Formation agile

1. Qui sommes-nous ?

1.1. Pierre-Emmanuel Dautreppe

Pierre-Emmanuel est spécialiste .NET depuis 2002 et travaille actuellement comme expert technique et architecte agile dans des projets exclusivement .NET.

Il s'est dirigé vers les méthodologies agiles (et eXtreme Programming en particulier) en 2005. En commençant à travailler avec une équipe qui n'était pas sensibilisé à l'agilité, il a pu voir bon nombre d'anti-patterns appliqués par des équipes venues du milieu traditionnel. Depuis ce jour, il partage son expérience et ses recettes en essayant d'inculquer les valeurs et la culture agile.

Il donne des conférences et des formations sur la technologie .NET et sur les méthodologies Agiles.
En 2009, il a fondé l'association DotNetHub (http://www.dotnethub.be) pour promouvoir .NET et l'agilité au Bénélux et en France. Il organise également La Journée Agile (http://www.journeeagile.be) chaque année.

1.2. Norman Deschauwer

Norman Deschauwer est analyste et chef de projet. Il s'est dirigé vers l'agilité depuis 2007 et tente depuis lors de s'améliorer sans cesse. Son but est maintenant de faire partager au plus grand nombre les principes agiles et faire changer les mentalités rigides. Les méthodologies agiles ne sont pas réservées au secteur IT.

En 2009, il a fondé l'association DotNetHub (http://www.dotnethub.be) pour promouvoir .NET et l'agilité au Bénélux et en France. Il organise également La Journée Agile (http://www.journeeagile.be) chaque année.

1.3. L'association DotNetHub

C'est une communauté, fondée par six francophones, dont le but est de partager connaissances et retours d'expérience autour de 2 pôles:

- Un pôle technologique : soit .NET, ainsi que les technologies et outils liées à Microsoft.NET. Nous pouvons citer, de manière non exhaustive
  - C#
  - Windows Communication Foundation
  - Windows Presentation Foundation
  - Silverlight
  - Windows Azure
  - Windows Phone 7
  - Team Foundation Server
  - ...

- Un pôle méthodologique autour des méthodologies agiles. Nous pouvons citer, de manière non exhaustive
    - eXtreme Programming
    - Scrum
    - Lean
    - Des introductions à la méthodologie
    - Des sessions pratiques sur les méthodologies de test
    - Des sessions pour les débutants ou les experts
    - ...
- Ces sessions seront à destination
    - des participants de tout niveau, des débutants aux experts
    - des participants de tout horizon, des développeurs au chef de projet, en passant par le responsable produit
    - ...

Nos sessions sont toujours données en Français et se déroulent

Pour tout savoir sur cette association, rendez-vous sur http://www.dotnethub.be. Inscrivez-vous sur le site web pour recevoir la newsletter et être tenu au courant de nos activités.

L'association organise chaque année **La Journée Agile**. Il s'agit du seul évènement francophone traitant de l'agilité en Belgique : une journée complète de conférences, proposant plusieurs tracks en parallèles.
Ces sessions sont données par des experts reconnus, et traitent de sujets techniques (méthodologies de test, outils, intégration continue, …), et de sujet plus théoriques (retours d'expérience, principes agiles, …).

Pour tout savoir sur cet évènement, et vous y inscrire, rendez-vous sur http://www.journeeagile.be.

2. Introduction

La définition de l'agilité n'est pas chose simple. Il ne faut pas confondre le terme agile avec une méthodologie telle que « Scrum » ou encore « Extreme Programming ». L'agilité est une façon de penser, une façon de faire. C'est un ensemble de pratiques qui conviennent dans un contexte précis. Un synonyme d'agilité serait le bon sens.
Scrum et XP sont des méthodes agiles, mais elles n'expliquent et ne résument pas à elles seules l'agilité.
L'agilité permet de voir les choses différemment, elle permet de se poser des questions tout le temps sur l'utilité de ce qu'on fait et la manière dont on le fait.
L'agilité vous permet de revenir aux sources et de manière incrémentale tenter d'atteindre la perfection. Perfection qu'il est préférable de ne jamais atteindre ou plutôt ne jamais croire qu'on l'a atteinte.
Retenez toujours ceci : « le contexte est toujours le paramètre le plus important ».
L'agile n'est pas réservé à une élite, l'agile n'est pas non plus réservé qu'au projet R&D ou en pure régie. L'agilité au forfait fonctionne aussi.

Nous étudierons quelques pratiques et adopterons la vision de Scrum pour une partie des processus, mais ce n'est pas la seule, c'est certainement la plus populaire, mais pas pour autant la meilleure pour votre contexte.

> "Une méthode agile est une approche itérative et incrémentale, qui est menée dans un esprit collaboratif avec juste ce qu'il faut de formalisme. Elle génère un produit de haute qualité tout en prenant en compte l'évolution des besoins des clients"
>
> **Veronique Messager Rota**, dans Gestion de projet : Vers les méthodes agiles

3. Agile Manifesto

Le manifeste agile (www.agilemanifesto.org) est un regroupement de pratiques permettant de développer du logiciel autrement. Cette charte a été signée par les plus grands du monde informatique en 2001.
Les valeurs sont celles-ci :

- **L'Individus et les interactions** sont plus importants que les processus et les outils
- **Un logiciel qui fonctionne (bien)** est plus important que la documentation
- **La collaboration du client** est plus importante qu'une négociation du contrat
- **S'adapter aux changements** est plus important que de suivre le plan

> ***That is, while there is value in the items on the right, we value the items on the left more.***

4. Terminologie

Pour comprendre certains passages il est indispensable de s'accorder sur certaines définitions.

4.1. Itération

Une itération est une fenêtre de temps pendant laquelle l'équipe va développer ce qu'elle a estimé dans cette durée définie. Cette durée n'excède pas 30 jours calendrier, mais peut descendre à une semaine (5J) dans certaines équipes.
Après une itération le produit peut être livré pour test ou en production.
En scrum une itération est appelée « Sprint », le mot semble mal choisi pour celui qui comprend qu'il faut se dépêcher à travailler.

5. Scrum

5.1. Historique

Scrum est issu du RAD (rapid application development) dans les années 90. La principale personne qui va promouvoir Scrum est Jeff Sutherland cosignataire de l'agile manifeste en 2001.

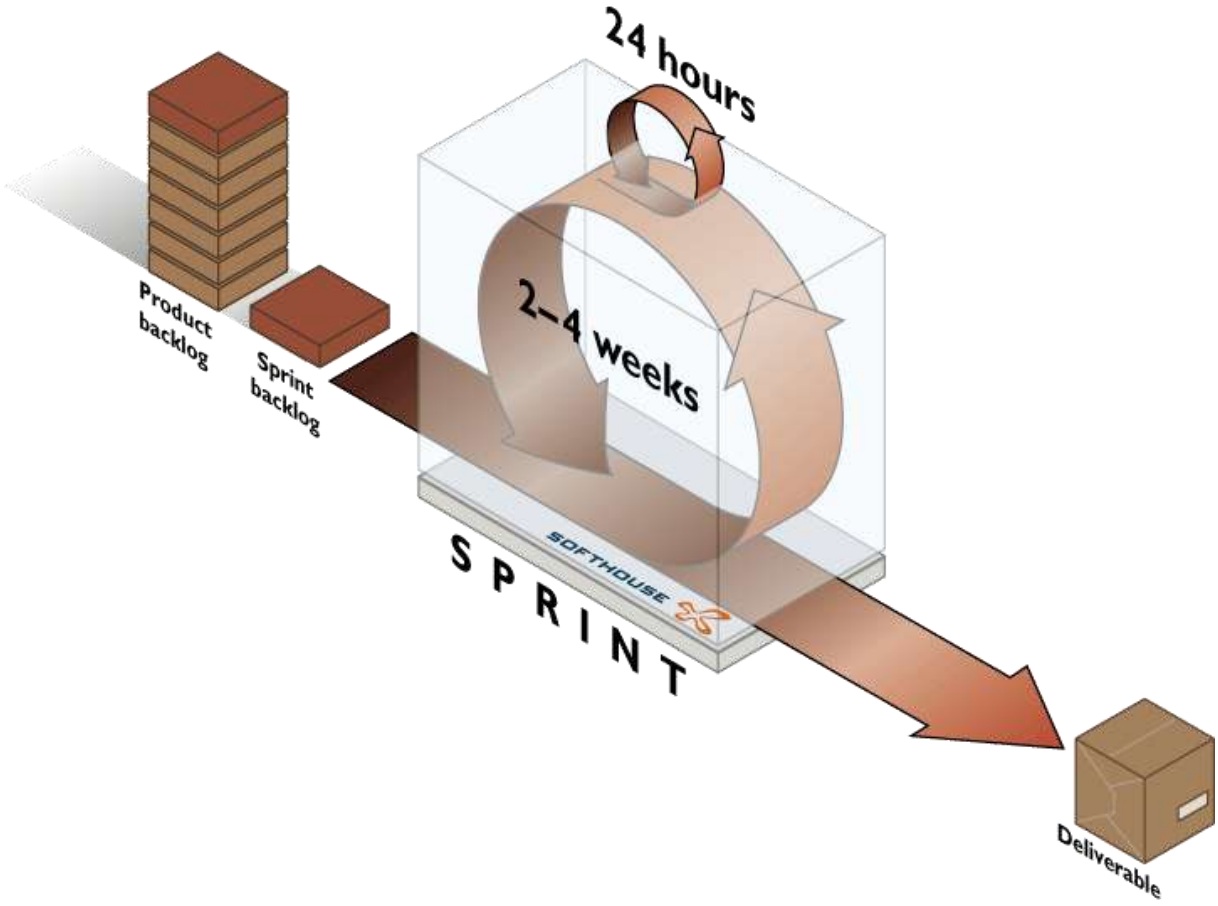
5.2. Scrum Alliance

5.3. Avant de débuter

Scrum est simple du prime abord, mais le mettre en place correctement peut s'avérer révélateur de beaucoup de changements inattendus.
Scrum, comme toute méthode agile est un accélérateur. Elle mettra en évidence le meilleur, comme le pire et ce très vite de par sa cadence.
Il ne faut pas confondre ici vitesse et précipitation. Les managers ne doivent pas comprendre ici que cela va accélérer les développements, on ne va pas faire rentrer 12h de travail dans 8h.

5.4. Scrum : synthèse

- 3 rôles
  - Scrum Master
  - Product Owner
  - Equipier
- 3 réunions
  - Daily scrum (stand-up)
  - Planning Game
  - Revue de sprint
- 2 listes
  - Product backlog
  - Sprint backlog



5.4.1. Scrum master (SM)

Le SM peut être assimilé au rôle du chef de projet s'il a bien compris qu'il n'est pas chef. Il n'y a aucune hiérarchie dans un projet agile. Il y a une hiérarchie dans chaque société, mais au sein même du projet, il y a une équipe.
Le rôle du SM est de veiller à la méthode, il est là pour aider ses coéquipiers, c'est un facilitateur. C'est aussi un animateur lors de certaines réunions.
C'est un coach pour les autres.

5.4.2. Product owner (PO)

Le PO est considéré comme le client. Il peut être le client ou un représentant du client dans l'équipe (analyste). C'est une personne qui dispose des réponses ou des moyens de se les procurer et si possible il doit également avoir un pouvoir de décision.
Un nouveau développeur dans l'équipe n'est peut-être pas la bonne personne, mais le PDG du client n'est certainement pas une meilleure solution. Car au-delà des réponses et décisions, il se doit d'être disponible pour l'équipe.
Il a la vision de la solution et des buts à atteindre.

5.4.3. Coéquipiers

Toute personne dans l'équipe qui n'est ni SM ou PO est un coéquipier.

5.4.4. Taille de l'équipe

La littérature et l'expérience nous apprennent qu'il ne faut pas plus de 9 personnes par équipe.

5.4.5. Scrum de Scrum

Au-delà de 10 personnes on peut séparer les équipes et dans chaque équipe il doit y avoir un SM et un PO qui peuvent se partager plusieurs équipes si cela les maintient toujours disponible.
Le Scrum de scrum est également appliqué dans le cas où les équipes sont physiquement distantes (USA/Europe par exemple). Le principe reste le même chaque équipe contient les 3 rôles et les équipes communiquent l'avancement entre elles via les PO et SM de chaque équipe.

5.4.6. Daily scrum

Le daily scrum ou stand-up meeting est la réunion quotidienne qui permet d'avoir une vision claire de l'avancement de l'itération et donc du projet, mais aussi de déceler les difficultés que l'équipe peut rencontrer.
De manière formelle, l'idéal est de répondre à trois questions :

- Qu'ai-je terminé la veille ?
- Quels sont les blocages que je rencontre ?
- Que vais-je commencer/finir aujourd'hui

Il faut voir cette réunion comme une place de marché, on expose ce qu'on a et on se sert dans ce que les autres exposent.
Mais il ne faut pas sortir de cette réunion avec un blocage que personne ne traite. Même si ce problème n'est pas résolu immédiatement, il est répertorié et sera traité.
Elle se fait debout et ne doit durer que 10 minutes maximum.
Ce n'est pas du reporting pour le chef de projet. Si le seul but est de satisfaire le chef de projet c'est qu'il y a un problème dans cette réunion. Il y a d'autres moyens/outils prévu pour le reporting.

Qui vient à cette réunion ? Toute l'équipe.

5.4.7. Planning game

Cette réunion intervient normalement une fois par itération, et regroupe les différents intervenants du projet (client, chef de projet, développeur, …) => l'équipe.
Il s'agit d'une réunion de planification, au cours de laquelle :

- Le PO va expliquer les fonctionnalités qu'il désire
- Il va également fixer la valeur business de ces différentes fonctionnalités
- Les développeurs vont alors discuter sur ces différentes tâches. Leur but est de s'assurer qu'ils comprennent correctement le besoin, et de proposer des solutions techniques/fonctionnelles et le coût associé
- En fonction de ce coût et de la valeur business, le client peut alors fixer la priorité entre les tâches, et ainsi fixer l' « iteration backlog » (ie le panier de tâches) de l'itération.

Ce planing game peut être effectué de beaucoup de façons différentes. La façon la plus connue est sans doute le « planning poker » au cours de laquelle les différents participants vont noter chaque tâche à l'aide d'une carte.

Il est important de noter que cette notation est agnostique du temps de réalisation. On parle en général de « nombre de points », correspondant à une complexité relative entre les différentes tâches : autrement dit, une tâche de « 8 » sera deux fois plus complexe qu'une tâche de « 5 ».
On va traditionnellement utiliser les valeurs de la suite de Fibonacci, soit 1, 2, 3, 5, 8, 13, 21, … Le choix d'une échelle non linéaire mets davantage en lumière les incertitudes de l'estimation : plus une tâche est « grosse », moins son estimation est précise.

Pourquoi utiliser des points et non chiffrer en temps ?
Cela a deux avantages principaux : dans la relation entre l'équipe de développement et le client, cela permet de réduire les tensions liées à la présence de temps non productif dans le projet, que ce soit du temps administratif (eg réunion) ou du temps de non qualité (résolution de bug par exemple). Si on réalise 30 heures de tâches fonctionnelles en 40 heures de travail effectif, le malaise est évident.
De plus cela permet d'avoir une échelle constante tout au long du projet. En effet, quel que soit le niveau des personnes travaillant dans les équipes, la complexité relative des tâches ne va pas évoluer. En revanche, c'est la vélocité (c'est-à-dire la quantité de tâche réalisée par l'équipe) qui va évoluer avec le temps.

Cette réunion est très importante et la réussite se joue en partie lors de cette réunion.

5.4.8. Terminé (Done)

Ce mot est d'une importance capitale. En commençant votre premier planning game, il est important de clarifier ce que signifie une tâche « terminée ».
La définition doit sortir de l'équipe elle-même. Elle doit se construire tout au long des itérations. Car la notion de terminé peut évoluer si le projet s'étale sur plusieurs mois/années et si des besoins nouveaux sont identifiés.
Exemple de terminé pour une équipe, ma tâche est considérée comme terminée si :

- Le besoin fonctionnel est couvert
- Le client a testé et approuvé
- Il y a des tests automatisés
- La documentation est faite

Cette liste va s'enrichir itération après itération, quelques semaines plus tard voici la liste :

- Le besoin fonctionnel est couvert
- Le client a testé et approuvé
- Il y a des tests automatisés
- La documentation est faite
- L'application répond en moins de 2 secondes
- Une revue de code a été faite
- …

Cette liste est propre à chaque équipe, projet, client, …
La notion de terminé se retrouve au quotidien lors du daily scrum, lorsqu'un équipier dit : « j'ai terminé cette tâche » cela signifie qu'il a appliqué toute la définition du mot « terminé » pour sa tâche.

5.4.9. Revue de sprint

La revue de sprint est une réunion essentielle dans le mode agile. L'agilité permet d'avancer de manière incrémentale et itérativement. La revue de sprint permet d'appliquer ce que l'on appelle l'amélioration continue. C'est une réunion où l'équipe va exposer ses points faibles et prendre des mesures correctives.
Il est important que les personnes s'approprient les changements. On parle d'équipe autogérée, mais croire que la bonne volonté suffira à résoudre les problèmes est une illusion.
Il faut donc sortir de cette réunion avec quelques points d'améliorations et surtout une personne qui se sera proposée à soutenir ces changements. Cela ne veut pas dire qu'elle sera la seule à s'impliquer, mais son rôle sera de mettre tout en œuvre pour appliquer les changements.
Il existe une multitude de façons d'animer la revue de sprint, et c'est au SM de préparer cette réunion.
Le but est de sortir de cette réunion avec des solutions et un responsable du suivi de ces solutions.
Il ne faut toutefois pas espérer résoudre tout d'un coup, il faut éviter de se disperser.
Mettez des priorités sur vos actions et n'en prenez que les trois plus importantes. C'est vous et votre équipe qui verrez le meilleur choix à faire.

5.4.10. Product Backlog

C'est la liste MACRO des fonctionnalités à implémenter dans l'application. Cette liste est élaborée avant ou en début de projet, mais comme pratiquement tout en agile, elle est susceptible d'être modifiée en cours de route.
Tous les éléments sont priorisés avec le client. Cela donne dès le départ du projet les priorités des semaines/mois à venir.
Les éléments sont des user stories. Ses user stories ne vont pas dans le détail, mais permettent d'apprécier le travail. Elles peuvent déjà avoir fait l'objet d'estimations. Estimations qui seront revues en fonction des détails qui seront énoncés lors des différents planning game.
Les estimations de ces Macro-tâches seront aussi précises que de détails disponibles au moment de l'estimation.

5.4.11. Sprint Backlog

Le sprint backlog est la liste des tâches à faire pendant le sprint (itération) courant. Le sprint backlog est créé pendant le planning game. Cette liste est une sous-liste plus détaillée du product backlog.
Cette liste est aussi priorisée avec le client. Toutes les listes de tâches en mode agile est priorisée. Le contenu des tâches sur le sprint backlog ne laisse plus de questions en suspens.
Cela signifie que lors du planning game, l'idéal est de ne laisser aucune zone d'ombre dans le contenu d'une tâche qui est planifiée.

5.5. Les outils

Les outils se veulent d'accès direct pour favoriser la communication et l'aspect visuel prime sur l'outillage informatique. Rien de tel qu'une feuille de papier, un stylo en lieu et place d'un outil informatique très performant mais coûteux et nécessitant l'outil sur tous les postes et un accès.
L'avantage d'un panneau accroché au mur est sa visibilité et sa compréhension visuelle.
Sur un tableau d'avancement d'itération, même un enfant de 10 ans pourra voir si l'équipe est en retard ou pas. Voici quelques exemples d'outils qui deviennent indispensables de mettre en place.

5.5.1. Vignettes autocollantes (Post-it)

Ces vignettes ont pris le nom de la marque comme passe-partout, nous parlerons donc de Post-it, mais il ne faut pas y voir une promotion de notre part.
Le post-it est donc le moyen visuel de créer une tâche. Celle-ci sera collée sur le mur, un tableau ou un Kanban. Le plus important c'est qu'il reste à portée de tous. Il faut donc éviter que les post-it s'accumulent dans le bureau du chef de projet !
Les post-it contiennent les infos nécessaires pour identifier la tâche. On ne sait pas tout écrire sur un post-it, le contenu détaillé de la tâche doit être stocké sur un outil (Jira, Word, Wiki, feuilles papier). Le Post-it peut également contenir des informations telles que l'estimation de complexité (story points), la business value,…

5.5.2. KANBAN

La prononciation importe peu, le tout est de comprendre que cet outil est à la fois un indicateur d'avancement et un état des lieux. Cela correspond à la chaîne de production.
Dans la pratique c'est un tableau en papier/carton sur lequel l'équipe dispose les tâches à faire pour délivrer le produit. Il contient au minimum 3 colonnes (à faire, en cours, terminé) dans lesquelles les tâches vont suivre le flux création.
Mais le kanban peut être beaucoup plus complet. Certaines équipes ont 10 colonnes au tableau et de la même manière les tâches passent de colonnes en colonnes (gauche à droite) jusqu'à une livraison possible en production.
Vision : à tout moment vous pouvez voir ce qui est en cours, ce qu'il reste à faire et ce qui est terminé (à tester, à livrer,..)
Il est important que les membres de l'équipe ne travaille pas sur plusieurs tâches simultanément. Cela permet d'augmenter la concentration et diminuer les perturbations, il en résultera normalement une augmentation de la productivité.

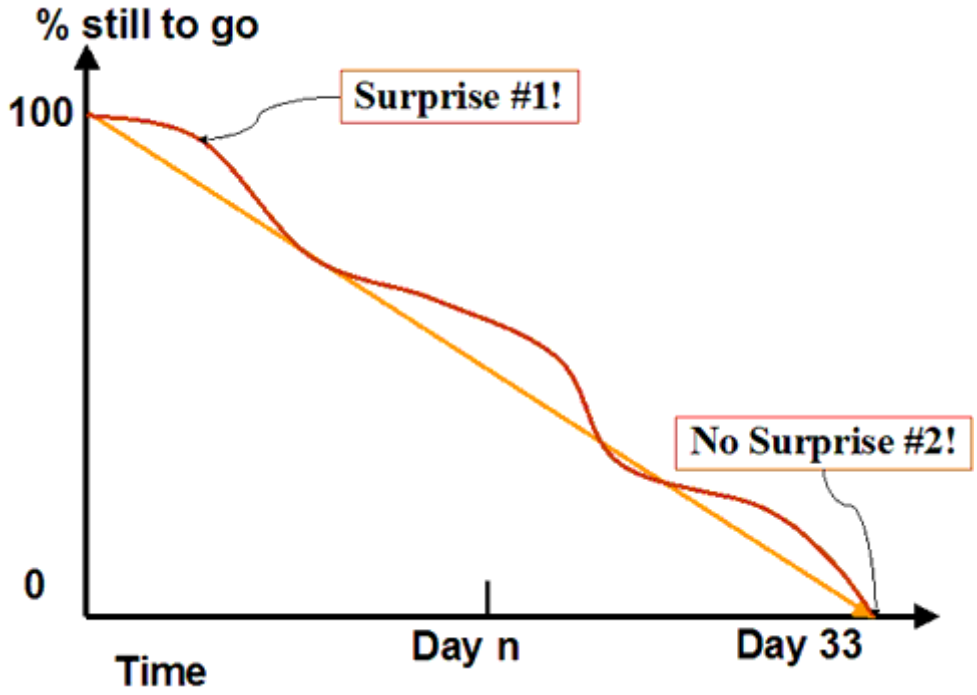
Exemple de Kanban :

Celui-ci ne contient que trois colonnes, mais celles-ci suffisent à l'équipe pour mener à bien leur projet.

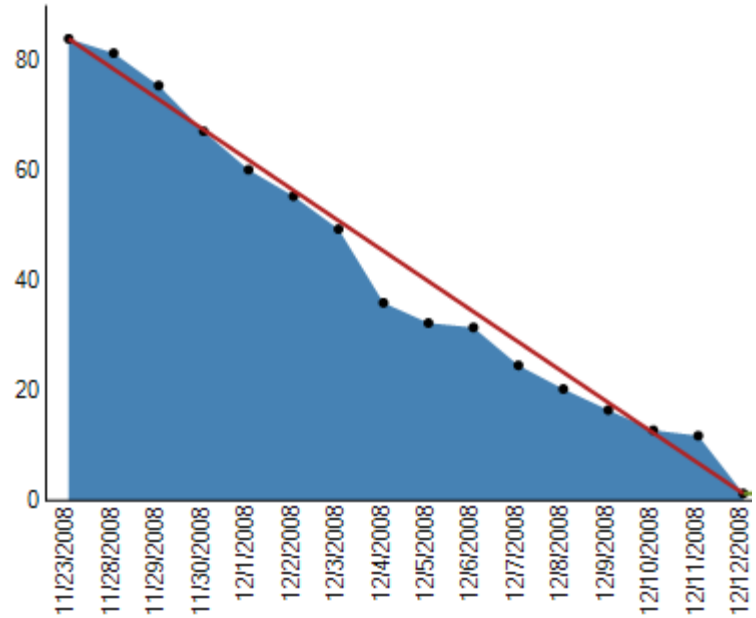
Autre exemple de Kanban plus complexe

5.5.3. Burndownchart (BDC)

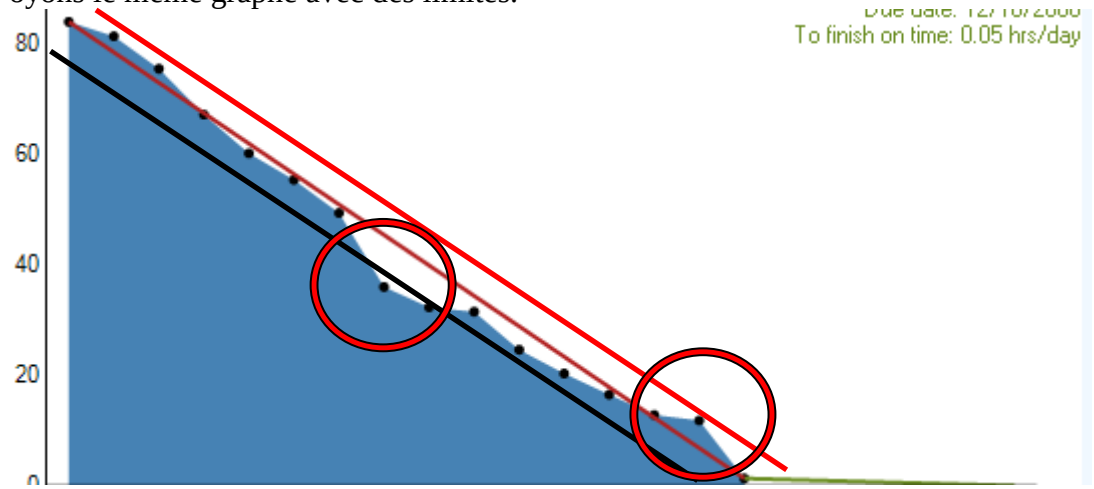
Le BDC est un graphique donnant une vision du travail restant à faire sur une itération.



Cela représente le nombre de points sur le temps disponible de l'itération.



Comment le mettre à jour ? lors du stand-up par exemple le tableau est mis à jour, cela permet de le maintenir au quotidien.
En ordonnée on retrouve les jours ouvrés et en abscisse les points de l'itération.
Cette exemple nous montre que 80 points sont à faire pendant l'itération. En 30 jours l'équipe doit donc terminé 80/30 point par jour pour suivre la tendance moyenne. On peut noter que sur ce graphe, le début était légèrement en retard, mais l'équipe a réussi à rattraper le retard et même à prendre une petite avance en milieu de tableau.
Voir un tableau tel quel n'est pas indicateur pour en avoir une bonne lecture, il faudrait apposer des limites dans lesquels les retards/avances sont tolérés.
Si la zone bleu est au dessus de la tendance, il y a un retard, et inversement pour l'avance.
Voyons le même graphe avec des limites.



On peut noter qu'à la moitié de l'itération, l'équipe était « trop » à l'avance. Le tout est de convenir d'une mesure lors de ce genre de dépassement ? Que fait-on en cas de retard ou d'avance ? Et surtout lors de la rétrospective, il faut aborder ce point et le comprendre ?
Un Dev est rentré de congé ? Le tâche était plus simple que prévue, un refactoring est à la base de ce gain de temps ?
On peut noter aussi que vers la fin l'équipe a frôlé la zone rouge. Le même raisonnement est à appliquer à ce retard : pq ? comment ? éventuellement qui ? il faut partager les infos pour ne plus le reproduire.