner les deux. La transparence générée contribue aussi au changement de culture. Les méthodes agiles apportent des bénéfices en favorisant la transparence sur l'avancement des travaux en cours et terminés et en produisant de nouveaux indicateurs, comme la vélocité (la quantité de travail réalisée dans une itération). Toutefois, Kanban va un peu plus loin et assure la transparence dans le processus et son flux. Kanban expose les goulets d'étranglement, les files d'attente, la variabilité et le gaspillage. Toutes ces choses qui influent sur la performance de l'organisation en termes de mesure des éléments à valeur ajoutée produits et de temps de cycle nécessaire pour les produire. Aux membres de l'équipe et aux intervenants externes, Kanban offre la visibilité sur l'effet de leurs actions (ou inactions). Ainsi, les premières études de cas ont montré que Kanban modifie les comportements et encourage une plus grande collaboration au sein des organisations. La visibilité sur les goulets d'étranglement, les gaspillages et le manque de régularité, ainsi que leur impact, encourage aussi les débats sur les progrès possibles et les équipes commencent rapidement à mettre en œuvre des tâches d'amélioration de leurs processus.

En conséquence, Kanban encourage l'évolution progressive des processus existants et cette évolution correspond globalement aux valeurs de l'Agilité et du Lean. Kanban n'exige pas de révolutionner de fond en comble la façon dont les gens travaillent, il encourage plutôt un changement progressif. Le changement est compris et adopté par consensus parmi les employés et leurs collaborateurs.

Kanban, basé sur un système à flux tiré, encourage également un engagement au plus tard, à la fois sur la priorisation du travail et la livraison du travail en cours. Typiquement, les équipes vont convenir d'une fréquence pour rencontrer les parties prenantes et décider sur quoi travailler dans la suite. Ces réunions peuvent être tenues régulièrement parce qu'elles sont généralement très courtes. On peut y aborder une

question très simple, quelque chose comme : "Depuis notre dernière réunion, deux créneaux de travail se sont libérés. Notre temps de cycle actuel jusqu'à la livraison est de 6 semaines. Quelles sont les 2 choses que vous aimeriez voir livrées dans 6 semaines ?". Cela a une double incidence. Poser une question simple se traduit généralement par une réponse de bonne qualité obtenue rapidement et permettant de maintenir une réunion courte. La nature de cette question signifie que l'engagement de sur quoi travailler est retardé jusqu'au dernier instant. Cela améliore l'agilité en gérant les exigences, en raccourcissant les temps de cycle entre l'engagement et la livraison, et en éliminant le travail de remise à niveau puisque le risque d'un changement dans les priorités sera minimisé.

Un dernier mot sur Kanban : l'effet de limiter le TAF assure la prédictibilité du temps de cycle et produit des livrables de meilleure qualité. L'approche “Arrêter la chaîne” lorsque des obstacles et des bugs sont détectés semble aussi encourager un haut niveau de qualité et une baisse rapide du retravail.

Bien que tout cela apparaisse évident avec les explications merveilleusement claires fournies dans ce livre, comment nous en sommes arrivés là reste obscur. Kanban n'a pas été conçu en un seul après-midi par quelque démonstration stupéfiante . Au lieu de cela, il est apparu sur plusieurs années. Un grand nombre de ses grands impacts psychologiques et sociologiques sur la culture, la capacité et la maturité des organisations n'avaient pas été imaginés. Ils ont plutôt été découverts. Bon nombre des résultats obtenus avec Kanban sont contre-intuitifs. Ce qui semble être une approche très mécanique - limiter le TAF et tirer le flux de travail - a effectivement eu des effets profonds sur les gens et sur la façon dont ils interagissent et collaborent entre eux. Je n'avais pas prévu cela, ni d'ailleurs qui que ce soit aux premiers jours de son implication dans Kanban.

J'ai appliqué ce qui est devenu Kanban comme une approche du changement provoquant une résistance minimale. C'était clair pour moi dès 2003. Je l'ai aussi mis en œuvre pour ses bénéfices mécaniques. À la même époque, j'étais en train de découvrir les applications des techniques Lean et, si la gestion du TAF avait un sens, le limiter en avait encore plus. Ainsi en 2004, j'ai décidé d'essayer de mettre en œuvre un système à flux tiré à partir des premiers principes. J'en ai eu l'opportunité lorsqu'un

directeur de Microsoft m'approcha et me demanda de l'aider à changer son équipe en charge des mises à jour de maintenance sur les applications informatiques internes. La première mise en œuvre était basée sur la Théorie des Contraintes d'un système à flux tiré connue sous le nom de Drum-Buffer-Rope. Ça a été un énorme succès : le temps de cycle a chuté de 92%, le débit a été multiplié au moins par 3 et la prédictibilité (respect de la date d'échéance) affichait un très acceptable 98%.

En 2005, Donald Reinertsen m'a convaincu de mettre en œuvre un système Kanban à plus grande échelle. J'en ai eu l'occasion en 2006 lorsque j'ai pris en charge le Département d'ingénierie logiciel chez Corbis à Seattle. En 2007, j'ai commencé à communiquer sur les résultats. La première présentation a été faite au New Product Development Summit à Chicago en Mai 2007. J'ai poursuivi avec un Open Space lors de l'Agile 2007 à Washington DC en août. 25 personnes étaient présentes. 3 d'entre elles étaient de Yahoo! Aaron Sanders, Karl Scotland et Joe Arnold. Ils rentrèrent en Californie, en Inde et au Royaume-Uni pour mettre en œuvre Kanban avec leurs équipes, qui étaient déjà aux prises avec Scrum. Ils ont également lancé un groupe de discussion Yahoo! qui, au moment où j'écris, est constitué de presque 800 membres. Kanban commençait à se propager et les premiers praticiens partageaient leurs retours d'expérience.

Aujourd'hui, en 2009, Kanban est de plus en plus adopté et de nombreux retours remontent du terrain. Nous avons beaucoup appris sur Kanban dans les 5 années passées et nous continuons tous à apprendre chaque jour davantage. J'ai consacré mes propres travaux à la pratique de Kanban, à écrire sur Kanban, à parler de Kanban et à réfléchir sur Kanban afin de mieux le comprendre et l'expliquer aux autres. J'ai volontairement laissé de côté le fait de comparer Kanban avec les méthodes agiles existantes, même si, en 2008, certains de mes efforts ont été consacrés à expliquer pourquoi Kanban méritait d'être considéré comme une approche agile.

J'ai laissé à d'autres, qui avaient une expérience plus large, le soin de répondre à des questions comme “Comment Kanban se compare-t-il à Scrum ?” Je suis très heureux qu'Henrik Kniberg et Mattias Skarin aient émergé comme des leaders dans ce domaine. Vous, les praticiens, avez besoin d'informations afin de prendre des décisions éclairées et d'avancer dans votre travail. Henrik et Mattias sont au service de vos besoins, et d'une manière que je n'aurais jamais pu mettre en œuvre. Je suis

particulièrement impressionné par l'approche réfléchie d'Henrik en matière de comparaison et de livrable factuel et neutre. Ses dessins et illustrations sont particulièrement perspicaces et vous permettent souvent d'économiser beaucoup de temps de lecture. L'étude de cas faite sur le terrain par Mattias est importante car elle démontre que Kanban est beaucoup plus qu'une théorie : il vous montre par l'exemple comment Kanban pourrait vous être utile dans votre organisation.

J'espère que vous apprécierez ce livre comparant Kanban avec Scrum et qu'il vous donnera un meilleur aperçu sur l'Agilité en général et sur Kanban et Scrum en particulier. Si vous voulez en savoir plus sur Kanban, je vous invite à visiter notre site web communautaire, The Limited WIP Society, http://www.limitedwipsociety.org/

David J. Anderson

Sequim, Washington, Etats-Unis
8 Juillet 2009

Introduction

Nous n'avons pas l'habitude d'écrire des livres. Nous préférons passer notre temps au fond des tranchées à aider les clients à optimiser, déboguer et remanier les processus de développement de leur organisation. Cependant, ces derniers temps, nous avons clairement remarqué une tendance et nous souhaitions vous faire part de quelques réflexions sur ce sujet. Voici un exemple typique :

- **Claude :** “Finalement nous sommes passés à Scrum !”
- **Nicolas :** “Et comment ça va ?”
- **Claude :** “Eh bien, c’est beaucoup mieux que ce que nous avions avant…”
- **Nicolas :** “… mais ?”
- **Claude :** “… mais en fait je suis dans une équipe de support et maintenance.”
- **Nicolas :** “Oui, et alors ?”
- **Claude :** “Ça nous plaît de mettre en œuvre les principes de priorisation du backlog de produit, d'équipe auto-organisée, de scrums quotidiens, de rétrospectives, …”
- **Nicolas :** “Et donc, quel est le problème ?”
- **Claude :** “Nous ne parvenons pas à tenir nos sprints.”
- **Nicolas :** “Pourquoi ça ?”
- **Claude :** “Parce c’est dur de s’engager sur un planning de deux semaines. Le principe de l’itération n’a pas beaucoup de sens pour nous, nous travaillons juste sur ce qui est le plus urgent chaque jour. Peut-être faudrait-il réduire les sprints à une semaine ?”

- **Nicolas :** “Pouvez-vous vous engager sur une semaine de travail ? Va-t-on vous laisser travailler en paix pendant une semaine ?”

- **Claude :** “Pas vraiment, de nouveaux problèmes surviennent chaque jour. Peut-être que si nous réduisions les sprints à 1 jour…”

- **Nicolas :** “Est-ce que vous arrivez à résoudre vos problèmes en moins d’une journée ?”

- **Claude :** “Non, cela prend parfois plusieurs jours.”

- **Nicolas :** “Donc cela ne fonctionnera pas mieux avec des sprints de 1 journée. Avez-vous envisagé de vous passer des sprints ?”

- **Claude :** “Eh bien, franchement, on aimerait bien. Mais est-ce que cela ne va pas contre Scrum ?”

- **Nicolas :** “Scrum est juste un outil. C’est vous qui choisissez quand et comment l’utiliser. N’en soyez pas esclave !”

- **Claude :** “Que devons-nous faire alors ?”

- **Nicolas :** “Avez-vous entendu parler de Kanban ?”

- **Claude :** “Qu’est-ce que c’est ? Quelle est la différence avec Scrum ?”

- **Nicolas :** “Lisez ce livre !”

- **Claude :** “Mais j’aime beaucoup les autres principes de Scrum, est-ce qu’il faut vraiment changer ?”

- **Nicolas :** “Non, vous pouvez combiner les deux techniques !”

- **Claude :** “Ah bon ? Et comment ?”

- **Nicolas :** “Il suffit de lire la suite…”

Objectif de ce livre

Si vous vous intéressez aux méthodes agiles pour le développement de logiciel, vous avez probablement entendu parler de Scrum, et vous avez peut-être également entendu parler de Kanban. La question que nous entendons de plus en plus souvent est “C'est quoi Kanban, quelle différence avec Scrum ?” Sont-ils complémentaires et de quelle façon ? Dans quels cas peuvent-ils rentrer en conflit ?

Le but de ce livre est de vous donner une vision plus claire qui vous permettra de comprendre comment Kanban et Scrum peuvent être utiles dans votre environnement de travail.

Faites-nous savoir si nous avons réussi !

1ère Partie - Comparaison

Cette première partie du livre consiste en un essai de comparaison objective et pratique de Scrum et Kanban. Il s'agit d'une petite mise à jour de la version de l'article original "Kanban vs Scrum" publié en avril 2009. Cet article étant devenu populaire, j'ai alors décidé d'en faire un livre et j'ai demandé à mon collègue Mattias de l'enrichir avec une étude de cas "depuis les tranchées" de l'un de nos clients. Excellent boulot ! Vous pouvez passer directement à la Partie II si vous préférez commencer par l'étude de cas, je ne serai pas vexé. En fait, peut-être juste un peu.

/Henrik Kniberg

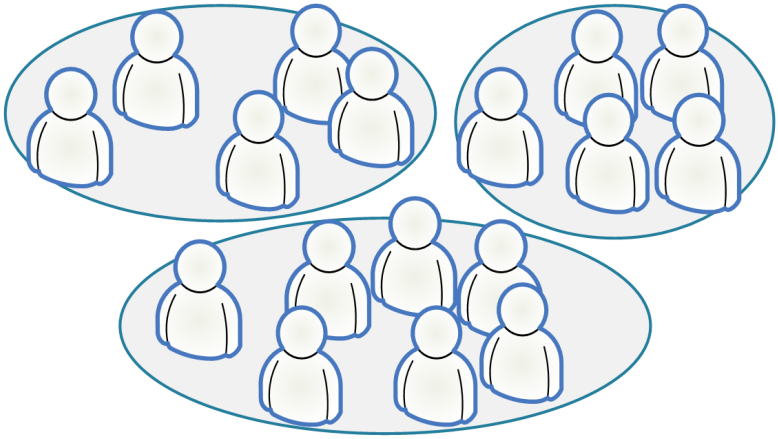
1

Que sont Scrum et Kanban ?

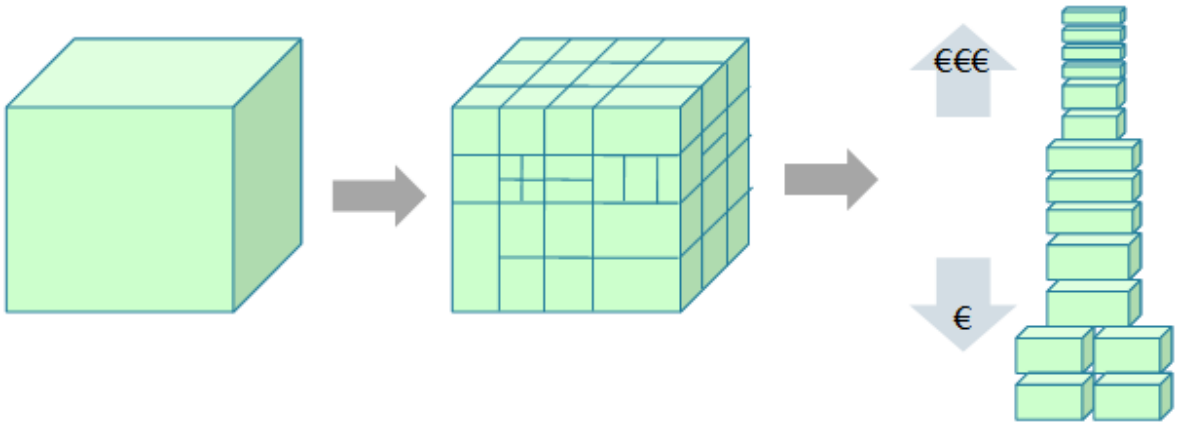
OK, essayons de résumer Scrum et Kanban, chacun en moins de 100 mots.

Scrum en bref

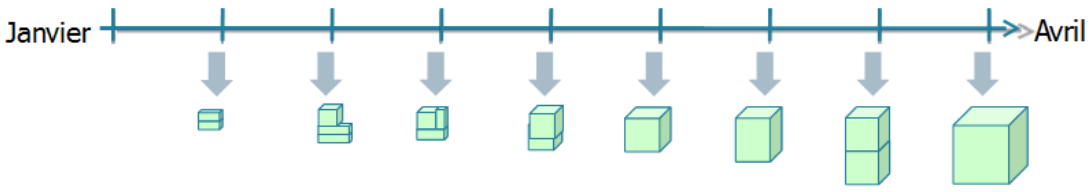
- **Divisez votre organisation** en petites équipes multidisciplinaires et auto-organisées.



- **Divisez votre travail** en une liste de petits livrables concrets. Triez cette liste par priorité et estimez la taille relative de chaque élément.



- **Divisez le temps** en petites itérations de durée fixe (appelées des sprints et durant habituellement de 1 à 4 semaines) et faites une démonstration à l'issue de chaque sprint avec un produit potentiellement livrable.



- **Optimisez le planning de la version** et mettez à jour les priorités en collaboration avec le client, sur la base de ce que vous avez appris après chaque sprint.

- **Optimisez le processus** en organisant une rétrospective après chaque sprint.

Ainsi, au lieu d'avoir un **grand groupe** passant **beaucoup de temps** sur la construction d'une **grande chose**, nous avons une **petite équipe** passant **un peu de temps** à construire une **petite chose**… mais **intégrant régulièrement** pour voir l'ensemble.

150 mots … on y est presque.

Pour plus de détails consultez "Scrum and XP from the Trenches"[1]. Le livre est disponible gratuitement en ligne. Je connais l'auteur, il est sympa ☺

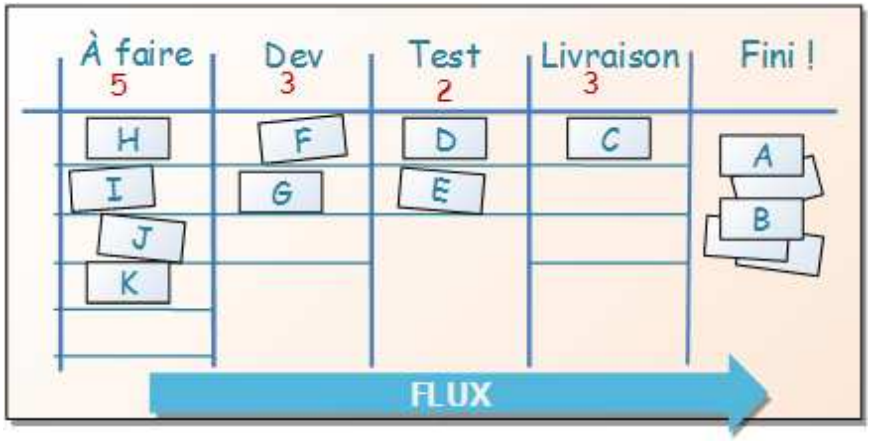
http://www.crisp.se/ScrumAndXpFromTheTrenches.html

Pour plus de liens sur Scrum, je vous renvoie sur http://www.crisp.se/scrum

[1] NdT : le livre a été traduit en français : http://www.infoq.com/resource/news/2007/06/scrum-xp-book/en/resources/ScrumAndXpFromTheTrenches_French.pdf

Kanban en bref

- **Visualisez le workflow :**
    - Divisez votre travail, décrivez chaque élément sur une fiche et mettez-la au mur.
    - Tracez des colonnes, donnez-leur le nom des étapes du workflow et placez y les éléments de travail.
- **Limitez le TAF** : fixez des limites précises indiquant combien d'éléments peuvent être placés dans chaque étape du workflow.
- **Mesurez le temps de cycle** (temps moyen pour traiter complètement un élément, appelé "lead time" en anglais), optimisez le processus pour que le temps de cycle soit aussi court et prévisible que possible.



Vous trouverez des liens utiles sur Kanban à : http://www.crisp.se/kanban

2

Quels sont les liens entre Scrum et Kanban ?

Scrum et Kanban sont tous les deux des outils de processus

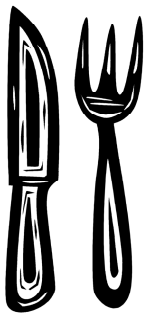
Outil = n'importe quoi pouvant être utilisé pour accomplir une tâche ou atteindre un but.

Processus = façon dont vous travaillez.

Scrum et Kanban sont des *outils de processus* en ce sens qu'ils vous aident à travailler plus efficacement en vous disant, dans une certaine mesure, quoi faire. Java est un outil, il vous fournit un moyen plus simple de programmer un ordinateur. Une brosse à dent est aussi un outil, elle vous aide à atteindre vos dents afin de pouvoir les nettoyer.

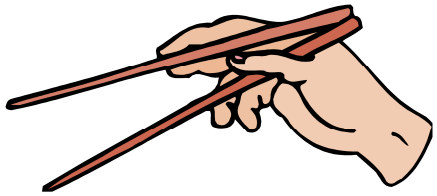
Comparer les outils pour mieux les comprendre et non pour les juger

Couteau et fourchette : quel est le meilleur outil ?



Cette question n'a pas trop de sens, n'est-ce pas ? Parce que la réponse dépend de votre contexte. Pour manger des boulettes de viande, la fourchette est probablement le meilleur choix. Pour hacher des champignons, le couteau est probablement le meilleur choix. Pour jouer

du tambour sur la table, les deux feront l'affaire. Pour manger un steak, vous aurez vraisemblablement besoin d'utiliser les deux outils ensemble. Pour manger du riz, … eh bien … certains préfèrent la fourchette tandis que d'autres préfèrent les baguettes.



Donc, en comparant des outils, on se doit d'être prudent. Comparez pour mieux comprendre et non pour juger.

Aucun outil n'est complet, aucun outil n'est parfait

Comme tout outil, Scrum et Kanban ne sont ni parfaits ni complets. Ils ne vous disent pas *tout* ce que vous avez à faire, ils vous fournissent juste quelques contraintes et quelques directives. Par exemple, Scrum vous contraint à avoir des itérations de durée fixe et des équipes multidisciplinaires, et Kanban vous contraint à utiliser des tableaux visibles et à limiter la taille de vos files d'attente.

Fait intéressant, l'intérêt d'un outil est qu'il *limite les options*. Un outil de processus qui vous permet de faire n’importe quoi n'est pas très utile. On pourrait appeler ce processus “Faire N’importe Quoi” ou même “Faire La Bonne Chose”. Le processus “Faire La Bonne Chose” est garanti pour marcher, c'est comme une baguette magique ! Si ça ne marche pas, c'est parce que vous n'êtes manifestement pas en train d'appliquer le processus ☺

Utiliser les bons outils vous aide à réussir, mais ce n'est pas une garantie de succès pour autant. Il n'est pas facile de discerner entre la réussite/l’échec d’un *projet* et la réussite/l'échec d’un *outil*.

- Un projet peut réussir grâce à un très bon outil.
- Un projet peut réussir malgré un mauvais outil.
- Un projet peut échouer à cause d’un mauvais outil.
- Un projet peut échouer malgré un très bon outil.

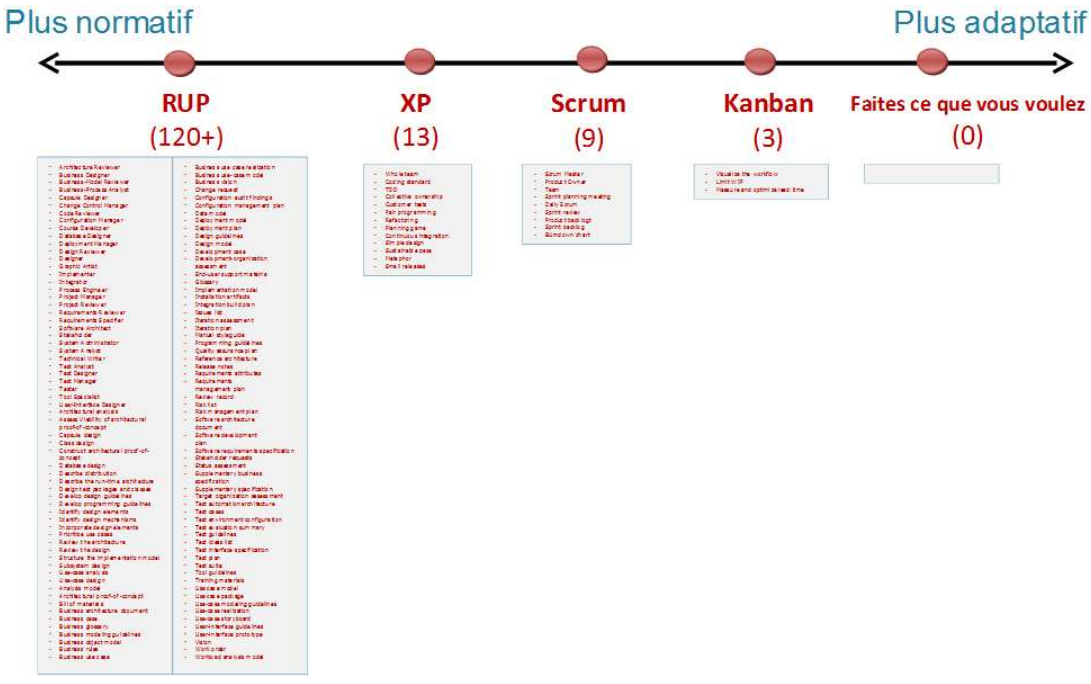
Scrum est plus normatif que Kanban

On peut comparer les outils en regardant le nombre de règles qu'ils fournissent. *Normatif* signifie "plus de règles à suivre" et *adaptatif* signifie "moins de règles à suivre". 100% de normatif signifie que vous n'avez pas besoin d'utiliser votre cerveau, il y a une règle pour tout. 100% d'adaptatif équivaut à faire n'importe quoi, il n'y a aucune règle ni aucune contrainte. Comme vous pouvez le deviner, les deux extrêmes de l'échelle sont ridicules.

Les méthodes agiles sont parfois qualifiées de méthodes *légères*, en particulier parce qu'elles sont moins normatives que les méthodes traditionnelles. D'ailleurs, le premier principe du Manifeste Agile est "Les Individus et leurs Interactions plutôt que les Processus et les Outils".

Scrum et Kanban sont tous les deux très adaptatifs, mais, *relativement parlant*, Scrum est plus normatif que Kanban. Scrum vous donne plus de contraintes et par conséquent vous laisse moins de possibilités. Par exemple Scrum prescrit l'utilisation d'itérations de durée fixe, Kanban ne le fait pas.

Comparons quelques outils sur une échelle normatif *vs* adaptatif :



RUP est assez normatif – avec plus de 30 rôles, plus de 20 activités et plus de 70 artefacts ; une énorme quantité de choses à apprendre. Vous n'êtes cependant pas censé tout utiliser, RUP encourage à sélectionner le sous-ensemble adapté à votre projet. Malheureusement, cela semble être difficile dans la pratique. "Hmmmm… aurons-nous besoin de l'artefact

Conclusion de l'Audit des Configurations ? Aurons-nous besoin d'un rôle de *Responsable de contrôle du changement* ? Pas sûr, mais prenons les juste au cas où." C'est une des raisons pour lesquelles les implémentations de RUP sont finalement généralement très lourdes par rapport à des méthodes agiles telles que Scrum et XP.

XP (eXtreme Programming) est plus normatif que Scrum. Il inclut une majorité des règles de Scrum + un ensemble de pratiques d'ingénierie bien spécifiques comme le développement piloté par les tests et la programmation en binôme.

Scrum est moins normatif que XP puisqu'il ne recommande aucune pratique d'ingénierie particulière. Scrum est plus normatif que Kanban cependant, car il impose des choses comme les itérations et les équipes multidisciplinaires.

Une des différences principales entre Scrum et RUP est qu'avec RUP vous vous retrouvez avec beaucoup trop de choses et vous êtes censé supprimer ce dont vous n'avez pas besoin. Avec Scrum, vous partez petit, et vous êtes censé ajouter ce qui manque.

Kanban laisse tout ouvert. Les seules contraintes sont de Visualiser Votre Workflow et de Limiter Votre TAF. Tout près du "Faire N'importe Quoi", mais étonnamment puissant.

Ne vous limitez pas à l'emploi d'un seul outil !

Combinez les outils dont vous avez besoin ! Je peux difficilement imaginer une équipe Scrum réussir sans inclure la plupart des éléments de XP, par exemple. Beaucoup d'équipes Kanban font des scrums quotidiens (une pratique Scrum de réunion avec l'équipe debout). Certaines équipes Scrum rédigent leurs éléments de backlog sous forme de cas d'utilisation (une pratique RUP) ou limitent la taille de leurs files d'attente (une pratique Kanban). Prenez tout ce qui peut marcher pour vous.

Musashi (célèbre samouraï du XVIIe siècle, connu pour sa technique de combat à deux sabres) le disait plus élégamment : "Ne vous attachez pas à une seule arme ou à un seul style de combat en particulier."

Ne vous attachez pas à une seule arme ou à un seul style de combat en particulier.

- Miyamoto Musashi

Faites cependant attention aux contraintes de chaque outil. Par exemple, si vous utilisez Scrum et décidez de cesser d'utiliser les sprints de durée fixe (ou tout autre aspect central de Scrum), alors ne dites pas que vous utilisez Scrum. Scrum est assez minimaliste tel qu'il est, alors si vous supprimez quelque chose et que vous appelez encore ça du Scrum, le mot sera dénué de sens et créera de la confusion. Appelez-ça par exemple “inspiré de Scrum” ou “sous-ensemble de Scrum” ou encore “Scrumayonnaise” ☺

3

Scrum impose des rôles

Scrum est basé sur 3 rôles : le Product Owner (qui donne la vision du produit et définit les priorités), l'Équipe (qui développe le produit) et le ScrumMaster (qui supprime les obstacles et guide l'équipe pour le suivi du processus).

Kanban ne prescrit aucun rôle.

Cela ne signifie pas que vous ne pouvez pas ou ne devez pas avoir un rôle de Product Owner dans Kanban ! Cela signifie simplement que vous n'êtes *pas obligé*. Avec Scrum et Kanban vous êtes libre d'ajouter tous les rôles supplémentaires dont vous avez besoin.

Soyez cependant prudent lorsque vous ajoutez des rôles. Assurez-vous que les rôles supplémentaires apportent effectivement de la valeur et n'entrent pas en conflit avec d'autres éléments du processus. Êtes-vous sûr d'avoir besoin d'un rôle de Chef de Projet ? Dans un grand projet c'est probablement une bonne idée, peut-être que cela aidera les nombreuses équipes et les Product Owners à se synchroniser entre eux. Dans un petit projet, ce rôle pourrait être du gaspillage, ou pire, pourrait conduire à une sous-optimisation et du micro-management.

L'état d'esprit général dans Scrum et Kanban est “maximiser le minimalisme”. Donc en cas de doute, commencer avec le minimum.

Dans le reste du livre, j'utiliserai le terme “Product Owner” pour parler de la personne qui fixe les priorités d'une équipe, quel que soit le processus utilisé.

4

Scrum impose des itérations de durée fixe

Scrum est basé sur des itérations de durée fixe, appelées sprints. Vous pouvez choisir la durée du sprint, mais l'idée générale est de conserver la même durée sur une période de temps significative et ainsi d'établir un *rythme de travail*.

- **Début de sprint** : un backlog de sprint est créé, c'est-à-dire que l'équipe extrait plusieurs éléments spécifiques du backlog de produit, sur la base des priorités fixées par le Product Owner et la quantité d'éléments que l'équipe pense pouvoir traiter dans ce sprint.

- **En cours de sprint** : l'équipe se concentre sur la réalisation des éléments sur lesquels elle s'est engagée. Le périmètre du sprint est fixe.

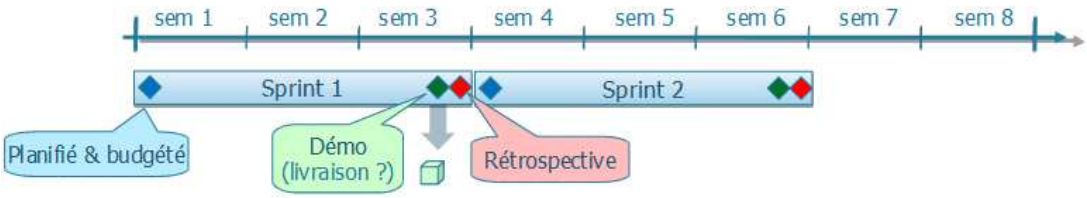
- **Fin de sprint** : l'équipe organise la démonstration du produit pour les parties prenantes concernées, le produit doit être *potentiellement livrable* (c'est-à-dire testé et prêt à être déployé). L'équipe fait ensuite une rétrospective pour discuter de son processus et l'améliorer.

Ainsi, un sprint Scrum représente un rythme de travail unique de durée fixe et combinant trois activités différentes : la planification, l'amélioration des processus, et (idéalement) la livraison d'une version.

Avec Kanban, les itérations de durée fixe ne sont pas imposées. Vous pouvez choisir à quel moment faire de la planification, de l'amélioration des processus et livrer des versions. Vous pouvez faire ces activités sur une base régulière ("livraison tous les lundis") ou à la demande ("livraison chaque fois que nous avons quelque chose d'utile à livrer").

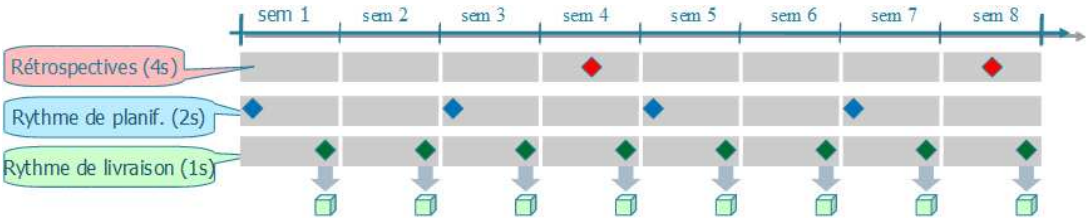
Équipe n°1 (rythme unique)

"Nous pratiquons des itérations comme les sprints de Scrum"



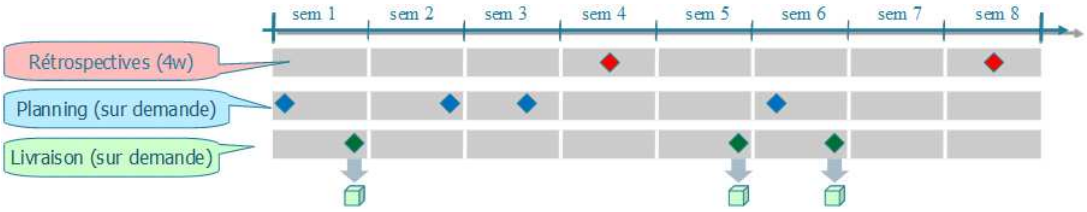
Équipe n°2 (trois rythmes)

“Nous avons trois rythmes différents. Chaque semaine, nous livrons ce qui est prêt à être livré. Toutes les deux semaines, nous avons une réunion de planification, de mise à jour de nos priorités et des plannings de livraison. Toutes les quatre semaines, nous avons une réunion (rétrospective) pour améliorer nos processus.”



Équipe n°3 (rythme essentiellement piloté par les événements)

“Nous déclenchons une réunion de planification chaque fois que nous commençons à ne plus rien avoir à faire. Nous déclenchons la livraison d’une version dès que nous disposons d’un ensemble de MMF (“Minimum Marketable Feature” = “fonctionnalité minimale apportant de la valeur à l’utilisateur”). Nous déclenchons spontanément un cercle de qualité à chaque fois que nous tombons sur un même problème pour la deuxième fois. Nous faisons aussi une rétrospective plus approfondie toutes les quatre semaines.”

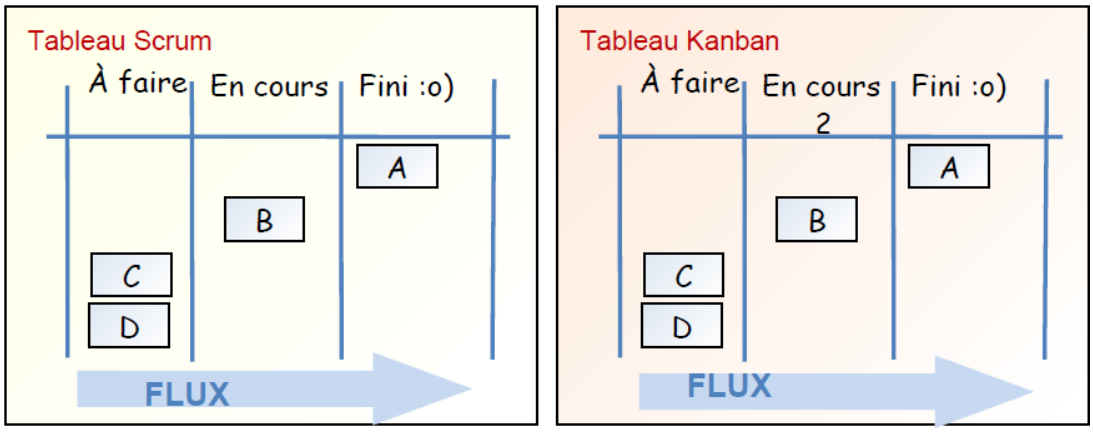


5

Kanban limite le TAF à chaque étape du workflow, Scrum limite le TAF à chaque itération

Avec Scrum, le backlog de sprint montre les tâches qui doivent être exécutées au cours de l'itération. Il est généralement concrétisé par des post-it collés sur un mur, que l'on appelle tableau Scrum ou tableau des tâches.

Alors, quelle est la différence entre un tableau Scrum et un tableau Kanban ? Commençons avec un projet très simple pour comparer les deux :



Dans les deux cas, nous suivons la progression d'un groupe d'éléments à travers un workflow. Nous avons défini trois états : "À faire", "En cours", et "Terminé". Vous pouvez définir autant d'états que vous voulez – certaines équipes ajoutent des états tels que "Intégré", "Testé", "Livré", … N'oubliez cependant pas le principe "*maximaliser le minimalisme*".

Alors, quelle est la différence entre ces deux exemples de tableaux ? Ouaip – le petit 2 dans la colonne du milieu sur le tableau Kanban. C'est tout. Ce chiffre 2 signifie que "il n'y a pas plus de 2 éléments dans cette colonne à un instant donné".

Avec Scrum, il n'existe pas de règle empêchant l'équipe de mettre tous les éléments dans la colonne "En cours" en même temps ! Toutefois, la limite

est implicite puisque le périmètre du sprint est figé. Dans ce cas, la limite implicite est de 4, car il y a seulement 4 éléments dans l'ensemble du tableau. Donc Scrum limite le TAF indirectement, alors que Kanban limite le TAF directement.

La plupart des équipes Scrum finissent par apprendre que c'est une mauvaise idée d'avoir un trop grand nombre d'éléments en cours, et développent une culture visant à terminer les éléments commencés avant d'en démarrer de nouveaux. Certaines ont même décidé de limiter explicitement le nombre d'éléments autorisés dans la colonne "En cours" et donc - tadaaa ! – le tableau Scrum est devenu un tableau Kanban !

Ainsi, Scrum et Kanban limitent tous les deux le TAF, mais de manière différente. Les équipes Scrum mesurent habituellement une vélocité – le nombre d'éléments (ou unités correspondantes telles que les "story points") réalisés par sprint. Une fois que l'équipe connaît sa vélocité, cela devient sa limite de TAF (ou tout au moins une indication). La plupart du temps, une équipe qui a une vélocité moyenne de 10 ne traitera pas plus de 10 éléments (ou "story points") au cours d'un sprint.

Ainsi, avec Scrum, le TAF est *limité par unité de temps*. Avec Kanban, le TAF est *limité par étape du workflow*.

Dans l'exemple Kanban ci-dessus, 2 éléments au plus peuvent être dans l'état "En cours" à un moment donné, indépendamment de toute durée. Vous avez besoin de définir la limite à appliquer à chaque état du workflow, sachant que l'idée générale est de limiter le TAF de tous les états, en commençant le plus tôt possible et en terminant le plus tard possible dans le flux de valeur. Donc dans l'exemple ci-dessus, nous devrions ajouter une limite TAF à l'état "À faire" (ou tout autre nom que vous donnez à votre file d'entrée). Une fois que nous avons défini nos limites de TAF, nous pouvons commencer à mesurer et estimer le temps de cycle, c'est-à-dire le temps qu'il faut à un élément, en moyenne, pour traverser tout le tableau. Disposer de temps de cycle prédictibles nous permet de nous engager sur des niveaux de service (en anglais SLA : Service Level Agreement) et de planifier de façon réaliste les livraisons.

Si la taille des éléments varie de façon spectaculaire alors il est possible d'envisager de définir les limites de TAF en story points, ou tout autre unité de taille que vous utilisez. Certaines équipes préfèrent investir du temps à re-découper en éléments d'à peu près la même taille ce qui permet d'éviter ce type de considération et réduit le temps passé à estimer les choses (vous pourriez même considérer que passer du temps à estimer est un gaspillage). Il est plus facile de créer un système fluide si les éléments sont à peu près de la même taille.

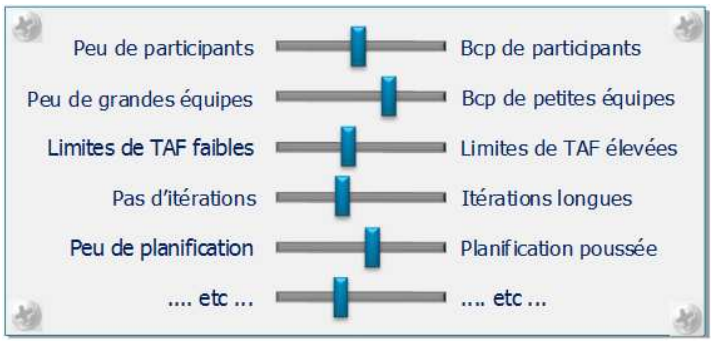
6

Les deux sont empiriques

Imaginez qu'il y ait des boutons sur ces compteurs, et que vous puissiez configurer votre processus en tournant simplement ces boutons. "Je veux une haute capacité, un temps de cycle faible, une qualité et une prédictibilité élevées. Je vais donc tourner les boutons sur 10, 1, 10 et 10, respectivement."

Ce serait géant, n'est-ce pas ? Malheureusement, il n'y a pas de boutons de contrôle de ce type. Aucun que je connaisse du moins. Faites-moi savoir si vous en trouvez.

Au lieu de cela, ce que nous avons, c'est un tas de boutons de contrôle *indirect*.



Scrum et Kanban sont tous les deux empiriques dans le sens où vous expérimentez le processus puis vous l'adaptez à votre environnement de travail. En fait, vous *devez* l'expérimenter. Scrum et Kanban ne sont prévus pour vous fournir toutes les réponses – ils vous donnent juste un ensemble de contraintes pour piloter votre propre amélioration de processus.

- Scrum explique que vous devez disposer d'équipes multidisciplinaires. Donc, qui devrait être dans quelle équipe ? Si vous ne savez pas, expérimentez-le.

- Scrum avance que c'est l'équipe qui choisit la quantité de travail à réaliser dans un sprint. Donc, quelle quantité de travail peut-elle faire ? Si vous ne savez pas, expérimentez-le.

- Kanban dit que vous devez limiter le TAF. Donc, quelle doit être cette limite ? Si vous ne savez pas, expérimentez-le.

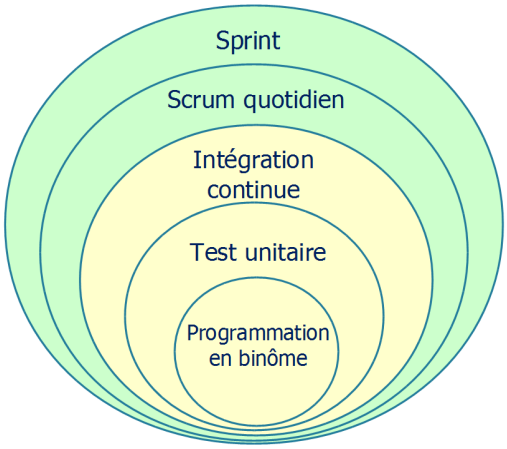
Comme je l'ai mentionné plus haut, Kanban vous impose moins de contraintes que Scrum. Ce qui signifie que vous devez penser à ajuster plus de paramètres, tourner plus de boutons. Cela peut être un désavantage ou un avantage, en fonction de votre contexte. Lorsque vous ouvrez l'interface de configuration d'un logiciel, préférez-vous avoir 3 options ou 100 options de paramétrage ? Probablement quelque part entre les deux. Cela dépend de ce que vous souhaitez paramétrer et de la compréhension que vous avez de l'outil.

Alors, disons que nous allons réduire la limite de TAF, en se basant sur l'hypothèse que cela améliorera notre processus. On observe ensuite comment les choses telles que la capacité, le temps de cycle, la qualité et la prédictibilité évoluent. À partir des résultats, nous en tirons des conclusions, puis nous changeons encore d'autres choses, améliorant constamment notre processus.

Il y a plusieurs noms pour cela. Kaizen (amélioration continue dans le langage Lean), Inspection et Adaptation (dans le langage Scrum), Processus de Contrôle Empirique, ou pourquoi pas La Méthode Scientifique.

L'élément le plus critique est le *feedback*. Changer quelque chose ⇒ Découvrez comment cela s'est passé ⇒ Tirez-en des leçons ⇒ Changez encore quelque chose. D'une manière générale, le feedback doit être le plus rapide possible, afin que vous puissiez adapter votre processus rapidement.

Avec Scrum, le feedback est rythmé par le sprint. Il y a plus court, cependant, surtout si vous combinez avec XP (eXtreme Programming) :



Lorsqu'il est correctement appliqué, le couple Scrum + XP vous offre de nombreuses possibilités de feedback à forte valeur ajoutée.

Le feedback interne - la programmation en binôme permet un feedback en quelques secondes. Les anomalies sont détectées et résolues dans les secondes qui suivent leur apparition (“Hé, cette variable n'est pas censée valoir 3 ?”). Autrement dit, il s'agit d'un feedback pour la question “Sommes-nous en train de bien construire le produit ?”.

La feedback externe - le sprint permet un feedback en quelques semaines. Autrement dit, il s'agit d'un feedback à la question “Sommes-nous en train de construire le bon produit ?”.

Mais qu'en est-il de Kanban ? Eh bien, tout d'abord vous pouvez (et probablement vous devriez) utiliser tous les types de feedback mentionnés ci-dessus dans votre processus que vous utilisiez Kanban ou non. Ce que Kanban vous offre en plus, ce sont des mesures temps réel très utiles.

- Temps de cycle moyen. Mis à jour chaque fois qu'un élément atteint l'état "Fini" (ou le nom que vous donnez à votre colonne la plus à droite).

- Les goulets d'étranglement. Un symptôme connu apparait lorsque la colonne “X” est bourré d'items tandis que la colonne “X+1” est vide. Recherchez dans ce cas des “bulles d'air” dans votre tableau.

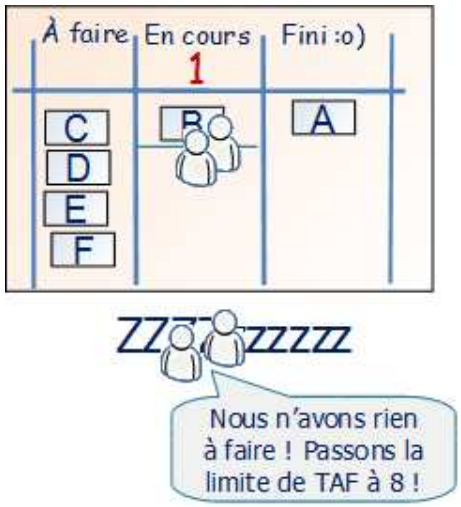
Ce qui est intéressant, concernant les mesures temps réel, c'est que vous pouvez choisir la durée de votre feedback, basée sur la fréquence à laquelle vous êtes prêt à analyser les mesures et à apporter des modifications. Un feedback trop long signifie que votre amélioration du processus se fera lentement. Un feedback trop rapide fait que votre processus n'aura peut être pas le temps de se stabiliser entre chaque changement, ce qui peut causer du gaspillage.

En fait, vous pouvez également expérimenter la durée du feedback … un peu comme un méta-feedback. Bon, je n'irai pas plus loin sur ce sujet.

Exemple : expérimentez les limites de TAF avec Kanban

L'un des moyens typiques d'ajustement de Kanban est la limite du TAF. Mais comment pouvons-nous savoir si l'on a vu juste ?

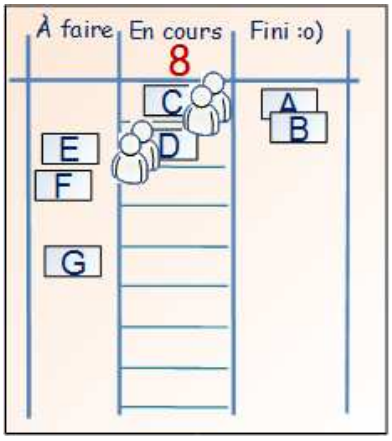
Supposons que nous avons une équipe de 4 personnes, et que nous décidons de commencer par une limite de TAF à 1.



Chaque fois que nous commençons à travailler sur un élément, nous ne pouvons pas démarrer un nouvel élément tant que le premier n'est pas Fini. Donc, cela va très vite.

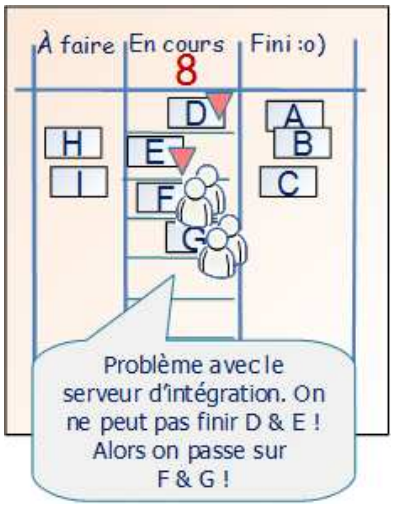
Super ! Mais il s'avère que c'est le plus souvent impossible pour 4 personnes de travailler sur le même élément (dans cet exemple), donc nous aurons des gens inactifs. Si cela ne se produit qu'une fois ce n'est pas un problème, mais si cela se produit régulièrement, le temps de cycle moyen va augmenter. Fondamentalement, un TAF de 1 signifie que les éléments vont rapidement traverser l'état "En cours", par contre ils vont rester coincés dans l'état "À faire" plus longtemps que nécessaire, de sorte que le temps de cycle total (= temps de traversée du workflow complet) sera inutilement élevé.

Alors, si un TAF de 1 est trop faible, qu'est-ce-que cela donne avec un TAF de 8 ?

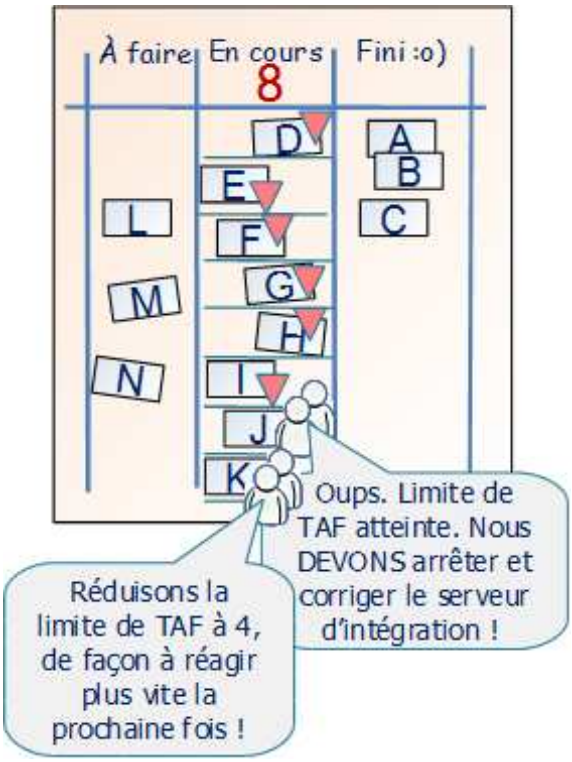


Cela fonctionne mieux pendant un temps. Nous constatons que le travail en binôme est plus rapide en moyenne. Ainsi avec une équipe de 4 personnes, nous avons habituellement 2 éléments dans l'état "En cours" à un moment donné. Un TAF de 8 est juste une limite haute, donc si on le baisse c'est mieux !

Supposez maintenant que nous avons un problème avec le serveur d'intégration, donc nous ne pouvons pas finir de traiter les éléments (notre définition de "Fini" comprend l'intégration). Ce genre de problème arrive parfois, non ?

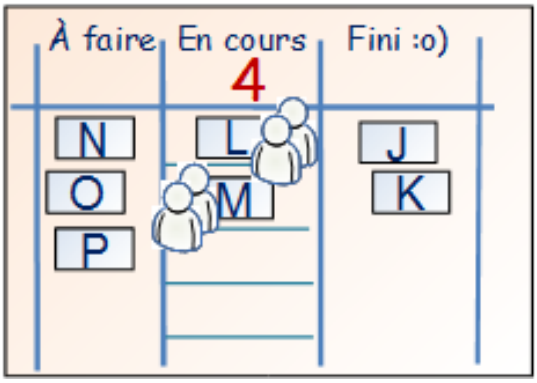


Puisque nous ne pouvons pas terminer les éléments D et E, nous commençons à travailler sur l'élément F. Nous ne pouvons pas non plus intégrer celui-ci, donc nous traitons un nouvel élément G. Au bout d'un moment, nous atteignons notre limite Kanban - 8 éléments dans "En cours".



À ce stade, nous ne pouvons plus ajouter d'éléments. Eh, nous ferions mieux de régler ce problème de serveur d'intégration ! La limite de TAF nous a invités à réagir et à éliminer le goulet d'étranglement au lieu d'empiler du travail non fini.

C'est pas mal. Mais si la limite de TAF était de 4, nous aurions réagi beaucoup plus tôt, nous donnant ainsi un meilleur temps de cycle moyen. Donc c'est un équilibre à trouver. Nous mesurons le temps de cycle moyen et essayons d'ajuster constamment nos limites de TAF pour optimiser le temps de cycle.



Au bout d'un moment, on peut constater que des éléments s'entassent dans l'état "À faire". Peut-être serait-il temps d'ajouter une limite de TAF à cet endroit également.

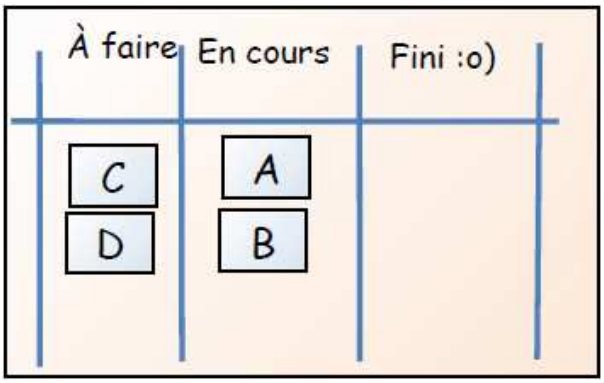
D’ailleurs pourquoi avons-nous besoin d'une colonne “À faire” ? Eh bien, si le client était toujours disponible pour dire à l’équipe quoi faire la fois suivante, alors la colonne “À faire” ne serait pas nécessaire. Mais dans notre cas, le client n'est pas toujours disponible, de sorte que la colonne “À faire” donne à l'équipe un petit buffer de travail.

Expérimentez ! Ou, comme les Scrumologistes le disent : Inspectez et Adaptez !

7

Scrum autorise peu de changement dans une itération

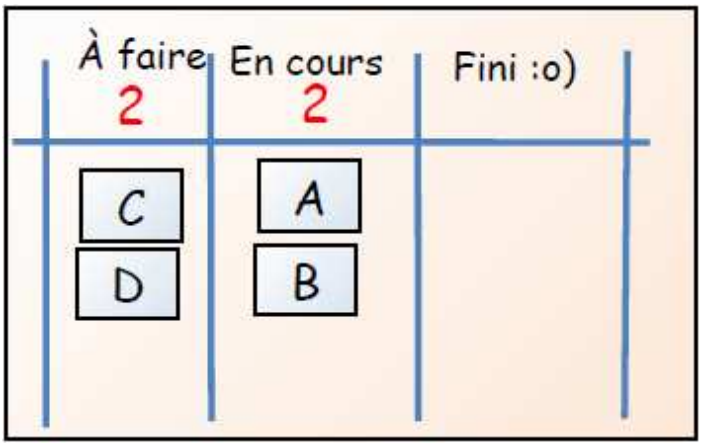
Disons que notre tableau Scrum ressemble à cela :



Que se passe-t-il si quelqu'un se présente et souhaite ajouter l'élément E au tableau ?

Une équipe Scrum répondra typiquement quelque chose comme "Non, désolé, nous nous sommes engagés sur le périmètre A+B+C+D pour ce sprint. Mais n'hésitez pas à ajouter l'élément E au backlog du produit. Si le Product Owner estime que c'est un élément de forte priorité, nous le traiterons dans le sprint suivant." Des sprints de bonne durée donnent à l'équipe tout juste assez de temps pour obtenir une version du produit, tout en permettant au Product Owner de gérer et mettre régulièrement à jour les priorités.

Et que dirait l'équipe Kanban ?



Une équipe Kanban répondrait : “N'hésitez pas à ajouter l’élément E dans la colonne “À faire”. Mais la limite est de 2 pour cette colonne, donc vous devrez dans ce cas retirer C ou D. Nous travaillons en ce moment sur A et B et, dès que nous en aurons la capacité, nous dépilerons un élément de la colonne “À faire”.

Ainsi, le temps de réponse (le temps qu'il faut pour répondre à un changement de priorité) d'une équipe Kanban correspond au temps pour retrouver de la capacité à traiter, suivant le principe général de “un élément qui sort = un élément qui rentre” (pilotage par les limites du TAF).

Le temps de réponse d'une équipe Scrum est en moyenne équivalent à la moitié d’un sprint.

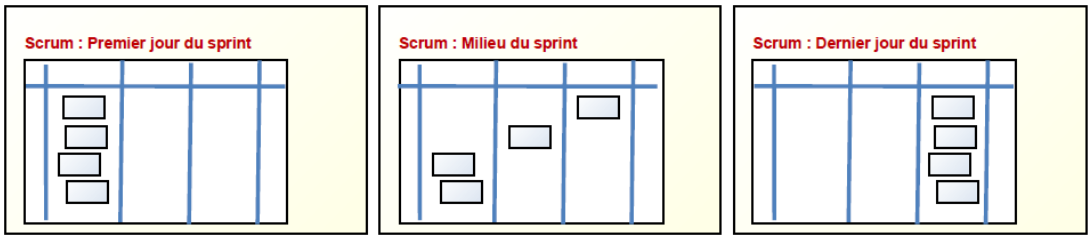
Avec Scrum, le Product Owner ne peut pas toucher au tableau Scrum puisque l'équipe s'est engagée sur un ensemble spécifique d’éléments dans le sprint. Avec Kanban, vous définissez vos propres règles de base précisant qui est autorisé à changer quoi sur le tableau. Typiquement, on attribue au Product Owner une colonne “À faire” ou “Prêt” ou “Backlog” ou “Proposé” à l’extrême gauche où il peut apporter des changements quand il le veut.

Ces deux approches ne sont cependant pas exclusives. Une équipe Scrum *peut* décider d'autoriser un Product Owner à changer les priorités au milieu d’un sprint (même si c’est normalement considéré comme une exception). Et une équipe Kanban *peut* décider d'ajouter des restrictions sur le moment où les priorités peuvent être changées. Une équipe Kanban peut même décider de s’engager sur des itérations de durée fixe avec un périmètre fixé, tout comme en Scrum.

8

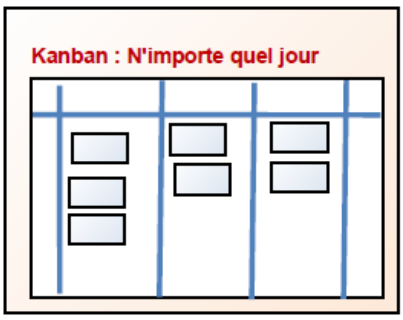
Le tableau Scrum est réinitialisé à chaque début d'itération

Un tableau Scrum ressemble généralement à quelque chose de ce genre au cours des différents moments d'un sprint.



Lorsque le sprint est fini, le tableau est effacé et tous les éléments sont enlevés. Un nouveau sprint est lancé et après la réunion de planification du sprint suivant, nous avons un nouveau tableau Scrum, avec de nouveaux éléments dans la colonne la plus à gauche. Techniquement, il s'agit d'un gaspillage mais pour des équipes Scrum expérimentées, cela ne prend généralement pas trop de temps, et le processus de réinitialisation du tableau peut apporter un sentiment agréable d'accomplissement et de fin officielle. Un peu comme laver la vaisselle après le dîner - le faire est une épreuve, mais c'est agréable une fois que c'est fini.

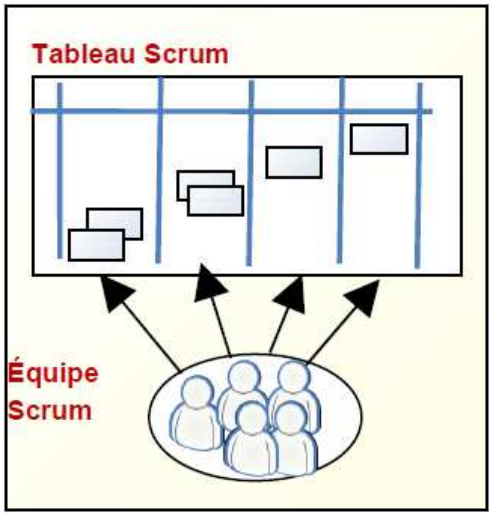
Avec Kanban, le tableau est généralement persistant : vous n’avez donc pas besoin de le réinitialiser et de démarrer avec autre chose.



9

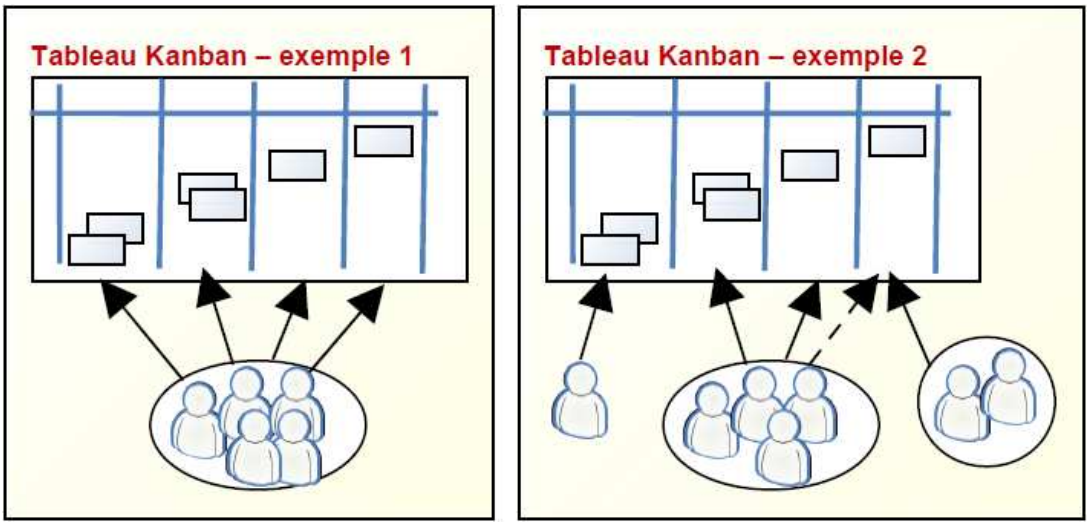
Scrum impose des équipes multidisciplinaires

Un tableau Scrum est détenu par une et une seule équipe Scrum. Une équipe Scrum est multidisciplinaire, elle possède toutes les compétences nécessaires pour mener à bien tous les éléments du sprint. Un tableau Scrum est généralement visible de toute personne intéressée, mais seule l'équipe Scrum peut le modifier : c'est l'outil qui lui permet de maîtriser ses engagements sur le sprint en cours.



Avec Kanban, les équipes multidisciplinaires sont facultatives, et un tableau peut ne pas être détenu par une équipe spécifique. Un tableau est lié à un workflow unique, pas nécessairement à une seule équipe.

Voici deux exemples :



Exemple 1 : l'ensemble du tableau est géré par une équipe multidisciplinaire. Comme en Scrum.

Exemple 2 : le Product Owner définit les priorités dans la colonne 1. Une équipe de développeurs multidisciplinaire développe le produit (colonne 2) et le teste (colonne 3). La livraison (colonne 4) est assurée par une équipe de spécialistes. Il y a un léger chevauchement de compétences, donc si l'équipe de livraison devient un goulet d'étranglement, l'un des développeurs va les aider.

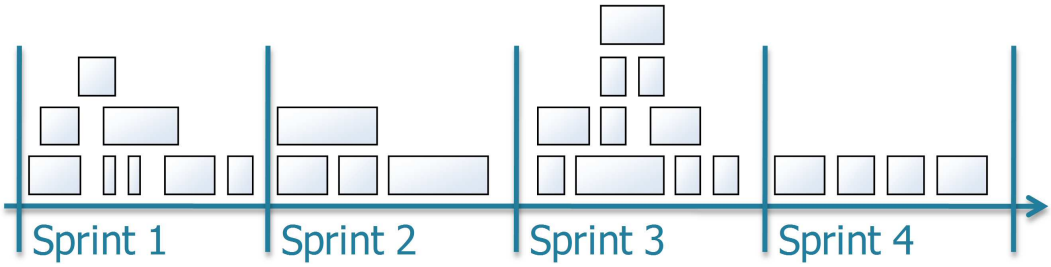
Avec Kanban, vous devez donc établir des règles de base pour définir qui utilise le tableau et de quelle façon, puis vous expérimentez les règles pour optimiser le flux.

10

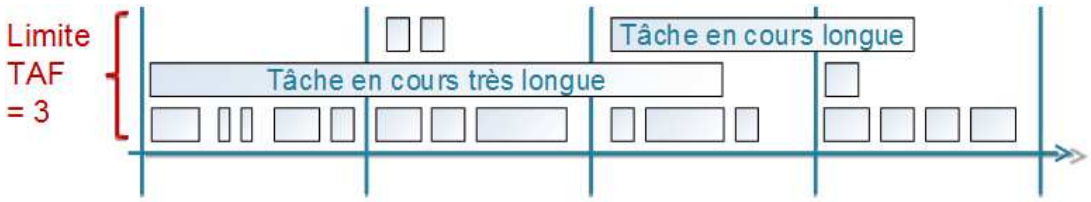
Les éléments du backlog Scrum doivent tenir dans un sprint

Scrum et Kanban sont tous les deux basés sur le développement incrémental, c'est-à-dire la division du travail.

Une équipe Scrum s'engagera uniquement sur des éléments qu'elle pense pouvoir traiter dans un sprint (en ayant préalablement défini le "Fini"). Si un élément est trop gros pour tenir dans un sprint, l'équipe et le Product Owner essayeront de trouver des moyens de diviser cet élément en plusieurs éléments plus petits jusqu'à ce que cela tienne dans un sprint. Si les éléments sont tous globalement gros, les sprint seront plus longs (mais généralement pas plus de 4 semaines).



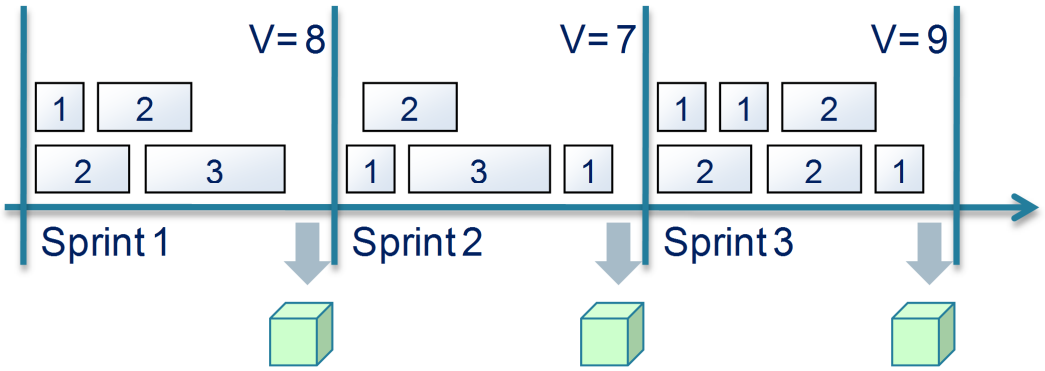
Les équipes Kanban essayent de minimiser le temps de cycle moyen et de maximiser le flux, ce qui incite indirectement à diviser les éléments en plus petits éléments. Mais il n'y a pas de règle explicite indiquant que les éléments doivent être suffisamment petits pour correspondre à une durée fixe. Dans le même tableau nous pourrions avoir un élément qui prend 1 mois à traiter et un autre élément qui prend 1 jour.



11

Scrum impose estimation et vélocité

En Scrum, les équipes sont censées évaluer la taille relative (= quantité de travail) de chaque élément sur lequel elles s'engagent. En additionnant la taille de chaque élément terminé à la fin de chaque sprint, nous obtenons la vélocité. La vélocité est une mesure qui sert à estimer la capacité : la quantité d'éléments que nous pouvons livrer par sprint. Voici un exemple d'équipe avec une vélocité moyenne de 8.



Sachant que la vélocité moyenne est de 8, nous pouvons alors prévoir une capacité de 8 pour les éléments que nous pourrons traiter dans les prochains sprints, et donc planifier de façon réaliste les livraisons.

En Kanban, l'estimation n'est pas imposée. Si vous souhaitez vous engager vous devez décider de la manière de prévoir les choses.

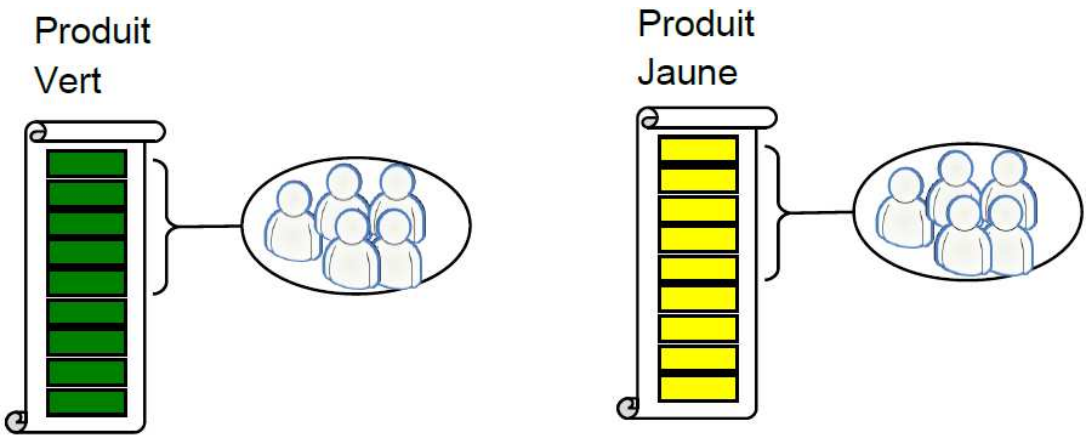
Certaines équipes choisissent de faire des estimations et de mesurer la vélocité, tout comme en Scrum. D'autres équipes choisissent de ne pas faire d'estimation, mais essayent de diviser les choses à faire en éléments d'à peu près de la même taille : ils peuvent alors mesurer la vélocité en terme de nombre d'éléments traités par unité de temps (par exemple, nombre de fonctionnalités par semaine). Certaines équipes groupent les éléments pour obtenir un ensemble de MMF (Minimal Marketable Feature), mesurent le temps de cycle moyen par MMF, et l'utilisent pour établir des SLA (Service Level Agreements). Exemple de SLA : "quand nous nous engageons sur une MMF, elle sera toujours livrée dans les 15 jours".

Il y a toutes sortes de techniques intéressantes pour la planification de release et la gestion des engagements dans le style Kanban, mais aucune technique spécifique n’est prescrite ; alors allez-y, essayez différentes techniques jusqu'à ce que vous en trouviez une qui convient à votre contexte. Vous verrez probablement émerger des bonnes pratiques au fil du temps.

12

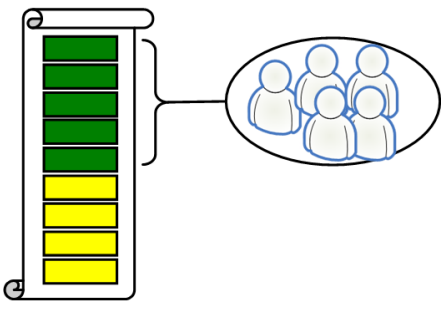
Les deux permettent de travailler sur plusieurs produits simultanément

En Scrum, le “Backlog de produit” est un nom plutôt regrettable, car il implique que tous les éléments doivent concerner le même produit. Voici deux produits, vert et jaune, chacun avec son propre backlog de produit et sa propre équipe :

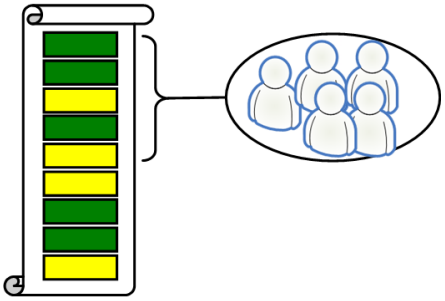


Que faire si vous n'avez qu'une seule équipe ? Eh bien, envisagez que le Backlog de produit devienne le Backlog d'équipe. Il liste les priorités pour les prochains sprints d'une équipe (ou un ensemble d'équipes). Si cette équipe maintient plusieurs produits, fusionnez donc les deux produits dans une seule liste. Cela oblige à établir des priorités entre les deux produits, ce qui est utile dans certains cas. Il existe plusieurs façons de le faire dans la pratique :

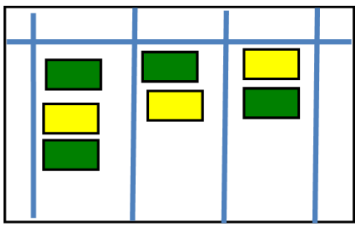
Une stratégie serait d'avoir l'équipe dédiée à un seul produit par sprint :



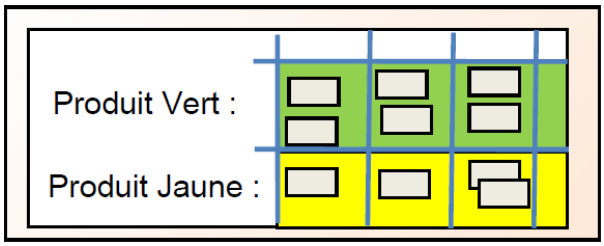
Une autre stratégie serait que l'équipe travaille sur les fonctionnalités des deux produits pendant chaque sprint :



C'est la même chose en Kanban. Nous pouvons avoir plusieurs produits circulant à travers le même tableau. On pourrait les distinguer à l'aide de cartes de différentes couleurs :



… ou en ajoutant des “lignes de nage” :



13

Les deux sont Lean et Agile

Je ne vais pas reprendre ici la Pensée Lean ni le Manifeste Agile, mais d'une manière générale, Scrum et Kanban reposent tous les deux sur les mêmes valeurs et principes. Par exemple :

- Scrum et Kanban sont orientés Juste à temps (JIT = Just In Time) qui est un principe Lean. Cela signifie que l'équipe choisit le moment et la quantité de travail sur laquelle s'engager, et “tire” le travail quand elle est prête, plutôt que de voir ce travail “poussé” de l'extérieur. Exactement comme une imprimante qui tire la page suivante uniquement au moment où elle est prête à l'imprimer (même s'il n'y a qu'un petit stock limité de papier qu'elle peut utiliser).

- Scrum et Kanban sont basés sur une optimisation continue et empirique des processus, qui correspond au principe Kaizen du Lean.

- Scrum et Kanban mettent l'accent sur la réactivité aux changements plutôt qu’au suivi d’un planning (même si Kanban donne en général une réponse plus rapide que Scrum), l'une des quatre valeurs du Manifeste Agile.

… et plus encore.

D'un autre côté, Scrum n’est pas aussi Lean que cela, car il prévoit de grouper la réalisation des éléments dans des itérations à durée fixe. Mais cela dépend de la durée de votre sprint et de ce que vous comparez. Par rapport à un processus traditionnel où l'on intègre et livre 2 à 4 fois par an, une équipe Scrum développant un produit potentiellement livrable toutes les 2 semaines est très Lean.

Si vous continuez avec des sprints de plus en plus courts, vous vous rapprochez de Kanban. Lorsque vous commencez à parler de durée inférieure à 1 semaine, vous pouvez envisager de laisser entièrement tomber le principe de l’itération à durée fixe.

Je l'ai déjà dit et je le dis encore : expérimentez jusqu'à ce que vous trouviez ce qui fonctionne pour vous ! Ensuite, continuez encore à expérimenter ☺

14

Des différences mineures

Voici quelques différences qui semblent moins importantes que celles mentionnées précédemment. Il vaut mieux en prendre conscience tout de même.

Scrum recommande un backlog produit priorisé

En Scrum, la gestion des priorités est toujours faite par le tri du backlog de produit et les changements de priorités prennent effet dans le sprint suivant (pas dans le sprint en cours). En Kanban, vous pouvez choisir n'importe quel mécanisme de priorisation (ou même aucun), et les changements prennent effet dès que la capacité est disponible (plutôt qu'à un moment fixé). Il peut y avoir un backlog de produit ou non et il peut être priorisé ou non.

Dans la pratique, cela fait peu de différence. Sur un tableau Kanban, la colonne la plus à gauche répond en général aux mêmes objectifs que le backlog de produit de Scrum. La liste peut-être ou non triée par ordre de priorité et l'équipe doit définir une règle de décision pour savoir quels éléments traiter/tirer en premier. Exemples de règles de décision :

- Dépiler toujours le premier élément.
- Prendre toujours l'élément le plus ancien (chaque élément est horodaté).
- Prendre n'importe quel élément.
- Prendre environ 20% d'éléments relatifs à la maintenance et 80% de nouvelles fonctionnalités.
- Répartir la capacité de l'équipe de façon à peu près égale entre les produits A et B.
- Prendre toujours les éléments en rouge d'abord, s'il y en a.

En Scrum, un backlog de produit peut aussi être utilisé selon un mode Kanban. Nous pouvons limiter sa taille et créer des règles de décision pour dire comment le prioriser.

Scrum impose des points quotidiens

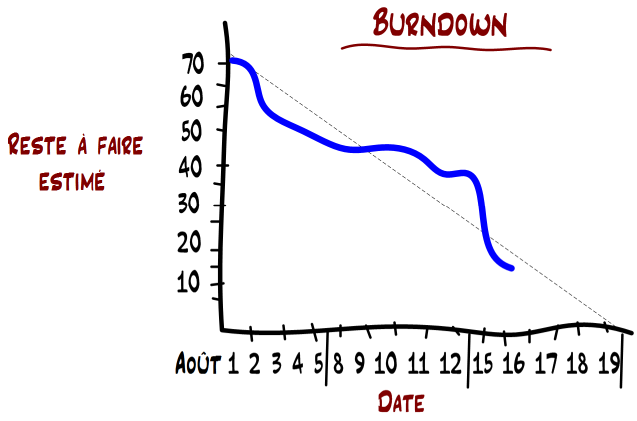
Une équipe Scrum tient une réunion courte (15 minutes au plus), appelée scrum, chaque jour à la même heure et au même endroit. Le but de cette réunion est de faire un point sur ce qui a été fait, de planifier la journée à venir et d'identifier tout problème significatif. On l'appelle aussi parfois standup puisqu'elle se tient habituellement debout (pour qu'elle reste courte et qu'elle maintienne un haut niveau de participation).

Kanban ne prescrit pas les scrums quotidiens, même si beaucoup d'équipes Kanban semblent tout de même les faire. C'est une excellente technique quel que soit le processus utilisé.

En Scrum le format de la réunion est axé sur la personne - chaque personne fait son rapport l'une après l'autre. De nombreuses équipes Kanban utilisent un format plus orienté tableau, en se concentrant sur les goulets d'étranglement et d'autres problèmes visibles. Cette approche est plus souple. Si vous avez 4 équipes partageant le même tableau et menant leurs réunions quotidiennes ensemble, il n'est pas nécessaire d'écouter tout le monde parler tant que l'on se concentre sur les goulets d'étranglement du tableau.

Scrum recommande des burndown charts

Un burndown chart de sprint montre, sur une base quotidienne, le travail restant à réaliser pour le sprint courant.



L'unité de l'axe Y est celle utilisée pour les tâches du sprint. Généralement des heures ou des jours (si l'équipe divise les éléments du backlog en tâches) ou en story points (si l'équipe ne divise pas les éléments). Il existe de nombreuses variantes.

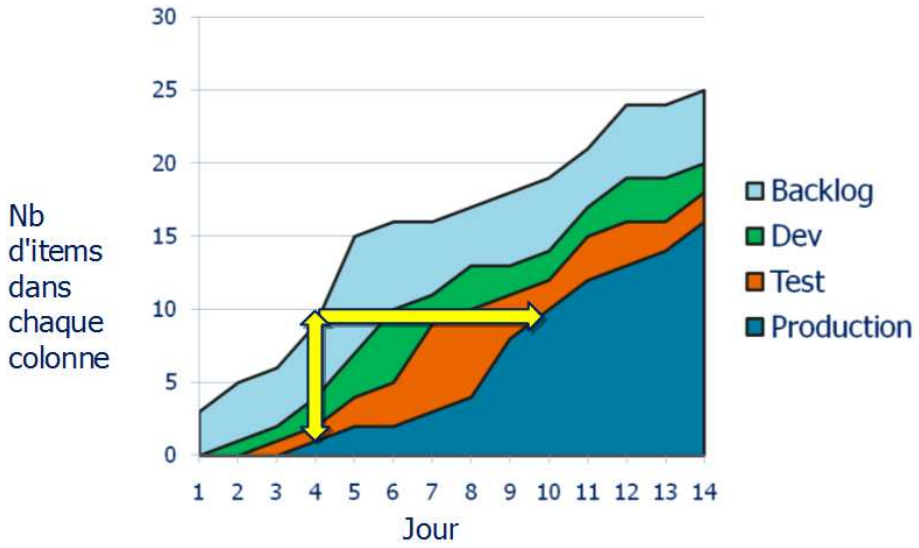
En Scrum, les burndown charts constituent l'un des principaux outils pour le suivi de la progression d'un sprint.

Certaines équipes utilisent également le burndown chart de release, qui suit le même format mais au niveau de la release : il montre généralement combien de story points restent à faire dans le backlog de produit après chaque sprint.

L'objectif principal est de se rendre compte facilement et le plus tôt possible si nous sommes en retard ou en avance, de sorte que nous pouvons nous adapter.

En Kanban, l'emploi des burndown charts n'est pas imposé. En fait, aucun type particulier de graphique n'est imposé. Mais ils sont bien sûr autorisés (y compris les burndowns).

Voici un exemple de diagramme de flux cumulé. Ce type de diagramme illustre parfaitement la pente de votre flux et la manière dont le TAF impacte votre temps de cycle.



Voici comment cela fonctionne. Chaque jour, faites le total du nombre d'éléments dans chaque colonne du tableau Kanban et empilez-les sur l'axe Y. Donc, au 4ème jour, il y avait 9 éléments dans le tableau. En partant de la colonne la plus à droite, il y avait 1 élément en Production, 1 élément en Test, 2 éléments en Dev et 5 éléments dans le Backlog. En traçant, et reliant, ces points tous les jours nous obtenons un beau diagramme comme celui ci-dessus. Les flèches verticales et horizontales illustrent la relation entre le TAF et le temps de cycle.

La flèche horizontale nous montre que les éléments ajoutés au backlog au 4ème jour ont pris en moyenne 6 jours pour atteindre la production. Environ la moitié de ce temps est passé en Test. Nous pouvons voir que si l'on arrivait à limiter le TAF en Test et en Backlog, cela permettrait de réduire significativement le temps de cycle.

La pente de la zone bleu foncé nous montre la vélocité (c'est-à-dire le nombre d'éléments déployés par jour). Au fil du temps on peut voir comment une vélocité supérieure peut réduire le temps de cycle, alors qu'un TAF supérieur accroît le temps de cycle.

La plupart des organisations souhaitent que les choses aillent plus vite (= réduire le temps de cycle). Malheureusement, beaucoup tombent dans le piège en supposant que cela signifie avoir plus de personnes ou faire des heures supplémentaires. Habituellement, la façon la plus efficace pour que les choses aillent plus rapidement est de lisser le flux et de limiter le travail à la capacité disponible, et non en ajoutant des personnes ou en les faisant travailler plus dur. Ce type de diagramme montre pourquoi, et augmente ainsi la probabilité que l'équipe et le management collaborent efficacement.

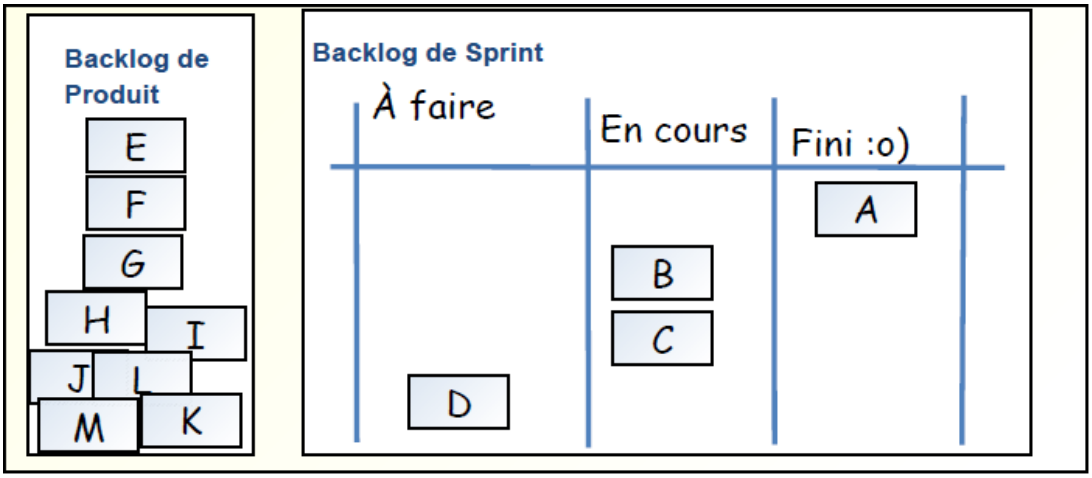
Cela devient encore plus évident si l'on matérialise les files d'attente entre les états du workflow (par exemple "en attente de Test") en plus des files de travail (par exemple "en cours de Test"). Nous voulons absolument minimiser le nombre d'éléments séjournant dans les files, et un diagramme de flux cumulé nous aide à trouver les pistes appropriées pour cela.

15

Tableau Scrum *vs* Tableau Kanban - un exemple moins trivial

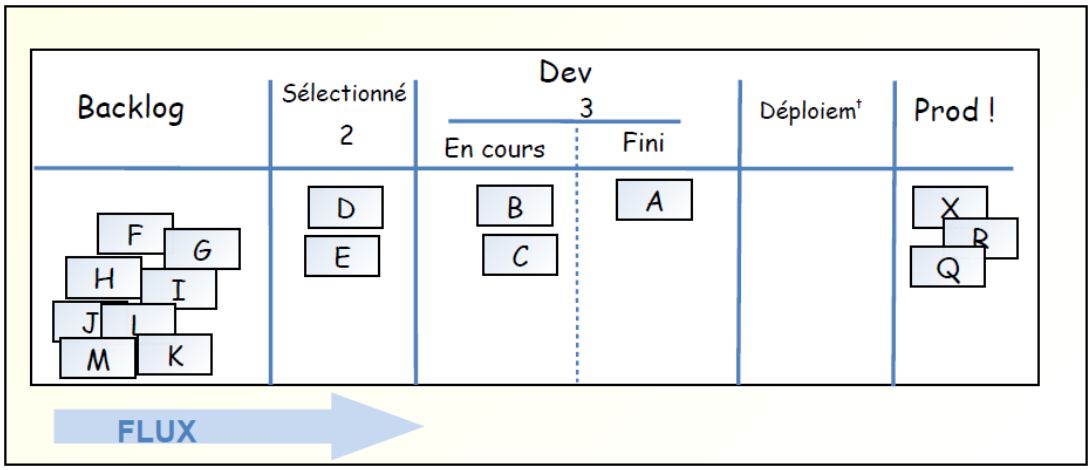
En Scrum, le backlog de sprint ne représente qu'une partie de la vue complète – la partie qui montre ce que l'équipe est en train de faire au cours du sprint. L'autre partie est le backlog de produit – la liste des choses que le Product Owner veut avoir dans les prochains sprints.

Le Product Owner peut consulter mais ne peut pas toucher au backlog de sprint. Il peut changer le backlog de produit autant de fois qu'il veut, mais les modifications ne prennent effet (c'est-à-dire affectent le travail en cours) que dans les sprints suivants.



Lorsque le sprint est terminé, l'équipe "fournit une version potentiellement livrable" au Product Owner. En fait l'équipe termine le sprint, fait une revue de sprint, et montre fièrement les fonctionnalités A, B, C et D au Product Owner. Le Product Owner peut alors décider s'il convient ou non de livrer cette version du produit. La dernière partie – la livraison réelle du produit – n'est habituellement pas incluse dans le sprint, et n'est donc pas visible dans le backlog du sprint.

Dans ce scénario, un tableau Kanban pourrait plutôt ressembler à quelque chose de ce genre :



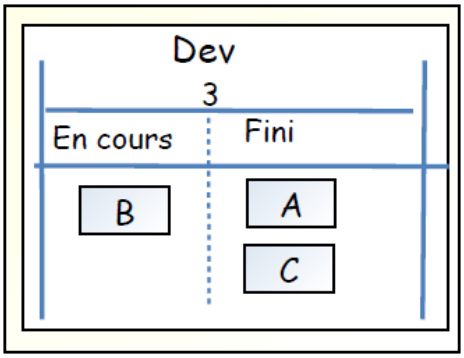
L'ensemble du workflow est représenté sur le même tableau : nous ne sommes pas simplement en train de regarder ce qu'une équipe Scrum est en train de faire dans une itération.

Dans l'exemple ci-dessus, la colonne "Backlog" est juste une liste de souhaits, sans ordre particulier. La colonne "Selected" contient les éléments de priorité haute, avec une limite Kanban à 2. Il peut donc y avoir seulement 2 éléments de priorité haute à un moment donné. Lorsque l'équipe est prête pour commencer à travailler sur un nouvel élément, elle dépile l'élément de plus haute priorité dans la colonne "Selected". Le Product Owner peut apporter des modifications aux colonnes "Backlog" et "Selected" à tout moment, mais pas aux autres colonnes.

La colonne "Dev" (scindée en deux sous-colonnes) montre ce qui est actuellement en cours de développement, avec une limite Kanban à 3. En termes réseau, la limite Kanban correspond à "la bande passante" et le temps de cycle correspond au "ping" (ou temps de réponse).

Pourquoi avons-nous scindé la colonne "Dev" en deux sous-colonnes "En cours" et "Fini" ? C'est pour donner à l'équipe de production la visibilité sur les éléments qu'ils peuvent mettre en production.

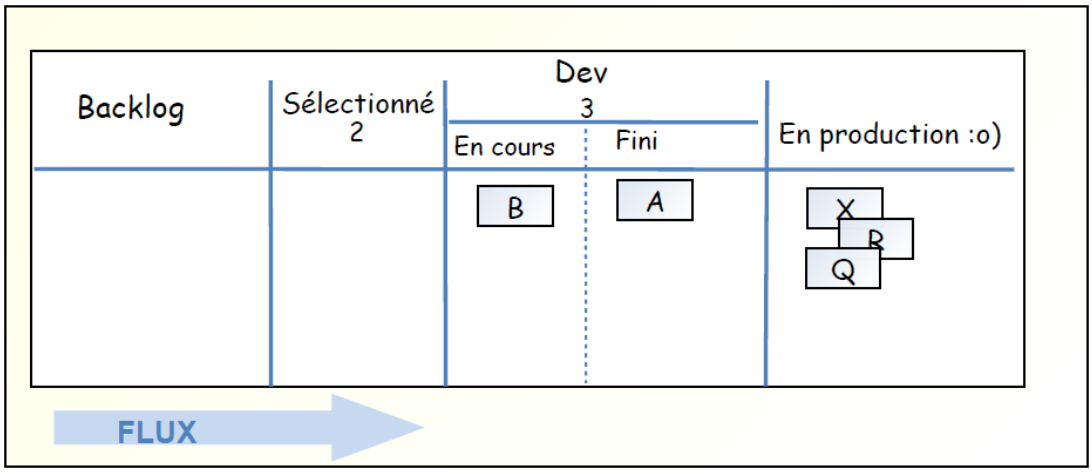
La limite "Dev" à 3 est partagée entre les deux sous-colonnes. Pourquoi ? Supposons qu'il y ait 2 éléments dans la colonne "Fini" :



Cela signifie qu'il ne peut y avoir qu'un seul élément dans la colonne "En cours". Cela implique qu'il y aura une capacité excédentaire, des développeurs qui *pourraient* commencer un nouvel élément, mais qui n'y sont pas autorisés à cause de la limite Kanban. Cela les incite donc fortement à aider à livrer des choses en production, afin de vider la colonne "Fini" et à maximiser le flux. Cet effet est positif et progressif – plus il y a de choses dans la colonne "Fini", moins il peut y avoir de choses dans la colonne "En cours" – ce qui permet à l'équipe de se concentrer sur de bonnes choses.

Flux à une pièce

Un flux à une pièce est une sorte de scénario pour un "flux parfait", où un élément traverse le tableau sans jamais être coincé dans une file d'attente. Cela signifie qu'à chaque instant il y a une personne travaillant sur cet élément. Voici à quoi pourrait ressembler le tableau dans ce cas :

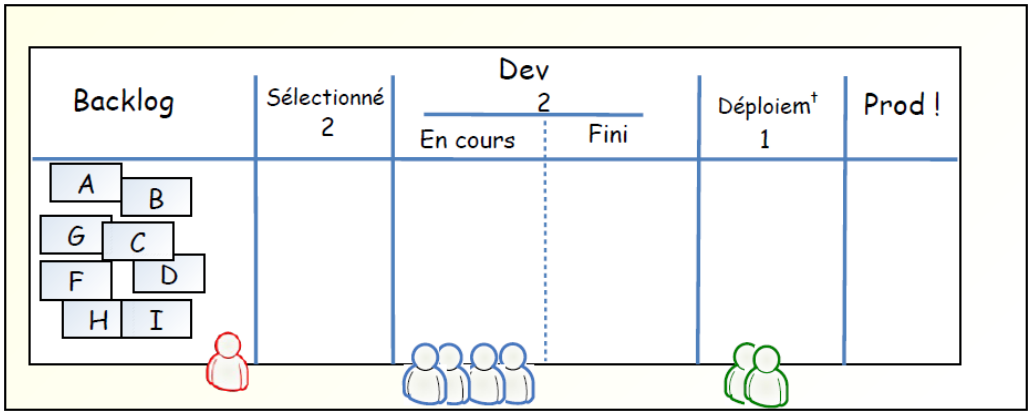


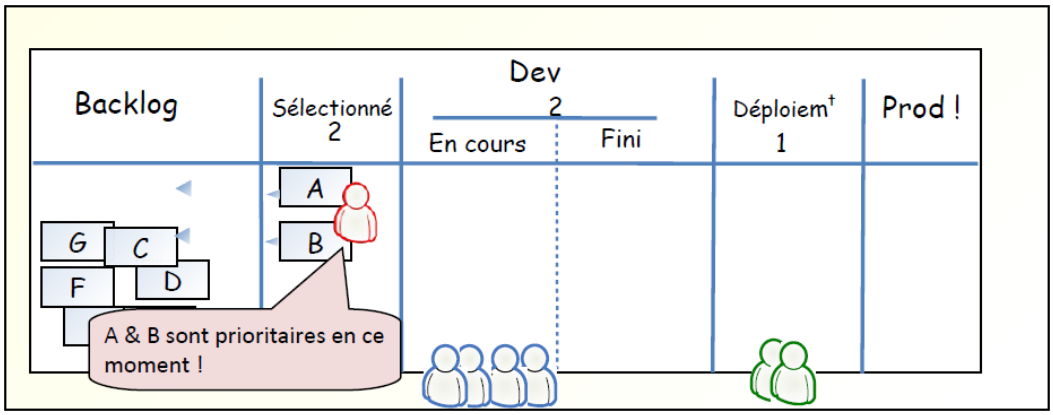
B est en cours d'élaboration, A est sur le point d'être mis en production. Chaque fois que l'équipe est prête pour traiter l'élément suivant, elle demande au Product Owner ce qui est le plus important et obtient une réponse immédiate. Si ce scénario idéal se maintient, nous pouvons nous débarrasser des deux files d'attente "Backlog" et "Sélectionné" et réellement obtenir un temps de cycle très court !

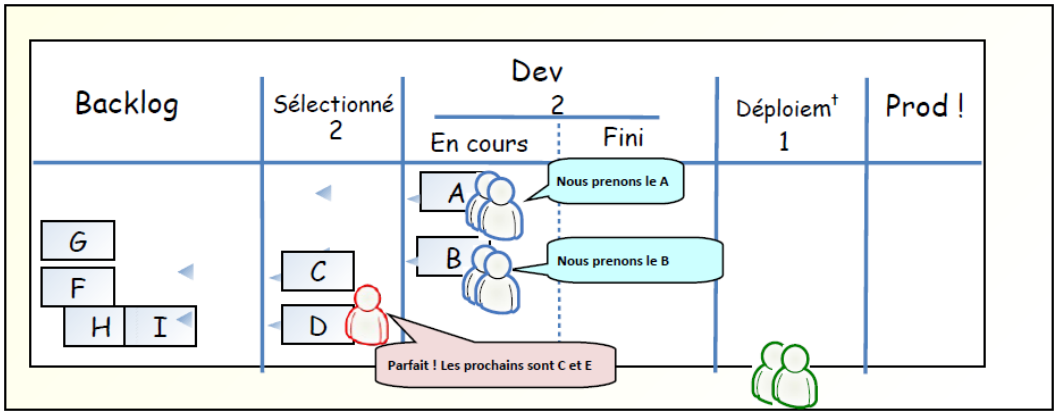
Cory Ladas le dit élégamment : "Le processus de planification idéal devrait toujours fournir à l'équipe de développement la meilleure chose à faire le coup suivant, ni plus ni moins".

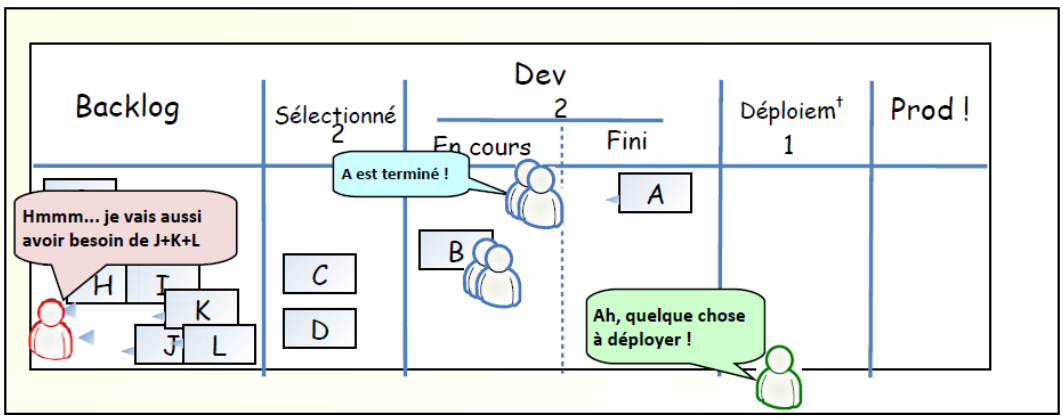
Les limites de TAF sont là pour maîtriser les problèmes, donc si les choses sont fluides les limites de TAF ne sont pas vraiment utiles.

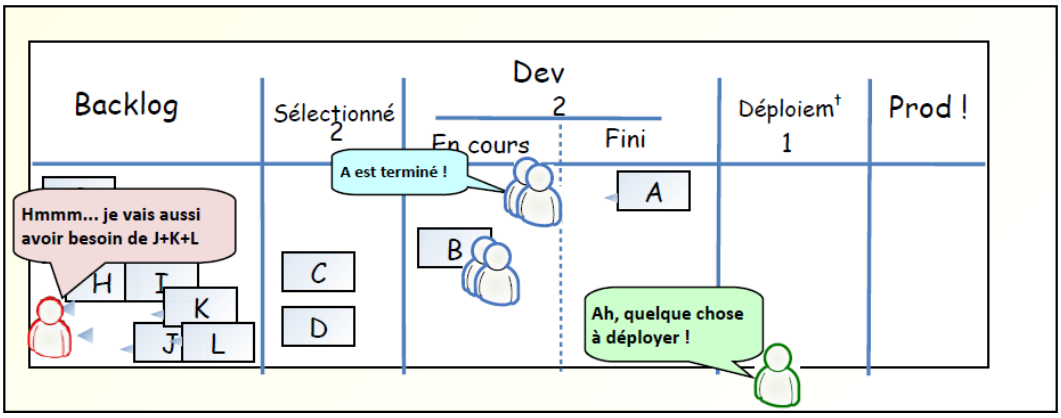
Un jour au pays de Kanban

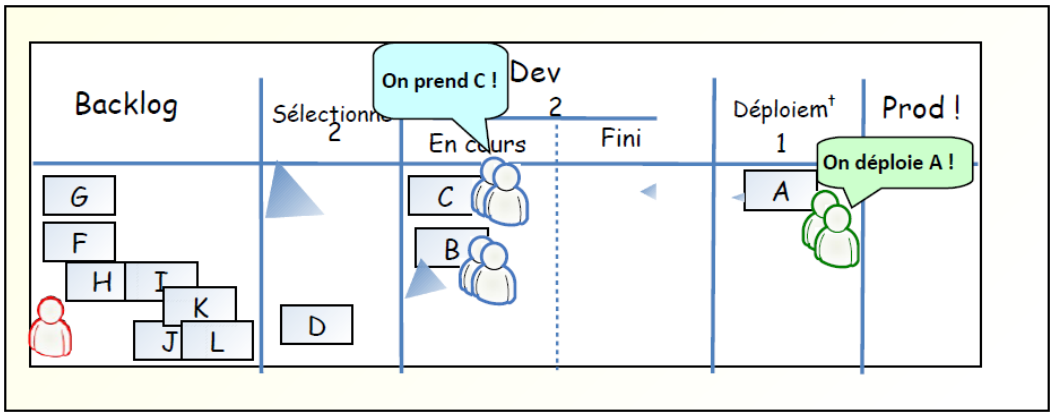


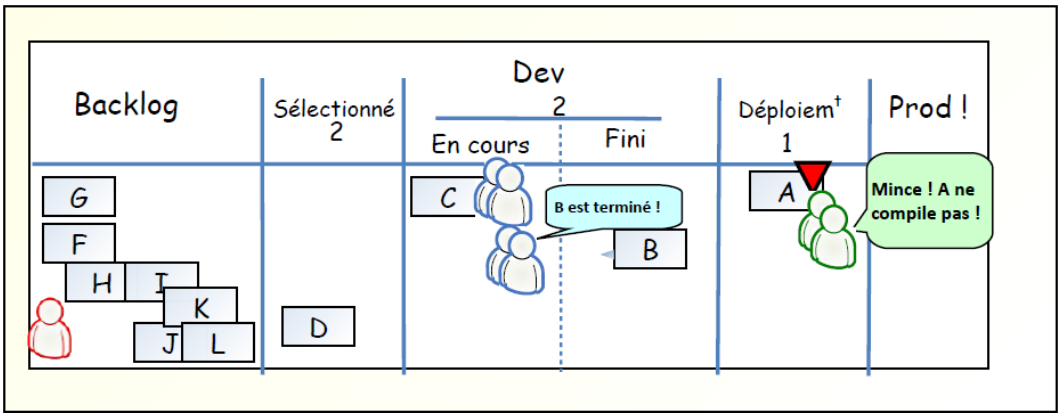


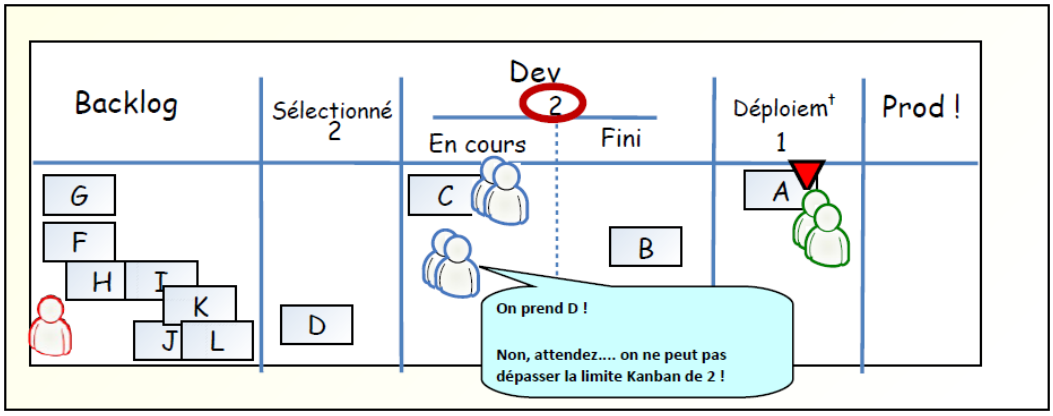


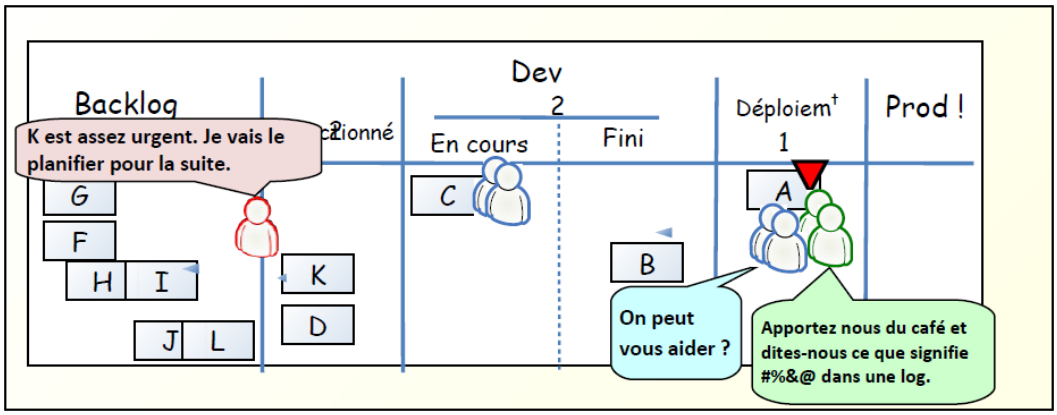


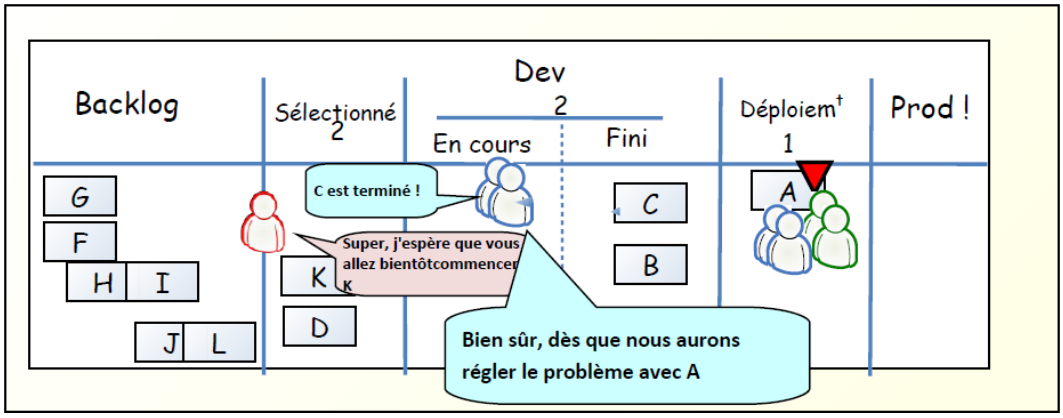

Backlog
Sélectionné 2
Dev 2
En cours
Fini
Déploiem† 1
Prod !
A est terminé !
Hmmm... je vais aussi avoir besoin de J+K+L
Ah, quelque chose à déployer !
A
B
C
D
H
I
K
J
L

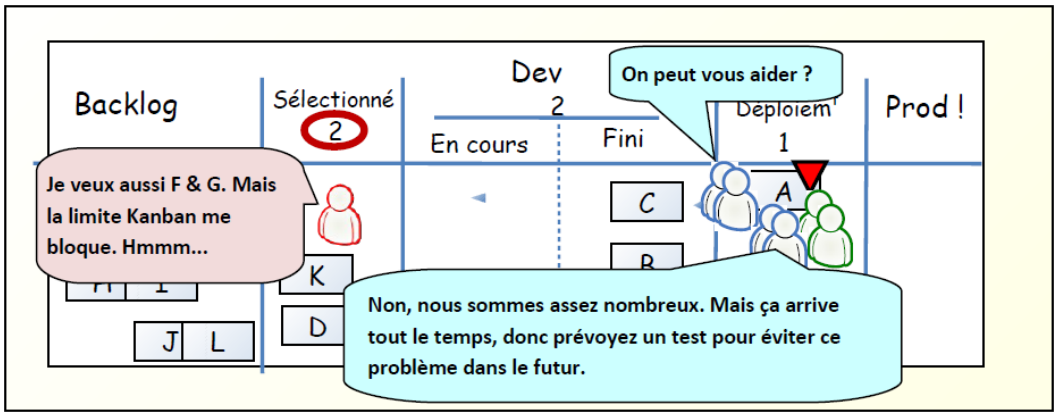

Backlog
Sélectionné 2
On prend C !
Dev 2
En cours
Fini
Déploiem† 1
Prod !
On déploie A !
G
F
H
I
K
J
L
D
C
B
A

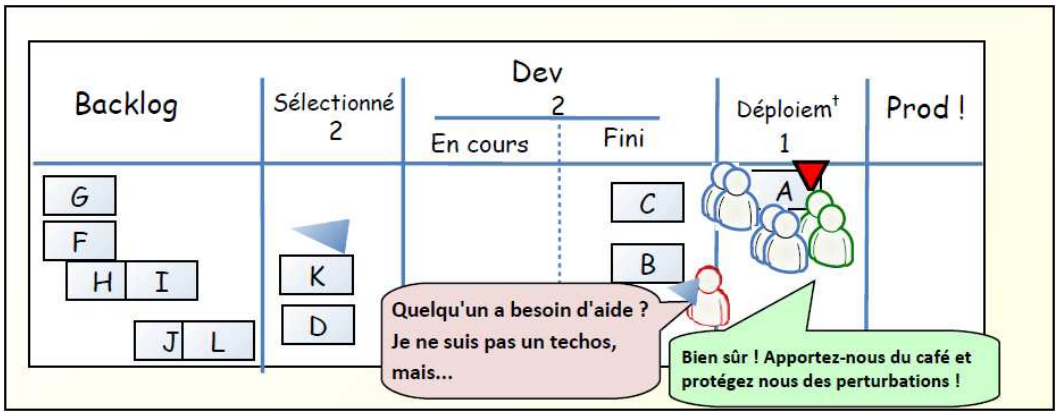

Backlog
Sélectionné 2
Dev 2
En cours
Fini
Déploiem† 1
Prod !
B est terminé !
Mince ! A ne compile pas !
G
F
H
I
K
J
L
D
C
B
A

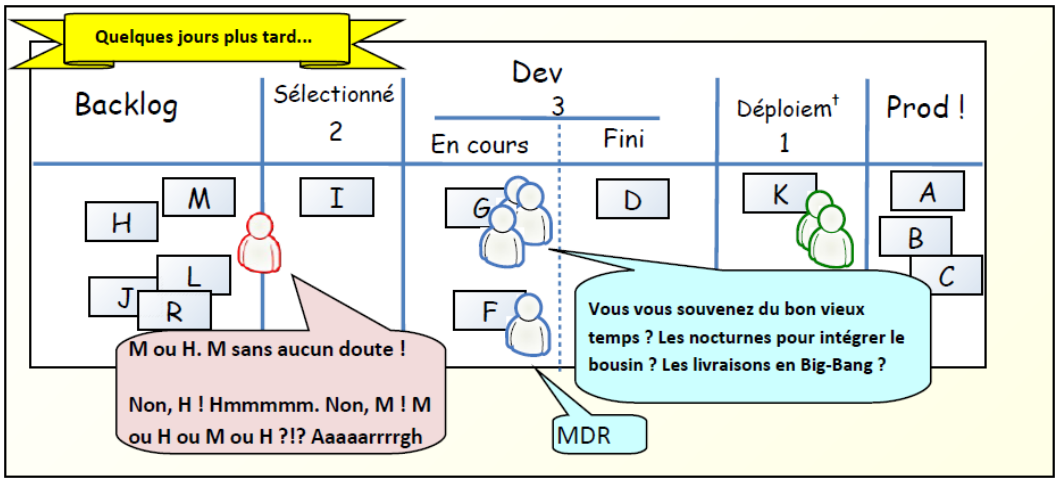

Backlog
Sélectionné 2
Dev 2
En cours
Fini
Déploiem† 1
Prod !
G
F
H
I
K
J
L
D
C
B
A
On prend D !
Non, attendez.... on ne peut pas dépasser la limite Kanban de 2 !


Backlog
Dev 2
En cours
Fini
Déploiem† 1
Prod !
K est assez urgent. Je vais le planifier pour la suite.
G
F
H
I
J
L
K
D
C
B
A
On peut vous aider ?
Apportez nous du café et dites-nous ce que signifie #%&@ dans une log.


Backlog
Sélectionné 2
Dev 2
En cours
Fini
Déploiem† 1
Prod !
C est terminé !
Super, j'espère que vous allez bientôtcommencer K
G
F
H
I
J
L
K
D
C
B
A
Bien sûr, dès que nous aurons régler le problème avec A


Backlog
Sélectionné 2
Dev 2
En cours
Fini
Déploiem† 1
Prod !
On peut vous aider ?
Je veux aussi F & G. Mais la limite Kanban me bloque. Hmmm...
K
D
J
L
C
A
Non, nous sommes assez nombreux. Mais ça arrive tout le temps, donc prévoyez un test pour éviter ce problème dans le futur.


Backlog
Sélectionné 2
Dev 2
En cours
Fini
Déploiem† 1
Prod !
G
F
H
I
J
L
K
D
C
B
A
Quelqu'un a besoin d'aide ? Je ne suis pas un techos, mais...
Bien sûr ! Apportez-nous du café et protégez nous des perturbations !


Quelques jours plus tard...
Backlog
Sélectionné 2
Dev 3
En cours
Fini
Déploiem† 1
Prod !
H
M
I
J
L
R
G
D
F
K
A
B
C
M ou H. M sans aucun doute !
Non, H ! Hmmmmm. Non, M ! M ou H ou M ou H ?!? Aaaaarrrrgh
Vous vous souvenez du bon vieux temps ? Les nocturnes pour intégrer le bousin ? Les livraisons en Big-Bang ?
MDR

Est-ce que le tableau Kanban doit ressembler à ça ?

Non, le tableau ci-dessus était juste un exemple !

Les seules choses que Kanban impose sont le workflow visuel et le TAF limité. Le but est de créer un bon débit à travers le système et de réduire le temps de cycle. Vous devez donc régulièrement soulever des questions telles que :

Quelles colonnes devons nous avoir ?
Chaque colonne représente un état du workflow, ou une file d'attente (buffer) entre deux états du workflow. Commencez simple et ajoutez des colonnes seulement si nécessaire.

Quelles devraient être les limites de Kanban ?
Lorsque la limite de Kanban pour “votre” colonne a été atteinte et que vous n'avez rien à faire, commencez à chercher un goulet d'étranglement en aval (c'est-à-dire des éléments s'empilant à la droite du tableau) et aidez à traiter ce goulet d'étranglement. S'il n'y a pas de goulet d'étranglement, c'est une indication que la limite Kanban pourrait être plus faible, puisque la raison d'avoir une limite est de réduire le risque de générer des goulets d'étranglement en aval.

Si vous remarquez que de nombreux éléments stagnent pendant une longue période sans être traités, c'est une indication que la limite Kanban pourrait être trop élevée.

- Limite Kanban trop faible ⇒ les gens sont inoccupés ⇒ Mauvaise productivité

- Limite Kanban trop élevée ⇒ tâches non traitées ⇒ Mauvais temps de cycle

Les limites Kanban sont-elles strictes ?
Certaines équipes les appréhendent comme des règles strictes (à savoir que l'équipe ne peut pas dépasser une limite), certaines équipes les considèrent comme des lignes directrices ou des motifs de discussion (à savoir que briser une limite Kanban est autorisé, mais cela doit être une décision volontaire avec une raison valable). Encore une fois, c'est vous qui choisissez. Je vous ai dit que Kanban n'était pas très normatif, n'est-ce pas ?

16

Résumé Scrum *vs* Kanban

Ressemblances

- Les deux sont Lean et Agile.
- Les deux utilisent le Juste à temps.
- Les deux limitent le TAF.
- Les deux utilisent la transparence pour piloter l'amélioration des processus.
- Les deux se concentrent sur la livraison rapide et fréquente du produit.
- Les deux sont fondés sur l'auto-organisation des équipes.
- Les deux requièrent de diviser le travail en éléments.
- Dans les deux cas, le planning de release est continuellement ajusté et basé sur des données empiriques (vélocité / temps de cycle).

Différences

Scrum	Kanban
Itérations à durée fixe.	**Itérations à durée fixe optionnelles.** Possibilité d'avoir des rythmes différents pour le planning, les versions et l'amélioration des processus. Peut-être piloté par les événements et non à durée fixe.
L'équipe s'engage sur une quantité spécifique de travail pour une itération.	**Engagement optionnel.**
Utilisation de la **Vélocité** en tant que mesure par défaut pour le planning et l'amélioration des processus.	Utilisation du **Temps de cycle** en tant que mesure par défaut pour le planning et l'amélioration des processus.
Équipes multidisciplinaires imposées.	Équipes multidisciplinaires optionnelles. **Équipes de spécialistes autorisées.**
Les éléments du backlog doivent être découpés afin de pouvoir être traités en 1 sprint.	Aucune taille imposée.
Burndown chart imposé.	Aucun diagramme particulier imposé.
Limitation indirecte du TAF (par sprint).	**Limitation directe du TAF** (par étape du workflow).
Estimation imposée.	**Estimation optionnelle.**
Impossible d'ajouter un item en cours d'itération.	**Possibilité d'ajouter de nouveaux items à chaque fois que la capacité le permet.**
Un backlog de sprint est traité par une seule équipe.	**Un tableau Kanban peut-être partagé par plusieurs équipes** ou individus.

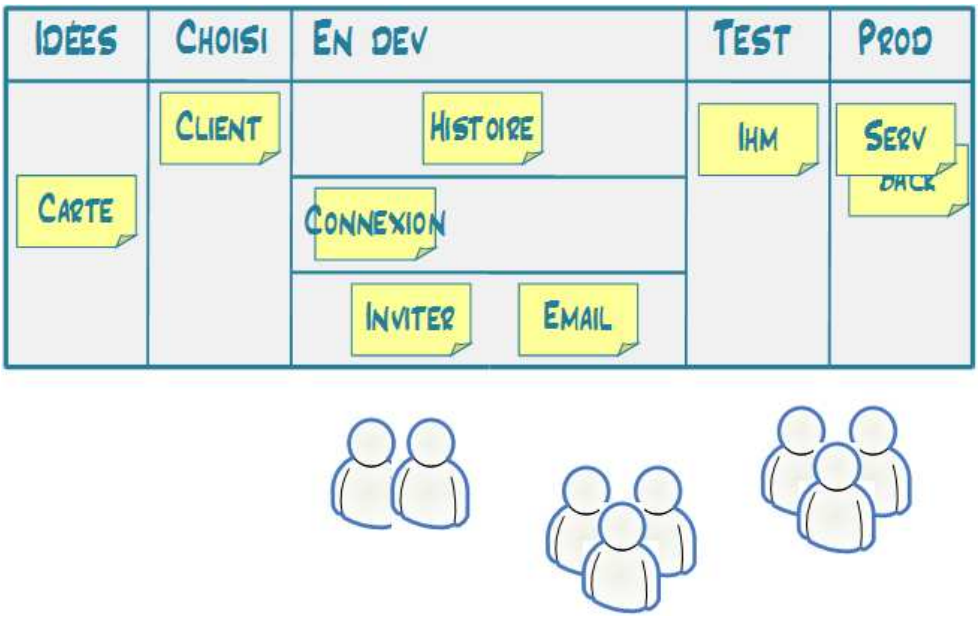
Scrum	Kanban
3 rôles imposés (PO/SM/Équipe).	**Aucun rôle imposé.**
Le tableau Scrum est réinitialisé à chaque début de sprint.	**Un tableau Kanban est persistant.**
Un backlog de produit priorisé imposé.	**Priorisation optionnelle.**

Voilà. On y est. Vous connaissez maintenant les différences.

Mais ce n'est pas encore terminé, la meilleure partie va commencer ! Chaussez vos bottes, il est temps de descendre au fond de la mine avec Mattias et de voir à quoi cela ressemble en pratique !

2ème Partie - Étude de cas

Kanban dans la vraie vie



Je vais raconter ici la façon dont nous avons appris à nous améliorer grâce à l'utilisation de Kanban. Lorsque nous avons commencé, nous n'avions pas beaucoup d'informations et Dr Google, pour une fois, nous a laissé les mains vides. Aujourd'hui, Kanban évolue avec succès et il existe une base émergente de connaissances. Je vous recommande fortement de jeter un coup d'œil sur le travail de David Anderson, par exemple sur les "classes de service". C'est ma première (et ma dernière) annonce (promis !). Quelle que soit la solution que vous déployez, assurez-vous qu'elle réponde à vos problèmes spécifiques. Voilà, c'est fait. Allons-y. Commençons notre histoire.

/Mattias Skarin

17

Le métier d'exploitant

Si vous avez déjà été d'astreinte 24h/24 et 7j/7, vous avez une bonne idée de ce que l'on ressent lorsqu'il faut gérer un environnement de production. On attend de vous d'être capable de régler un problème au milieu de la nuit, que vous soyez ou non à son origine. Personne ne sait quoi faire, c'est pourquoi on vous appelle. C'est un sacré défi puisque ce n'est pas vous qui avez construit le matériel, les drivers, le système d'exploitation ou les logiciels spécifiques. Vos possibilités se limitent le plus souvent à confiner le problème en limitant ses impacts et à conserver les traces nécessaires, en espérant que la personne responsable du problème dont vous avez été le témoin soit capable de le reproduire et de le résoudre.

La réactivité et la capacité à résoudre les problèmes sont fondamentales, ceci avec rapidité et précision.

18

Pourquoi diable changer ?

En 2008, l'un de nos clients, une société scandinave de développement de jeux, est passé par toute une série d'améliorations de ses processus. L'une de ces améliorations concernait le déploiement de Scrum au sein de l'organisation et l'élimination progressive des problèmes freinant l'équipe dans la livraison du logiciel. Lorsque la livraison du logiciel a commencé à devenir plus fluide et que la performance s'est accrue, la pression a commencé à monter en aval du côté du pôle exploitation. Auparavant, les équipes d'exploitation ne se sentaient pas vraiment concernées, aujourd'hui elles sont de plus en plus impliquées et jouent un rôle actif dans le processus de développement.

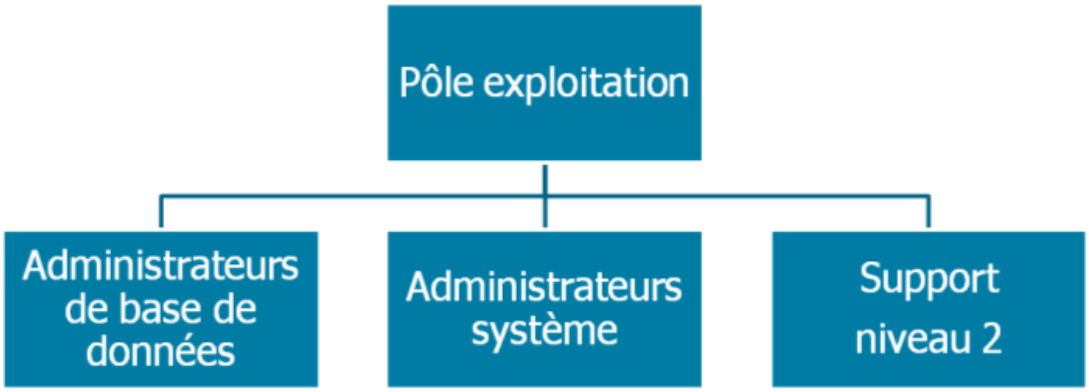


Figure 1. L'organisation du pôle exploitation comprenait trois équipes : des administrateurs de base de données (DBAs), des administrateurs systèmes et un support niveau 2.

Se contenter d'aider les équipes de développement n'était pas suffisant. Si nous avions continué à nous concentrer uniquement sur les équipes de développement, cela aurait pu provoquer des retards dans l'amélioration, indispensable, de l'infrastructure, menée par les équipes d'exploitation. L'amélioration était donc nécessaire dans les deux domaines.

En outre, les progrès réalisés dans les équipes de développement faisaient que les managers étaient de plus en plus sollicités pour aider à analyser les (nouvelles) idées et en donner leur feedback. Ils avaient donc de moins en moins de temps pour la priorisation des tâches au fil de l'eau et la

résolution des problèmes. L'équipe de management a donc réalisé qu'elle devait réagir avant que la situation devienne complètement ingérable.

19

Par où commencer ?

Un bon point de départ a été de questionner les équipes de développement, qui sont les clients du service exploitation.

Vision de l'exploitation par les développeurs

Je leur ai posé la question suivante : “Quelles sont les trois premières choses qui vous viennent à l'esprit quand vous pensez à l'exploitation ?”. Les réponses les plus fréquentes sont :

“Connaissance variable”

“Leur workflow est pourri”

“Très compétent quand il s'agit de l'infrastructure”

“Qu'est-ce qu'ils font ?”

“Ils veulent aider, mais en réalité il est difficile d'obtenir de l'aide”

“Beaucoup trop de mails pour que les choses se fassent”

“Les projets prennent trop de temps”

“Difficile à contacter”

Voilà en bref la vision que les développeurs avaient de l'exploitation. Maintenant, comparons ça à la vision que l'exploitation avait du développement…

Vision du développement par l'exploitation

“Pourquoi n'utilisez-vous pas les avantages de la plate-forme existante ?”

“Faites des versions beaucoup plus simples à livrer !”

“Nous ne comprenons pas que vous produisiez du logiciel de mauvaise qualité !”

“Ils doivent changer” a été le leitmotiv utilisé d'un côté comme de l'autre. Il est évident que l'état d'esprit devait changer pour essayer de régler les problèmes communs. Un point positif : “Très compétent quand il s'agit de l'infrastructure” (qui indique une confiance dans les compétences de base de l'exploitant) m'a laissé penser que l'on pouvait dépasser la mentalité “nous contre eux” si l'on créait les bonnes conditions de travail. Éliminer la surcharge de travail et se concentrer sur la qualité devenait maintenant un scénario viable.

20

Pour s'y mettre

Il fallait donc s'y mettre, mais par où commencer ? La seule chose que nous savions avec certitude était que notre point d'arrivée serait totalement différent de notre point de départ.

Mon parcours est celui d'un développeur donc j'en savais très peu sur le métier d'exploitant. Mon propos n'était pas de “remettre tout en cause et de commencer à tout changer”. Je devais adopter une approche moins conflictuelle qui nous permettrait d'identifier les choses pertinentes, de laisser de côté les choses sans importance et d'apprendre facilement.

Les approches possibles étaient les suivantes :

1. Scrum, qui fonctionne bien avec les équipes de développement.

2. Kanban, nouveau et non mis en pratique, mais en bonne adéquation avec les principes Lean qui nous faisaient défaut.

Lors des discussions avec les managers, Kanban et les principes Lean se sont avérés bien répondre aux problématiques que nous essayions de traiter. De leur point de vue, les sprints ne semblaient pas bien adaptés, car ils avaient l'habitude de redéfinir les priorités quotidiennement. Kanban a donc été logiquement le point de départ, même si c'était un nouveau truc pour nous tous.

21

Mise en ordre de marche des équipes

Comment mettre les équipes en ordre de bataille? Il n'y avait pas de manuel pour cela. S'y prendre mal dès le départ constituait un gros risque. Mis à part le risque de se planter sur les axes d'amélioration, les personnes de la plate-forme d'exploitation à qui nous avions affaire étaient hautement spécialisées et qualifiées, difficiles à remplacer. Se passer d'eux était une trèèès mauvaise idée.

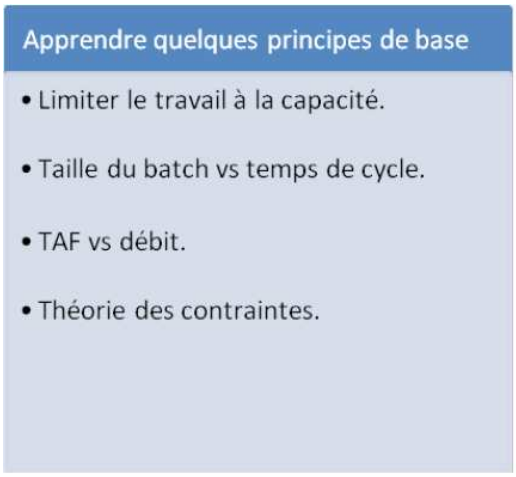
- Devions-nous simplement nous lancer ? ET assumer les conséquences au fur et à mesure ?

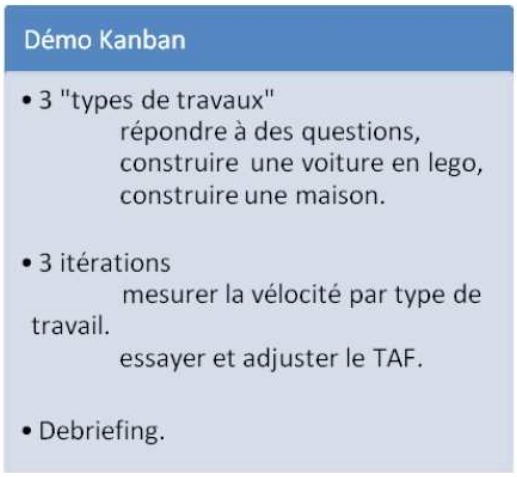
- Ou mener un atelier avant ?

Il était évident pour nous de démarrer par l'atelier, mais comment ? C'était un défi d'obtenir la participation de toute l'équipe d'exploitation à un atelier (qui répond si quelqu'un appelle ?). Finalement, nous avons décidé de faire un atelier d'une demi-journée, simple et basé sur des mises en situation.

L'atelier

Un des avantages de l'atelier était de faire remonter très tôt nos problèmes à la surface. C'était également l'occasion de proposer en toute confiance une séance de travail, pendant laquelle les implications pourraient être discutées directement avec les membres de l'équipe. Soyons clair : tout le monde n'était pas vraiment enthousiaste à l'idée de modifier les méthodes de travail actuelles mais la plupart des membres de l'équipe étaient prêts à essayer. Nous avons donc réalisé un atelier démontrant les principes les plus importants de Kanban avec une simulation à petite échelle.





À la fin de l'atelier, nous avons fait un vote à main levée (sur le principe du “Fist of Five”[2]) pour vérifier si les équipes étaient prêtes à se lancer dans la vie réelle. Aucune objection n'a alors été soulevée et nous avons donc obtenu le feu vert pour aller de l'avant.

[2] NdT : site intéressant détaillant le principe du « Fist of Five » : http://leadinganswers.typepad.com/leading_answers/2007/02/team_decision_m.html

22

Impliquer les parties prenantes

Il était probable que les parties prenantes seraient également impactées par la mise en œuvre de la démarche Kanban. Même si les changements qui allaient avoir lieu étaient faits pour améliorer les choses (l'équipe allait bientôt pouvoir répondre "non" pour un travail qui ne pourrait être terminé, axer son travail autour de la qualité et de la suppression des choses de très faible priorité dans le backlog), il était préférable d'en discuter avant avec elles.

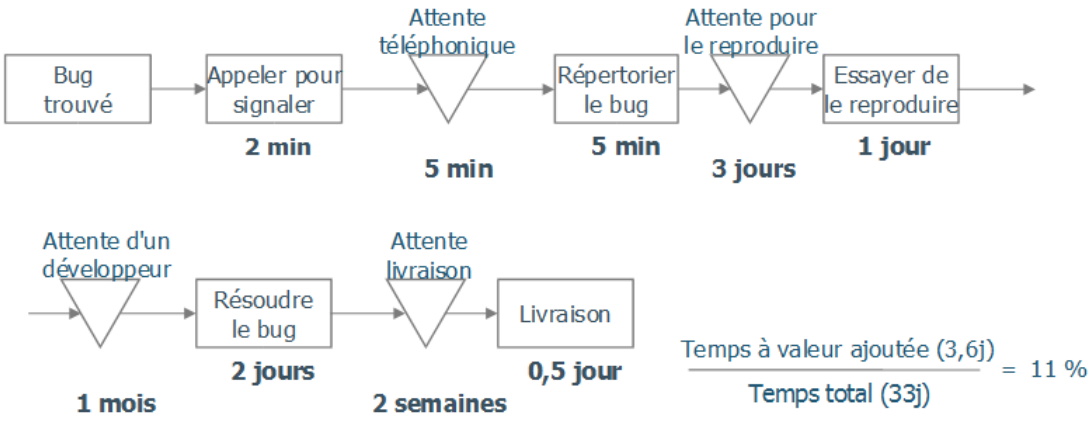
Les parties prenantes les plus proches étaient le support de niveau 1 et les managers du département. Ces derniers ayant participé à l'atelier, ils étaient déjà d'accord pour aller de l'avant. Même chose pour les équipes de développement (qui étaient plus ou moins en attente que les choses s'améliorent d'une manière ou d'une autre). Mais, pour une équipe, celle du support, les choses étaient différentes. Leur problème le plus important était qu'ils étaient surchargés de travail. Ils assuraient notamment le support client et étaient soumis à des engagements pour répondre à tout type de demande. Ces points allaient probablement changer avec la mise en place de Kanban et l'application des limitations du TAF.

Nous sommes donc allés voir les principales parties prenantes pour présenter notre démarche, les bénéfices attendus et les impacts possibles. À mon grand soulagement, nos idées ont été généralement bien reçues, avec parfois une remarque du type “super, nous allons enfin pouvoir enterrer ces problèmes”.

23

Construire le premier tableau

Une bonne façon de commencer la construction d'un tableau Kanban est de cartographier les flux de valeur. Il s'agit de visualiser la chaîne de valeur et de fournir un aperçu de l'état d'avancement des travaux, de l'écoulement du flux et du temps passé dans le système (temps de cycle).



Mais nous avons commencé par quelque chose de beaucoup plus simple ; nous avons dessiné avec le manager un simple tableau Kanban sur le papier. Nous l'avons révisé plusieurs fois de suite puis nous avons démarré avec. Les questions que nous nous sommes posées dans cette phase étaient les suivantes :

- Quels types de travaux réalisons-nous ?
- Qui les gère ?
- Devons-nous partager des responsabilités entre les différents types de travail ?
- Comment pouvons-nous faire face à une responsabilité partagée parmi des compétences spécialisées ?

Étant donné que les différents types de travaux ont des exigences de niveaux de services (SLAs) différents, il a semblé naturel de laisser à chaque équipe le soin de concevoir son propre tableau. Les équipes ont donc élaboré les colonnes et les lignes elles-mêmes.

La grande décision suivante fut de savoir si l'on passait par un partage de responsabilités entre différents types de travaux. “Devons-nous laisser une partie de l'équipe dédiée à la réponse aux questions directes (réactivité) et laisser le reste de l'équipe se concentrer sur les projets (pro-activité)?” Nous avons d'abord essayé le partage des responsabilités. Une des principales raisons était que nous avions identifié que l'auto-organisation et l'augmentation du nombre de personnes dans les équipes étaient indispensables pour soutenir la croissance. L'inconvénient avec ce choix était que chacun pouvait subir d'éventuelles perturbations, mais cela a été la meilleure solution que nous avons pu trouver au départ. Une petite remarque : lorsque nous avons mené l'atelier, les équipes s'étaient déjà organisées d'elles-mêmes pour traiter cette problématique. Une personne gérait les demandes immédiates et le reste de l'équipe traitait les problèmes de fond.

Le premier modèle Kanban

Vous trouverez ci-dessous le modèle de base que nous avons utilisé comme Kanban. Notez que l'équipe a décidé de gérer l'état des éléments du tableau du bas vers le haut (comme des bulles qui remontent à la surface de l'eau) plutôt que d'utiliser le modèle typique où les éléments circulent de gauche à droite.

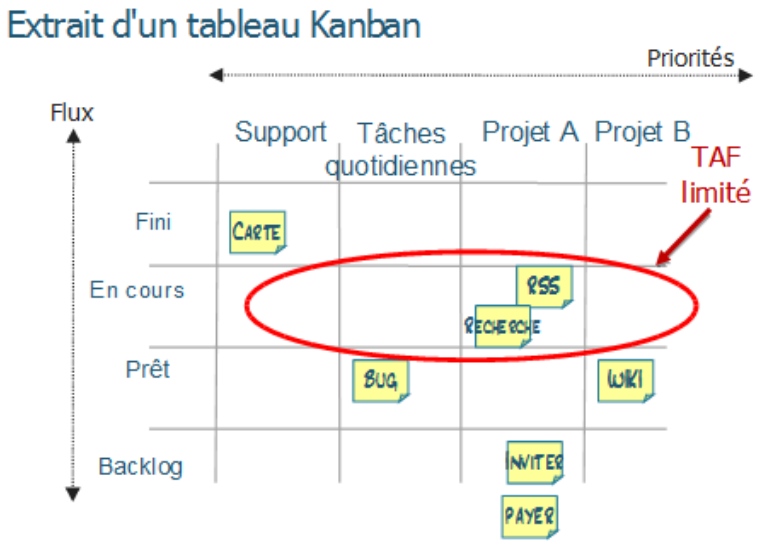


Figure 2. Ceci est notre premier modèle de Kanban. Les priorités vont de gauche à droite et le flux des items du bas vers le haut. Le TAF est comptabilisé par le nombre total de tâches présentes dans la ligne “En cours” (entourée en rouge). Ce modèle a été influencé par les différents retours d'expérience de Linda Cook.

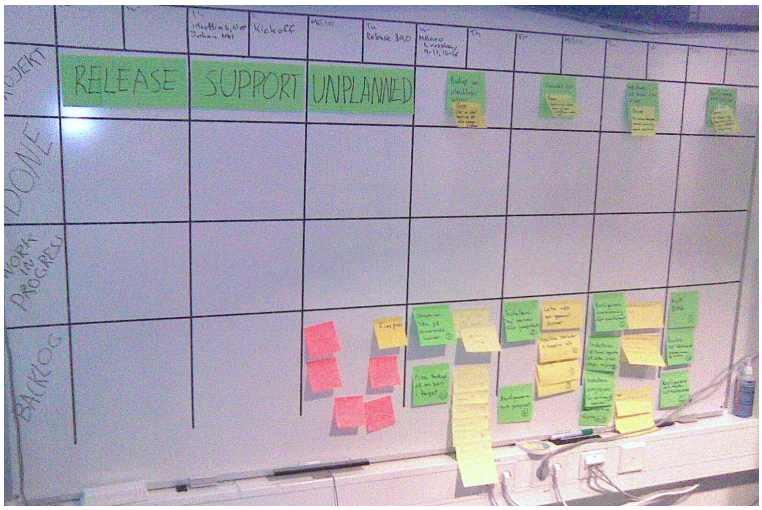


Figure 3. Premier tableau Kanban pour l'équipe administration système.

Lignes utilisées

État du workflow	Notre définition
Backlog	Stories déclarées nécessaires par le manager
Prêt	Les stories sont estimées et découpées en tâches d'une taille maximale de 8 heures
En cours	La ligne disposant d'une limite de TAF. Nous avons démarré en fixant la limite à 2 x la taille de l'équipe - 1 (-1 pour la charge de collaboration). Donc une équipe de 4 personnes a une limite de TAF de 7.
Fini	Exécutable par les utilisateurs.

Colonnes utilisées

Type de travail	Notre définition
Livraison	Assistance des équipes de développement pour les activités de livraison du logiciel.
Support	Petites demandes des autres équipes.
Non planifié	Travail non anticipé nécessitant d'être fait mais sans affectation claire. Par exemple, des améliorations mineures de l'infrastructure.
Projet A	Projet d'exploitation plus conséquent, par exemple, changer le matériel d'un environnement de recette.
Projet B	Un autre projet plus conséquent.

Les tableaux Kanban n'étaient pas tous similaires. Tous ont démarré avec un modèle simple et se sont adaptés au fur et à mesure.

24

Fixer une limite de TAF la première fois

Notre première limite de TAF pour En cours était assez généreuse. Nous pensions qu'en visualisant le flux, nous verrions de façon concrète ce qui se passait et qu'il était peu probable que nous soyons en mesure de deviner la meilleure limite dès le départ. Au fur et à mesure, nous avons ajusté les limites de TAF, à chaque fois que nous avons trouvé de bonnes raisons de le faire (tout ce que nous avions à faire était de l'indiquer sur le tableau).

La première limite de TAF que nous avons utilisée était 2n-1 (n = nombre de membres de l'équipe, -1 pour encourager la coopération). Pourquoi ? C'est simple, parce que nous n'avions pas de meilleure idée ☺. En outre, cette valeur nous semblait incontestable. La formule fournissait une explication simple et logique à toute personne tentant de donner du travail en surcharge à l'équipe : ”… donc étant donné que chaque membre de l'équipe peut travailler sur au plus deux choses à la fois, une chose en cours et une chose en attente, pourquoi voudriez-vous qu'ils en prennent plus ?” Avec le recul, n'importe quelle limite confortable aurait fait l'affaire au démarrage. En observant le tableau Kanban, il est facile de faire ressortir les bonnes limites au fur et à mesure.

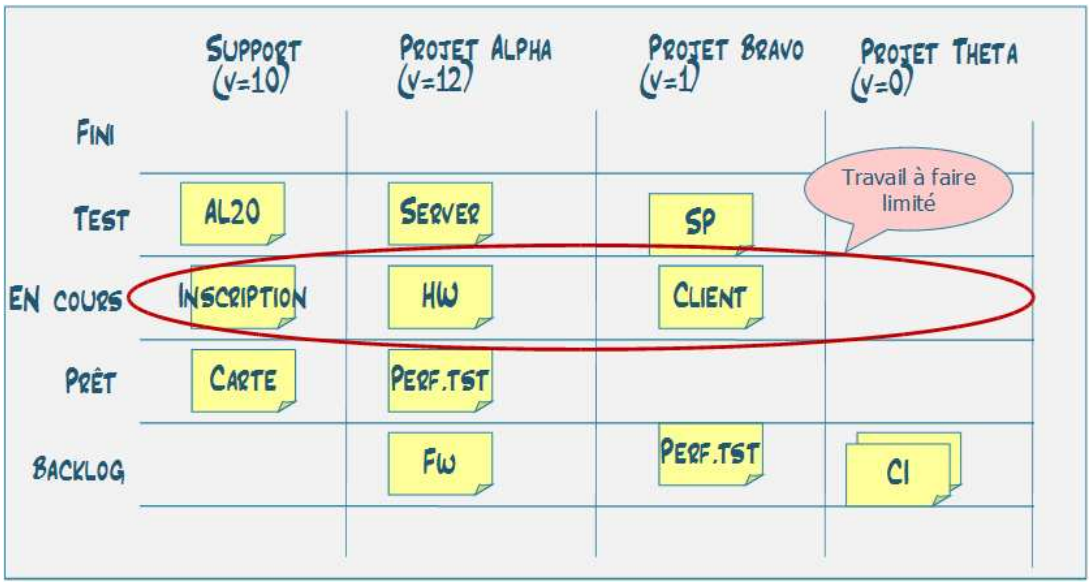


Figure 4. Comment nous avons appliqué la limite de travail en cours pour les DBA et l'équipe d'administration système : une limite partagée pour tous les types de travail.

Nous avons observé qu'il était inutile de définir la limite de TAF en story points. C'était tout simplement trop difficile d'en garder la trace. La seule limite assez facile à suivre était de simplement comptabiliser le nombre d'items (= tâches parallèles).

Pour l'équipe support, nous avons défini un TAF par colonne (donc par type de travail) parce que nous avions besoin d'une réactivité plus grande si la limite venait à être dépassée.

25

Respecter la limite de TAF

Respecter une limite de TAF semble facile en théorie, mais très difficile dans la pratique. Cela signifie pouvoir dire “non” à un certain stade. Nous avons essayé différentes approches pour traiter ce problème.

En discuter devant le tableau

Lorsqu'une limite n'était pas respectée, nous pouvions amener les parties prenantes devant le tableau et leur demander ce qu'ils souhaitaient terminer parmi les travaux en cours. Au début, le cas le plus fréquent de non respect de la limite était juste l'inexpérience. Dans certains cas, nous sommes tombés sur des visions différentes des priorisations, un cas typique étant un spécialiste de l'équipe travaillant dans un domaine spécifique. Ce sont les seules fois où nous avons connu des confrontations, la plupart du temps les problèmes ont été identifiés et discutés devant le tableau.

Créer une colonne Débordement

Comme dire “non” était trop agressif et retirer des items du tableau était trop compliqué, nous déplacions les items de priorité plus faible vers une colonne “débordement” sur le tableau à chaque fois que les limites de TAF étaient dépassées. Deux règles s'appliquaient dans ce cas :

1. Les items concernés n'ont pas été oubliés ; si nous avons du temps, nous les traiterons.

2. Quand nous les supprimons, nous le faisons savoir.

Tout juste deux semaines après, il était évident que les items en débordement ne seraient jamais traités ; donc, avec l'appui du manager de l'équipe, nous les avons finalement supprimés.

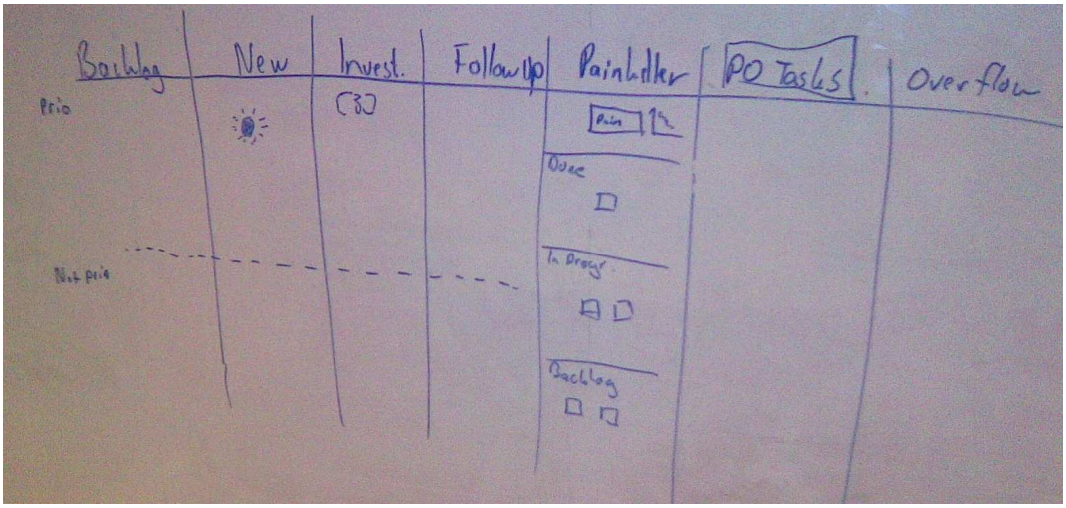


Figure 5. Une esquisse du tableau Kanban pour l'équipe support ; la colonne Débordement (Overflow) est à l'extrême droite.

26

Quelles tâches afficher sur le tableau ?

Nous avons décidé très tôt de ne pas ajouter sur le tableau *tout* le travail effectué par l'équipe. Inclure des tâches comme répondre au téléphone ou prendre un café transformerait le modèle Kanban en un monstre administratif. Nous étions là pour *résoudre* des problèmes et non pour en créer :-). Nous avons donc décidé d'afficher sur le tableau uniquement les tâches de taille > 1 heure, tout ce qui était inférieur étant considéré comme du "bruit". La limite de 1h a effectivement assez bien fonctionné et a été l'une des rares choses qui soit demeurée inchangée. (Nous avons dû revoir nos prévisions quant à l'impact qu'avait réellement le bruit de fond, plus de détails ultérieurement sur ce sujet.)

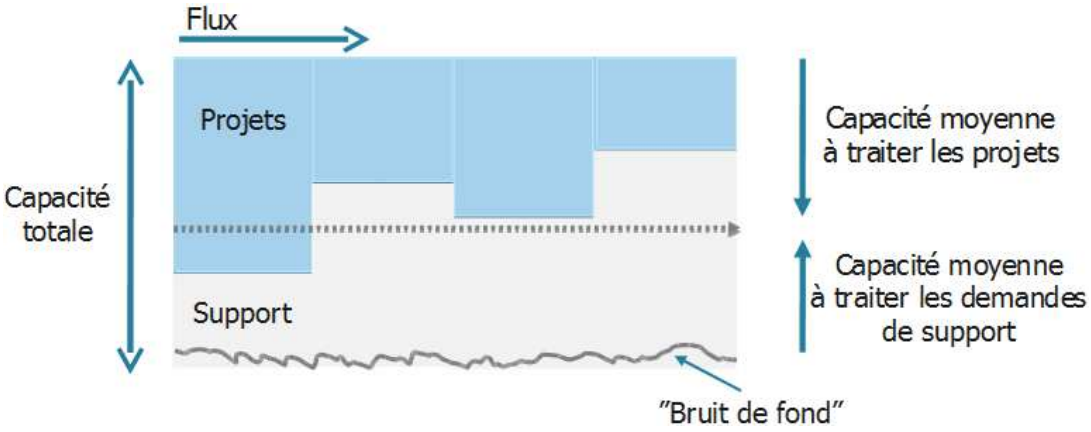


Figure 6. Nous avons démarré en supposant que la capacité totale était principalement consommée par deux types de travaux : les projets dans une grosse proportion et dans une moindre mesure le support. Estimer et suivre la vélocité sur les projets nous a donné une indication de la date de livraison lorsque c'était nécessaire. Le "bruit" (demandes de support < 1h, réunions, prendre un café, aider les collègues) était censé être inclus dans la marge.

27

Comment estimer ?

C'est un sujet toujours ouvert, et il y a certainement plus d'une réponse :

- Estimer régulièrement.
- Estimer quand le besoin s'en fait sentir.
- Faire les estimations en jours idéaux ou en story points.
- Si les estimations sont incertaines, utiliser des tailles de T-shirt (S, M, L, XL).
- Ne pas estimer, ou estimer uniquement lorsque le coût d'un retard le justifie.

Légèrement influencés par Scrum (puisque c'est là d'où nous venions, après tout) nous avons décidé de démarrer les estimations avec des story points. Mais dans la pratique, les équipes ont traité les story points comme si c'était des heures/homme (cela leur semblait plus naturel). Au début, toutes les stories étaient estimées. Avec le temps, les managers ont appris que s'ils conservaient un nombre faible de projets menés de front, ils ne faisaient pas attendre les parties prenantes. Ils ont également appris que dans le cas d'un changement soudain, ils pourraient revoir les priorités et traiter le problème.

Le besoin de prévoir la date de livraison n'était plus un vrai problème. Les managers en vinrent à ne plus demander d'estimations préalables. Ils ne l'ont fait que lorsqu'ils craignaient de faire attendre des personnes.

Au cours des premiers temps, un manager, stressé suite à un appel téléphonique, a promis la livraison d'un projet "avant la fin de cette semaine". Le projet étant sur le tableau Kanban, il s'avéra facile d'évaluer l'état d'avancement (comptabiliser les stories terminées) et de conclure qu'il serait réalisé à environ 25% à la fin de la semaine. Du coup, un délai supplémentaire de trois semaines semblait nécessaire. Face à ce constat, le manager changea la priorité du projet, stoppa les

autres travaux en cours et rendit ainsi la livraison possible dans les délais prévus. Vérifiez toujours le tableau! ☺

Que représente la taille estimée ? Le temps de cycle ou le temps de travail ?

Nos story points reflétaient un temps de travail, c'est à dire le nombre d'heures de travail ininterrompu que cette story prendrait selon notre estimation ; et non pas un temps de cycle (ni le temps calendaire, ni le nombre d'heures d'attente). En mesurant le nombre de story points atteignant la ligne "fini" (donc livrés) chaque semaine (vélocité), on pouvait en déduire le temps de cycle.

Nous avons estimé chaque nouvelle story une seule fois, nous n'avons pas revu les estimations des stories pendant leur traitement. Cela nous a permis de réduire au minimum le temps passé par l'équipe sur les estimations.

28

Alors, comment avons-nous travaillé, concrètement ?

Le système Kanban pose vraiment très peu de contraintes, vous pouvez travailler de toutes sortes de façons. Vous pouvez laisser l'équipe s'appuyer sur un déclenchement planifié des activités ou vous pouvez choisir de commencer une activité lorsqu'elle est suffisamment précise pour la justifier.

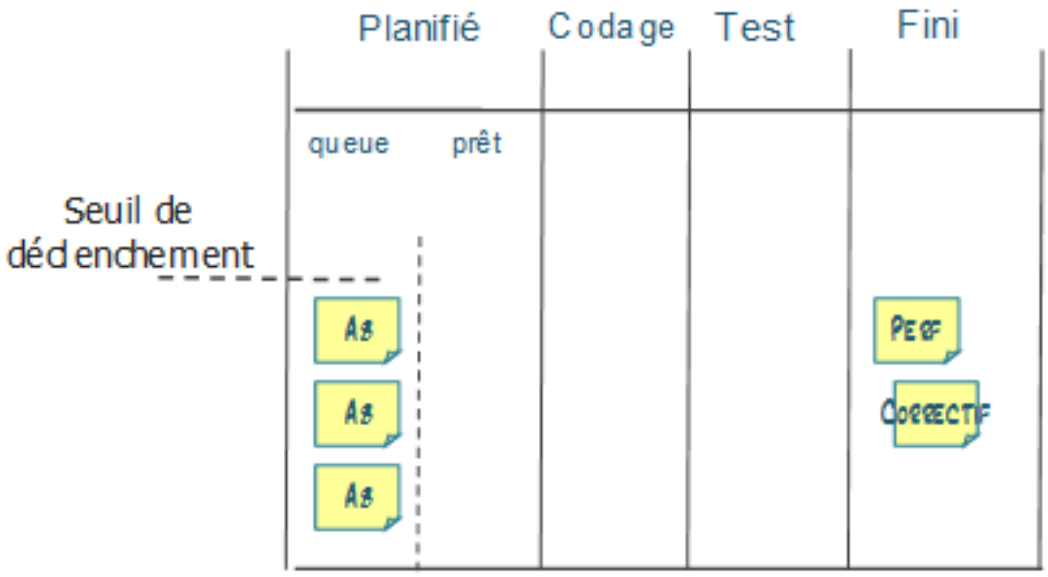
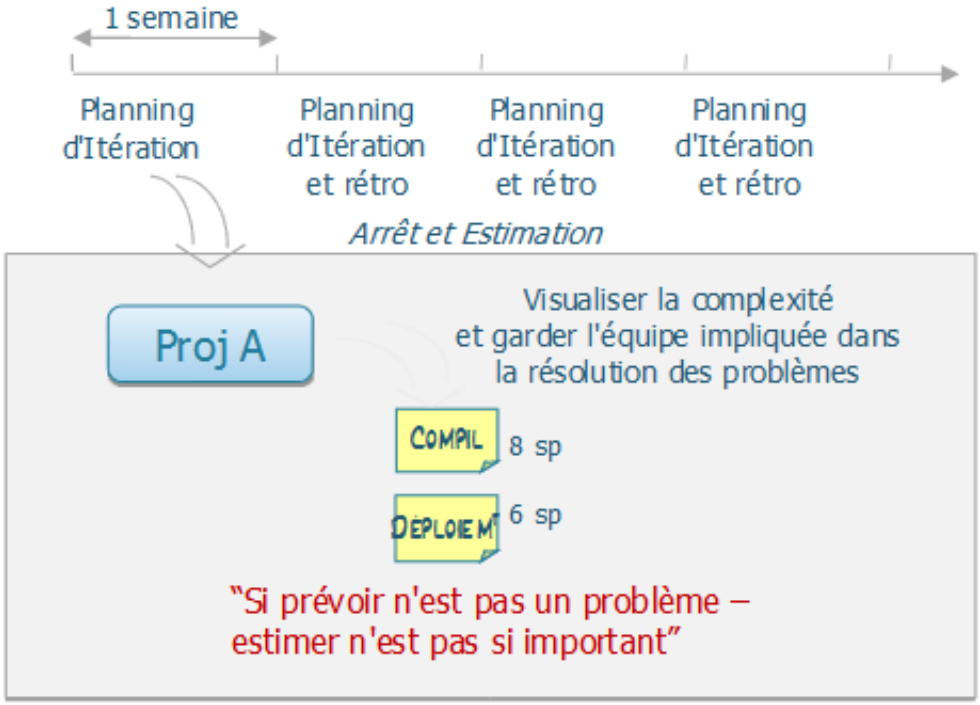


Figure 7. Quand trois tâches sont placées dans le backlog, cela déclenche une session d'estimation/planification.

Nous avons décidé de programmer deux évènements récurrents :

- Le point quotidien - avec toute l'équipe, devant le tableau, pour identifier les problèmes et aider à créer une “vision transversale partagée” des tâches des membres des autres équipes.

- Le planning hebdomadaire d'itération, avec pour objectifs la planification et l'amélioration continue.



Cela a bien fonctionné pour nous.

Le point quotidien

Le point quotidien était similaire à celui pratiqué en Scrum. Cela se déroulait après la réunion quotidienne de "Scrum de Scrums" qui impliquait toutes les équipes (développement, tests, exploitation). Le Scrum de Scrums fournissait des entrées importantes aux équipes Kanban, telles que l'identification des problèmes à traiter en premier ou l'équipe de développement la plus en difficulté à ce moment là. Au début, les managers assistaient régulièrement à ces points quotidiens, proposant des solutions et prenant des décisions vis-à-vis des priorités. Au fil du temps, comme les équipes se sont améliorées en terme d'auto-organisation, les managers ont espacé leur présence (mais n'étaient jamais très loin en cas de besoin).

La planification d'itération

Une fois par semaine, nous organisions une réunion pour planifier l'itération. Nous avons conservé cette fréquence d'une semaine car nous avons constaté que si nous ne prenions pas ce temps pour planifier, il était consommé par d'autres priorités ☺. Et puis nous avions besoin de plus de discussions d'équipe. L'ordre du jour typique était le suivant :

- Mettre à jour les diagrammes et le tableau (les projets terminés étaient déplacés vers le "Mur des Terminés").

- Bilan de la semaine écoulée. Que s'est-il passé ? Pourquoi cela s'est passé ainsi ? Que pourrions nous faire pour améliorer les choses ?

- Ajustement de la limite de TAF (si nécessaire).

- Répartition des tâches et estimation des nouveaux projets (si besoin était).

En gros, la réunion de planification d'itération était une combinaison d'estimations et d'amélioration continue. Les problèmes de petite ou de moyenne taille étaient résolus sur le champ avec l'appui des managers de premier niveau. Mais garder le rythme sur des problèmes complexes d'infrastructure était une épreuve plus ardue. Pour gérer ça, nous avons introduit la possibilité pour chaque équipe de remonter 2 “obstacles pour l'équipe” à son manager.

Les règles étaient les suivantes :

1. Un manager peut travailler sur 2 obstacles en parallèle.

2. Si les 2 places sont prises, on peut ajouter un nouvel obstacle à condition de retirer celui d'importance la plus faible.

3. C'est l'équipe qui décide quand un obstacle est éliminé.

Ce fut un changement positif. Tout à coup, les équipes pouvaient voir que leurs managers travaillaient pour les aider, même sur des problèmes complexes. Ils pouvaient pointer le tableau des obstacles et demander “comment ça se passe ?” Ils ne pouvaient plus être oubliés ou “doublés” par une nouvelle stratégie plus prioritaire.

Un exemple d'obstacle sérieux : l'équipe exploitation n'arrivait pas à obtenir l'aide dont elle avait besoin de la part des développeurs quand ils suspectaient un bug. Ils avaient besoin d'aide pour identifier quelle partie du système était responsable du problème ; mais comme les développeurs

étaient pris dans leur sprint à développer de nouvelles choses, les problèmes continuaient à s'empiler. Ce n'est donc pas étonnant que les exploitants aient eu l'impression que les développeurs ne s'occupaient pas assez de la qualité.

Quand cet obstacle a été mis en évidence, il a été remonté, d'abord au manager de premier niveau, puis plus tard au manager de département. Celui-ci a alors programmé une réunion avec le responsable des développements. Dans les débats qui suivirent, les managers décidèrent de mettre en priorité première la qualité, et ils ont imaginé une solution de support tournant, des développeurs vers les exploitants - chaque sprint, une équipe de développement serait “joignable” et se rendrait instantanément disponible pour assister l'exploitation. Après avoir obtenu le soutien de ses managers, le responsable des développements a fourni une liste de contacts aux équipes de support. Ils ont testé immédiatement la solution, suspectant qu'elle ne fonctionnerait pas. Mais les devoirs avaient été bien faits cette fois et le problème a été considéré comme résolu. Cela a été d'un grand secours pour les équipes d'exploitation.

29

Trouver un concept de planification qui fonctionne

Une histoire

Je me souviens d'une histoire qui a influencé le comportement de l'une des équipes. J'étais assis avec eux lors de leur deuxième session d'estimation. L'équipe était bloquée sur un projet qu'ils ne savaient pas comment estimer. Il y avait trop d'inconnues et la session d'estimation ralentit jusqu'à s'arrêter. Plutôt que de rentrer dans le jeu et prendre les choses en main, je leur ai demandé de retravailler leur processus pour trouver une meilleure solution. Conduits par leur manager, ils ont relevé le défi et ont commencé à concevoir leur propre solution. Cet événement a vraiment constitué un tournant, une "victoire" importante à partir de laquelle ils ont vraiment constitué une équipe en confiance. À partir de ce moment là, ils ont évolué si rapidement qu'il a presque fallu que nous nous écartions de leur chemin !

Deux mois plus tard, leur manager s'est approché de moi après une rétrospective : "J'ai un problème" m'a-t-il dit en pointant le tableau de Kanban. "Nous n'avons pas de vrais problèmes, que devons-nous faire ?"

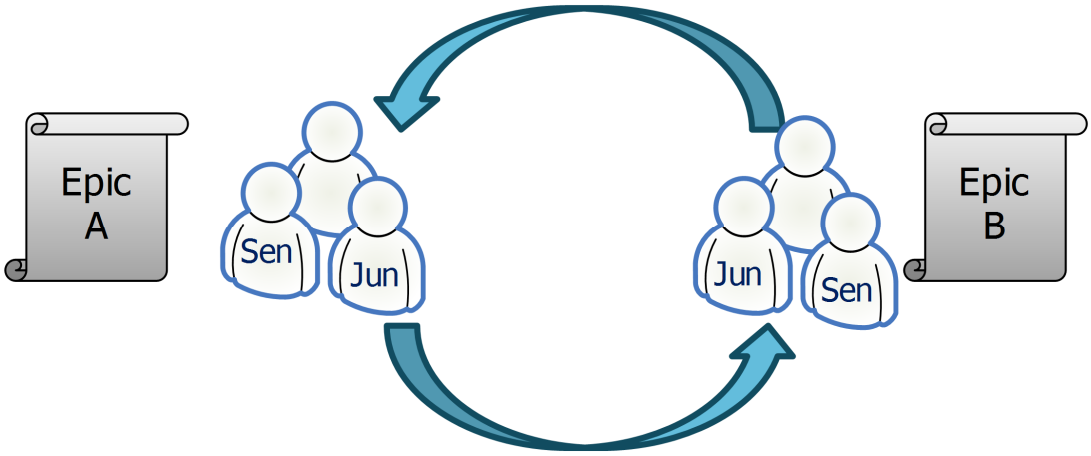
Réinventer la planification

Les sessions d'estimation par planning poker impliquant l'ensemble des membres d'une équipe n'ont marché pour aucune des équipes d'exploitation. Quelques explications :

1. La connaissance était trop inégalement répartie dans l'équipe.
2. La plupart du temps, une seule personne prenait la parole.
3. Les membres de l'équipe avaient l'esprit aux problèmes urgents à traiter à leur bureau.

Mais, par l'expérimentation, les équipes ont construit indépendamment deux processus d'estimation différents. Chacun d'entre eux a bien fonctionné pour les équipes respectives.

Approche 1 - Echanger et revoir



- Pour chaque projet/story, désigner une paire senior + junior pour estimer (c'est à dire une personne qui connait particulièrement bien la story et une autre qui ne la connait pas bien). Cela aide à diffuser la connaissance.

- Les autres membres de l'équipe choisissent quelle story ils veulent aider à estimer (mais avec une limite de quatre personnes par story pour conserver une discussion efficace).

- Chaque équipe d'estimation découpe sa story en tâches et, si nécessaire, l'estime.

- Puis, les équipes échangent les stories et revoient le travail des autres (une personne par équipe était “détachée” pour expliquer le travail de son équipe aux “relecteurs”).

- Terminé !

En général, la session d'estimation durait environ 45 minutes et le niveau de concentration restait élevé tout au long de la réunion. 1 à 2 ajustements étaient généralement effectués lors de la rotation et revus par un ensemble de regards nouveaux.

Approche 2 - Revue de haut niveau puis estimation

Deux membres expérimentés de l'équipe mènent une revue de haut niveau sur la story / le projet avant la planification. Ils analysent les solutions d'architecture et font un choix. Une fois lancée, l'équipe découpe la story en tâches en utilisant la solution proposée comme point de départ.

Figure 8. Découpage en tâches avec revue par une autre équipe lors de la planification d'itération.

30

Quoi mesurer ?

De nombreux points pourraient être mesurés : le temps de cycle (temps écoulé entre le moment où le besoin est découvert et celui où il est satisfait), la vélocité, les files d'attente, les burndowns... La question essentielle est de savoir quels indicateurs peuvent être *utilisés* pour améliorer le processus. Mon conseil est d'expérimenter et de trouver ce qui marche pour vous. Nous avons appris que les burndown charts étaient surdimensionnés pour les petits projets de 1 à 4 semaines. La progression peut être examinée en regardant le tableau Kanban (nombre de stories qui sont dans le backlog et nombre de stories qui sont terminées).

Indicateur proposé	**Pour**	**Contre**
Temps de cycle	Facile à mesurer. Pas d'estimation requise. Démarre et finit avec le client.	Ne tient pas compte de la taille.
Vélocité totale (agrège les types de travaux)	Un indicateur brut mais facile pour mesurer l'amélioration et la variation.	N'aide pas à prévoir les dates de livraison pour des types de travaux spécifiques.
Vélocité par type de travail	Plus précis que la vélocité totale.	Pour être utile, nécessite d'être mesurée à partir du besoin utilisateur jusqu'à sa livraison. Prend plus de temps pour suivre l'évolution que la vélocité totale.
Longueur des files d'attente	Un indicateur rapide pour mesurer la fluctuation des demandes. Facile à visualiser.	Cela ne vous dit pas si la cause est liée à une demande inégale ou une capacité inégale. Une file d'attente vide peut en réalité indiquer une surcapacité.

Nous avons commencé par mesurer la “Vélocité par type de travail” et la “Longueur des files d'attente”. La vélocité par type de travail est facile à mesurer et est suffisante. La longueur des files d'attente est un indicateur de premier plan, car il peut être repéré instantanément (une fois que vous savez où regarder).

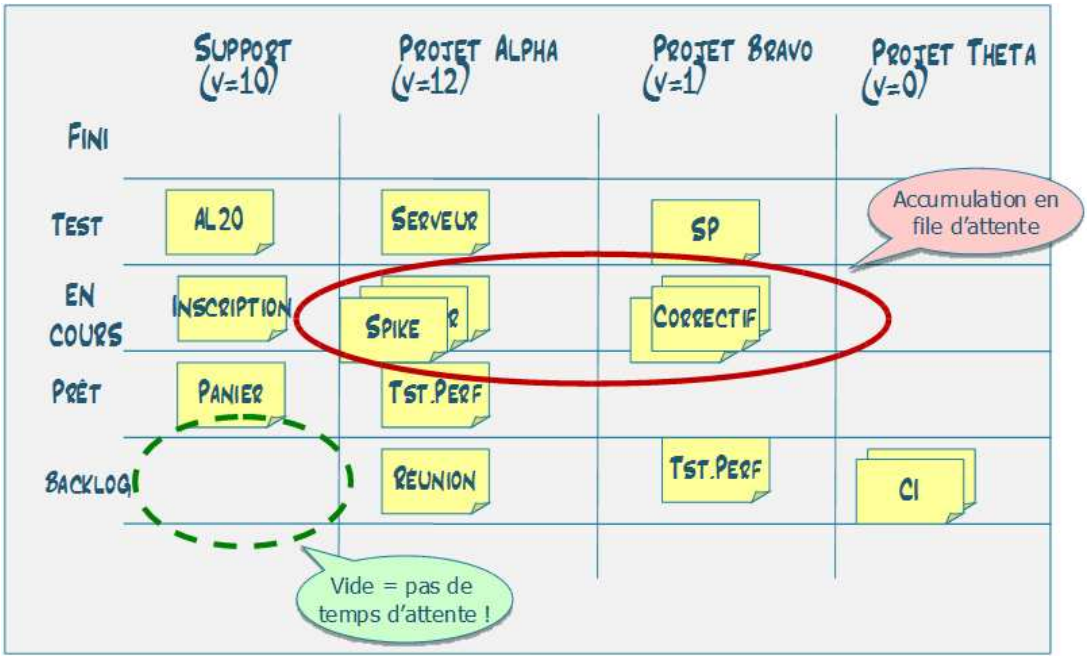
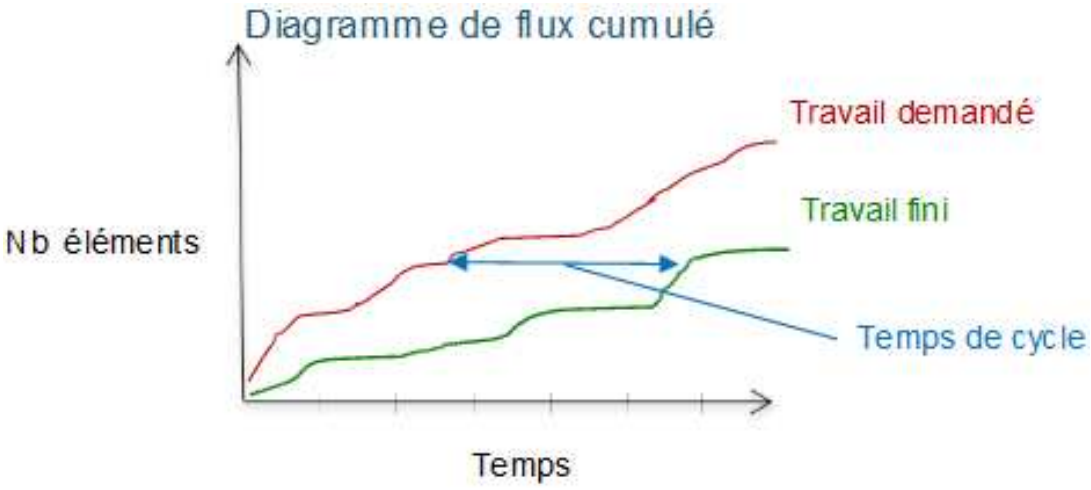


Figure 9. Goulets d'étranglement et opportunités. La zone rouge montre comment les files d'attente se sont accumulées pour faire apparaître un goulet d'étranglement sur le test. L'absence de file d'attente dans la ligne “Backlog” du support indique qu'il n'y a pas de temps d'attente pour une nouvelle demande de support. C'est bon signe pour un niveau de service élevé.

Nous n'avons pas utilisé de diagramme de flux cumulé, mais cela aurait été intéressant.



Nous n'avons pas utilisé les diagrammes de flux cumulés puisque le tableau Kanban et le graphe de vélocité nous ont donné suffisamment d'informations, tout au moins dans nos premières phases de montée en compétence. Les goulets d'étranglement, l'inégalité du flux et la surcharge de travail ont pu être facilement identifiés et la résolution de ces choses-là nous a occupés pendant les six premiers mois.

31

Comment les choses ont commencé à changer

Trois mois après l'introduction de Kanban, l'équipe d'administration système a été nommée “équipe la plus performante” du département informatique par le management. Dans le même temps, cette équipe a également été élue comme l'une des trois “expériences les plus positives” au cours de la rétrospective de la société, une manifestation à l'échelle de l'entreprise qui a lieu toutes les six semaines. C'était la première fois qu'une *équipe* se plaçait au top 3 ! Ceci alors qu'il y encore 3 mois, cette équipe était reconnue comme un goulet d'étranglement dont la plupart des gens se plaignaient.

La qualité de service s'est clairement améliorée. Comment est-ce donc arrivé ?

Le moment essentiel a été quand tout le monde a commencé à bouger ensemble dans le même sens. Les managers ont donné une orientation claire et ont protégé l'équipe du boulot qui ne la concernait pas, et les équipes se sont engagées sur la qualité et les délais. Il a fallu environ trois à quatre mois pour en arriver là, mais après tout s'est passé en douceur. Ce n'est pas comme si tous les problèmes avaient disparu dans le monde (si c'était le cas, on serait tous au chômage n'est-ce pas ? ☺) - mais nous avons dû relever de nouveaux défis tels que “comment garder une équipe motivée pour s'améliorer (quand elle n'est plus le goulet d'étranglement) ?”

Une pièce importante du puzzle de l'auto-organisation a été l'introduction du principe “un contact de l'exploitation par équipe”. Cela signifie que chaque équipe de développement se voyait affecter leur propre interlocuteur au support exploitation. Kanban a rendu cela possible en permettant aux membres de l'équipe d'exploitation de s'auto-organiser autour du travail, de prévenir la surcharge de travail et de faciliter l'amélioration continue. Avant, une personne au hasard dépilait un problème de la file d'attente, essayait de le résoudre au mieux de ses compétences et enchaînait sur le problème suivant. Tout défaut de communication signifiait qu'il fallait tout recommencer avec une nouvelle

demande de support. Lorsque le principe “un-pour-un” a été mis en place, l'équipe support eut tout à coup l'opportunité de répondre rapidement lorsque des données erronées et des problèmes de qualité menaçaient le système.

Pour anecdote, après un certain temps, les procédures de communication ont évolué vers des méthodes plus personnalisées : les membres de l'exploitation ont commencé à utiliser la messagerie instantanée avec les développeurs qu'ils connaissaient bien, le courrier électronique pour ceux qui écrivaient mieux qu'ils parlaient et le téléphone si c'était le moyen le plus rapide pour résoudre le problème ☺

Avant

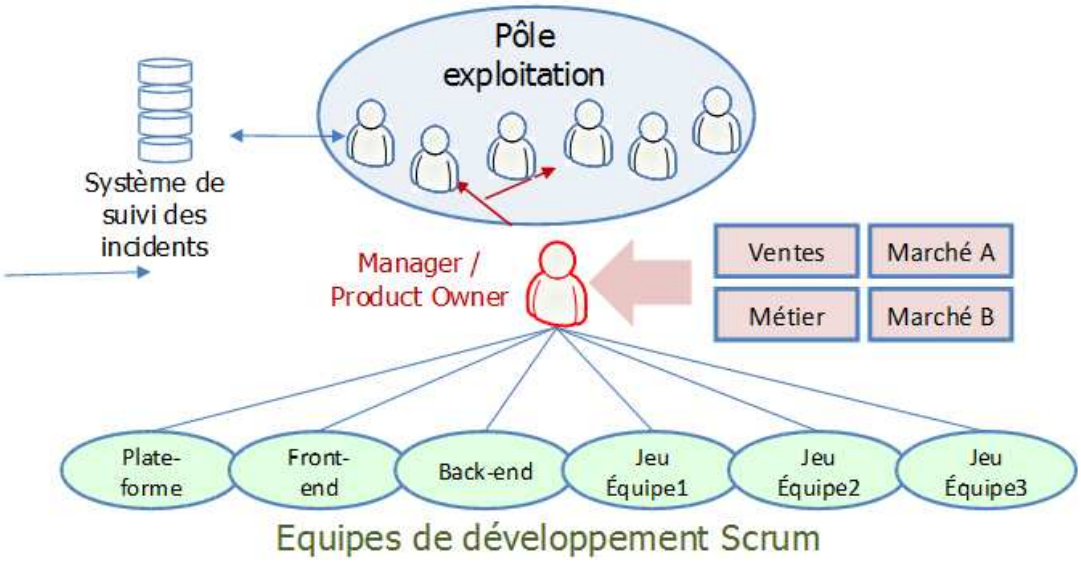


Figure 10. Avant : le manager en première ligne est le point de contact principal pour l'équipe. Tout ce qui doit se faire et qui est important passe par lui. Les problèmes de moindre importance, typiquement ceux des développeurs, sont reçus via le système de suivi des problèmes. Très peu d'interactions directes entre les personnes.

Après

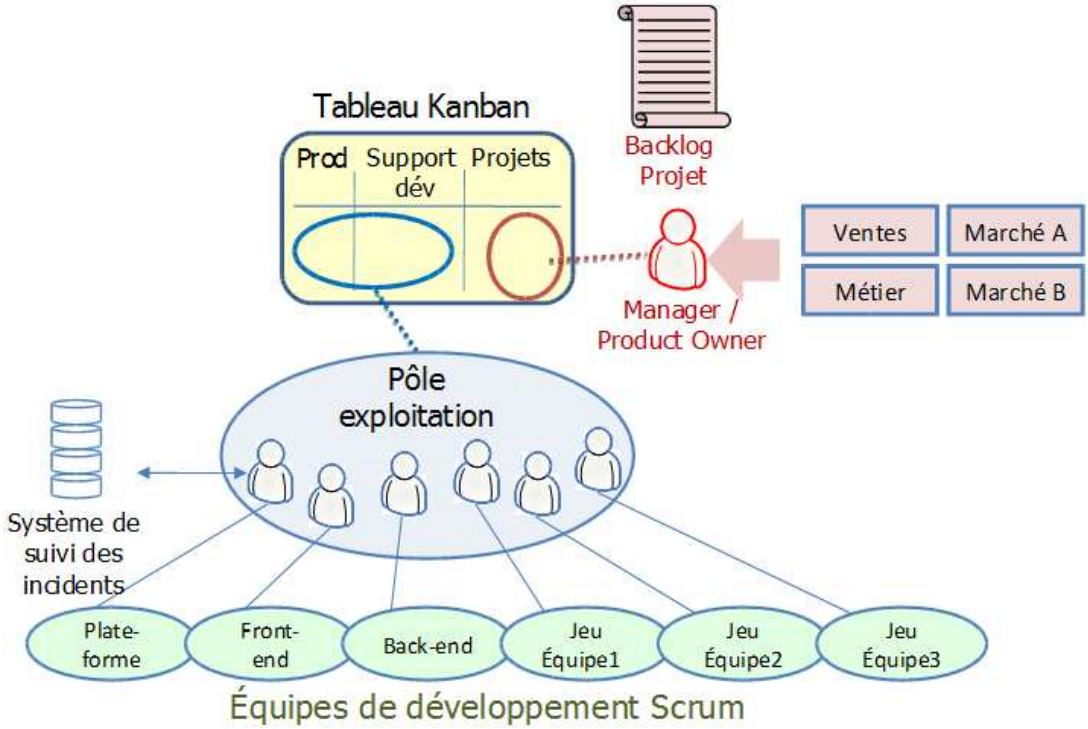


Figure 11. Après : “un contact de l'exploitation par équipe”. Les équipes de développement parlent directement à une personne définie à l'exploitation. Beaucoup d'interactions entre les personnes. Les membres de l'équipe d'exploitation organisent eux-mêmes leur travail en utilisant le tableau Kanban. Le manager se concentre sur la priorisation des gros projets et sur le renfort de l'équipe lorsque des problèmes difficiles se posent.

Quels sont alors les impacts sur la performance de l'équipe ?

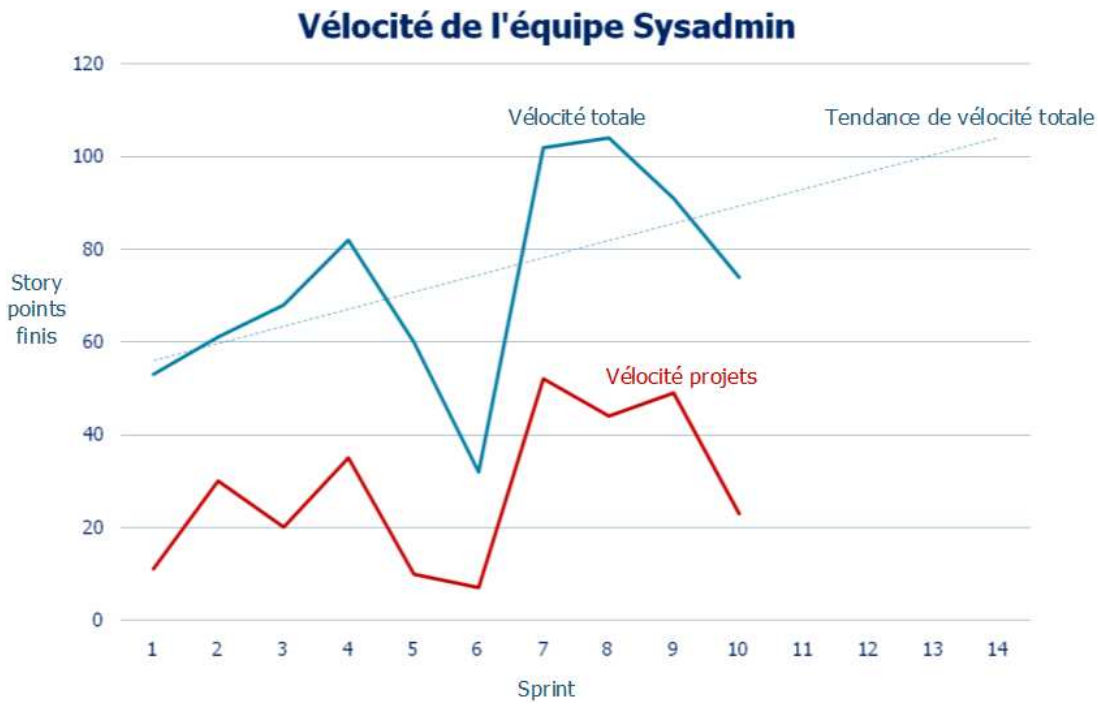


Figure 12. La vélocité totale et la vélocité projets, mesurées en story points "finis" par semaine. La vélocité totale est la somme de toutes les colonnes, la vélocité projets représente la partie consacrée aux "projets" (gros travaux, par exemple la mise à niveau d'une plate-forme matérielle). Les deux creux correspondent à 1) une semaine où presque tous les membres de l'équipe étaient en voyage et 2) une version majeure de l'équipe de développement.

Ainsi, l'équipe a affiché une tendance globalement positive. Dans le même temps l'équipe a lourdement investi dans le partage des connaissances en utilisant la programmation en binôme.

Jetons un coup d'œil à la performance de l'équipe des DBAs :

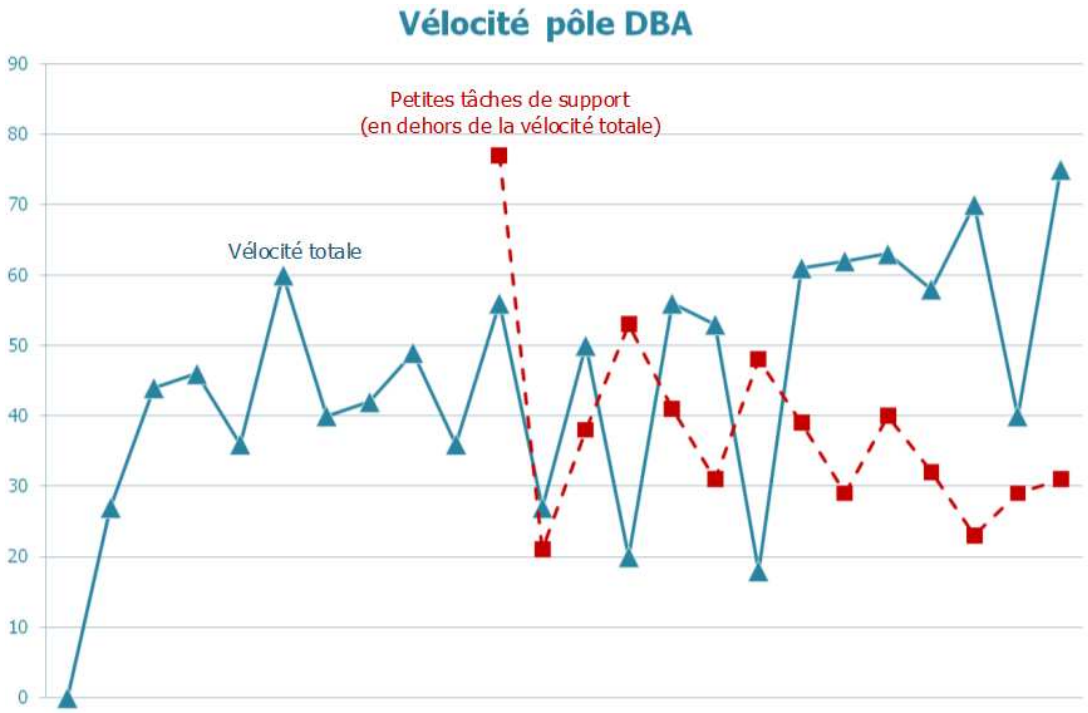


Figure 13. Vélocité totale et petites activités de support. Le creux du milieu correspond à la période de Noël.

La vélocité totale a une tendance à la hausse même si la variance est significative. Ce qui a conduit l'équipe à surveiller le nombre de petites tâches de support (tâches normalement trop petites pour être affichées sur le tableau Kanban). Comme vous le pouvez le constater, le graphique montre une corrélation inverse entre le nombre de petites tâches de support et la vélocité totale.

L'équipe de support ayant commencé à appliquer Kanban plus tard que les deux autres équipes, nous n'avons pas encore de données fiables pour le moment.

Mûrissement

Lorsque nous avons commencé, trouver les zones de problèmes a été facile. Mais trouver les plus grandes opportunités d'amélioration a été dur. Le tableau Kanban nous a apporté un tout nouveau niveau de transparence. Non seulement il était plus facile de repérer les problèmes, mais d'importantes questions ont été soulevées concernant le flux de travail, la variance et les files d'attente. Nous avons commencé à utiliser les files d'attente comme un outil pour isoler les problèmes. Quatre mois après avoir commencé à appliquer Kanban, les gestionnaires partaient à la chasse aux générateurs de variance qui perturbaient leurs équipes.

Comme les équipes évoluaient d'un niveau individuel vers un niveau auto-organisé, les managers ont réalisé qu'ils étaient confrontés à de nouveaux défis en termes de leadership. Ils devaient davantage s'occuper des problèmes des gens - traiter les réclamations, définir des objectifs partagés, résoudre les conflits et négocier des accords. Ce ne fut pas une transition sans douleur - ils ont ouvertement fait remarquer que l'apprentissage exigeait compétences et énergie. Mais ils ont relevé le défi et ont fini par devenir de meilleurs leaders.

32

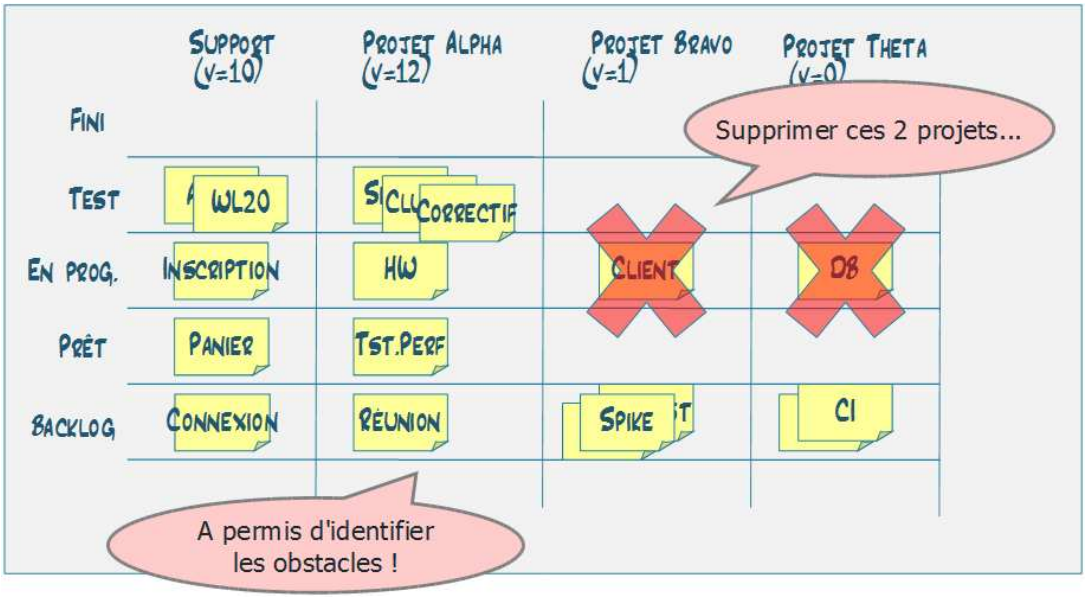
Leçons apprises

Des contraintes émergent lorsque le travail à faire diminue

Toutes les équipes ont commencé avec des limites de TAF assez confortables. À cette époque, les efforts ont été employés pour créer un flux de travail et pour s'assurer que l'organisation obtenait le soutien dont elle avait besoin.

Au début, les managers voulaient avoir de nombreux projets en parallèle, mais après quelques semaines il devint évident qu'il n'y avait pas la capacité suffisante pour traiter les projets de plus faible priorité. Il a suffi de jeter un rapide coup d'œil au tableau pour constater qu'aucun travail n'était démarré sur les projets de faible priorité. Cela a incité les managers à réduire le nombre de projets par équipe.

Au fil du temps, le flux devenant de plus en plus stable pour les travaux prioritaires, nous avons commencé à resserrer les limites de TAF. Cela a été réalisé en réduisant le nombre de projets en cours (colonnes) de trois à deux, puis à un. À ce moment là, des contraintes extérieures à l'équipe commencèrent à apparaître. Certains membres d'équipes signalèrent qu'ils ne recevaient pas d'aide dans les temps, si bien que les managers commencèrent à y prêter attention.



Une autre chose qui est remontée à la surface a été à quel point la mauvaise qualité des livrables produits par d'autres équipes pouvait dégrader les performances. Il a été difficile de maintenir un flux de travail régulier et rapide lorsque les éléments en entrée avaient en permanence besoin d'être corrigés.

Ces problèmes étaient connus avant que nous commencions. La question était alors de savoir “quel problème traiter en premier” et ceci avec l'accord de tout le monde. À l'aide du tableau Kanban, tout le monde a pu se rendre compte du problème spécifique qui impactait le *flux*, ce qui a permis de bénéficier de l'effort commun pour traiter la question globalement quelles que soient les différentes entités en présence.

Le tableau va changer en cours de route, ne figez pas sa conception

Tous les tableaux Kanban changent en cours de route. La conception du tableau est généralement revue deux ou trois fois avant qu'une équipe le considère fonctionnel. Donc, investir beaucoup de temps dans la première version du tableau est sans doute inutile. Assurez-vous que vous pouvez réorganiser le tableau facilement. Nous avons utilisé des bandes de ruban noir pour les bordures du tableau. C'était facile à réorganiser et utilisable aussi bien sur les murs que sur les tableaux blancs. Un autre moyen possible est de dessiner le tableau avec des marqueurs épais (mais assurez-vous qu'ils soient effaçables ! ☺)

Vous trouverez ci-dessous un exemple typique d'une optimisation de la conception d'un tableau. L'ordre des priorités changeaient souvent au début, donc afin d'éviter d'avoir à déplacer des colonnes entières de post-

its, l'équipe a choisi d'afficher un numéro de priorité au dessus de chaque colonne.

Figure 14. Tableau Kanban avec des post-its pour voir les priorités

N'ayez pas peur d'expérimenter et d'échouer

La leçon que je tire de cette aventure est que c'est sans fin. Nous n'avons pas réussi à voir la fin. Il y a uniquement un mode d'expérimentation et d'apprentissage qui ne s'arrête jamais. Ne jamais échouer signifie ne jamais apprendre. Nous avons échoué à plusieurs reprises le long de la route (mauvaise conception des tableaux, mauvaises estimations, burndown redondants, ...) mais à chaque fois nous avons appris quelque chose de nouveau et d'important. Si nous avions cessé d'essayer, comment aurions-nous pu apprendre ?

Le succès de Kanban a motivé les équipes de management et les équipes de développement Scrum à expérimenter les tableaux Kanban. Peut-être que ce livre vous y aidera !

Points à retenir

Commencez par des rétrospectives !

Beaucoup de choses auxquelles il faut penser hein ? Espérons que ce livre ait permis d'éclaircir ce qui était nébuleux. En tout cas, ça a fonctionné pour nous ☺

Si vous êtes intéressés par le changement et l'amélioration de vos processus, permettez-nous de prendre une bonne décision pour vous dès maintenant. Si vous ne faites pas de rétrospectives régulièrement, alors commencez par ça ! Et assurez-vous qu'elles débouchent sur un véritable changement. Faites appel à un facilitateur externe si nécessaire.

Une fois que vous avez mis en place des rétrospectives efficaces, vous commencez un voyage pour évoluer vers le processus parfaitement adapté à votre contexte – qu'il soit basé sur Scrum, XP, Kanban, une combinaison de ceux-ci, ou quoi que ce soit d'autre.

N'arrêtez jamais d'expérimenter !

Kanban ou Scrum n'est pas l'objectif final, c'est l'apprentissage continu qui l'est. L'un des grands principes dans le monde du logiciel est le feedback très rapide : c'est la clé de l'apprentissage. Donc utilisez ce feedback ! Posez des questions sur tout, expérimentez, échouez, apprenez, puis expérimentez à nouveau. Ne vous inquiétez pas de bien faire les choses dès le début, parce que vous ne réussirez pas ! Il suffit de commencer à un endroit et de progresser à partir de là.

Le seul *véritable* échec est de ne pas réussir à apprendre de ses échecs.

Mais bon, vous pouvez aussi apprendre en le sachant.

Bonne chance et profitez du voyage !

Henrik et Mattias, Stockholm le 24/06/2009

H : C'est tout ce que nous avons ?

M : Je pense que oui. Arrêtons-nous ici.

H : Peut-être devrions-nous leur dire qui nous sommes ?

M : Exact. Si nous le faisons, nous passerons pour des gars sympas et on nous proposera peut être des missions de consultant.

H : Alors on le fait ! Ensuite, on s'en va.

M : Oui, on a autre chose à faire, ainsi que nos lecteurs d'ailleurs.

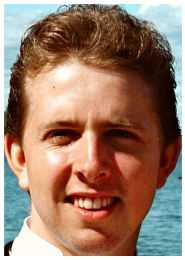
H : En fait, mes vacances commencent maintenant. ☺

M: OK, n'insiste pas.

Les Auteurs

Henrik Kniberg et Mattias Skarin sont des consultants de la société Crisp à Stockholm. Ils aident les entreprises à réussir dans le développement logiciel à la fois sur les plans techniques et humains. Ils ont aidé des dizaines d'entreprises à mettre en œuvre les principes Lean et Agile pour travailler efficacement.

Henrik Kniberg



Au cours de ces dix dernières années, Henrik a été Directeur Technique de 3 sociétés informatiques suédoises et en a aidé bien d'autres à améliorer leur processus. Il est Certified Scrum Trainer et travaille régulièrement avec les pionniers du Lean et de l'Agile tel que Jeff Sutherland, Mary Poppendieck et David Anderson.

Le précédent livre de Henrik “Scrum et XP depuis les Tranchées” a conquis plus de 150 000 lecteurs et est l'un des livres les plus populaires sur le sujet. Il a été distingué à plusieurs reprises en tant que meilleur orateur lors de conférences internationales.

Henrik a grandi à Tokyo et vit maintenant à Stockholm avec son épouse Sophia et ses trois enfants. Il est musicien à ses heures perdues, compose et joue de la basse ainsi que du clavier avec des groupes locaux.

henrik.kniberg<at>crisp.se

http://blog.crisp.se/henrikkniberg

http://www.crisp.se/henrik.kniberg

Mattias Skarin

Mattias travaille en tant que coach Lean et fait profiter les entreprises des avantages du Lean et de l'Agile. Il coache toutes les couches, des développeurs aux managers. Il a aidé une société qui développe des jeux à réduire son temps de développement de 24 mois à 4 mois, a restauré la confiance dans un département de développement entier et fut l'un des pionniers de Kanban.

En tant qu'entrepreneur, il a cofondé et dirigé deux entreprises.

Mattias a un diplôme d'ingénieur qualité et a travaillé pendant 10 ans comme développeur sur des systèmes critiques.

Il vit à Stockholm et adore danser le rock n'roll, courir et skier.

mattias.skarin<at>crisp.se

http://blog.crisp.se/mattiasskarin

http://www.crisp.se/mattias.skarin

Les Traducteurs

Claude Aubry

Développeur en SSII, architecte logiciel, chef de projet puis consultant, Claude Aubry a créé sa société de conseil en 1994. Depuis 2005, il se consacre à Scrum et aux méthodes agiles.

Il est président de l'association toulousaine SigmaT dédiée à la promotion des méthodes agiles, et membre du bureau du Scrum User Group français.

Il est également professeur associé pour l'IUP ISI à l'Université Paul Sabatier de Toulouse (où il enseigne Scrum) et à l'initiative du logiciel OpenSource IceScrum (dédié à Scrum). Il est l'auteur du premier livre en français sur Scrum et publie régulièrement sur son blog "Scrum, Agilité et Rock'n roll". Vous pouvez également le suivre sur Twitter @claudeaubry.

Il vit à Castanet-Tolosan, près de Toulouse, avec Ruth. Maintenant que les enfants ont quitté la maison et qu'il ne reste que Suissi (le chat), il a plus de temps pour se consacrer à Scrum, au rugby (après une carrière modeste d'ailier, il soutient le Stade Toulousain au stade ou devant la télé avec des amis et des bières) et au rock'n roll.

http://www.aubryconseil.com/

Frédéric Faure

Architecte Java/JEE avec plus de 10 ans d'expérience, Frédéric Faure a découvert l'agilité en 2006 au sein du groupe Excilys.

Il a créé Equitalis en 2007 pour porter le modèle Excilys et promouvoir l'agilité dans la région bordelaise. Il est devenu par expérience consultant agile tout en gardant un pied dans la technique.

Il s'investit dans diverses communautés informatiques sur bordeaux : French Scrum User Group, Okiwi. Il a activement participé à l'Agile Tour 2009 à Bordeaux en tant que sponsor, co-organisateur et intervenant !

Vous pouvez le suivre sur Twitter @ffaure32.

Il vit dans la région bordelaise avec sa petite famille (marié, 2 jeunes enfants). Pour être cohérent avec sa passion pour Scrum, il pratique aussi régulièrement que possible le rugby ☺

https://www.excilys.com

http://www.agiletour.org/fr/at2009_bordeaux.html

http://twitter.com/ffaure32

Antoine Vernois

Antoine a un doctorat de l'ENS Lyon en Informatique et se spécialise dans les systèmes distribués.

Il pratique intuitivement les grands concepts de l'agilité pendant plusieurs années avant d'en découvrir leurs formalisations en 2008. Il devient alors ScrumMaster et s'implique activement dans les communautés Agile et Scrum. Il est notamment membre de l'association SigmaT et membre du conseil d'administration du Scrum User Group France. Il a également implémenté le Nokia Test/Scrum But.

Il vit actuellement sur Toulouse, mais qui sait où il sera demain ? Il s'intéresse à la photographie et aux littératures de l'imaginaire.

Vous pouvez le suivre sur son blog personnel où l'on parle parfois d'agilité "Mon blog à moi que j'ai" et sur Twitter @avernois.

http://antoine.vernois.net/dotclear

http://twitter.com/avernois

Fabrice Aimetti

Fabrice est passionné par l'Agile et par Scrum en particulier.

Il est impliqué dans des communautés telles que le Scrum User Group bordelais, Okiwi et SigmaT.

Régulièrement, il publie un billet de rétrospective ou une traduction sur son bloc-notes agile.

Il vit à Bordeaux avec son épouse Stéphanie et ses deux enfants. De temps en temps, il fait une pause en pratiquant l'Aïkido ou en dégustant un bon Single Malt :-o

fabrice.aimetti <at> gmail.com

http://groups.google.fr/group/sug-bordeaux?hl=fr

http://www.okiwi.org/

http://www.sigmat.fr/

http://www.fabrice-aimetti.fr/dotclear/index.php?tag/Agile