L'organisation d'une équipe de développement logiciel
(6ème édition)

Emmanuel CHENU

emmanuel.chenu@gmail.com
http://emmanuelchenu.blogspot.com/

Cet article décrit l'organisation vers laquelle une équipe de développement de logiciels a convergé pour améliorer son efficacité et sa motivation. Il ne s'agit pas d'un standard d'organisation à priori pouvant être mise en place sur un projet. Il s'agit plutôt d'une solution empirique décrite à postériori. Cette description est une photo des pratiques d'une équipe à un moment donné dans l'évolution de son organisation pour mener un projet.
Dans leur démarche, les développeurs se sont très largement inspirés du Lean et du développement logiciel Agile, en particulier de Scrum et de l'eXtreme-Programming.

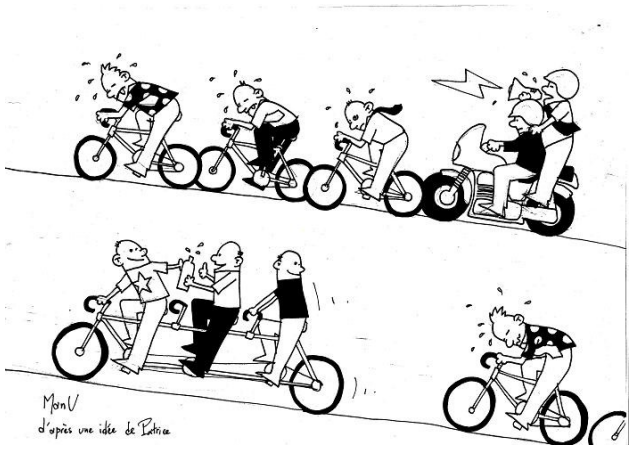
Des projets

Nous développons des logiciels temps-réel critiques embarqués pour l'avionique. La complexité des besoins des clients, le volume et la durée des développements à réaliser, la tenue de jalons ambitieux et la rigueur imposée par les exigences de sécurité ne peuvent être appréhendés que par un travail d'équipe. C'est en confrontant les points de vues, en partageant les expériences et en parallélisant certains travaux au sein d'une équipe compétente que nous pouvons développer des logiciels de qualité dans les délais attendus.

Travailler en équipe est nécessaire à la réussite de nos projets.

Des équipes

Par «équipes compétentes», nous entendons des équipes qui s'engagent en groupe sur des objectifs ambitieux, qui apprennent sans réapprendre, qui développent les compétences de tous, qui améliorent continument leurs pratiques, qui peuvent intégrer rapidement de nouveaux membres en les accompagnant pour les rendre vite contributeurs, qui mesurent leur avancement et identifient les difficultés, qui communiquent en toute transparence sur leur avancement et leurs difficultés, qui prennent l'initiative pour s'adapter, qui recherchent la pluridisciplinarité et qui restent motivées sur le long terme.
Un tel travail en équipe exige un groupe de développeurs solidaires convergeant vers un même objectif et convaincus que la dynamique du groupe est supérieure à la somme des contributions individuelles de ses membres.



Pour travailler en équipe, il faut une équipe!

Cela semble évident, mais si on est réellement convaincu de la force du travail d'équipe, alors il faut inciter et entretenir l'esprit d'équipe.

Des pratiques d'équipe

Nos équipes adoptent et adaptent en continue un ensemble cohérent de pratiques qui incitent à la communication, au retour d'information, à la simplicité et au respect[1]. Ces pratiques ont pour objectifs d'améliorer l'équipe et de bien développer le bon produit.

Le travail en équipe est stimulé par des pratiques d'équipes.

Les pratiques décrites dans cet article se renforcent mutuellement. Certaines dépendent d'ailleurs de la mise en place préalable d'autres pratiques. En tout cas, nous avons constaté sur une succession de projets que l'efficacité de l'équipe a augmenté en appliquant la combinaison de ces pratiques.

Un environnement adapté à l'équipe et ses pratiques

Cultiver une équipe compétente n'est pas une activité déterministe[2]. Néanmoins, certaines conditions aident une équipe à se construire. Notamment, le style de management et l'espace de travail de l'équipe doivent constituer un environnement propice à la fusion des développeurs et à la libre application de leurs pratiques.

L'espace de travail doit être adapté à l'esprit d'équipe et aux pratiques de l'équipe.
Le management doit être adapté à l'esprit d'équipe et aux pratiques de l'équipe.

Suivez le guide

Voici une organisation vers laquelle une de nos équipes à convergé.

1 On retrouve les valeurs de l'eXtreme-Programming. Voir [Beck]
2 Alors que le contraire l'est, malheureusement ... Il est facile de détruire une équipe.

L'espace partagé

Avant tout, l'équipe de développement partage un même espace de travail. Cette proximité nous permet de privilégier la communication orale de face à face et renforce la cohésion de l'équipe. Les développeurs logiciel sont tous installés autour d'une grande table afin que personne ne se tourne le dos. Nous communiquons toujours de face à face, sans avoir à interrompre les activités en cours. Ainsi tout membre de l'équipe peut aisément solliciter ses collègues pour le moindre conseil.
De plus, partager le même bureau favorise la communication passive: chacun est à tout moment au courant des activités des autres, même s'il n'est pas directement impliqué. Ceci a deux principaux avantages: tout membre peut intervenir s'il connait une solution à un problème naissant et tout membre développe une compréhension même vague de l'ensemble des travaux en cours
Aussi, un lieu dédié à une équipe favorise la fusion de l'équipe, pourvu qu'on lui accorde la liberté de s'approprier son espace de travail.[3]

L'équipe s'approprie un espace partagé de travail.

L'espace de travail partagé.
Une même pièce regroupe le bureau des experts du domaine (au premier plan), le bureau des développeurs (au second plan) et le poste d'intégration continue (au fond).

L'espace partagé est une pratique dont je ne pourrais plus me passer tant elle contribue à souder l'équipe et à améliorer la communication entre ses membres. Aussi, de nombreuses autres pratiques ne sont pleinement efficaces, voire même possibles, que si celle-ci est déjà en place[4]. Par contre, elle est accompagnée d'un inconvénient majeur: le bruit. Ce problème est évoqué plus loin. Enfin, les membres de l'équipe doivent être assignés intégralement au même et unique projet afin de que la communication ne soit pas parasitée par des informations ne concernant qu'une minorité. *Il me semble que cette pratique d'équipe garantit un grand retour sur investissement.*

L'espace de travail doit être adapté pour les pratiques de l'équipe.

3 Pour plus d'information sur l'espace de travail partagé, voir la pratique du **Sit-together** dans [Beck].
4 Pair-programming, métriques, war-room et gizmo.

Le pair-programming

Dans l'espace partagé, chaque poste est adapté au travail en binôme et aux relectures de code. En effet, chaque table peut accueillir deux développeurs qui s'échangent le clavier et la souris devant le même écran.

Lors des séances de pair-programming[5], le conducteur est au clavier et à la souris. Il s'occupe principalement de la syntaxe du code, de la compilation et du rejeu des tests. Le navigateur s'occupe principalement de ce que le code doit faire, de la conception et de relire le code. La solution émerge d'une discussion entre les deux équipiers. Le binôme veille à ce que le travail soit découpé en petits bouts livrables et soit synchronisé aussi souvent que possible avec la base commune de code.
Régulièrement, le binôme permutera les rôles. Aussi, l'équipe permutera les binômes. Ainsi, les développeurs ont tous travaillé ensemble et ont eu l'occasion de contribuer à de multiples fonctionnalités de l'application. Cette pratique facilite donc le partage des connaissances et la pluridisciplinarité.

Une telle pratique contribue à fusionner l'équipe par le travail collaboratif et l'entraide. Néanmoins, le temps de travail comprend également des périodes de travail individuel. En effet, les séances de pair-programming exigent beaucoup de concentration et ne laisse pas l'occasion de traiter les tâches personnelles et administratives telles que répondre au téléphone ou consulter sa messagerie électronique.

Un binôme pratiquant le pair-programming.
Pour travailler confortablement en binôme, il faut éviter les bureaux croissant de lune et à tiroirs. De simples tables et des chaises à roulettes font parfaitement l'affaire.

Ceux qui n'ont pas expérimenté le pair-programming évoquent souvent l'idée préconçue et intuitive selon laquelle le travail en binômes réduit la productivité de l'équipe. Lorsqu'on est confronté à de telles objections, il est judicieux de recommander de se renseigner auprès de praticiens. En effet, on constate que ceux qui ont essayé cette démarche ne l'ont pas abandonné.

Le pair-programming me semble une pratique très rentable pour souder l'équipe, pour garantir la propriété collective du code et pour converger plus vite vers un logiciel de qualité. Aussi, je ne vois pas meilleure organisation pour intégrer au plus vite de nouveaux intervenants. Ainsi cette pratique d'équipe offre un grand retour sur investissement. Par contre, il faut bien veiller à permuter les binômes.
Attention, cette pratique contribue sensiblement au niveau de bruit généré par une équipe agile.
On remarque enfin que le travail à deux n'est pas systématique. Des activités comme la documentation ou l'implémentation de certaines fonctionnalités jugées simples et réduites sont menées hors binômes.

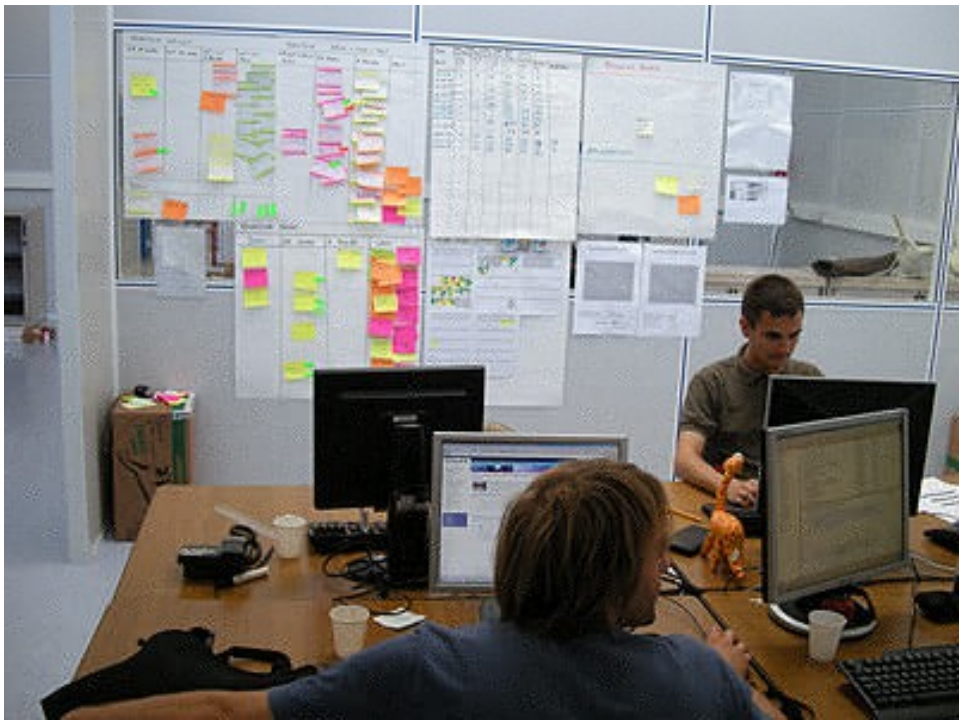
5 Pour plus d'informations sur la pratique du **Pair-programming**, voir [Beck]

La war-room

L'espace de travail commun et le «pair-programming» facilitent la communication orale. Néanmoins, l'équipe s'organise également pour formaliser la communication.

Dans l'espace de travail, toutes les informations dont l'équipe a souvent besoin sont affichées. De même, les éléments utiles pour communiquer sur le projet, sur son avancement et ses difficultés sont visibles sur les murs[6]. L'objectif est d'être en mesure de présenter un court avancement à l'improviste à un visiteur en se basant entièrement sur des informations affichées, pertinentes et à jour.
L'aménagement de l'espace partagé de travail en war-room fait partie des pratiques qui aident une équipe à fusionner. L'équipe s'approprie son espace, qui devient le quartier général du projet.

Ce qui est important pour le projet est visible.



La war-room.
L'espace partagé héberge des réunions: avancement quotidien, planification mensuelle, revue mensuelle et rétrospective mensuelle. Les informations les plus utiles à l'équipe ainsi que celles reflétant l'avancement et les difficultés du projet sont affichées.

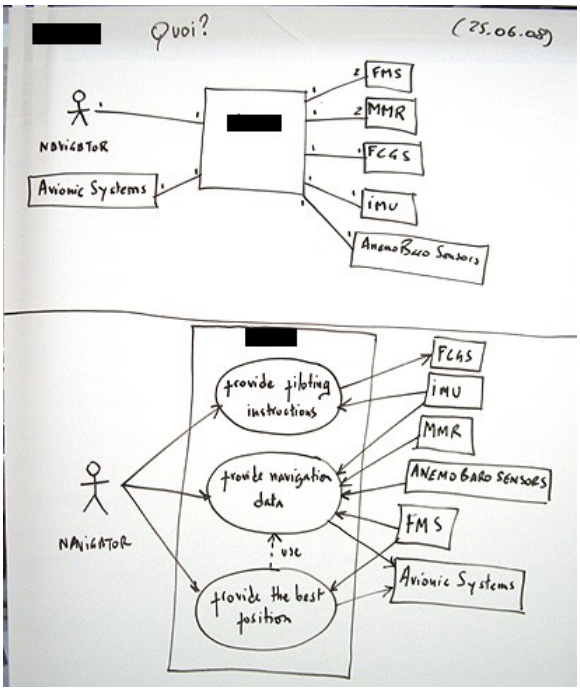
La war-room avec ses outils de management visuel du projet et de l'équipe entretient une transparence qui plait aux managers et aux clients. L'avancement et la santé du projet deviennent tangibles. Aussi, cela garantit des opportunités de présentations et démonstrations d'une équipe auto-organisée en mode agile.

Parmi les informations affichées dans la war-room, nous trouvons:

6 Pour plus d'information, lire la pratique **Informative-workspace** dans [Beck].

Le produit

Un post-it géant[7] présente le produit à développer. Un diagramme illustre le produit dans son contexte opérationnel avec utilisateurs et interfaces externes. Un autre présente les principaux cas d'utilisation du produit.

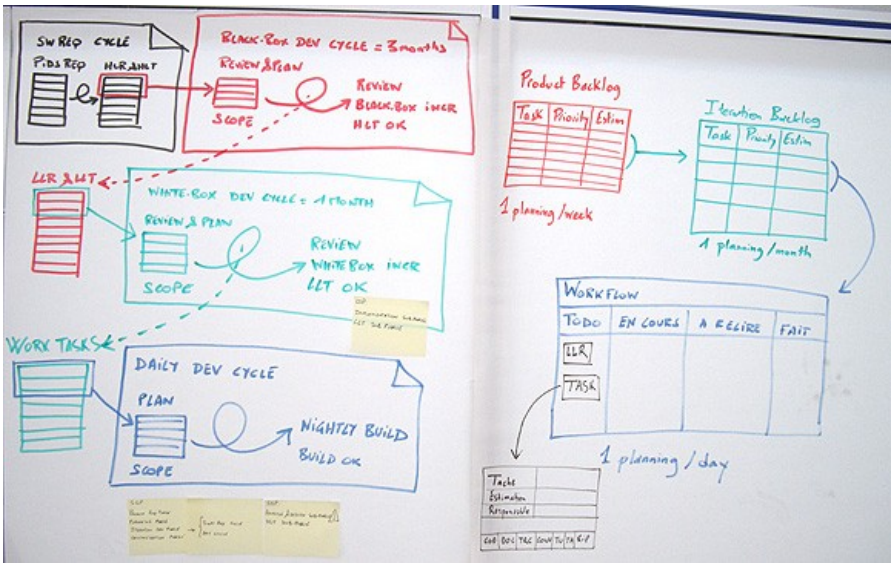


Vulgarisation du produit.

Un diagramme UML décrit le contexte statique du système (son environnement avec utilisateurs et interfaces externes). Un autre identifie ses principaux cas d'utilisation (sa raison d'être pour les utilisateurs).

Le processus de développement

Un second post-it géant présente une vulgarisation du processus de développement actuellement utilisé pour construire le produit.

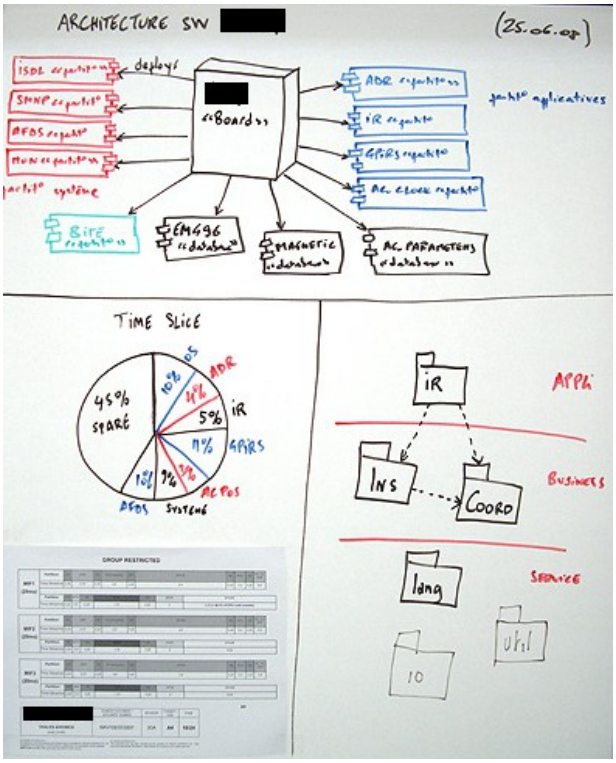


Vulgarisation du processus de développement.

7 Référence des post-it géants.

L'architecture

Un troisième post-it géant décrit synthétiquement l'architecture actuelle de la solution retenue.



Vulgarisation de l'architecture de la solution retenue.
Entre autre, cette vulgarisation comprend un diagramme UML de déploiement et un diagramme UML de décomposition en packages.

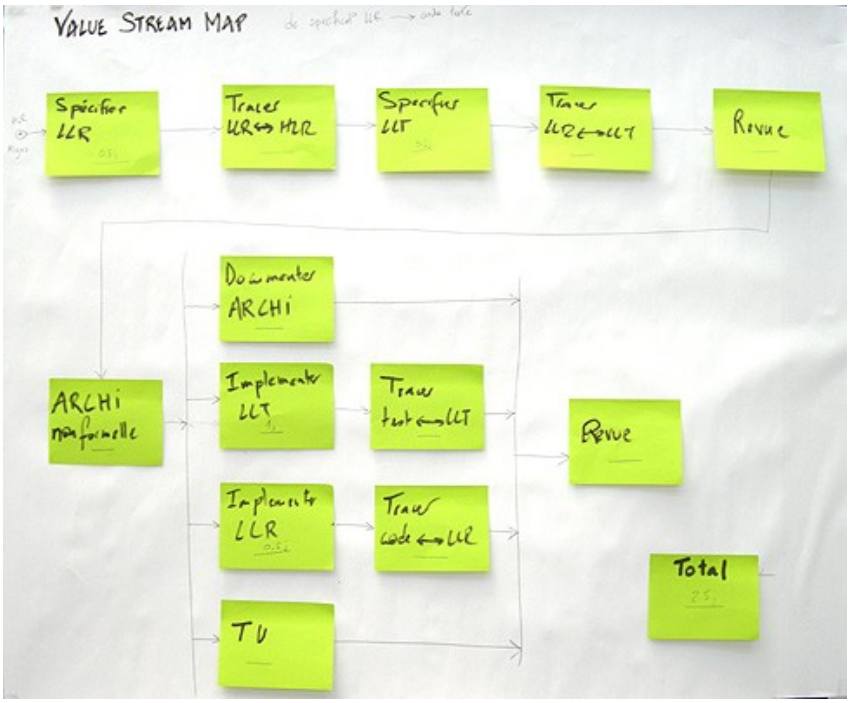
Ces trois affichages permettent de présenter rapidement le projet à un nouveau venu en répondant aux trois questions suivantes: Que développons-nous? Comment développons-nous? Quelle solution développons-nous? Aussi ces vulgarisations permettent de s'assurer que l'équipe entretient une vision claire et à échelle humaine de ses activités.

Un membre de l'équipe présente le projet à un nouvel arrivant en s'aidant des affichages.

Ces vulgarisations ne sont pas vitales. Néanmoins, je pense qu'il est dommage de s'en passer alors que le coût de mise en place et d'entretien est si faible.

Value-Stream-Mapping

Concernant le processus de développement appliqué par l'équipe, une autre pratique est utilisée: le « value stream mapping ».
Toutes les étapes qui permettent de créer le produit sont identifiées et enchainées chronologiquement. Certaines activités se déroulent en séquence et d'autres en parallèle. Des métriques de temps consommé par poste sont mesurées. Ainsi, il est possible d'identifier des goulots d'étranglement dans le flux de travail. Aussi, cette pratique permet d'expliquer à des nouveaux membres comment l'équipe procède pour créer le produit.
Nous avons choisi de limiter le périmètre actuel de la « value-stream-map » à la spécification et à l'implémentation d'une exigence sur le logiciel en boîte-blanche. Ainsi, nous savons combien de temps met un besoin en boîte-blanche pour être identifié et transformé en logiciel utilisable.



Identification de la « value-stream-map »[8]

Attention, chaque étape du « value-stream-map » est une petite activité contribuant à transformer un besoin en logiciel opérationnel et non une phase de transformation de nombreux besoins en un logiciel. L'équipe privilégie la traversée complète du flux par une exigence avant d'en commencer une nouvelle. Il ne s'agit surtout pas de phases appliquées en séquence sur un lot d'exigences.

Nous n'avons pas encore assez d'expérience (et de maturité?) pour tirer pleinement profit du value stream mapping. Cette démarche a surtout donné lieu à un intéressant exercice initial après plusieurs mois de développement. Nous devrions sûrement rejouer l'exercice périodiquement.
Toutefois, nous avons remarqué que la « value-stream-map » est appréciée par les nouveaux intervenants pour sa description synthétique de la procédure de création de logiciel. Aussi, cet exercice aide à définir les critères du travail fini (« done » [9]).

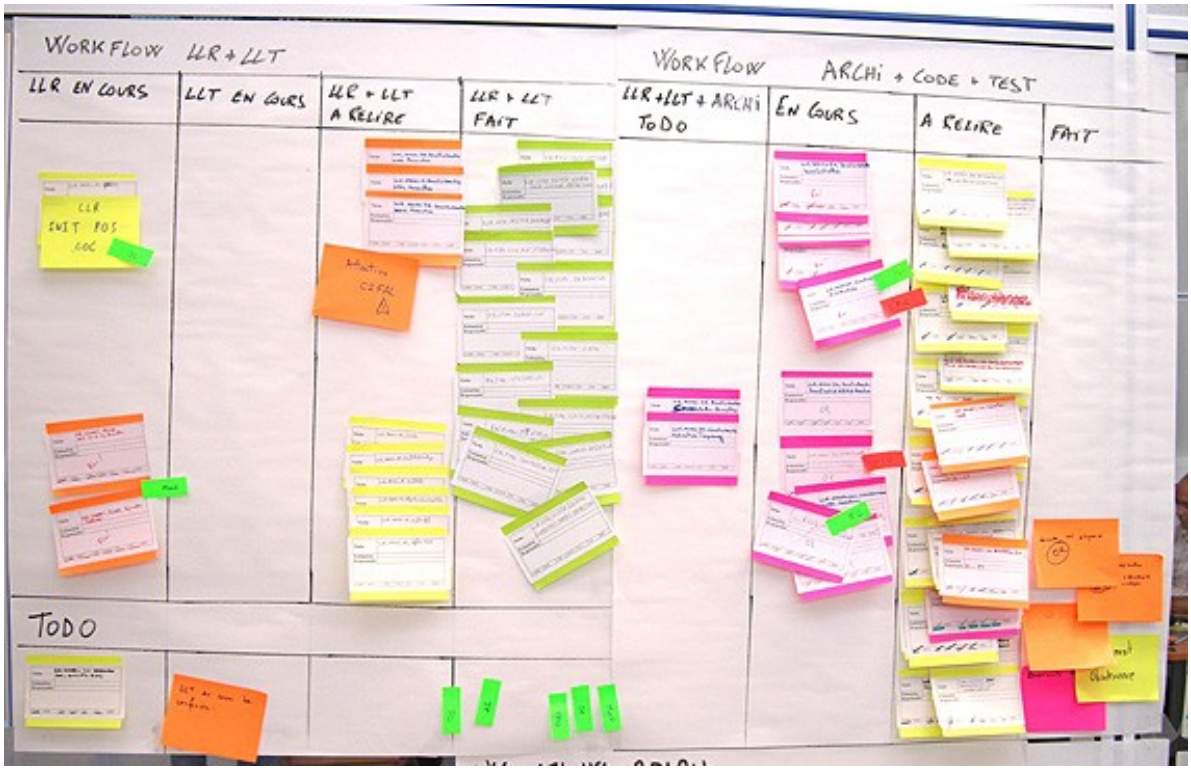
8 Pour plus d'information sur le **value-stream-mapping**, se référer à des ouvrages sur le Lean, comme [Pop1] et [Pop2]
9 La définition du travail fini sera abordé par la pratique « Done, c'est Done »

Le task-board

Une fois que le flux de travail étant identifié à postériori par une « value-stream-map », il reste à suivre les tâches qui le traversent.

Un des tableaux du bureau est un grand Kanban[10] représentant l'état de notre flux de travail à un moment donné. Lors de réunions quotidiennes de moins de dix minutes, nous faisons avancer les tâches, matérialisées par de petits post-it répartis le long des cases du Kanban. Ces cases correspondent grossièrement aux étapes « à faire », « en cours », « à relire » et « terminé ». Cette pratique nous permet de visualiser quotidiennement l'avancement de l'équipe et de l'afficher en toute transparence.
De plus, ce tableau sert de support matériel à la courte réunion d'avancement quotidienne. Les post-it aident les membres de l'équipe à parler de leurs travaux.

En fait, le flux de travail de l'équipe est réparti en deux Kanbans distincts. Le premier est dédié à la spécification des exigences fonctionnelles et leur transformation en logiciel opérationnel. Chaque post-it représente une exigence sur le logiciel. Des cases à cocher permettent de s'assurer que les travaux prévus pour cette exigence sont terminés. Ces cases à cocher sont: « spécification », « recette », « traçabilité de l'exigence vers un besoin amont », « couverture de l'exigence par le moyen de vérification », « documentation de l'architecture », « codage », « test-unitaire », « traçabilité du code vers l'exigence » et « relectures ». Une pastille verte identifie une exigence jugée pertinente par les experts du métier. A l'inverse, une pastille rouge signale les exigences à retravailler.

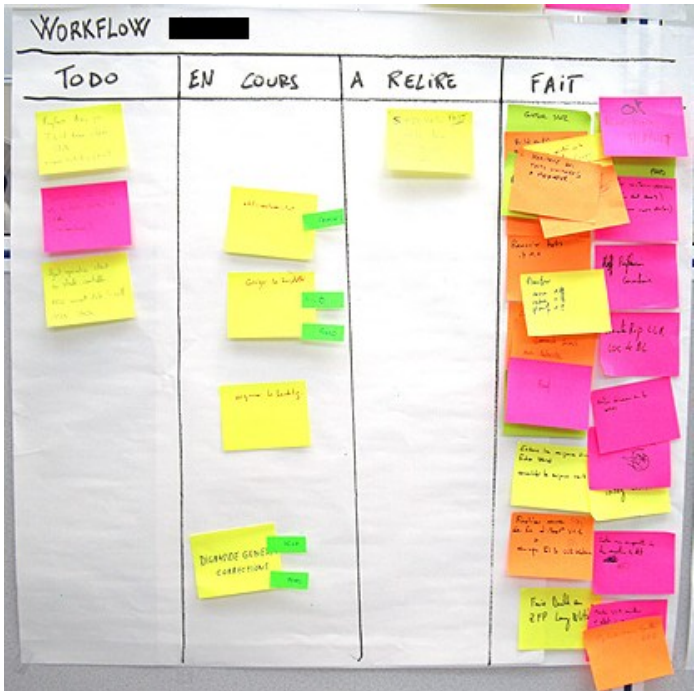


Le Kanban des exigences sur le logiciel.

Les différentes étapes le long du Kanban sont celles identifiées par le Value-Stream-Mapping. En fait, on peut envisager le Value-Stream-Map comme un plan statique du flux de travail et le Kanban comme un plan dynamique du flux de travail qui évolue dans le temps avec les exigences qui le traversent.

10 Pour plus d'information sur la pratique du **Kanban**, se référer au Lean et notamment à [Pop1] et [Pop2].

Le second Kanban est dédié aux activités techniques qui ne sont pas de l'ajout de fonctionnalité au logiciel. Un post-it représente une activité de moins d'une semaine. Ces activités concernent les outils, les maquettes, les réunions à planifier et les problèmes techniques à résoudre.



Le Kanban des activités techniques non-fonctionnelles.
Une petite étiquette nominative indique qui dans l'équipe est responsable de la tâche décrite sur le post-it.

En plus d'organiser et de rendre visibles les flux de travail, les Kanban permettent d'identifier les goulots d'étranglement et de détecter la dispersion de l'effort. Idéalement, un développeur doit mener une tâche à terminaison avant d'en entamer une nouvelle. Les post-it nominatifs aident l'équipe à identifier la dispersion et à se réorganiser pour recentrer l'effort pour terminer les tâches en cours.
De plus, les Kanban aident à l'équipe à s'auto-organiser[11]. Si un développeur a terminé les tâches qu'il s'était attribué pour la journée, il peut de lui-même sélectionner une nouvelle activité à mener parmi les post-it de la rubrique « à faire » ou porter assistance à un autre équipier sans avoir à en référer à un chef de projet.

L'équipe se gère elle-même au jour le jour.

Lorsqu'une équipe se gère elle-même, cela a un effet très bénéfique sur la motivation et l'implication des ses membres.

Le Kanban du projet me semble désormais indispensable tant il contribue à la transparence des activités en cours et à l'auto-organisation de l'équipe. Cette pratique garantit un grand retour sur investissement.

11 Pour plus d'information sur l'**auto-organisation** des équipes de développement, se référer à [Beck], [Schwab], [SchwaBee], [Pop1] ou [Pop2].

Le daily-stand-up-meeting

Pour conduire le projet et rester réactif, l'équipe se pilote lors du « daily stand-up meeting ».

Tous les matins, à heure fixe et en lieu fixe, toute l'équipe se réunit debout pour mener une réunion limitée à 10 minutes. L'objectif est que les développeurs synchronisent leurs travaux et se réorganisent en fonction de la situation.
A tour de rôle, chaque membre présente ce qu'il a fait depuis la dernière réunion et identifie les éventuels obstacles qui le gênent. Ensuite, à tour de rôle, chaque membre présente ce qu'il s'engage à faire dans la journée. Les séances de travail en binômes sont alors planifiées pour la journée.
Les informations échangées ne sont pas destinées au chef de projet ou au ScrumMaster mais à tous les membres de l'équipe.
Cette pratique contribue à fusionner l'équipe en réservant un moment pour que tous s'expriment, s'organisent et s'entraident.

L'équipe reste synchronisée.

Daily stand-up meeting[12].
Une réunion est menée par sous-équipe de 7 +/- 2 membres[13]. Afin de ne pas de ne pas déborder dans le temps, l'équipe utilise un minuteur réglé à 10 minutes. La sonnerie annonce la fin de la réunion.

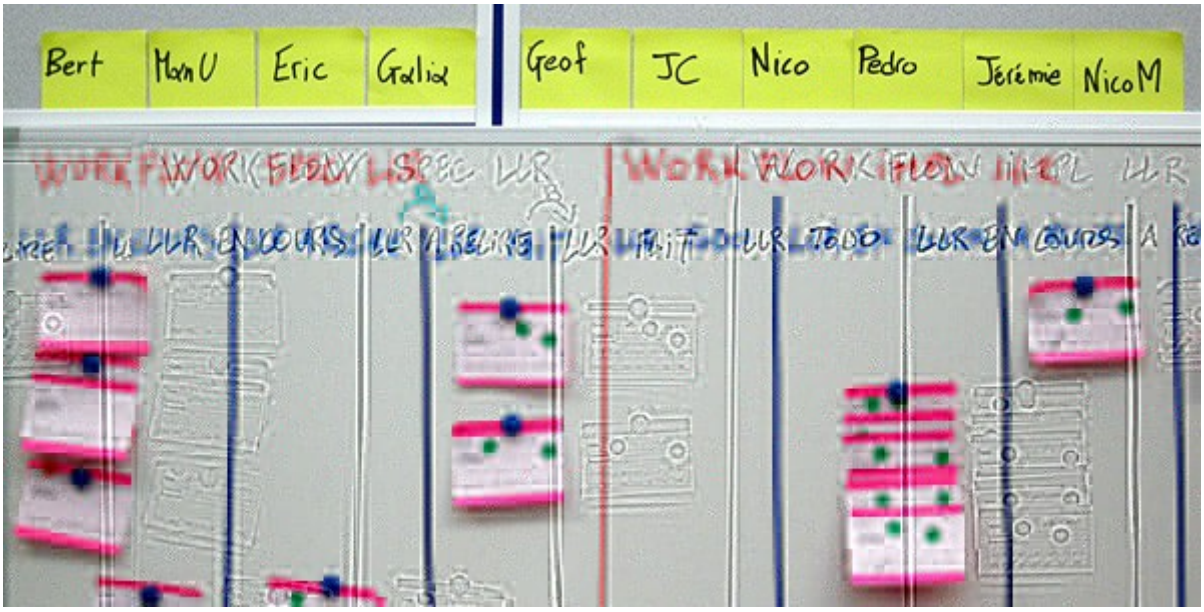
Certaines informations échangées sont formalisées en actualisant les Kanban et « burn-down-charts ». D'autres sont formalisées sur le « blocking-board » et la boîte à idées, décris ci-après.

La courte réunion de pilotage quotidien garanti un grand retour sur investissement pourvu qu'on l'applique de manière très disciplinée. Il faut à tout prix en limiter la durée et en éliminer toute discussion technique ou hors-sujet. Il est si facile de laisser cette pratique dériver hors de son objectif initial pour ensuite l'abandonner en prétextant un manque d'efficacité.

12 Pour plus d'information sur le **daily-stand-up-meeting** ou **daily-scrum-meeting**, se référer à [Schwab] ou [SchwaBee].
13 Voir le nombre idéal de membres par équipe dans [Schwab]

Composition des équipes

Pour clairement identifier tous ceux dont la présence au « stand-up-meeting » est obligatoire et pour signaler ceux qui doivent prendre la parole, les membres de l'équipe sont identifiés au dessus de son Kanban.



Cette équipe est composée des 10 membres identifiés au dessus de son Kanban. Ces équipiers participent à la même réunion de pilotage quotidien.

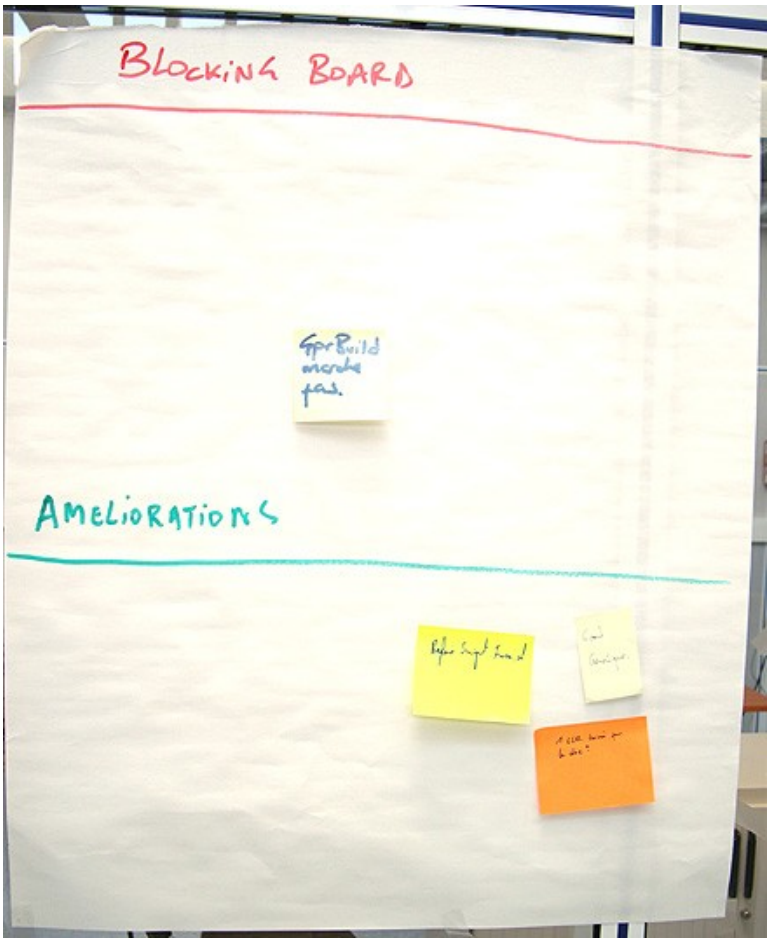
Nous avons introduit cette pratique au moment où l'équipe de développement a été réorganisée en trois sous-équipes. En fonction des itérations, certains membres passent d'une équipe à l'autre. Chaque sous-équipe mène son propre « stand-up-meeting ». L'identification des membres permet de signaler de qui les trois sous-équipes sont composées pour l'itération en cours.

Le blocking-board

L'équipe colle sur le blocking-board des post-it décrivant des problèmes qui nuisent à l'efficacité du travail. Ces problèmes sont discutés lors des réunions quotidiennes et lors de la réunion mensuelle de rétrospective d'itération.

La boite à idée

De même, l'équipe colle sur la boite à idée des post-it décrivant des idées d'amélioration du logiciel, du processus de développement et d'organisation de l'équipe. Ces idées sont discutées lors des réunions quotidiennes et lors de la réunion mensuelle de rétrospective d'itération.



Le blocking-board et la boite à idées.

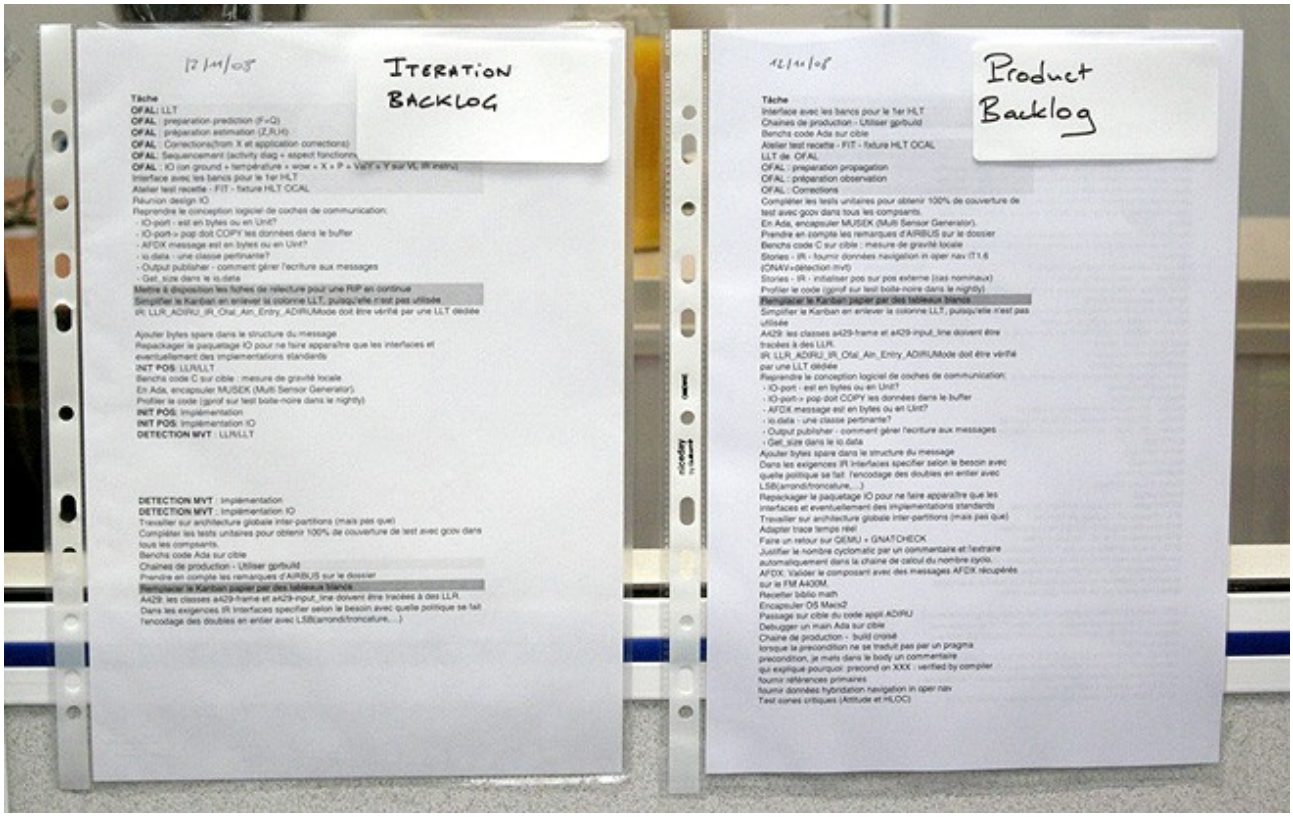
Régulièrement, lors des réunions mensuelles de planification ou lors des réunions quotidiennes, l'équipe peut décider d'essayer certaines idées d'amélioration et de supprimer des obstacles. Ces tâches traversent alors le Kanban des activités techniques non-fonctionnelles.
Ces pratiques permettent donc à l'équipe de s'approprier sa démarche travail en la faisant évoluer en continue par la prise en compte d'idées d'amélioration et la correction de problèmes gênants.

L'organisation de l'équipe s'adapte en continu à la situation.

Je pense qu'il est dommage de se passer du blocking-board et de la boîte à idée tant leur coût de pratique est faible pour une contribution non-négligeable à l'auto-organisation de l'équipe.

Les backlogs

Pour planifier les activités, l'équipe entretien le backlog produit et le backlog de l'itération en cours. Il s'agit de la liste priorisée et estimée du restant à faire à l'échelle du projet et plus finement à l'échelle de l'itération en cours. Ces documents sont maintenus au format électronique, mais des extractions datées sont imprimées et mis à disposition dans des pochettes plastiques attachées au mur. Ainsi, il est possible de rapidement consulter les priorités du projet pour prendre une décision de pilotage.



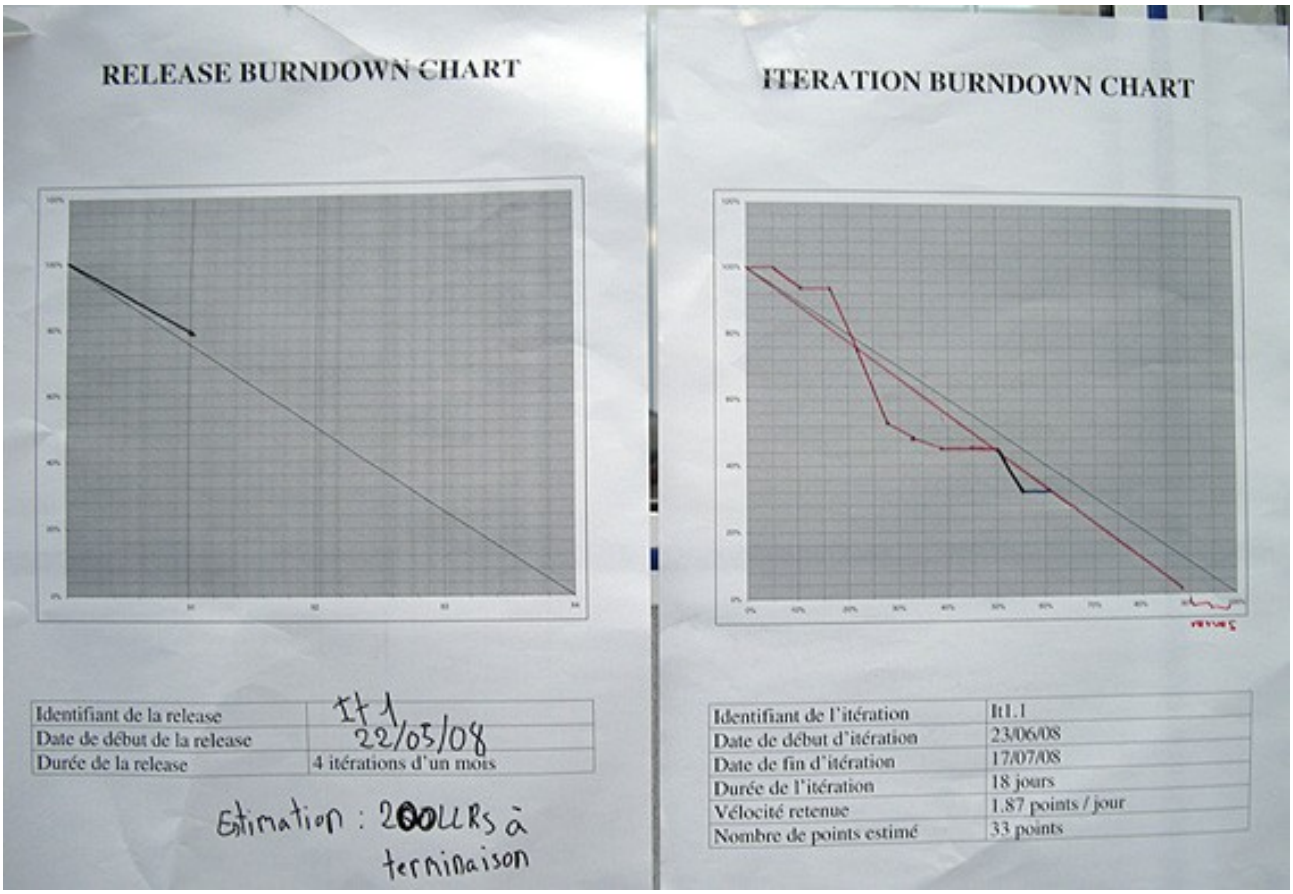
Le backlog produit et le backlog de l'itération en cours sont accessibles dans le bureau

Chaque équipier a rapidement accès aux priorités du projet.

L'élaboration des backlogs est une manière efficace d'entretenir une vision du déroulement du projet et de planifier les travaux. Cela permet également de mesurer l'avancement. Le fait d'afficher publiquement ces artefacts permet à l'équipe de s'approprier les objectifs du projet et de prendre des décisions en connaissance des priorités. Cela permet également de discuter des travaux avec un client, manager ou équipier sans avoir à s'installer devant un écran. Il s'agit encore d'une pratique très facile à mettre en place, pourvu que l'équipe entretienne des backlogs.

Les burndown-charts

L'équipe mesure son avancement avec deux outils: le « release burn-down chart » et l' « iteration burn-down chart ». Les « burn-down charts » présentent en abscisse le temps alloué pour tenir un objectif et en ordonné l'effort consommé pour tenir cet objectif. A la fin de la période prévue, l'effort estimé doit être consommé et l'objectif doit être tenu. L'avancement idéal est symbolisé par la droite à 45 degrés. Si la courbe réelle est au dessus de la droite à 45°, le projet prend du retard. Au contraire, si la courbe réelle est en dessous de la courbe à 45°, l'avancement est meilleur que prévu.
L' « iteration burn-down chart » est actualisé tous les matins lors de la réunion d'avancement en fonction du nombre d'exigences qui passent dans la rubrique « à relire » du task-board. Le « release burn-down chart » est actualisé mensuellement à chaque fin d'itération[14].



Le « release burn-down chart » et l' « iteration burn-down chart ».

Les « burn-down charts » offrent un avancement visuel très facile à interpréter: si la courbe réelle est au dessus de la droite à 45° alors le projet est en retard sur l'estimation. Si la courbe reste en dessous, l'avancement est meilleur que l'estimation.

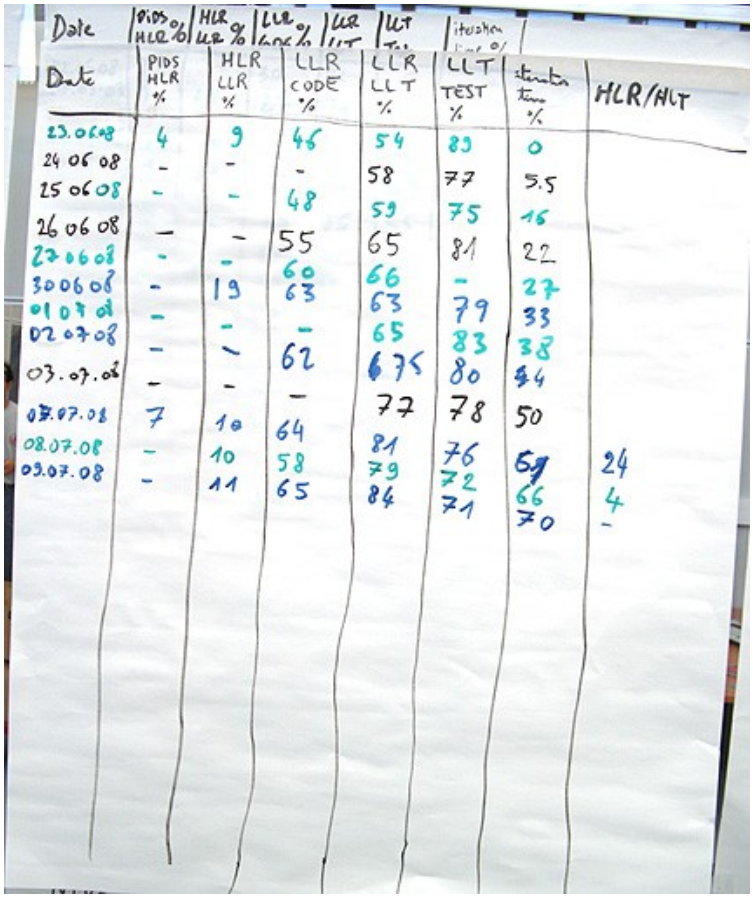
Chaque membre de l'équipe sait où et comment va le projet.

Ne vous passez pas d'un tel moyen de mesure du restant à faire. Je pense qu'il s'agit de l'outil le plus simple et le plus intuitif pour estimer l'avancement du projet. Bien sûr, le plus difficile n'est pas de mettre à jour ces courbes, mais bien de disposer d'estimations du restant à faire ...

14 Pour plus d'information sur les **burn-down charts**, se référer à [Schwab] et [SchwaBee].

Les métriques

Les métriques de pilotage du projet sont automatiquement actualisées quotidiennement *(par le « build » de nuit)*. Le développeur le plus matinal relève sur le site intranet d'intégration continue les valeurs calculées pendant la nuit et les écrit sur un post-it géant. Lors de la courte réunion quotidienne d'avancement, l'équipe commente ces métriques et s'organise pour une éventuelle action corrective.



Les métriques fraiches du jour.
L'historique des métriques est automatiquement tracé en courbes consultables sur le site d'intégration-continue.

Le fait de suivre des indicateurs de pilotage est tout à fait indispensable. Par contre, on peut se passer d'écrire ces métriques sur un post-it géant d'autant plus qu'elles proviennent d'artefacts électroniques. Néanmoins, nous avons remarqué qu'on néglige vite des données publiées sur un site web. L'affichage de ces données à l'avantage de forcer la discussion entre les membres de l'équipe.

Les jalons

Des post-it représentant les jalons du projet sont collés sur un grand calendrier affiché au mur. Ainsi, l'équipe visualise le temps qui la sépare des moments clés du projet.

Le niko-niko

Chaque soir, en quittant le travail, les membres de l'équipe collent une pastille de couleur sur le calendrier dans la case du jour révolu. Une pastille verte indique que le développeur a passé une bonne journée sur le projet. Une pastille jaune représente une journée neutre. Une pastille rouge marque une mauvaise journée. Une tendance de pastilles jaunes et rouges signale que l'équipe doit s'adapter pour redistribuer les tâches ou repenser son organisation. Cette pratique permet de suivre un indicateur sur le « sustainable pace[15] ». Elle permet aussi à l'équipe de fusionner en la rendant maître de sa propre organisation.

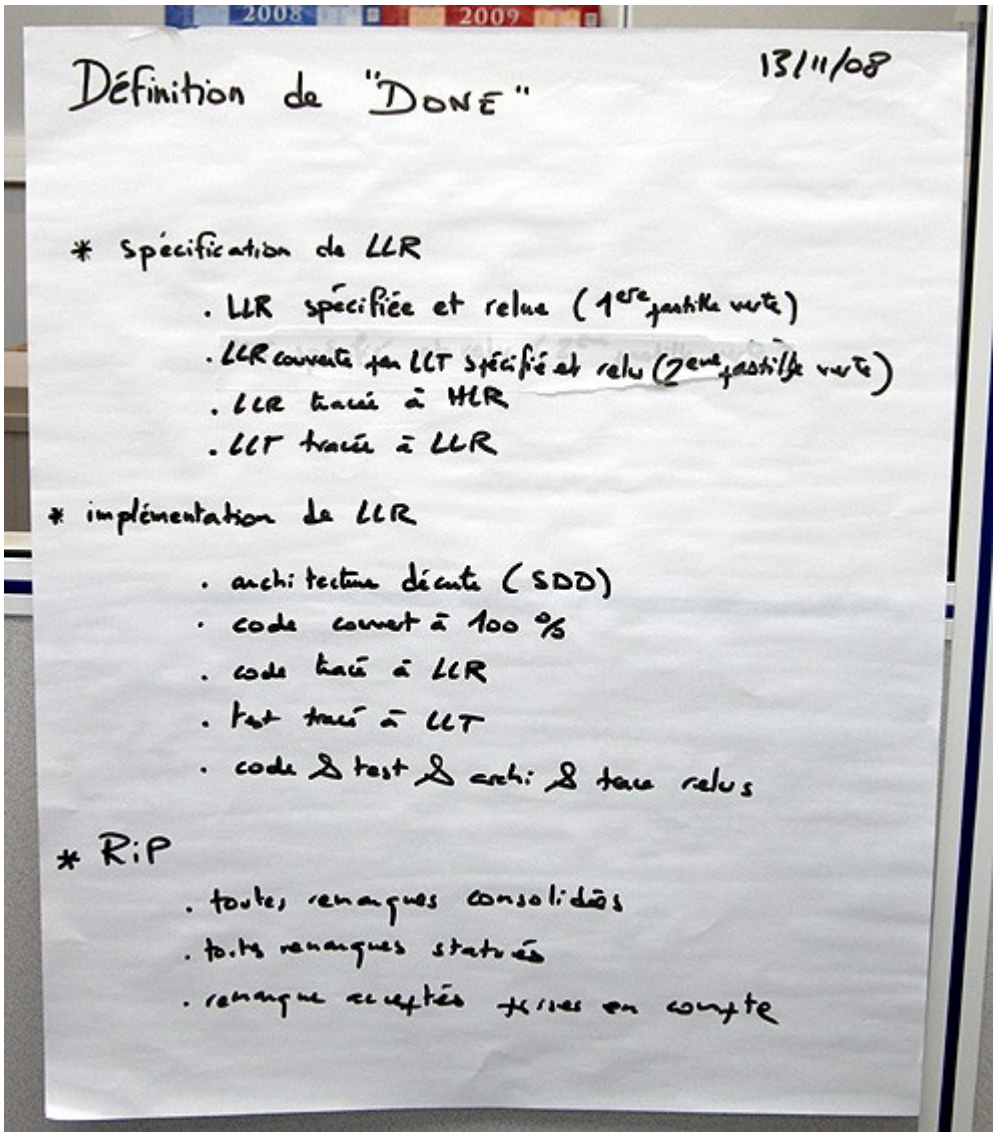
Le calendrier des jalons et de la santé de l'équipe.

Pour le moment, nous ne savons pas encore ce qu'il faut retirer de cette pratique en cours d'expérimentation. En tout cas, elle permet de vérifier la corrélation entre la santé de l'équipe et la santé du projet. Elle permet aussi à chacun de s'exprimer sur la manière dont il vit le projet, sans avoir à attendre la rétrospective d'itération mensuelle.

15 Pour plus d'information sur le **sustainable-pace**, se référer à [Beck].

« Done » c'est « Done »

Lorsque nous subissons la pression des délais, nous développeurs avons tendance à réduire le niveau de qualité ou à perdre en rigueur sur la notion de travail terminé. D'une part, le respect partagé de la définition de travail partagé permet de maintenir le niveau de qualité. D'autre part, cela permet de savoir objectivement où en est le projet et d'utiliser des métriques pertinentes de vélocité pour se projeter dans l'avenir. Ainsi, l'équipe élabore, partage et affiche dans le bureau la définition commune du travail terminé pour le projet.



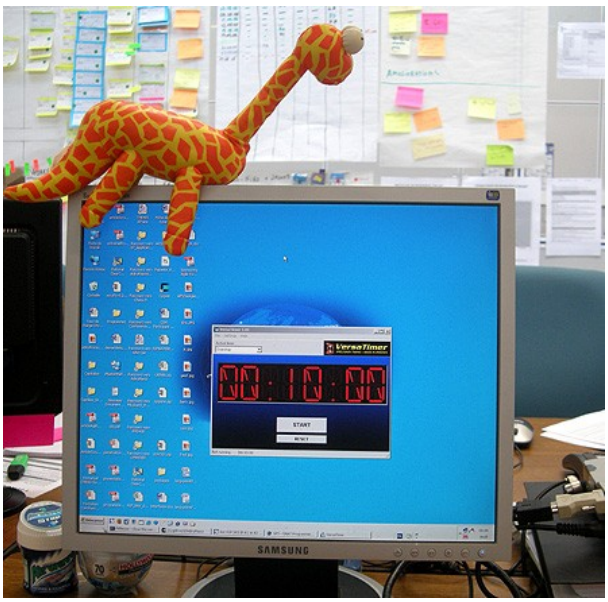
La définition partagée du travail terminé.

Le fait que tous les équipiers partagent une même définition du travail terminé est absolument essentiel pour maintenir un niveau de qualité, pour mesurer l'état du projet et pour se projeter dans l'avenir.

Le Gizmo

L'équipe cherche à avancer continuellement par petits pas pour entretenir un flux constant de travail terminé et pour éviter à tout prix de se désynchroniser longtemps par rapport à la base commune de code. Une des difficultés souvent rencontrées par les développeurs est d'apprendre à découper ses activités en petits lots de travail livrable, chaque lot ne prenant qu'une ou deux heures.
Une pratique pour apprendre à travailler ainsi est d'utiliser un sémaphore d'intégration. Dans notre équipe, il s'agit d'une peluche baptisée « Gizmo ». Tout développeur ou binôme qui délivre du travail doit « prendre le Gizmo » le temps de se synchroniser avec la base commune de code et d'intégrer ses travaux. Ainsi, il rend visible le fait qu'il se synchronise et contribue. Il n'existe qu'un seul Gizmo pour toute l'équipe, ce qui rend l'intégration exclusive.
Au bout d'un moment, grâce au Gizmo, l'équipe apprend d'elle-même à détecter et aider les membres qui se désynchronisent trop longtemps au risque de subir des intégrations douloureuses. Aussi, les développeurs recherchent à prendre le Gizmo aussi souvent que possible fluidifiant ainsi leurs activités de manière ludique. Le travail est alors ponctué d'appels « Gizmo! ».

Le Gizmo, ou sémaphore d'intégration.
Le Gizmo sur l'écran signifie que le développeur est en train de se synchroniser avec la base commune de code et d'intégrer ses modifications.



Le Gizmo est pris. L'équipe constate quel poste est en cours de synchronisation et contribution.

Cette pratique souvent assimilée à du folklore est pourtant une vraie aide à l'intégration continue et fluide des développements. Elle permet également de détecter les développeurs enlisés dans des activités et d'inciter l'entre-aide au sein de l'équipe. Son coût se limite au prix d'une peluche.

Les minuteurs et le time-boxing

Une autre manière de fluidifier le travail en multipliant les séquences de travail terminé est d'utiliser la technique du « time-boxing ». Il s'agit d'estimer le temps nécessaire pour remplir un petit objectif, comme prendre une décision en réunion technique et de s'imposer de ne pas dépasser cette limite de temps en utilisant un minuteur. Ainsi, toutes les réunions sont minutées: dix minutes pour la réunion quotidienne d'avancement, une heure pour la réunion de mise à jour du backlog produit[16], une heure pour la revue d'itération, une heure pour la rétrospective d'itération mensuelle et deux heures pour la réunion de planification d'itération mensuelle. Cette pratique permet de respecter le temps de chacun et d'arrêter les séances qui dérivent. Certains poussent cette technique plus loin en s'inspirant de la technique du Pomodoro[17]. Pour cela, des applications minuteur sont installées sur les postes et de vrais minuteurs de cuisine sont disponibles dans le bureau.

Minuteur sur PC et minuteurs individuels.

Souvent jugée comme du folklore, l'utilisation des minuteurs est pourtant un outil pragmatique et économique pour contribuer à des séances de travail efficace.

Safe deliver

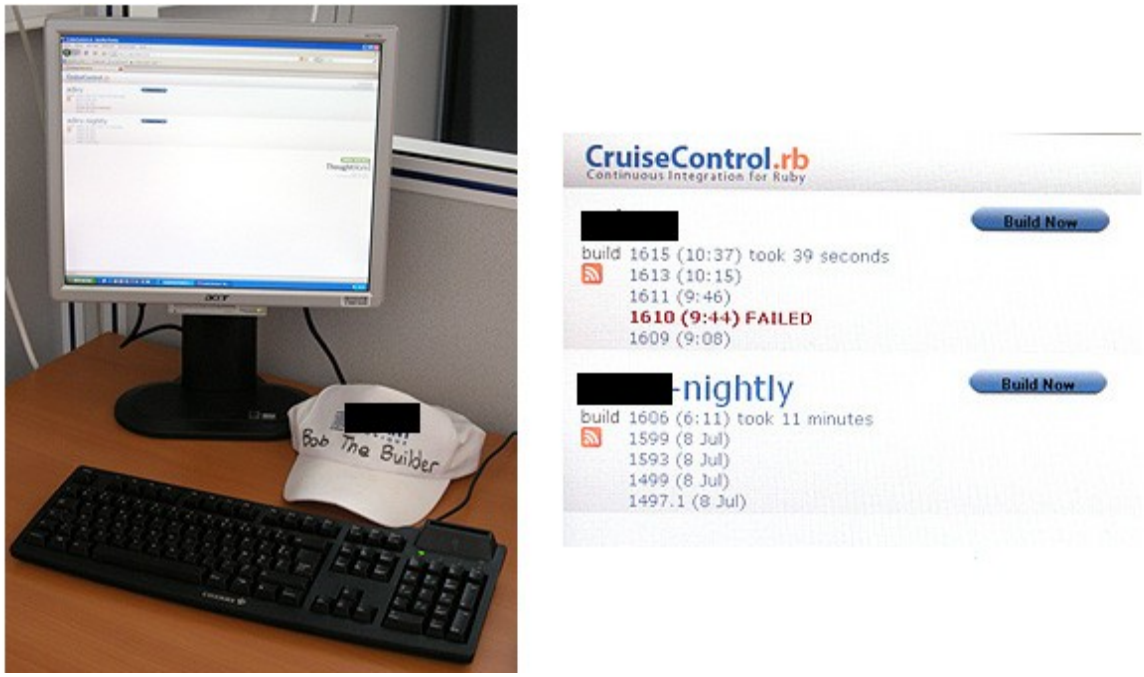
Toute livraison de modifications vers la base commune de code se fait au moyen d'un script. Ce script ne réalise effectivement la livraison que si tous les tests passent avec succès. Cette pratique évite à priori l'introduction de régressions et de de modifications incompatibles dans la base de code. Ainsi, l'équipe est libérée du stress de la contribution destructive. La seule manière de « contribuer » en « cassant le build » est de livrer des modifications dépendant d'éléments non gérés en configuration. Cette mauvaise manipulation n'est pas grave car elle est très vite détectée et alertée par Bob the Builder.

Cette pratique contribue à la prévention des défauts pour un très faible coût de mise en œuvre. En effet, un tel script s'écrit et se teste en une ou deux heures.

16 Pour découvrir ce qu'est un **product backlog**, se référer à [Schwab] ou [SchwaBee].
17 Pour plus d'information sur la technique du **Pomodoro**, se référer à http://www.tecnicadelpomodoro.it/docs/francesco-cirillo/2007/ThePomodoroTechnique_v1-3.pdf.

Bob the Builder

L'équipe utilise un poste d'intégration continue[18] pour réaliser deux constructions automatisées du logiciel.
Une construction incrémentale avec rejeu des tests est déclenchée toutes les minutes sur détection d'une modification livrée dans la base commune de code.
Une construction complète avec rejeu des tests, mesure de la couverture structurelle et production des métriques est déclenchée toutes les nuits.
Un échec de construction provoque l'émission d'un e-mail d'alerte à tous les membres de l'équipe. Les développeurs doivent alors corriger ce problème en toute priorité[19].
Bien sure, la pratique efficace de l'intégration continue sous-entend que l'équipe pilote le développement par les tests (TDD[20]).



Le poste d'intégration continue avec l'outil CruiseControl.

La pratique de l'intégration continue est essentielle au bon déroulement d'un projet de développement. L'énergie déployée dans la gestion de configuration, les tests, la construction en une action et la mise en place d'un outil d''intégration continue est un investissement des plus rentables.

Le groove[21]

Les itérations et les réunions de planification, de revue et de rétrospective d'itération définissent un cycle mensuel de travail[22]. Les réunions d'avancement, le Niko-Niko et les constructions automatisées de nuit définissent un cycle quotidien de travail. Ces cycles, ainsi que le Gizmo et le time-boxing donnent du rythme à l'équipe. Ainsi, il existe un flux constant de travail terminé qui accorde la satisfaction du travail accompli[23].

18 Pour plus d'information sur la pratique de l'**intégration continue**, se référer à [Beck].
19 Pour plus d'information sur la pratique du **Stop-the-line**, se référer au Lean ou plus particulièrement à [Pop1] et [Pop2].
20 Pour plus d'information sur la pratique du **Test-Driven Development**, se référer à [Beck].
21 Le rythme.
22 Pour plus d'information sur les pratiques de la démarche Scrum, se référer à [SchwaBee] et/ou [Schwab].
23 Pour plus d'information sur l'importance du flux de travail, se référer au Lean et à [Pop1] et [Pop2] en particulier.

Les tableaux blancs et les post-it géants

Nous modélisons beaucoup en groupe et au marqueur en utilisant la notation UML. Les conceptions sur lesquelles nous sommes en train de travailler sont ainsi affichées en grand format. Ceci nous permet de revenir fréquemment sur nos décisions de design pour les conforter ou les remanier au fur et à mesure des informations que nous collectons en codant et en testant.
Cette pratique aide l'équipe à fusionner puisque l'architecture est une activité menée en groupe et non sous la responsabilité exclusive et l'autorité d'un architecte.

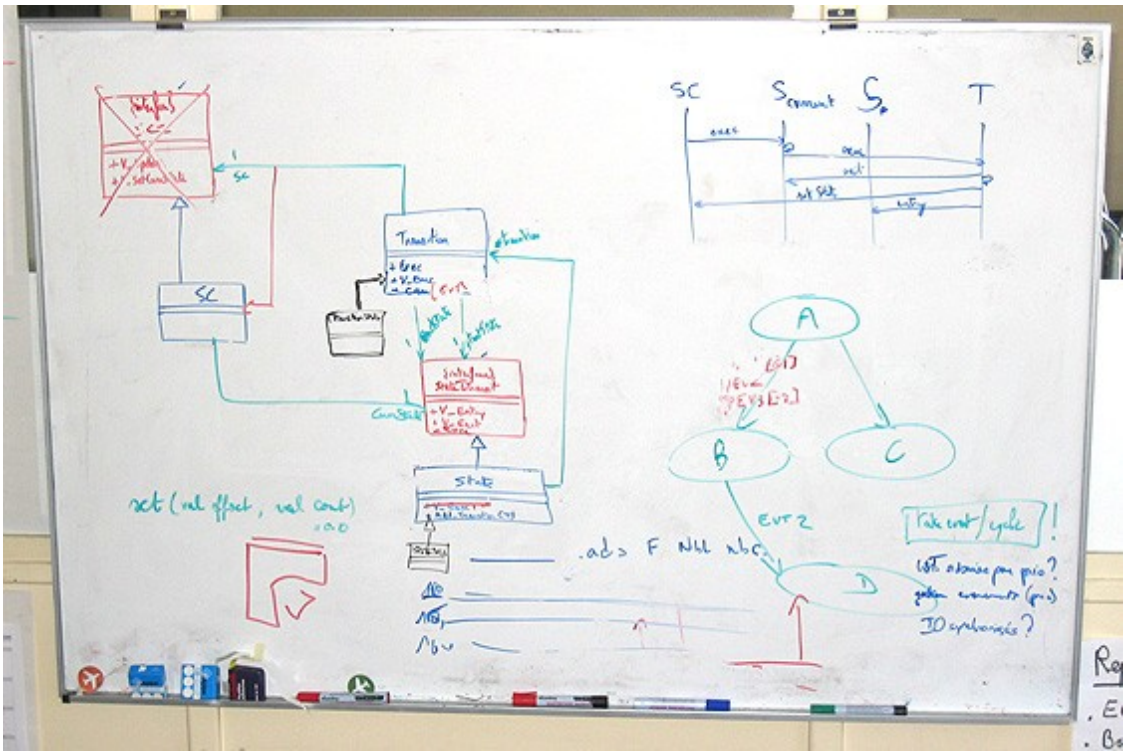
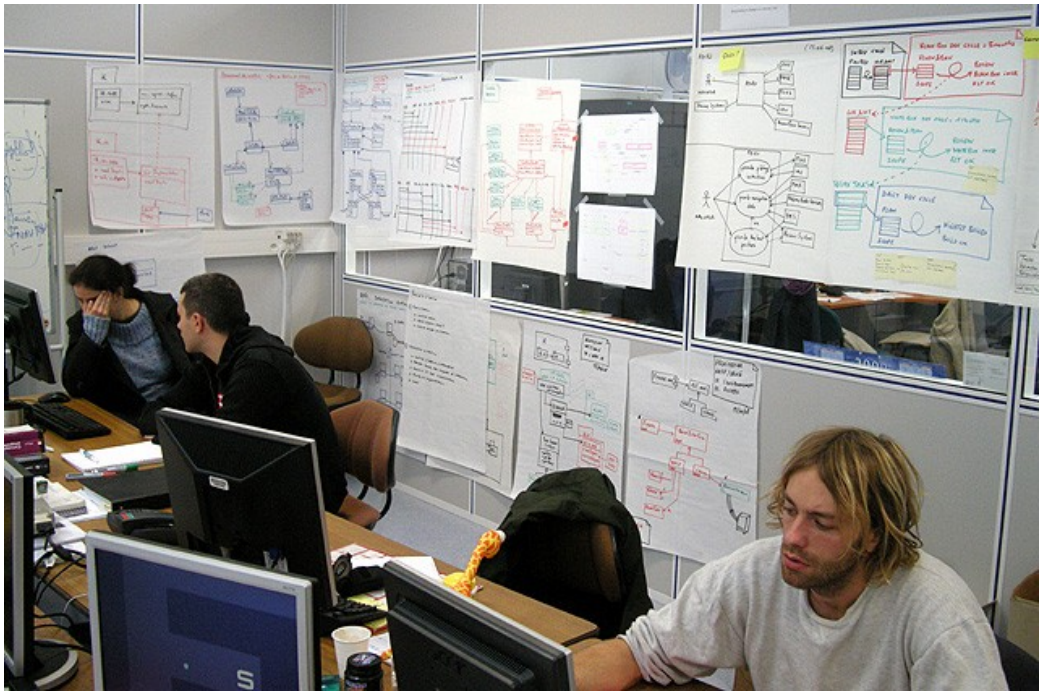


Tableau blanc vivant.



Conception murale

Le vidéo-projecteur

L'équipe dispose d'un vidéo-projecteur installé à demeure dans la war-room. Cet équipement facilite grandement la conduite des réunions mensuelles de planification, de revue et de rétrospective d'itération. Il en est de même pour la réunion hebdomadaire de mise-à-jour du backlog produit. Le vidéo-projecteur est également utilisé pour des présentations et relectures de code en équipe sur des points particulièrement importants. Un tableau blanc sert alors d'écran de projection.

Vidéo projecteur à demeure.

Chef de projet, ScrumMaster, coach et ProductOwner

Il y a peu de rôles distribués dans l'équipe. Comme dans de nombreuses équipes dites Agiles, le rôle du chef de projet est grandement modifié. Il perd le rôle de chef d'orchestre directif, de médiateur et de pilote. En effet, l'équipe s'organise elle-même, choisit ses pratiques et pilote ses travaux avec les indicateurs qu'elle sélectionne et collecte. Le ScrumMaster joue souvent le rôle de médiateur et facilitateur.
L'équipe assure la gestion du projet à raison de huit heures par itération mensuelle et par développeur. Il s'agit du temps passé par chacun en réunions quotidiennes d'avancement, en réunions de planification, de revue et de rétrospective d'itération mensuelle et en réunions hebdomadaires de mise à jour du backlog produit.
La démarche Scrum étant pratiquée à l'initiative de l'équipe, les rôles de ScrumMaster et ProductOwner ne sont pas formellement et officiellement attribués. En quelque sorte, ces rôles sont distribués parmi des personnes auto-proclamés. Aussi, pour une équipe expérimentée de huit membres, ces responsabilités ne correspondent pas du tout à des plein-temps.
Le fait que ces rôles ne soient pas officiellement alloués permet à différents membres de s'essayer à tenir ces responsabilités. Par contre, cela est éprouvant pour ceux qui tiennent ces rôles car ils ne sont reconnus en tant que ScrumMaster et ProductOwner qu'au mérite.

Le chaos

L'espace partagé, le travail en binômes et les réunions dans la war-room impliquent un niveau non-négligeable de bruit et d'agitation. Il s'agit de l'énergie d'une équipe en pleine activité. Clairement, cela peut nuire à la concentration de certains et même donner une impression de chaos. Ces inconvénients devraient être relativisés par le gain en communication au sein de l'équipe, l'émulation, l'entraide et l'esprit d'équipe. A cause du bruit et de l'agitation engendrés, cette configuration est parfois critiquée par ceux qui ne font pas directement partie de l'équipe. Il revient alors au management d'isoler de telles équipes dans un local spacieux et insonorisé afin qu'elles ne gênent pas leur entourage.

Enfin, le quartier général partagé et les pratiques forment un folklore qui aide l'équipe à fusionner, à s'affirmer et à s'identifier.

La bande à part

Parce que l'équipe est affirmée et soudée, certaines personnes extérieures se sentent exclues. Ce sentiment est courant et doit être pris en compte[24].

24 Pour plus d'information sur ce sentiment, lire [DeMarList]. L'auteur pense même que ce sentiment d'exclusion vécu par certains est un prix à payer pour disposer d'équipes soudées et efficaces.

Le management

Le style de management doit être adapté à des équipes solidaires.

Typiquement, face à une équipe motivée et efficace, le management doit être délégatif[25] (ce qui ne veut pas dire absent). Cela se traduit notamment en accordant à l'équipe la liberté de s'auto-organisr au quotidien. En contrepartie l'équipe communique en toute transparence sur son développement. Aussi le management ne doit pas chercher à entraver tout ce qui permet à l'équipe de se construire son folklore et son identité, même si cela peut être peu apprécié par ceux qui n'en sont pas membres. Enfin, le management devrait adopter un pilotage par des objectifs d'équipe.

Objectifs individuels versus objectifs d'équipe

Comme Edward Deming l'a signalé il y a maintenant cinquante ans, le pilotage par les objectifs individuels nuit à l'efficacité de l'équipe[26]. Un tel management pousse à l'individualisme en officialisant le fait que malgré que le travail en équipe soit nécessaire, les membres sont reconnus et rémunérés individuellement sur la base de performances personnelles. Les membres sont alors tous conscients que chacun doit jongler entre la tenue d'objectifs individuels (et souvent secrets) et sa loyauté envers l'équipe.

La main courante

La stabilité de l'équipe est un objectif – mais elle n'est pas assurée. Il faut donc faciliter l'intégration de nouveaux membres et ne pas perdre les connaissances acquises par l'équipe. Là encore, la communication doit être privilégiée pour capitaliser ce que l'équipe apprend et pour le transmettre à de nouveaux intervenants. C'est pourquoi, les développeurs consignent toutes les décisions avec leur justification dans une main courante consultable et éditable dans un wiki.
Cette pratique permet de conserver les raisons qui ont amené les choix. Un tel historique argumenté facilite l'intégration de nouveaux intervenants et évite la perte d'information lorsque des membres quittent l'équipe.

La main courante est une pratique très difficile à entretenir d'autant plus que son retour sur investissement se produit que sur le long terme ou lors d'un changement d'équipe.

25 Pour plus d'information, se référer au Management Situationnel: http://en.wikipedia.org/wiki/Situational_leadership_theory

26 Deming's 14 points: Point 12.b *"Remove barriers that rob people in management and in engineering of their right to pride of workmanship. This means, inter alia, abolishment of the annual or merit rating and of management by objective"*. Pour plus d'information, voir le chapitre 3 de [Dem]. D'ailleurs, parmis les « Seven Deadly Diseases », en 3ème position on retrouve *: "Evaluation by performance, merit rating, or annual review of performance".*

La montée en charge de l'équipe

Désormais la montée en charge des capacités d'un système se fait couramment de manière itérative et incrémentale. Nous appliquons la même démarche pour l'organisation qui produit le système. Pour absorber la charge de travail, l'équipe croit par petit incrément au fur et à mesure des itérations. Nous constatons un rythme maximal de croissance de un développeur par mois pour une équipe de moins de neuf personnes. L'équipe s'est scindée en deux lorsqu'elle a dépassé le seuil critique des neuf développeurs.
Désormais, une troisième équipe intervient sur le projet. Il s'agit d'organiser les équipes selon des critères de forte cohésion fonctionnelle et de faible couplage[27] (comme pour le logiciel) et non sur des critères de spécialité ou d'architecture.

Les trois équipes partagent un même backlog produit et un même backlog d'itération. Chaque tâche du backlog d'itération est allouée à une équipe spécifiquement. Le poste d'intégration continue intègre les développements de toutes les équipes car elles contribuent toutes au même produit.

Il nous reste encore à mettre ne place les courtes réunions périodiques de synchronisation des équipes entre elles, après qu'elles aient chacune conduit leur « stand-up-meeting ».

27 Les critères de forte cohésion et de faible couplage s'appliquent de la même manière aux équipes et à leurs logiciels. En effet, les logiciels sont organisés de manière semblable aux organisations qui les ont conçus. Pour plus d'information sur cette loi, dite de Conway, se référer à [Bro].

Apprendre

Pour que l'équipe devienne une organisation qui apprend continuellement, les membres lisent des ouvrages, surfent le Web, participent à des conférences et partagent leurs découvertes. Un wiki est disponible pour publier sans contrainte.
L'essentiel de nos pratiques techniques proviennent de l'eXtreme-Programming. [Beck] reste la référence même si [BenBosMedWil] présente deux intérêts: il est en français et très pédagogique.
Nous avons puisé une grande partie de nos pratiques de gestion de projet et d'équipe de la démarche Scrum. [SchwaBee] ou [Schwab] décrivent complètement la méthode. Le premier est plus théorique alors que le second est rempli d'exemples concrets. Aussi, il donne de bons conseils pour mettre en place Scrum sur de grands projets. La lecture de [Pop1] et [Pop2] nous ont permis d'absorber certains enseignements du Lean, tels que la notion de flux continus, de Kanban et de Value-Stream-Mapping.
[DeMarList] et [Bro] sont plein d'informations concernant l'équipe.

Je trouve extrêmement difficile d'entretenir la curiosité et la soif d'apprendre d'autant plus qu'une très grande partie de ces démarches ont lieu hors temps de travail. Pourtant, le retour sur investissement de ces initiatives est impressionnant: nous n'aurions pas basculé vers le développement logiciel agile si nous n'avions pas passé autant de temps à lire.

Bibliographie, pour aller plus loin

Référence	Auteur(s)	Titre	ISBN
[Beck]	Kent Beck et Cynthia Andres	Extreme Programming Explained, embrace change	ISBN 0-321-27865-8
[Pop1]	Mary et Tom Poppendieck	Lean Software Development, An Agile Toolkit	ISBN 0-321-15078-3
[Pop2]	Mary et Tom Poppendieck	Implementing Lean Software development, From Concept to Cash	ISBN-10: 0321437381
[Schwab]	Ken Schwaber	Agile Project Management With Scrum	ISBN 0-7356-1993-X
[SchwaBee]	Ken Schwaber et Mark Beedle	Agile Software Development With Scrum	ISBN 0-13-067634-9
[DeMarList]	DeMarco et Lister	PeopleWare	ISBN-10: 0932633439
[Dem]	Edward Deming	Out of the Crisis	ISBN-10: 0911379010
[BenBosMedWil]	Jean-Louis Bénard, Laurent Bossavit, Régis Medina et Dominic Williams	Gestion de pojet eXtreme-Programming	ISBN 2-212-11561-X
[Bro]	Frederick P. Brooks, Jr	The Mythical Man-Month	ISBN 0201835959
[HunTho]	Andrew Hunt et David Thomas	The Pragmatic Programmer, from journeyman to master	ISBN020161622X
[Yip]	Jason Yip	It's Not Just Standing Up: Patterns of Daily Stand-up Meetings	http://martinfowler.com/articles/

Liste des pratiques et références

Pratiques évoquées	Origine	Bibiographie
Espace partagé	eXtreme-Programming	Sit-Together, dans [Beck]
Pair programming	eXtreme-Programming	Pair programming, dans [Beck]
War-room	eXtreme-Programming	Informative Workspace, dans [Beck]
Produit/Architecture/Processus	Lean	Standard A3
Value-Stream-Mapping	Lean	Value Stream Mapping, voir [Pop1]
Task-board	Lean	Kanban, voir [Pop1]
Daily Stand-Up Meeting	Scrum	Scrum-Meeting, dans [Schwab]
Blocking-Board	-	-
La Boîte à Idées	-	-
Burndown-Charts	Scrum	Burndown Charts, dans [Schwab]
Métriques affichées	-	-
Jalons affichés	-	-
Niko-Niko	-	Niko niko calendar, voir http://www.infoq.com/articles/agile-kanban-boards
Gizmo	-	Sémaphore d'intégration
Minuteurs et Time-Boxing	Scrum / eXtreme-Programming	Sprint, dans [Schab] Weekly and Quarterly cycles, dans [Beck] Pomodoro, voir http://www.tecnicadelpomodoro.it/docs/francesco-cirillo/2007/ThePomodoroTechnique_v1-3.pdf
Safe deliver	-	-
Bob the builder	eXtreme-Programming	Intégration continue, dans [Beck]
La main courante	-	-

Se projetter dans l'avenir

Les éditions futures de cet article pourront décrire les réunions de travail sur le backlog produit, les réunions de planification d'itération, les rétrospectives d'itération, les règles d'équipe et les pratiques techniques.

Conclusion

La dynamique du groupe est supérieure à la somme des contributions individuelles de ses membres. De même, nous avons constaté que l'efficacité de la combinaison de ces pratiques est supérieure à la somme de l'efficacité de chaque pratique prise individuellement. Après une succession de projets, l'équipe entretien toujours ces démarches alors qu'il est si facile de relâcher la discipline. L'explication de cette persévérence réside sans doute dans l'efficacité atteinte, mais aussi dans la motivation des membres, cultivée et entretenue par ce travail d'équipe.

Remerciements

Je tiens à remercier l'équipe de développement qui me supporte, joue le jeu et pousse toujours plus loin notre démarche: Nicolas Blanpain, Jean-Christophe Piau, Geoffroy Jabouley, Pierre Vince, Eric Moussy Bertrand Lesot, Frédéric Bonnot et François Brun. Ne l'oublions pas, tout ce qui est décrit ici est le résultat d'un travail d'équipe!
Je remercie aussi des membres d'autres équipes qui répondent toujours présent pour conseiller, partager leur avis et confronter nos points de vue: Jean-Pierre Decoux, Dominique David et Emmanuel Hervé.
Je remercie enfin Rémy Sanlaville et mes responsables hiérarchiques pour leurs remarques de relecture.