

Secteur Tertiaire Informatique
Filière « Etude et développement »

Séquence : Travailler avec une démarche de développement

Agilité et Scrum

Apprentissage

Mise en Pratique

Evaluation

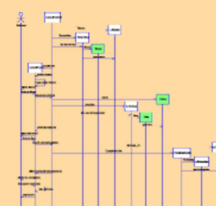
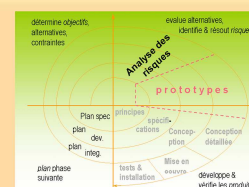
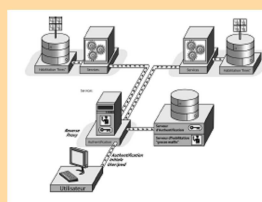


TABLE DES MATIERES

Table des matières	2
1. Gérer un projet de développement.....	5
1.1 Gérer un projet ?	5
1.2 Les méthodes classiques en informatique et leurs résultats.....	6
2. Méthodes agiles de gestion de projet de développement informatique	9
2.1 Agilité ?	9
2.2 Le Manifeste Agile.....	9
2.3 XP et Scrum.....	13
3. Scrum	13
3.1 Principes et vocabulaire Scrum	13
3.2 Equipe, Artefacts et Cérémonies Scrum.....	14
3.3 Le Scrum Master.....	14
3.4 Le Product Owner Scrum	15
3.5 L'équipe de développement	15
3.6 Sprint Zéro, Roadmap, User Stories, Personas et Product Backlog	17
3.7 La vie d'un sprint.....	22
3.7.1 Scrum Board ➔ <i>Transparence</i>	22
3.7.2 Backlog Grooming - Affinage de Backlog.....	24
3.7.3 Sprint Planning Meeting - Planification du Sprint	25
3.7.4 Daily Scrum Meeting - Mêlée quotidienne ➔ <i>Transparence et Adaptation</i>	26
3.7.5 Revue de Sprint - Sprint Review	27
3.7.6 Rétrospective - Sprint Retrospective Meeting ➔ <i>Adaptation</i>	29
3.7.7 Pour résumer, les cycles Scrum en image.....	30
3.8 Outils d'aide à la gestion de projets Scrum.....	33
3.8.1 IceScrum	34
3.8.2 JIRA Agile.....	36
3.8.3 Autres produits	36
3.9 Scrum et contractualisation	37
3.10 En guise de bilan	38

Objectifs

A l'issue de cette séance d'apprentissage, vous serez à même :

- de vous repérer dans une organisation mise en place autour d'un projet de développement informatique et d'identifier la méthode de conduite de projet adoptée ;
- de collaborer efficacement au sein d'une équipe adoptant une méthode « Agile » et plus particulièrement « Scrum ».

Pré requis

Avoir des notions sur la problématique d'un développement en informatique de gestion et sur les différentes tâches nécessaires pour faire aboutir le projet (programmation, design de l'interface homme-machine, stockage des données...).

Outils de développement

Aucun. La découverte et la mise en œuvre de IceScrum sur un projet d'étude peut être envisagée.

Méthodologie

Ce support d'apprentissage se propose d'exposer les grands principes des méthodes Agiles de gestion de projet, et en particulier de Scrum ; vous serez amené à étudier des documents divers, à effectuer des recherches individuelles ou en groupe et à vous entraîner sur certaines techniques, sans autres outils que papiers, crayons, feutres, Post it, paperboard, traitement de textes ou éditeur de slides.

Mode d'emploi

Symboles utilisés :



Renvoie à des supports de cours, des livres ou à la documentation en ligne constructeur.



Propose des exercices ou des mises en situation pratiques.



Point important qui mérite d'être souligné !

Ressources

Présentation générale : z-presentation-agile.pptx ; z-gestion-projets-informatiques-intro-scrum.pptx

Mises en pratique : P-agilite_et_scrum.pdf (P-agilite_et_scrumQCM.pdf, S-agilite_et_scrumQCM.pdf, P-agilite_et_scrumUS.pdf)

Pistes d'animation en groupe à l'aide de serious games :

- Estimation : « population de pays »

- Auto-organisation : « compter de 1 à 10 », « chaises non-musicales », voire « puzzle » (Agile à l'échelle)

Agilité et Scrum

Afpa © 2015 – Section Tertiaire Informatique – Filière « Etude et développement »

- ...

Lectures conseillées

<http://agilemanifesto.org/iso/fr/>

Livre Blanc Smile : « z-LB_Smile_Methode et Agilite.pdf » (52p)

Guide officiel Scrum en français : « z-Scrum-Guide-FR.pdf » (19p)

Présentation officielle en français de Scrum Alliance : « z-corescrum-fr-v1.1.pdf » (12p)

Présentation en français de Mountain Goat Software : « z-French-Redistributable-Intro-Scrum.ppt » (42 slides)

Présentation en français 'Scrum Primer' : « z-fr_scrumprimer20.pdf » (22p)

Exemple de contrat Agile : z-Contrat_Agile_V1.1.pdf

Et aussi :

« Scrum et XP depuis les tranchées » : « z-ScrumAndXpFromTheTrenches_French.pdf » (160p)

« Kanban et Scrum - tirer le meilleur des deux » : « z-KanbanAndScrum-French.pdf » (128p)

« Lean Primer » : « z-Lean_Primer_fr.pdf » (46p)

« Un guide de Survie à l'Adoption... » : « z-un_guide_de_survie_.pdf » (91p)

Sites Web :

- <http://www.qualitystreet.fr> (Jean-Claude Grosjean)
- <http://www.aubryconseil.com/> (Claude Aubry)
- <http://www.agiliste.fr/fiches> (Florent Lothon)
- <http://scrumtrainingseries.com/>
- <http://www.planningpoker.com/>

Vidéos en ligne :

- <https://www.youtube.com/watch?v=kZTLIWkxFN4> (Scrum pour les nuls - 7mn)
- <https://www.youtube.com/watch?v=cQzefbh10sU> (Les 3 rôles de Scrum - 5mn)
- <https://www.youtube.com/watch?v=3qMpB-UH9kA> (la gestion de projet Agile en 2 mots - 16mn)
- <https://www.youtube.com/watch?v=XICcwvMioSQ> (Scrum - une méthode Agile - 28mn)
- <https://www.youtube.com/watch?v=70sDPSR5j1Y> (Présentation Scrum à Agile Tour 2012 - 24mn)

Agilité et Scrum

Afpa © 2015 – Section Tertiaire Informatique – Filière « Etude et développement »

1. GERER UN PROJET DE DEVELOPPEMENT

1.1 GERER UN PROJET ?

Programmer, c'est bien beau ; développer une application, ça se fait, surtout si on développe pour soi-même. Mais qu'en est-il quand on développe à plusieurs, et au profit de tiers dont les exigences sont proportionnelles à la facture qu'on leur adresse ?

Pour poser la problématique, on peut oser un parallèle avec la construction d'un bâtiment. Dans l'histoire des trois petits cochons, Nif-Nif, Naf-Naf et Nouf-Nouf ont pu construire seuls leur maison car le chantier n'était pas très compliqué (surtout pour celui des trois qui n'a utilisé qu'une botte de paille) ; parions que celui qui a entrepris une construction en dur a déjà dû organiser un peu son travail. Plus proche de notre monde, un bon bricoleur commencera par bâtir les plans de la maison de ses rêves avant d'attraper sa truelle et un artisan-maçon discutera longuement du projet avec son client avant d'engager les travaux et de solliciter d'autres artisans spécialisés. Quant à la construction d'une centrale nucléaire ou d'un hôpital, qui nécessite l'intervention simultanée de centaines d'ouvriers et de machines ainsi que la livraison continue des matériaux nécessaires, on peut imaginer qu'elle est murement organisée et pilotée par toute une équipe.

Quoi de commun dans tout cela ? Et quand parle-t-on de projet ?

Chacun des cas correspond bien à un **projet** en ce sens que chacune des constructions est **unique, n'existait pas avant et ne se reproduira jamais à l'identique** ; en effet même si Areva enchaîne la construction de centrales nucléaires, chaque cas est spécifique en terme de puissance, de situation géographique, de normes locales de sécurité... (et parions que les cochons paresseux ne recommenceront pas les mêmes erreurs !).

💡 En d'autres termes, **un projet, c'est toujours nouveau, inconnu.**

Si un bon bricoleur peut construire seul sa maison, il vaut mieux pour lui qu'il ne soit pas trop pressé... L'artisan-maçon s'arrangera pour faire intervenir le plombier, l'électricien ou le carreleur en même temps que lui-même terminera ses travaux de maçonnerie de manière à réduire le délai de livraison (et gageons que le plombier ne devra pas effectuer des tranchées dans le carrelage tout neuf).

💡 En d'autres termes, quand un projet devient **complexe en nombre de tâches et d'intervenants**, il nécessite une **organisation des activités** visant à réaliser les tâches en **parallèle**, dans la mesure du possible, de manière à **réduire le délai de livraison**. Il s'agit là de la **gestion du projet**.

La gestion d'un projet peut elle-même devenir tellement problématique (hôpital, centrale nucléaire, tunnel sous la Manche...) que des méthodes et

Objectifs

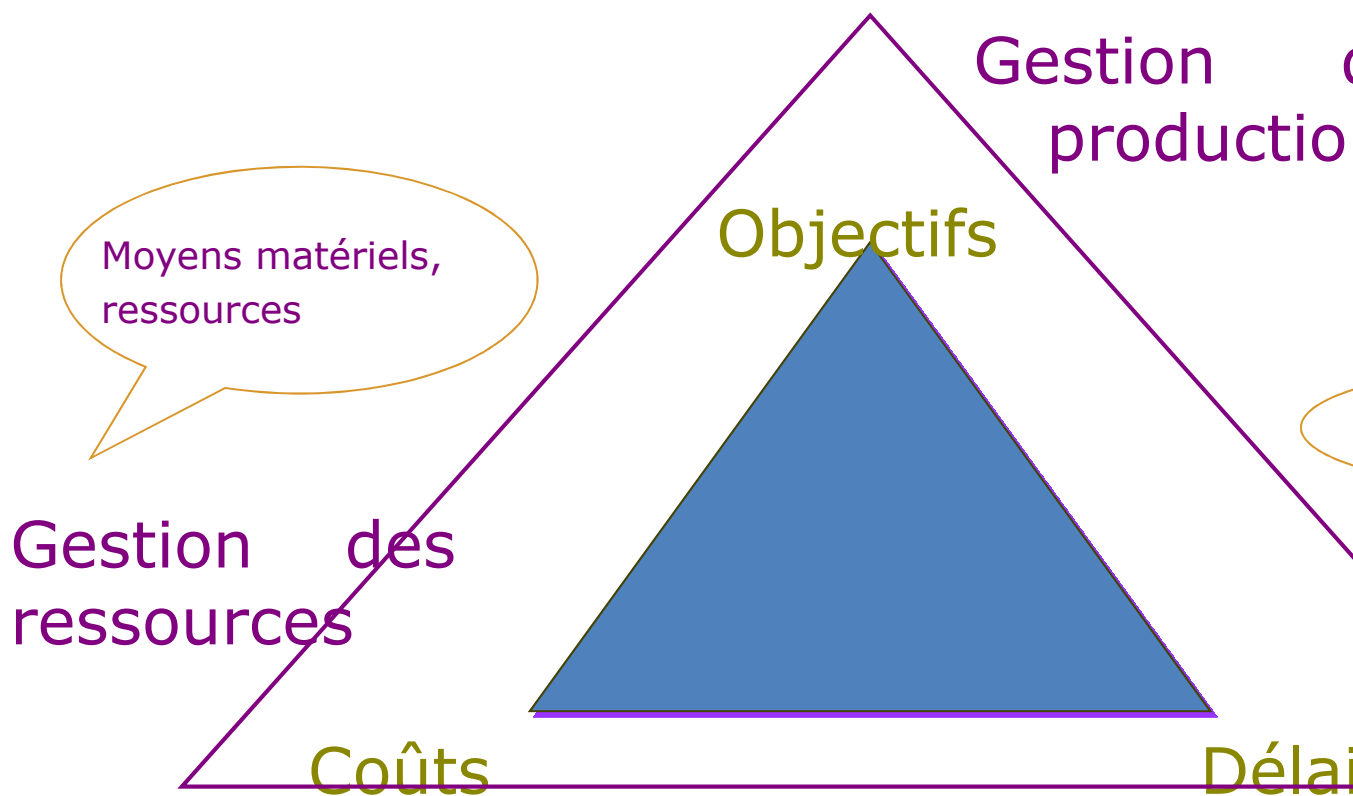
A

outils sont apparus et ont évolué au fil du temps, des expériences et des besoins. Il existe même des normes internationales concernant ce domaine comme la norme ISO 1006-2003 qui définit un projet comme :



« un processus **unique**, qui consiste en un **ensemble d'activités coordonnées** et maîtrisées comportant des **dates de début** et de **fin**, entrepris dans le but d'atteindre un **objectif conforme à des exigences** spécifiques telles que les contraintes de **délais**, de **coûts** et de **ressources** ».

Un projet est tiraillé en permanence entre ces **trois contraintes, coûts, délais et qualité** (le « triangle projet »), ce qui nécessite une **gestion des ressources** (matérielles, humaines), une **gestion du temps** (planning prévisionnel, suivi des réalisations) et une **gestion de la production** (mesure du réalisé et du restant à faire) ; ces trois activités (le « triangle de gestion du projet ») doivent être **déroulées en continu** de manière à **mesurer** en permanence où l'on en est, de **détecter des anomalies** par rapport aux prévisions afin de **prendre des mesures correctives** et de **prévoir** la suite avec un minimum d'incertitude.



1.2 LES METHODES CLASSIQUES EN INFORMATIQUE ET LEURS RESULTATS

Concernant les projets de développement informatique, « l'industrialisation » de la production de logiciels a bien entendu fait émerger des méthodes de gestion de ces projets.

Agilité et Scrum

Afpa © 2015 – Section Tertiaire Informatique – Filière « Etude et développement »

Historiquement, on a tout d'abord identifié des étapes-types qui font l'objet de validations successives :

- **définition des besoins** dans un Cahier des Charges contractuel et signé,
- **conception globale** de l'application,
- **conception technique** (écrans, états imprimés, bases de données...),
- **programmation** proprement dite, **tests** et corrections éventuelles
- puis, enfin, **livraison** au client (« recette »).

Le client est appelé « **maître d'ouvrage** » ou MOA, le fournisseur, « **maître d'œuvre** » ou MOE, et leurs responsabilités sont clairement définies ; on adjoint parfois une assistance au client (AMOA) quand les compétences techniques ou le temps lui manquent.

L'équipe de développement est typiquement dirigée par un **Chef de Projet** qui effectue les **estimations de charges** de travail et **l'attribution des tâches** aux différents développeurs, la **planification prévisionnelle** et le **suivi** du déroulement du projet.

On parle de « **méthodes prédictives** », de modèles ou « **cycles de vie** » « **en cascade** » ou « **en V** ». Ces méthodes sont caractérisées par un **long délai entre l'expression des besoins et la livraison** (on parle « **d'effet tunnel** »). Et durant ce laps de temps, les choses ont pu changer (nouveaux besoins, nouvelles règles de gestion, nouvelles contraintes réglementaires...). Et ce n'est qu'au bout de ce tunnel, à l'issue du développement, que le client peut enfin se rendre compte s'il a bien été compris par son prestataire (ah ! les aléas de la communication humaine...).

Des études statistiques ont montré la réelle fragilité de ces méthodes de gestion de projet. Ainsi le Standish Group identifie que, en 2014 :

- **16,2% des projets sont des succès,**
- **52,7% des projets sont des succès avec écarts de budgets, délais ou qualité,**
- **31.1% des projets sont des échecs ou ont été abandonnée en cours de route.**

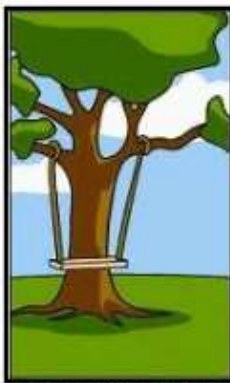
Pour ces deux dernières catégories, le **budget moyen tourne autour de 190% du budget prévu**, soit près du double ! Et le **dépassement moyen des délais**, tous types de projets confondus, est dans les mêmes ordres de grandeur.

De plus, il s'avère que certaines fonctionnalités définies dans le cahier des charges ne sont jamais utilisées (on a donc développé certaines parties pour rien).

☛ Ces études montrent que les causes principales ne relèvent pas de difficultés techniques ; **les échecs sont dus aux changements dans les besoins et au manque de compréhension entre les acteurs.**



Comment le client a exprimé son besoin



Comment le chef de projet l'a compris



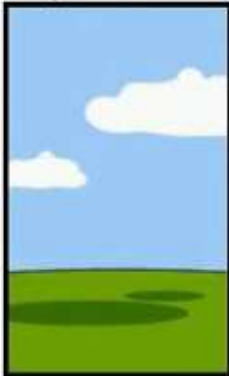
Comment l'ingénieur l'a conçu



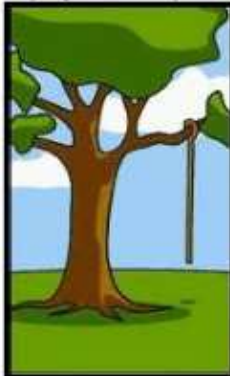
Comment le programmeur l'a écrit



Comment le responsable des ventes l'a décrit



Comment le projet a été documenté



Ce qui a finalement été installé



Comment le client a été facturé



Comment la hotline répond aux demandes



Ce dont le client avait réellement besoin

Face à ces difficultés, les années 1990 à 2000 on vu apparaître de nouvelles méthodes et de nouveaux outils permettant de **donner à voir au client tôt dans le projet** de manière à réduire cet effet tunnel pour rectifier au plus tôt. C'est le domaine du « **Rapid Application Design** » (ou RAD) utilisant des outils de maquettage graphiques comme Visual Studio et de la **méthode RUP**, adaptée à des développements « orientés objet » utilisant la modélisation UML.

2. METHODES AGILES DE GESTION DE PROJET DE DEVELOPPEMENT INFORMATIQUE

2.1 AGILITE ?

Et les « **méthodes agiles de gestion de projet** » dans tout cela ?

L'origine historique remonte au milieu du XX^e siècle avec des expérimentations industrielles par Toyota et Toshiba au Japon (Lean Management et Kanban). Mais ce n'est que depuis les années 2000 que l'on applique ces méthodes au développement de logiciels, en particulier avec **SCRUM** et **eXtreme Programming** (ou XP).

Les points communs sont avant tout :

- **L'acceptation du changement**, même tard dans le projet (le changement n'est plus un incident perturbateur) ;
- **La livraison continue** sur des **cycles courts** (2 à 4 semaines) de manière à valider au fur et à mesure (feedback et validation rapides) ;
- **La priorisation des développements par la valeur Métier à livrer** (le plus utile pour le business en premier) ;
- Un engagement sur les moyens mis en œuvre et non plus sur des résultats escomptés ;
- Un **engagement par cycle** et non plus sur l'ensemble du projet (en fin de cycle, on se demande 'Stop ou encore ?' et on arrête quand cela suffit).

Ces méthodes ont pu voir le jour grâce à l'apparition d'outils informatiques permettant en particulier **l'intégration continue** des développements et **le déploiement aisé** du logiciel, la **gestion fine des versions** du logiciel ou le **feedback continu** des utilisateurs.

 **En d'autres termes, l'agilité c'est quelques grammes de souplesse dans ce monde en mouvement.**

2.2 LE MANIFESTE AGILE

Le top départ des méthodes Agiles de gestion de projet de développement informatique est marqué par la publication en 2001 du « **Manifeste pour le développement Agile de logiciels** » signé par une quinzaine de développeurs expérimentés, innovants et fatigués des méthodes classiques ( <http://agilemanifesto.org/iso/fr/>).

Cette « profession de foi » repose sur 4 valeurs et 12 principes relativement abstraits ; Scrum et XP en sont des concrétisations.

Pour les valeurs, il s'agit de valoriser :

- **Les personnes et leurs interactions**, donc la confiance et le dialogue ;
- **Des logiciels opérationnels** (c'est bien le but du développement !) ;
- **La collaboration avec le client**, nécessitant de nouvelles modalités contractuelles et de travail ;
- **L'adaptation au changement** (dans une certaine limite bien entendu).

Le manifeste ne réfute pas pour autant la valeur des processus de développement, de la documentation du logiciel, de la négociation contractuelle ou du suivi d'un plan, mais il donne la priorité aux 4 valeurs fondamentales.

En d'autres termes, développer Agile ne veut pas dire programmer vite comme un cochon et tout changer en permanence, comme ont pu le dire certains détracteurs !

Pour préciser le sens de ce changement de mentalité, le manifeste agile énonce 12 principes à respecter qui peuvent se résumer à :

- **Livrer tôt** du logiciel opérationnel même incomplet, en commençant par **ce qui a le plus de valeur pour le client** ;
- **Livrer régulièrement** du logiciel selon des cycles réguliers de quelques semaines ;
- **Accepter le changement**, même tard dans le projet ;
- **Faire collaborer au quotidien client et équipe de développement** ;
- Motiver et faire confiance à **l'équipe de développement qui s'auto-organise** ;
- Communiquer directement par des **dialogues en face à face** ;
- Privilégier les **solutions techniques simples** ;
- Adopter et ajuster un **rythme de développement soutenable** par l'équipe ;
- Réfléchir régulièrement aux **moyens d'améliorer l'efficacité** de l'équipe.



Exercice : pour chacune des phrases ci-dessous, cochez s'il s'agit d'après vous d'une démarche Agile ou non :

	Agile	Non Agile
Après livraison partielle pour tests, quand le client envoie un mail pour rectifier une fonctionnalité développée, il faut tout d'abord passer par le service commercial pour établir l'avenant au contrat correspondant à cette modification.	<input type="checkbox"/>	<input type="checkbox"/>
Nous, on livre chaque mois du logiciel opérationnel. Le client a donc un mois pour valider ; après cela, toute modification est refusée, sinon, on ne s'en sort plus !	<input type="checkbox"/>	<input type="checkbox"/>
Dans notre ESN, avant de démarrer un cycle de développement, on se réunit et on détermine ensemble la charge de travail de chaque fonctionnalité à développer. Ça prend du temps mais au moins, on est tous bien au courant du travail à faire.	<input type="checkbox"/>	<input type="checkbox"/>
Dans notre ESN, on pratique Scrum maintenant et l'équipe de développement s'auto-organise pour faire le travail afin de livrer chaque mois. Au début, le client voulait disposer d'un bureau dans notre openspace mais notre chef de projet lui a fait comprendre que sa présence n'était utile que lors des recettes partielles en fin de cycle, dans la salle de réunion.	<input type="checkbox"/>	<input type="checkbox"/>
Nous on pratique Scrum dans l'équipe de développement interne à notre entreprise. On n'a pas vraiment de client puisqu'on fait partie de la même boutique. Et pour échanger des infos, comme on est tous sur le même réseau local, on s'envoie des mails, ça gagne du temps.	<input type="checkbox"/>	<input type="checkbox"/>
Il paraît qu'après livraison partielle, on doit analyser nos pratiques pour faire mieux la prochaine fois. D'abord, on n'a jamais le temps, et ensuite, ça ne sert à rien car on est toujours suffisamment efficace pour livrer dans les temps.	<input type="checkbox"/>	<input type="checkbox"/>
Après livraison d'une fonctionnalité, il est fréquent que le client nous demande de modifier quelque chose pour faire encore mieux, alors on met ça sous le coude pour le prochain cycle. Par contre, s'il nous demande un changement sur ce que l'on est en train de développer, on refuse, sinon, on ne s'en sort pas !	<input type="checkbox"/>	<input type="checkbox"/>

Réponses page suivante.

Agilité : réponses et commentaires :

	Agile	Non Agile
Après livraison partielle pour tests, quand le client envoie un mail pour rectifier une fonctionnalité développée, il faut tout d'abord passer par le service commercial pour établir l'avenant au contrat correspondant à cette modification. Les principes Agiles reposent sur l'acceptation du changement et une contractualisation des moyens plutôt que des résultats ; le classique « cahier des charges » contractuel n'existe pas plus que la notion de « pénalité ».	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Nous, on livre chaque mois du logiciel opérationnel. Le client a donc un mois pour valider ; après cela, toute modification est refusée, sinon, on ne s'en sort plus ! Il ne s'agit pas d'accepter une demande de changement à tout moment. Mais est-il raisonnable de refuser une meilleure solution ou une nouvelle idée sous prétexte qu'on y a pensé « trop tard » ?	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Dans notre ESN, avant de démarrer un cycle de développement, on se réunit et on détermine ensemble la charge de travail de chaque fonctionnalité à développer. Ça prend du temps mais au moins, on est tous bien au courant du travail à faire. En méthode classique, c'est en général le Chef de projet qui estime les charges de travail et attribue les tâches. En début de cycle, Scrum préconise une réunion de préparation au cours de laquelle la charge de travail de chaque fonctionnalité est définie collectivement par l'équipe.	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Dans notre ESN, on pratique Scrum maintenant et l'équipe de développement s'auto-organise pour faire le travail afin de livrer chaque mois. Au début, le client voulait disposer d'un bureau dans notre openspace mais notre chef de projet lui a fait comprendre que sa présence n'était utile que lors des recettes partielles en fin de cycle, dans la salle de réunion. Les principes Agiles reposent sur une collaboration permanente entre client et équipe de développement. Selon Scrum, le représentant du client peut ne pas être présent en permanence mais il doit être joignable au quotidien.	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Nous on pratique Scrum dans l'équipe de développement interne à notre entreprise. On n'a pas vraiment de client puisqu'on fait partie de la même boutique. Et pour échanger des infos, comme on est tous sur le même réseau local, on s'envoie des mails, ça gagne du temps. Un client interne est malgré tout un client ; le développement de logiciel est bien une activité de service au profit d'un tiers. De plus, les principes Agiles reposent sur une communication réelle en face à face bien plus efficace que des échanges asynchrones de mails.	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Il paraît qu'après livraison partielle, on doit analyser nos pratiques pour faire mieux la prochaine fois. D'abord, on n'a jamais le temps, et ensuite, ça ne sert à rien car on est toujours suffisamment efficace pour livrer dans les temps. Les principes Agiles reposent sur un processus d'amélioration continue par auto-inspection. Et il est toujours possible de faire mieux, non ?	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Après livraison d'une fonctionnalité, il est fréquent que le client nous demande de modifier quelque chose pour faire encore mieux, alors on met ça sous le coude pour le prochain cycle. Par contre, s'il nous demande un changement sur ce que l'on est en train de développer, on refuse, sinon, on ne s'en sort pas ! Selon Scrum, le contenu d'un cycle reste figé le temps du cycle et toute modification pourra être affectée au plus tôt au cycle suivant.	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Agilité et Scrum

2.3 XP ET SCRUM

XP et Scrum sont les deux vedettes actuelles du développement informatique Agile.

XP est véritablement une méthode en ce sens qu'elle **décrit des processus précis** à respecter alors **que Scrum dessine un cadre de travail (framework)** à adapter à chaque situation selon les caractéristiques du projet ou des équipes ; pour ce faire, l'assistance d'un **Coach Agile** peut être très bénéfique.

XP est totalement adapté au développement de logiciels et contient de nombreux aspects techniques. XP est basé sur le bon sens et l'observation de bonnes pratiques de programmation qui sont poussées à l'extrême. On peut citer les principes suivants :

- **Livraison fréquentes** à un rythme régulier ;
- **Présence du client** sur le site de développement afin de définir ou préciser en continu les fonctionnalités à développer ;
- **Estimation collective** des charges de travail pour les scénarios à développer ;
- Conception technique la plus simple possible ;
- Mise en place de **règles de codage** respectées par toute l'équipe ;
- **Programmation par binômes** et recherche permanente d'optimisation (par *refactoring*) ;
- Responsabilité collective du code produit ;
- **Importance des tests** (notion de développement piloté par les tests ou TDD) et de l'intégration continue.

Scrum reprend des principes du Lean Management, de Kanban et de XP. Scrum s'est vite révélé très efficace pour des projets de taille modeste impliquant peu de participants.

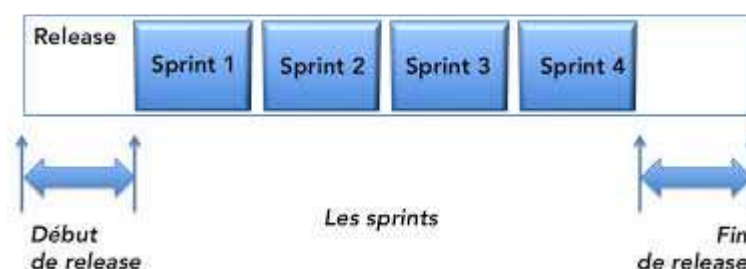
3. SCRUM

3.1 PRINCIPES ET VOCABULAIRE SCRUM

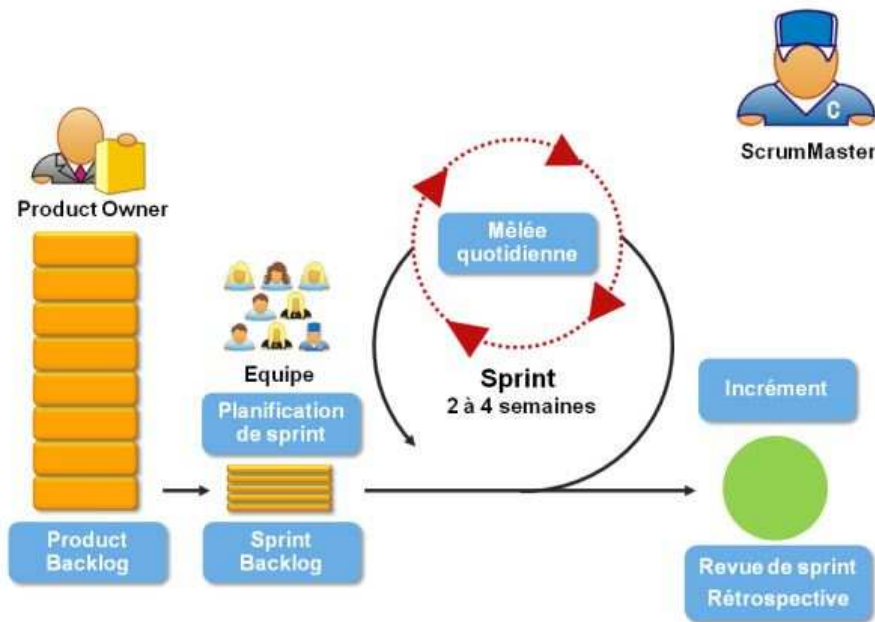
Dans un projet Scrum il y a 3 acteurs : le **Product Owner**, le **ScrumMaster** et **l'équipe de développement**.

Au début d'un projet, il n'est pas rédigé de document qui décrit en détail les spécifications fonctionnelles à développer. Le représentant du client (« Product Owner ») définit les fonctions essentielles sous forme d'un « **Backlog** » que l'équipe affine progressivement, au fur et à mesure des développements. Les cycles de développement sont appelés « **Sprint** » ; ils ont une durée constante et doivent aboutir à quelque chose de potentiellement livrable.

Une « **Release** » est une période de temps, composée de Sprints, qui permet de produire une version du logiciel, et de décider de poursuivre ou non les développements.



Agilité et Scrum



3.2 EQUIPE, ARTEFACTS ET CEREMONIES SCRUM

Scrum s'appuie sur 3 piliers, **transparence**, **inspection** et **adaptation**, et définit une organisation du travail pour y arriver basée sur :

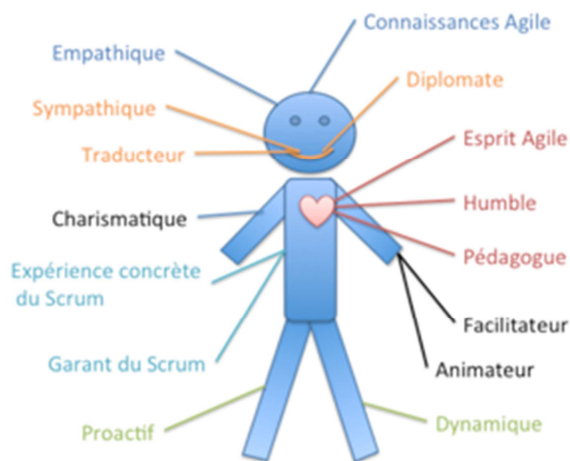
- la constitution de **petites équipes pluridisciplinaires auto-organisées** ;
- le principe de développement incrémental par le cadencement en **Sprints courts à durée constante** (2 à 4 semaines) aboutissant chacun à un « **livrable** » (du logiciel plus ou moins complet, plus ou moins livrable) ;
- le respect d'un certain nombre de **cérémonies** (*affinage du Backlog, planification du sprint, mêlée quotidienne, revue de Sprint, rétrospective de Sprint*), réunions aux objectifs précis et rythmant les Sprints de développement ; pour chacun des cérémoniaux, Scrum définit un cadre de temps (*timebox*) ;
- l'utilisation de 4 **artéfacts visuels** (document) permettant d'obtenir une visibilité permanente sur l'avancement du projet (*Product Backlog, Sprint Backlog, Scrum Board, Charts*).

L'**équipe de développement Scrum** est composée de développeurs plus ou moins polyvalents mais tous qualifiés du même titre **Développeur**, encadrés au quotidien par un **Scrum Master** (SM). A cela s'ajoute le client sous la forme du **Product Owner** (PO), lui aussi présent et actif au quotidien.

Il est temps de revenir plus en détail sur tous ces termes.

3.3 LE SCRUM MASTER

🔥 Au contraire des organisations classiques avec Chef de Projet, le **Scrum Master n'est pas un 'chef'**, il n'a pas de rôle hiérarchique envers les développeurs. C'est un **facilitateur** et le **garant de l'application de la méthode Scrum**. A ce titre il anime les différents cérémoniaux.



Le Scrum Master identifie au quotidien les difficultés rencontrées par les développeurs et **contribue à lever les obstacles** en intervenant aux bons niveaux afin d'assurer la bonne coopération de tous les acteurs concernés.

Le Scrum Master collabore avec le Product Owner à la définition opérationnelle du produit sous forme d'histoires ou **User Stories** répertoriées dans le **Product Backlog** (1° artéfact Scrum).

En bref, le Scrum Master doit **connaître Scrum** (formation et certification), être **impliqué**, être **communicatif** et **meneur d'équipe**, avoir des **capacités de médiation**, et rester **humble**

(c'est celui qu'on oublie quand tout va bien...).

3.4 LE PRODUCT OWNER SCRUM

💣 Le Product Owner **représente** à lui seul les besoins des différents **utilisateurs** ; il en est le **représentant unique** et **l'interlocuteur privilégié des développeurs**.

Le Product Owner doit avoir une **vision du produit final**, la faire partager à l'équipe de développement, la scinder selon un **plan de Releases** échelonnées dans le temps. Il est responsable de cette vision du produit et du retour sur investissement attendu par le client.

Le Product Owner décrit précisément au fur et à mesure des développements les fonctionnalités à développer sous forme de **User Stories accompagnées de critères d'acceptation** et les **priorise** dans le Product Backlog ; il doit donc être capable de rentrer dans le détail des fonctionnalités à développer et des règles de gestion à respecter. Au début de chaque Sprint il participe à la définition du contenu du Sprint, le **Sprint Backlog** (2° artéfact Scrum).

Le Product Owner doit connaître les fondamentaux de Scrum et savoir négocier.

3.5 L'EQUIPE DE DEVELOPPEMENT

Une équipe de développement Scrum est réduite : **5 à 7 personnes** semble être la taille idéale. **Sa première responsabilité est d'atteindre l'objectif du Sprint**. Et pour ne pas être "prise en traître", **l'équipe s'auto-organise et définit elle-même collectivement** le contenu du Sprint en estimant, collectivement toujours, la charge de travail correspondant à chaque User Story proposée par le Product Owner ; c'est l'objet de la réunion de planification initialisant chaque Sprint Scrum.

L'équipe se réunit chaque jour sur un temps court (Daily Scrum Meeting) pour **se coordonner** et **noter visuellement l'avancement** du Sprint grâce au **Scrum Board** (3° artéfact Scrum) qui affiche en permanence l'état des tâches à réaliser. *Transparence*.

En fin de Sprint, l'équipe réalise une **présentation des développements livrables** et **fait le point sur son fonctionnement passé** afin de **mesurer son efficacité** et décider d'actions ou d'orientation à adopter **pour améliorer son efficacité future**. *Inspection et adaptation*.



Exercice : qui fait quoi ?

	<i>SM</i>	<i>PO</i>	<i>Dév.</i>	<i>Autres</i>
Présenter les fonctionnalités du produit à développer				
Animer la mêlée quotidienne				
Définir les fonctionnalités du produit à développer				
Estimer la charge de travail d'une tâche				
Attribuer une tâche à un développeur				
Accepter la modification d'un besoin déjà exprimé dans une User Story				
Faire le bilan du Sprint				
Définir le contenu du Sprint				
En cas de difficulté, décider qui vient en aide à un développeur				

Réponses page suivante.

Réponses : qui fait quoi ?

	SM	PO	Dév.	Autres
Présenter les fonctionnalités du produit à développer		✓		
Animer la mêlée quotidienne	✓			
Définir les fonctionnalités du produit à développer C'est bien entendu le rôle principal du Product Owner mais l'équipe de développement est aussi force de proposition tout au long du projet		✓	✓	
Estimer la charge de travail d'une tâche Surtout pas le Scrum Master qui n'est pas un Chef de Projet			✓	
Attribuer une tâche à un développeur Surtout pas le Scrum Master qui n'est pas un Chef de Projet			✓	
Accepter la modification d'un besoin déjà exprimé dans une User Story Accepter le changement est un principe fort de méthodes Agiles ; pour Scrum toute modification est acceptable tant que l'équipe n'est pas en train de développer la fonctionnalité en question ; le contenu d'un Sprint en cours est figé		✓		
Faire le bilan du Sprint Toute l'équipe est concernée, et même au-delà ! Scrum sépare ce bilan en deux cérémonies car il s'agit à la fois de valider ce qui a été produit (Sprint Review pouvant être ouvert à d'autres 'parties prenantes') et d'améliorer le fonctionnement interne pour le prochain Sprint (Retrospective de Sprint)	✓	✓	✓	✓
Définir le contenu du Sprint Le PO liste les fonctionnalités à développer ordonnées par valeur Métier et l'équipe les prend en charge selon sa capacité de production auto-estimée.		✓	✓	
En cas de difficulté, décider qui vient en aide à un développeur Surtout pas le Scrum Master qui n'est pas un Chef de Projet			✓	

3.6 SPRINT ZÉRO, ROADMAP, USER STORIES, PERSONAS ET PRODUCT BACKLOG

Au contraire de la démarche classique qui commence par définir dans un Cahier des Charges contractuel le produit à réaliser, en démarche Agile, et avec Scrum en particulier, **le produit est défini, conçu et développé au fur et à mesure des développements.**

Le Product Owner doit bien entendu avoir une « vision » globale du produit final et la faire partager à l'équipe de développement. Le PO définit des étapes intermédiaires de mise en service, qualifiées de **Releases** ; chaque Release correspond à une version **homogène, livrable et opérationnelle** du produit (alors que chaque Sprint aboutit à une version intermédiaire, livrable ou seulement potentiellement livrable).

Chaque fonctionnalité à développer est décrite par le Product Owner (assisté du Scrum Master) dans une **User Story** dont l'ensemble constitue le **Product Backlog**.

Mais comment et quand tout cela est-il concrètement défini ?

Un projet Scrum commence en général par un « **Sprint Zéro** » qui est tout à la fois l'occasion de **constituer et consolider l'équipe** en réalisant une première itération complète (se connaître et apprendre à travailler ensemble) et aussi un **moment privilégié de partage de la vision du produit** (en cas de découpage en Releases, il peut y avoir un sprint zéro en début de chaque Release).

Bien souvent, le Product Owner n'a qu'une idée imprécise du produit final ; il en connaît seulement les grandes lignes et les objectifs ; et **ces objectifs ne sont pas techniques mais bien Métier, orientés business**, puisque le PO n'est pas un technicien informatique mais le représentant du client et des futurs utilisateurs du logiciel. On est donc bien loin d'une définition exhaustive des fonctionnalités à développer !

Dans la logique Agile, basée sur des principes de coopération, c'est bien l'équipe au complet qui définit, au fur et à mesure, le produit. Et plus on met de cerveaux ensemble, plus les bonnes idées fusent ! Ensemble, développeurs, Scrum Master et Product Owner vont donc concevoir, au cours du Sprint Zéro, une première vision du produit à réaliser et ce, à l'aide de jeux favorisant la créativité et la communication ; on parle de « **serious games** » et de « communication visuelle », beaucoup utilisés en démarche Agile.

Le Sprint Zéro n'a pas de durée fixe et reste assez différent des sprints suivants. Au-delà de cette démarche vers une vision partagée du produit, le Sprint Zéro a aussi pour but de vivre un premier Sprint complet, ponctué de ses autres cérémonies et aboutissant à une première production toute simple afin de rôder le fonctionnement.

Une multitude de jeux et techniques existent et il s'en crée en permanence ; on peut citer : *Brainstorming, Impact Mapping, Story Mapping, Design Thinking, Storyboard, Pitch Elevator*. La représentation visuelle du futur produit peut être réalisée par maquettage de l'emballage (*Box Vision*) ou encore simulation de journaux du lendemain vantant les mérites du produit. Il est aussi important de bien se représenter les différents utilisateurs, au niveau de leurs besoins mais aussi de leur personnalité, leur comportement ; c'est l'objet des « **Personas** ». Tout cela n'a d'autre but que favoriser l'émergence de bonnes idées et souder l'équipe autour d'un but commun. L'animation de ce type de jeux ne s'improvise pas et c'est là aussi qu'interviennent les Coach Agile.

Exemples de persona :

SOPHIE, la Gestionnaire



CONTEXTE

Gère ses comptes et devant une activité quasi-quotidienne
2 enfants - Requin Parisienne
Jougle entre son travail, sa vie de famille et ses loisirs qu'elle tient à conserver.
Ses préférés: Vente Privée - VDM
Cadre: 2800 euros/mois
→ On veut la Fidéliser
→ On veut la pousser à épargner
→ On veut Améliorer son Expérience Client

BUTS et COMPORTEMENTS

- Utilise beaucoup Internet dans le cadre de son travail ou chg elle pour ses services préférés.
- Viens 3 à 4 fois/semaine en soirée
- Veut consulter ses comptes
- Veut Faire des virements (y compris internationaux)
- Veut échanger avec son conseiller sur des questions précises (épargne, assurance, assurance...)
- Pourquoi pas faire une souscription simple via Internet


CE QUE CELA IMPLIQUE

- Un accès rapide à ses comptes
- Des raccourcis évidents pour ses actions simples et fréquentes
- Un Tableau de Bord simplifié centré sur la synthèse des comptes + simulateurs + Bloc conseiller (⇒ Engager la conversation)

Jc Grosjean - www.qualitystreet.fr

NAVIMA BYX

Client: Arnold

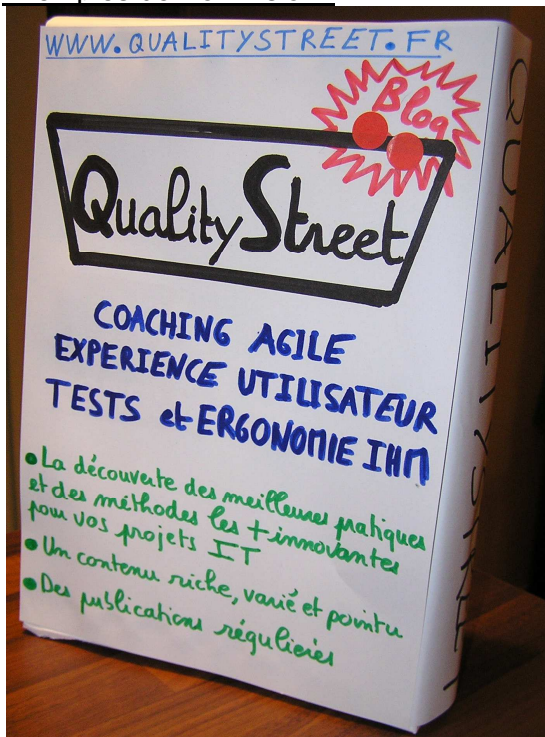


😊 Satisfaction 😊

5						
---	--	--	--	--	--	--

Arnold est du genre raisonnable. Il sait ce qu'il veut et comment l'obtenir. Tant qu'il est satisfait, travailler avec lui est un plaisir. Mais lorsque ce n'est plus le cas...

Exemples de Box Vision :



Agilité et Scrum

Afpa © 2015 – Section Tertiaire Informatique – Filière « Etude et développement »



A l'issue de ces séances créatives, l'équipe dispose d'une multitude de notes, en général **sous forme de Post-it** qu'il reste à organiser ; on y trouve pêle-mêle, User Stories déjà clairement énoncées, simples embryons de User Stories ou au contraire 'super' User Stories qu'il faudra décomposer plus tard (on parle alors d'**Epics**).

Des jeux et techniques d'animation aident à classer ces masses de User Stories de manière à obtenir une vision plus nette du produit sous forme de fonctionnalités, regroupées au sein de Releases, qui seront précisées au besoin et développées progressivement, au fur et à mesure des Sprints.

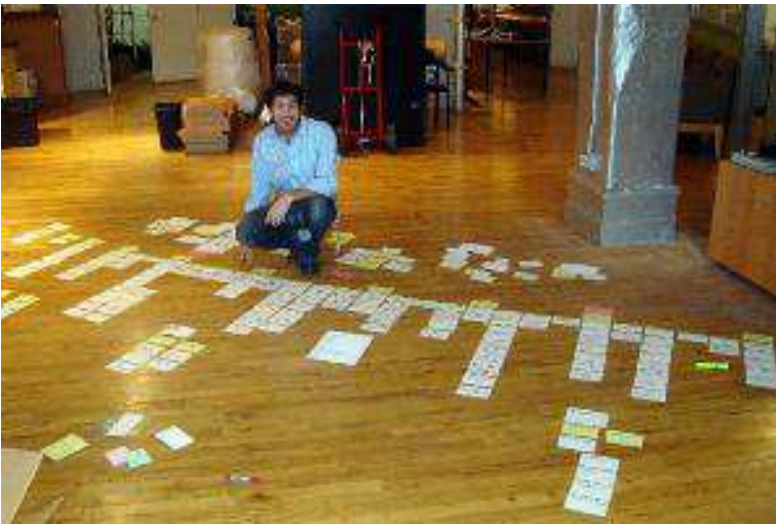
La difficulté du classement (**Story Mapping**) vient du fait que l'on travaille sur deux axes, l'axe horizontal, correspondant à l'**USAGE**, matérialise la succession (chronologique) des usages de l'utilisateur, l'axe vertical, l'axe **PRIORITE**, matérialise la priorité des fonctionnalités décrites en User Stories, plus ou moins détaillées, et **rangées par ordre d'importance pour l'activité**, en haut celles qui sont indispensables à l'usage, en bas celles qui sont annexes ou ne concernent qu'une fraction des flux d'informations que traite le produit. Enfin, on trace les limites prévisionnelles des différentes **Releases** sous forme de séparations horizontales.

Pour aider à prioriser les idées, on utilise fréquemment la technique dite « **MoSCoW** » : chaque item est ainsi qualifié de

- **M**(ust), soit **DOIT** être fait,
- **S**(hould), soit **DEVRAIT** être fait dans la mesure du possible,
- **C**(ould), soit **POURRAIT** être fait dans la mesure du possible,
- ou encore **W**(on't), soit **NE SERA PAS** fait cette fois mais plus tard.

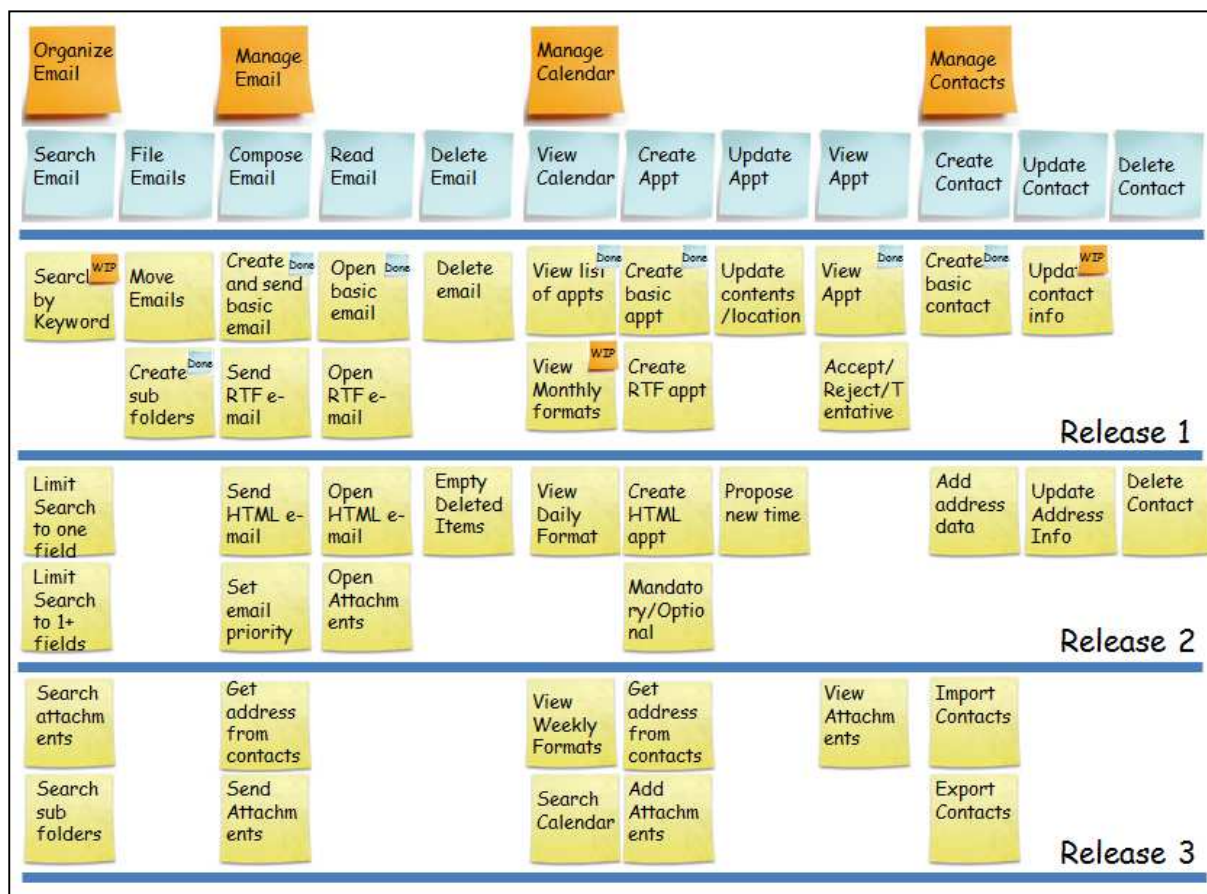
Cette série de questions peut être utilisée aussi bien pour établir cette *Roadmap* générale du produit que lors de la définition des items à développer pour un Sprint.

Exemple de séance de Story Mapping :



Exemples de **Roadmap** priorisant les User Stories dans des Releases :





Ces panneaux représentent donc visuellement le **Product Backlog**. Bien entendu, rien n'est jamais définitif aussi bien au niveau des User Stories, de leur précision, que de leur priorisation.

Enfin, il est important de souligner que certaines User Stories, voire certaines Releases, ne seront probablement jamais développées : il s'agit bien d'une vision, d'un projet, non d'une charte contractuelle !

Si un calendrier prévisionnel est souhaité par la direction, il reste préférable que les livraisons de Releases soient prévues à un rythme de 3 mois maximum (ce qui permet de limiter les différences entre les versions et d'obtenir des retours *-feedback-* rapides et précis sur le logiciel).

3.7 LA VIE D'UN SPRINT

La vie d'un Sprint Scrum est ponctuée par des cérémonies et utilise des artefacts visuels (documents) qui sont conçus pour aider à atteindre les objectifs. Le Scrum Master, garant de l'application de la méthode, est là pour s'assurer que ces événements ont bien lieu, que ces artefacts sont bien utilisés et à jour et qu'ils respectent les 'règles de l'art' Scrum.

3.7.1 Scrum Board → *Transparence*

Avant le démarrage d'un Sprint, il faut bien définir son contenu, c'est-à-dire **ce qui est à réaliser** lors du cycle à venir ; cela est défini sous forme d'une **liste de tâches inscrites au Sprint Backlog**.

Agilité et Scrum

Afpa © 2015 – Section Tertiaire Informatique – Filière « Etude et développement »

- **TODO** (à faire),
- **DOING** ou **IN PROGRESS** (en cours)
- **DONE** (fait) ;

☀ Chaque **tâche** y est identifiée par un **Post-it**, disposé à l'origine dans la colonne **TODO** puis déplacé dans les colonnes suivantes au fur et à mesure de l'avancement des travaux ; simple et efficace ! En voilà de la bonne communication visuelle !

The whiteboard is organized into three main columns: **To Do**, **In Progress**, and **Done**.

In Progress column features a large red title: **Summer of Bob**.

Sticky Notes Content:

- To Do:**
 - Branches - AMP - website? (Yellow)
 - AMP - 3.0 (Pink)
 - Big Weekly AMP Call (Pink)
 - Verit UrbanCode 3 (Green)
 - White paper 2.0 (Yellow)
 - AgileCycle Demo SCRIPT (Green)
 - Website page (Pink)
 - AgileCycle Demo SET-up (Green)
 - TFS COMP. App 14 (Pink)
 - RTC COMP. App 14 (Yellow)
 - AgileCycle Cost 3 (Yellow)
 - BLOG #2 1 (Yellow)
 - BLOG #3 1 (Yellow)
 - BLOG #4 1 (Yellow)
 - BLOG #4 1 (Pink)
- In Progress:**
 - BLOG #1 1 (Pink)
 - AMP Demo CA 14 (Yellow)
 - COMP MATRIX - 3 (Yellow)
 - BLOG #1 1 (Yellow)
 - AMP Demo SC 3 (Yellow)
 - Reimagine Learning Platform 1.5 (Yellow)
 - Website 4 (Yellow)
 - Terms 5 (Yellow)
 - BLOG #2 1 (Pink)
 - BLOG #3 1 (Pink)
 - Big Ideas:
 - Testing
 - Why CI/CD?
 - Learn & Share
 - Big Ideas:
 - Security
 - Change
- Done:**
 - AgileCycle Demo 1 (Yellow)
 - Website 6 (Yellow)
 - Website 2 (Yellow)
 - Website 8 (Yellow)
 - HP Website 3 (Yellow)
 - AgileCycle Pricing 1 (Yellow)
 - Website 6 (Yellow)
 - Website 00 (Yellow)
 - AgileCycle Demo 3 (Yellow)
 - AgileCycle Demo 3 (Yellow)
 - Q2:
 - AgileCycle Demo Training (Pink)
 - AgileCycle Demo (Pink)
 - Sales Training (Yellow)
 - AgileCycle Website page - 2018 (Yellow)

Comme un Sprint doit aboutir à du logiciel (potentiellement) livrable, il est nécessaire que toute la chaîne du développement soit assurée au cours du Sprint, de la conception au déploiement en passant par la documentation, l'intégration dans l'existant et les tests, y compris les techniques de relecture et d'amélioration du code (*nettoyage, refactoring*). Cette définition du « terminé », du « fini-fini », pour un Sprint est en général matérialisée par une liste de critères génériques affichée de façon visible et permanente sur le Scrum Board.

3.7.2 Backlog Grooming - Affinage de Backlog

Mais comment en arrive-t-on à la planification de *tâches* au Scrum Board, à partir du Product Backlog qui liste des *User Stories* et des *Epics* ?

C'est l'objet des 2 premières cérémonies Scrum, appelées **Backlog Grooming** ou **Product Backlog Refinement Meeting** (affinage du Backlog) et **Sprint Planning Meeting** (planification du Sprint).

Un **Backlog Grooming** réunit **Scrum Master**, **Product Owner** et **équipe de développement** pour une durée typique de 2 heures et a lieu régulièrement (en particulier en début de Sprint) pour 'toiletter' le Product Backlog de manière à :

- clarifier et décomposer les items prioritaires du Product Backlog (Epics et User Stories),
- ré-évaluer la priorité relative des items,
- diviser des User Stories qui ont une priorité élevée mais qui paraissent trop 'grosses' en terme de charge de travail pour tenir dans un Sprint à venir,
- enlever des User Stories qui n'apparaissent plus pertinentes,
- créer de nouvelles User Stories en réponse aux besoins nouvellement découverts,
- assigner des estimations aux User Stories qui n'en ont pas encore reçu,
- corriger les estimations à la lumière d'information nouvelles.

Contrairement à un Cahier des Charges contractuel et figé, un Backlog vit et évolue dans le temps.

Définir une User Story n'est pas un exercice évident car il s'agit de **décrire une fonctionnalité utile du logiciel et non une liste de tâches techniques à réaliser** pour la développer. On parle de découpage *vertical* en ce sens qu'une User Story pourra nécessiter toutes les tâches habituelles de développement (modélisation, implémentation en base de données, design d'interface homme-machine, programmation événementielle, programmation de services, tests, intégration, déploiement).

☀ Une User Story se définit toujours à l'aide du modèle « **en tant que... je peux/veux... afin de ...** » et doit vérifier les règles **INVEST** :

- Indépendante,
- Négociable,
- Valorisable,
- Estimable,
- Succincte -Small-
- et Testable)

Une User Story doit toujours correspondre à une **valeur Métier** (*Business Value* - un bénéfice pour le client, évalué entre 1 et 100) ; cette valeur Métier reste le **seul critère de priorisation** dans le Product Backlog.

☀ Une User-Story est complétée par un ou plusieurs **critères d'acceptation (Acceptance Criterias)** qui définissent la manière de tester en fin de développement qu'elle est bien conforme aux besoins ; cela rejoint la technique du **développement piloté par les tests** (ou TDD). Typiquement, un critère se définit à l'aide du modèle « **étant donné..., quand... alors...** » (soit 'given..., when... then...').

Un entraînement est nécessaire pour pouvoir définir correctement les User Stories et leurs critères d'acceptation ; c'est l'objet du Sprint Zéro pour les équipes débutantes et des exercices proposés en annexe.

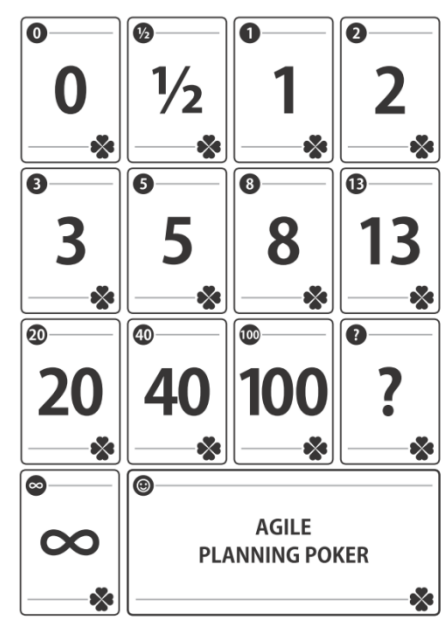
A ce niveau de préparation du Sprint, l'équipe Scrum établit déjà une **estimation collective** de la charge de travail, le **coefficient d'effort**, que représente chaque User Story prioritaire. **On utilise en général la méthode d'estimation relative Tee-Shirt Sizing** : quand la User Story candidate est bien comprise, chacun qualifie sa complexité de réalisation à l'aide d'une taille de tee-shirt, **S**(mall), **M**(edium), **L**(arge) ou **XL** (extra large) ; après confrontation et échanges sur les estimations, une valeur consensuelle est affectée à la User Story ou bien cette Story candidate est reformulée ou encore décomposée quand il s'agit d'une Epic.

Une autre méthode d'estimation utilisée fréquemment en démarche Scrum et le **Poker Planning**. Plutôt que de se limiter à 4 niveaux d'effort, S, M, L et XL, cette animation utilise un jeu de carte basé sur une 'suite de Fibonacci'.

Le principe reste le même, chaque développeur estimant l'effort relatif que représente la User Story puis se confrontant au reste de l'équipe pour arriver à une estimation consensuelle.

🔍 L'originalité de cette démarche par rapport aux techniques d'estimation de charge classiques, est **qu'il ne s'agit pas d'estimer le nombre de d'heures ou de jours-homme de travail, mais plutôt de situer l'effort à fournir, par rapport à un développement simple ordinaire**. Pour ce faire, le Sprint Zéro peut être l'occasion d'identifier une User Story prioritaire correspondant à un effort de base (Medium en Tee-Shirt sizing ou 1 en Planning Poker.

L'estimation de l'effort de développement est enregistrée sur la User Story et servira à limiter l'engagement et mesurer l'efficacité de l'équipe.



3.7.3 Sprint Planning Meeting - Planification du Sprint

Le **Sprint Planning Meeting**, quant à lui, a pour but de décider des items du Product Backlog qui constitueront le **Sprint Backlog** (2° artéfact Scrum) puis **de découper collectivement ces items en tâches** ; il est calé dans un *timebox* de 4h pour des Sprints de 2 semaines ou une journée pour des Sprints de 4 semaines.

Une réunion de planification ne peut se tenir que si le PO dispose d'un Product Backlog priorisé d'au moins 50 Stories définies, accompagnées de leurs critères d'acceptation, et que le PO comprend afin de pouvoir les expliquer clairement.

Le Product Owner présente les objectifs du Sprint (résultats attendus) et fournit un Product Backlog priorisé stable (issu du Product Backlog Grooming). Pour chaque User Story, **l'équipe de développement liste alors les tâches à réaliser et les consigne sur des Post-it placés en colonne TODO** face à la User Story correspondante. Ainsi on trouvera des Post-it concernant la modélisation des données et les implémentations nécessaires dans la base de données, d'autres pour le design des écrans, d'autres encore pour le développement de services distants fournis par des serveurs, pour la documentation, pour les tests d'intégration... Idéalement, une tâche ne devrait pas représenter plus d'une journée de travail. En fin d'analyse de la User Story, l'équipe se demande collectivement si la réalisation de ces tâches reste possible dans le temps restant du Sprint.

Agilité et Scrum

Afpa © 2015 – Section Tertiaire Informatique – Filière « Etude et développement »

Les User Stories sont ainsi décortiquées une à une, en ordre de priorité, et les tâches placées sur le Scrum Board, sans dépasser la **capacité de production** de l'équipe pour le Sprint à venir.

💣 **A la fin du Sprint Planning Meeting, le contenu du Sprint Backlog est figé** ; tout nouveau besoin, toute correction de bug, toute modification, sera placée dans le Product Backlog de manière à être affectée au(x) Sprint(s) suivant(s). Agilité, souplesse et acceptation du changement, oui, mais pas n'importe quand ! Sur le même principe, il est important que la constitution de l'équipe reste stable sur le temps d'un Sprint, voire, une Release.

Exemple de Scrum Board en cours de Sprint ; plus complet, il reprend les items ordonnés du Product Backlog sur 2 colonnes, non encore préparés en affinage de Backlog et prêts pour le démarrage (*Kick Off*) du prochain Sprint :



3.7.4 Daily Scrum Meeting - Mêlée quotidienne ➔ *Transparence et Adaptation*

La cérémonie majeure d'un Sprint Scrum est la **mêlée quotidienne**, le **Daily Scrum Meeting**, regroupant **équipe de développement** et **Scrum Master** chaque jour, pour **10 à 15mn**, debout en cercle. Il s'agit que chaque membre de l'équipe de développement réponde à **3 questions** :

- ✓ Qu'est-ce que j'ai fait hier ?
- ✓ Qu'est-ce que je pense faire aujourd'hui ?
- ✓ Quelles sont les difficultés que je rencontre ?

💣 **Le Scrum Master facilite** l'expression de chacun, fait préciser les difficultés éventuelles et les enregistre au besoin pour chercher des solutions externes à l'équipe ; en aucun cas, il ne doit jouer au classique Chef de projet et attribuer les tâches aux développeurs ni imposer à un

membre de venir en aide à un autre ! **Selon Scrum, l'équipe de développement s'auto-organise.**

Quand une tâche est terminée, le développeur déplace le Post-it correspondant en colonne DONE et propose de prendre en charge une autre tâche ; avec l'accord de l'équipe, il place le Post-it concerné en colonne DOING et y appose son nom.

Eventuellement, des tâches supplémentaires peuvent apparaître au cours du développement ; le Daily Meeting est l'occasion d'ajouter le Post-it correspondant dans la liste des tâches à faire. C'est ainsi que l'équipe s'auto-organise et effectue un suivi très fin de l'avancement du Sprint. **Selon Scrum, l'équipe de développement est collectivement responsable du code produit.**

3.7.5 Revue de Sprint - Sprint Review

Les Sprints Scrum ont une durée fixe, en général entre 2 et 4 semaines. **A l'issue du temps imparti se tient la Revue de Sprint**, étape de recette partielle et temps fort d'inspection du produit.

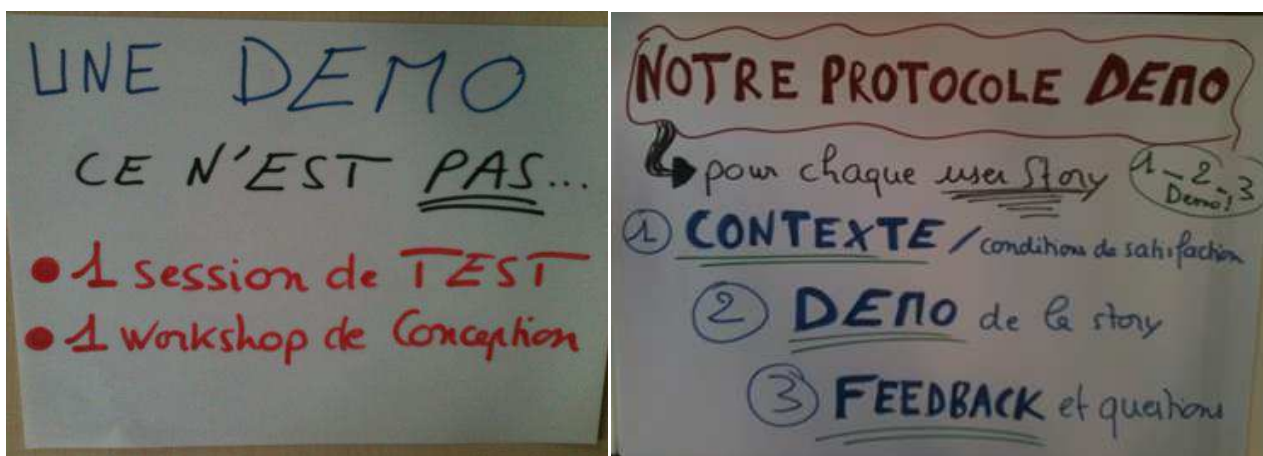
Elle réunit **l'équipe de développement**, le **Scrum Master**, le **Product Owner** et tout représentant du client jugé utile (les *parties prenantes* ou *stakeholder*, tels des utilisateurs ou des décideurs financiers), pour une durée de 2h ou 4h selon la durée des Sprints.

Cette revue comporte 6 étapes : la préparation de la démonstration, le rappel des objectifs du Sprint, la démonstration, la collecte du *feedback*, le calcul de la *vélocité* et un éventuel ajustement du plan de Release.

Le temps fort de cette réunion est la **démonstration du produit** potentiellement livrable, et en particulier **du nouvel incrément développé**. Il s'agit bien d'une étape de recette car c'est à ce moment que **le PO déclare quels items du Product Backlog** sont acceptés, c'est-à-dire sont **conformes aux critères d'acceptation** définis lors du Sprint Planning Meeting et, au final, **si les objectifs du Sprint sont atteints**. La « **Definition of Done** » et les « **Acceptance Criterias** » de la User Story sont les seuls éléments d'arbitrage.

Les items qui ne sont pas finis ou qui sont 'partiellement finis' retournent intégralement dans le Product Backlog pour un prochain Sprint.

Typiquement, le Sprint Review commence par un rappel des objectifs du Sprint puis vient la démonstration ; les parties prenantes invitées et le PO peuvent effectuer des manipulations et sont invités à réagir, voire à suggérer des améliorations. A l'issue, le PO valide les items du Product Backlog qui sont finis et ajoute éventuellement de nouveaux items dans le Product Backlog, qui resteront à prioriser dans les Sprints à venir.



Tous les items du Product Backlog acceptés par le PO sont comptabilisés pour mesurer l'efficacité de l'équipe. La production du Sprint peut être mesurée en **nombre de User Stories acceptées et refusées** mais ce n'est pas suffisant car toutes les User Stories ne représentent

pas le même effort de développement. On y ajoute donc en général l'**indicateur de vélocité** qui **pondère chaque User Story** acceptée par son **coefficient d'effort** (valeur de carte de Poker Planning ou valeur correspondant à la taille de Tee-shirt) de manière à calculer un nombre de **Story Points**.

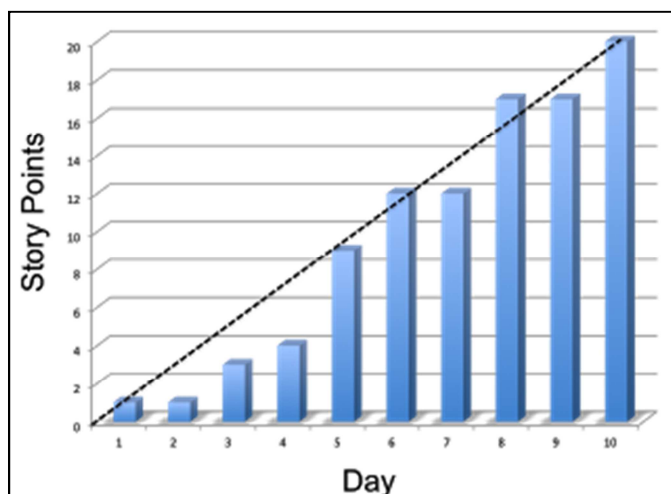
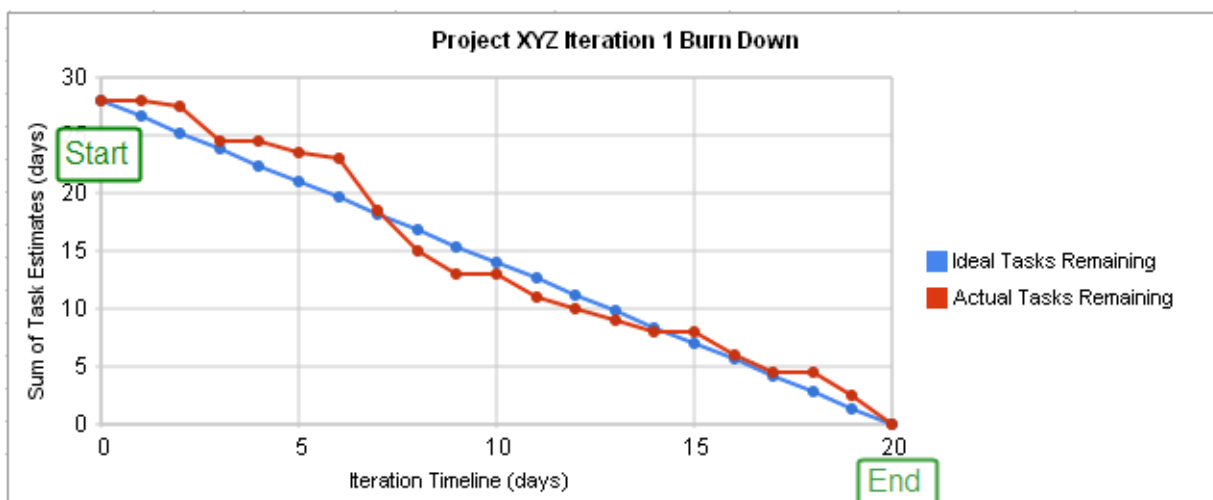
🔥 Vélocité de l'équipe = somme des points d'effort des Stories réalisées et acceptées.

Cette mesure n'a pas de signification intrinsèque ; elle permet juste de **comparer l'efficacité de l'équipe d'un Sprint à l'autre** de manière à tenter de l'améliorer et de **faire des prévisions** sur l'atteinte des objectifs de la prochaine Release, donc sur le calendrier prévisionnel ; en démarche Scrum, les engagements mutuels du client et de l'équipe portent sur les moyens mis en œuvre, non sur le résultat ou le calendrier de livraison.

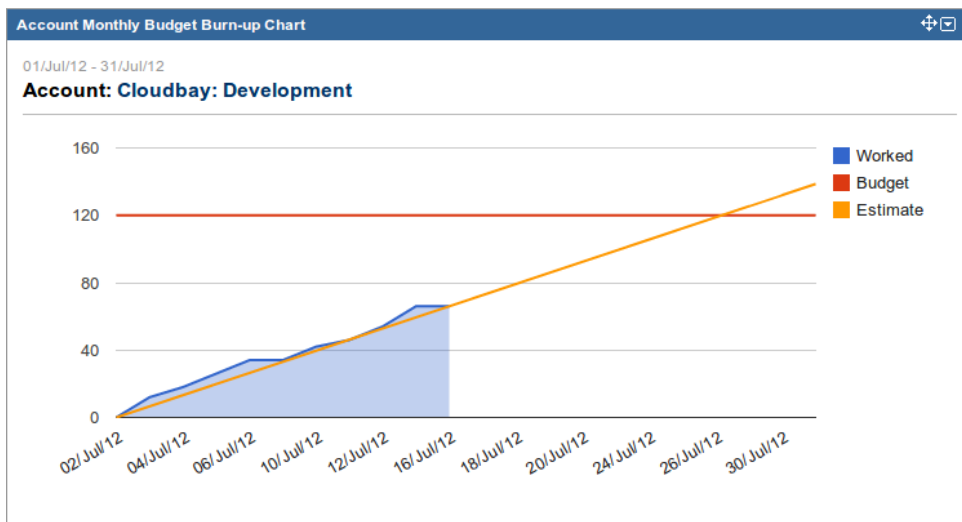
Après 3 Sprints, l'équipe dispose déjà de l'**expérience** suffisante pour procéder à une estimation de sa vélocité ; elle pourra alors d'autant mieux s'engager sur sa production pour les Sprints suivants.

Pour visualiser l'avancement du projet, Scrum propose des graphiques (4^e artéfact) comme le **Burndown Chart** et le **Burnup Chart**. Le premier part de la somme de travail à effectuer et montre sa diminution dans le temps ; le deuxième part de zéro et montre l'évolution de la production ; en général on ajoute à la courbe des productions réelles une droite représentant une production linéaire idéale. Ces graphiques peuvent aussi bien être établis sur la durée d'un Sprint (comptabilisation quotidienne des tâches réalisées) que sur la durée d'une Release (comptabilisation des Story Points réalisées en fin de chaque Sprint).

Exemples de graphiques Scrum :



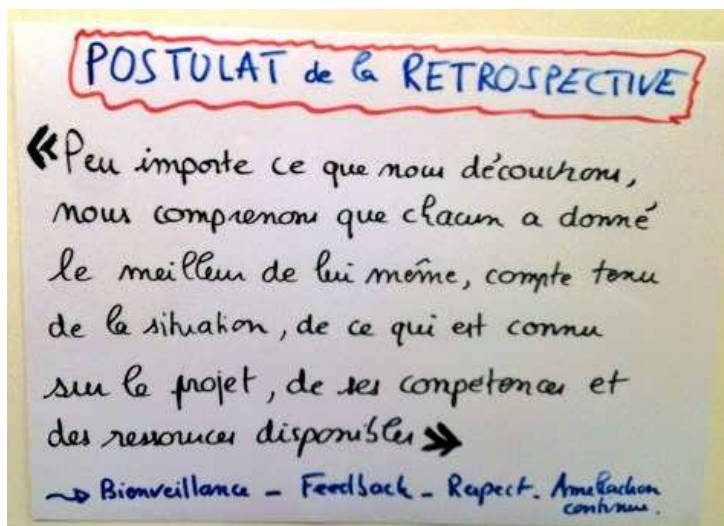
Agilité et Scrum



3.7.6 Rétrospective - Sprint Retrospective Meeting ➔ Adaptation

☀ Si la Revue de Sprint est l'occasion d'une inspection du produit, la **Rétrospective** est, quant à elle, le **moment privilégié d'inspection des processus mis en œuvre au cours du Sprint** de manière à identifier des **pistes d'amélioration** et prendre des mesures pour le Sprint suivant.

L'idée générale est que l'équipe de développement **apprend en continu**, de ses succès bien entendu, mais aussi de ses échecs et erreurs, et qu'il **est toujours possible de faire mieux** (principe d'amélioration continue).



Le Scrum Master anime cette réunion courte (typiquement 1/2 h à 3h) avec une neutralité bienveillante et en utilisant éventuellement des techniques favorisant la communication comme le *brainstorming*; il s'assure que chacun peut s'exprimer librement, sans agressivité ni jugement, en répondant aux 2 questions fondamentales : « Durant ce sprint, qu'est-ce qui a bien fonctionné et qu'avons-nous appris ? » et « Durant ce Sprint, qu'est-ce qui pourrait être amélioré, qu'est-ce qui m'a posé problème ? ».

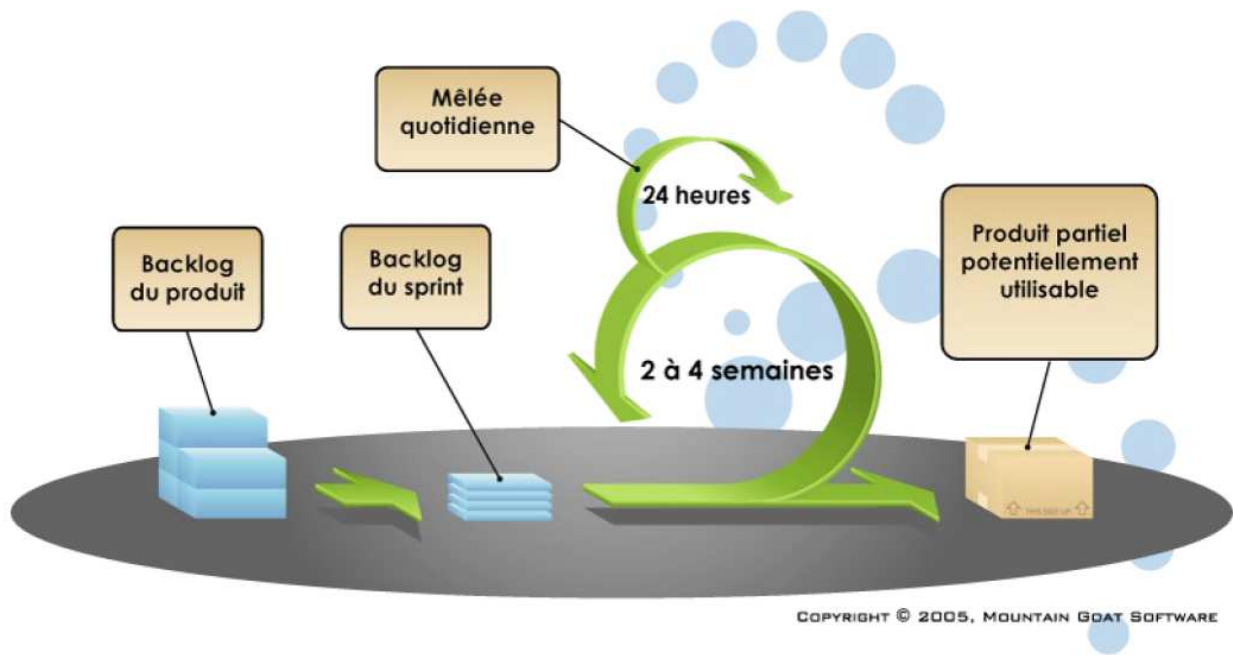
Le but est bien d'améliorer le fonctionnement, la productivité, la vélocité, mais pas de régler des comptes ou de vouloir tout améliorer d'un coup.

A l'issue de cette Rétrospective, l'équipe décide des actions à entreprendre lors du prochain Sprint, sur 3 axes : '*commencer à faire*', '*arrêter de faire*' et '*continuer à faire*'. Il s'agit bien de mesures très concrètes portant par exemple sur la tranche horaire du Daily Scrum Meeting, la clarification de la « Definition of Done », un processus de test ou la granularité (taille) des User Stories du Sprint Backlog. Il est conseillé que l'équipe s'engage sur 1 ou 2 décisions concrètes (que le Scrum Master s'attachera à faire appliquer lors du Sprint suivant).

Agilité et Scrum

Afpa © 2015 – Section Tertiaire Informatique – Filière « Etude et développement »

3.7.7 Pour résumer, les cycles Scrum en image



 Exercice Artéfacts et cérémonies Scrum : cocher la bonne case

Affirmation	Vrai	Faux
Le Product Backlog liste les fonctionnalités à développer lors du prochain Sprint	<input type="checkbox"/>	<input type="checkbox"/>
Une tâche terminée est supprimée du Scrum Board	<input type="checkbox"/>	<input type="checkbox"/>
Le Burndown Chart met en évidence les succès et les échecs	<input type="checkbox"/>	<input type="checkbox"/>
Le Backlog Grooming permet entre autres d'affiner des User Stories imprécises	<input type="checkbox"/>	<input type="checkbox"/>
Le Sprint Review est l'occasion de réfléchir sur les pratiques de l'équipe	<input type="checkbox"/>	<input type="checkbox"/>
Le Planning Poker permet de planifier les Releases	<input type="checkbox"/>	<input type="checkbox"/>
Tâche est le mot français pour User Story	<input type="checkbox"/>	<input type="checkbox"/>
Un Sprint Zéro est un Sprint raté n'aboutissant à aucune production	<input type="checkbox"/>	<input type="checkbox"/>
Un Persona ou un développeur, c'est pareil	<input type="checkbox"/>	<input type="checkbox"/>
Le Sprint Backlog liste les User Stories à développer lors du prochain Sprint	<input type="checkbox"/>	<input type="checkbox"/>
Un Sprint se termine quand toutes les tâches prévues sont placées en colonne DONE	<input type="checkbox"/>	<input type="checkbox"/>
Quand toutes les tâches d'une User Story sont placées en colonne DONE, on peut considérer la Story comme définitivement réalisée	<input type="checkbox"/>	<input type="checkbox"/>
L'équipe de développement prend en charge toutes les tâches d'une User Story avant de passer à la suivante	<input type="checkbox"/>	<input type="checkbox"/>
Lors des Rétrospectives, l'équipe de développement prend des mesures visant à conserver toujours la même vélocité	<input type="checkbox"/>	<input type="checkbox"/>

Agilité et Scrum

Afpa © 2015 – Section Tertiaire Informatique – Filière « Etude et développement »

Mis à jour le 15/07/2015 V1.1

30/40

La vélocité d'un développeur permet de décider de l'attribution de primes	<input type="checkbox"/>	<input type="checkbox"/>
Lors de la mêlée quotidienne, un développeur rencontrant une difficulté peut solliciter l'aide du Scrum Master	<input type="checkbox"/>	<input type="checkbox"/>
La Rétrospective, avec sa mesure de vélocité et ses engagements d'amélioration, a donc pour but d'améliorer la capacité et le fonctionnement de l'équipe pour le prochain Sprint	<input type="checkbox"/>	<input type="checkbox"/>

Réponses page suivante

Réponses : Artéfacts et cérémonies Scrum

Affirmation	Vrai	Faux
Le Product Backlog liste les fonctionnalités à développer lors du prochain Sprint Il contient l'ensemble priorisé des fonctionnalités envisagées pour produit à développer	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Une tâche terminée est supprimée du Scrum Board Elle est déplacée en colonne DONE.	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Le Burndown Chart met en évidence les succès et les échecs de l'équipe Les écarts entre la droite 'idéale' et la courbe des réalisations doivent être explicités avant d'être jugés ; ils peuvent être dus à d'erreurs d'estimation, à des difficultés non prévues ou l'absence de difficultés ou encore à des aléas imprévisibles ; en terme de gestion de projet, comme tout est nouveau, les estimations ne peuvent être fiables.	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Le Backlog Grooming permet entre autres d'affiner des User Stories imprécises	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Le Sprint Review est l'occasion de réfléchir sur les pratiques de l'équipe Non ; c'est le but des Rétrospectives.	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Le Planning Poker permet de planifier les Releases Non ; il permet juste d'estimer collectivement l'effort que nécessite le développement d'une User Story.	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Tâche est le mot français pour User Story Une User Story définit une fonctionnalité du point de vue d'un utilisateur alors qu'une tâche correspond à une opération technique nécessaire pour réaliser une partie d'une Use Story.	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Un Sprint Zéro est un Sprint raté n'aboutissant à aucune production livrable Un Sprint Zéro permet, en début de projet ou de Release, de partager la vision du produit et de s'entraîner à travailler ensemble. Il n'a pas de durée fixe ni d'objectif de production.	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Un Persona ou un développeur, c'est pareil Non ; un Persona est un personnage fictif décrivant personnalité et comportement d'un utilisateur du logiciel ; un développeur, ça existe !	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Le Sprint Backlog liste les User Stories à développer lors du prochain Sprint	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Un Sprint se termine quand toutes les tâches prévues sont placées en colonne DONE Un Sprint se termine à la fin de son timebox, que le travail prévu soit achevé ou non.	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Quand toutes les tâches d'une User Story sont placées en colonne DONE, on peut considérer la Story comme définitivement réalisée Non, car il reste encore à ce qu'elle soit acceptée par le PO (selon les critères d'acceptation de la Story et la définition du fini).	<input type="checkbox"/>	<input checked="" type="checkbox"/>
L'équipe de développement prend en charge toutes les tâches d'une User Story avant de passer à la suivante	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Agilité et Scrum

Non ; l'équipe de développement s'auto-organise pour assurer toutes les tâches du Sprint ; le Daily Scrum Meeting est un moment privilégié pour cette auto-organisation.		
Lors des Rétrospectives, l'équipe de développement prend des mesures visant à conserver toujours la même vitesse Le but des Rétrospectives est plutôt d'augmenter cette vitesse en adoptant des mesures d'amélioration.	<input type="checkbox"/>	<input checked="" type="checkbox"/>
La vitesse d'un développeur permet de décider de l'attribution de primes La vitesse ne se mesure qu'au niveau de l'équipe, qui est responsable collectivement du code produit ; de plus, la vitesse ne doit pas être utilisée pour comparer des équipes ni déclencher primes ou avertissements.	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Lors de la mêlée quotidienne, un développeur rencontrant une difficulté peut solliciter l'aide du Scrum Master Non ; le développeur sollicite l'aide de l'équipe afin qu'un autre membre de l'équipe se propose de lui venir en aide. De plus, le SM n'est pas un développeur ni une référence technique mais le garant de l'application de la méthode.	<input type="checkbox"/>	<input checked="" type="checkbox"/>
La Rétrospective, avec sa mesure de vitesse et ses engagements d'amélioration, a donc pour but d'améliorer la capacité et le fonctionnement de l'équipe pour le prochain Sprint C'est bien le but d'une Rétrospective, mais il ne s'agit pas d'une course à la vitesse ! Un principe de Scrum est que l'équipe doit adopter un 'rythme soutenable', et soutenable dans la durée ; à vouloir courir plus vite que ses jambes, on finit par chuter...	<input checked="" type="checkbox"/>	<input type="checkbox"/>

3.8 OUTILS D'AIDE A LA GESTION DE PROJETS SCRUM

Les méthodes Agiles et Scrum en particulier préconisent l'usage de **communication visuelle** à l'aide de Post-it, Paper Board et autre 'Sticky Wall' (mur collant), surtout pour l'équipe de développement qui œuvre dans un même espace. Il est malgré tout pratique et nécessaire de conserver le détail et la mémoire d'un certain nombre d'informations et de pouvoir les modifier ou préciser en tant que de besoin à l'aide d'outils informatiques ou bureautiques.


L'intérêt de tels outils réside dans leur capacité à croiser les informations (liste priorisée de Stories, liste de stories par Persona, liste de stories par Release...), ce qui est plus difficile à visualiser avec des outils papier.

Un **Tableur** comme Excel ou Google Sheet permet aisément d'enregistrer et mettre à jour Product Backlog et Sprint Backlog ; et les modules de graphiques de ces tableurs peuvent déjà représenter les Burndown et Burnup Charts.

Des **outils spécialisés** existent pour aider à la **gestion de projet classique** ; ils permettent de lister les tâches, de leur affecter des charges estimées, de les attribuer à des personnes, de calculer coûts et budgets, de suivre l'avancement des réalisations. C'est le domaine de Microsoft Project et autres Gantt Project. Mais ces outils ne sont pas très adaptés à la gestion Agile de projet (MS Project inclut maintenant des fonctionnalités orientées Scrum).

Des **outils spécifiques** apparaissent sur le marché. Parmi les plus utilisés, on peut citer **IceScrum** et **Jira Agile** qui s'appliquent aussi bien à des démarches Scrum que Kanban.

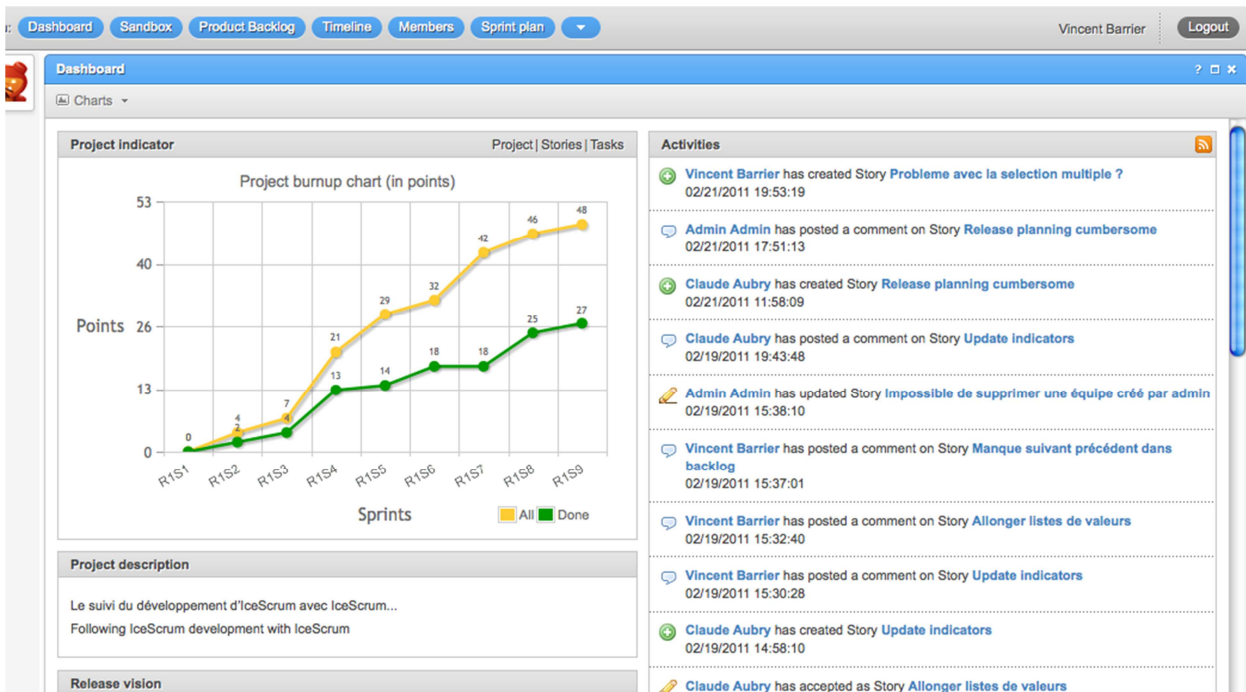
3.8.1 IceScrum

IceScrum, édité par l'association française iceScrum Foundation sous la direction avisée de Claude Aubry (référence française en Agilité et Scrum - voir <http://www.aubryconseil.com/>) est un **logiciel gratuit en version de base totalement orienté gestion de projets Scrum**. IceScrum est disponible en version serveur pour Windows ou MacOS, en version d'essai en ligne et en version complète *Cloud* payante. L'utilisation du logiciel ne nécessite qu'un simple navigateur Internet ( <http://www.icescrum.org/>).

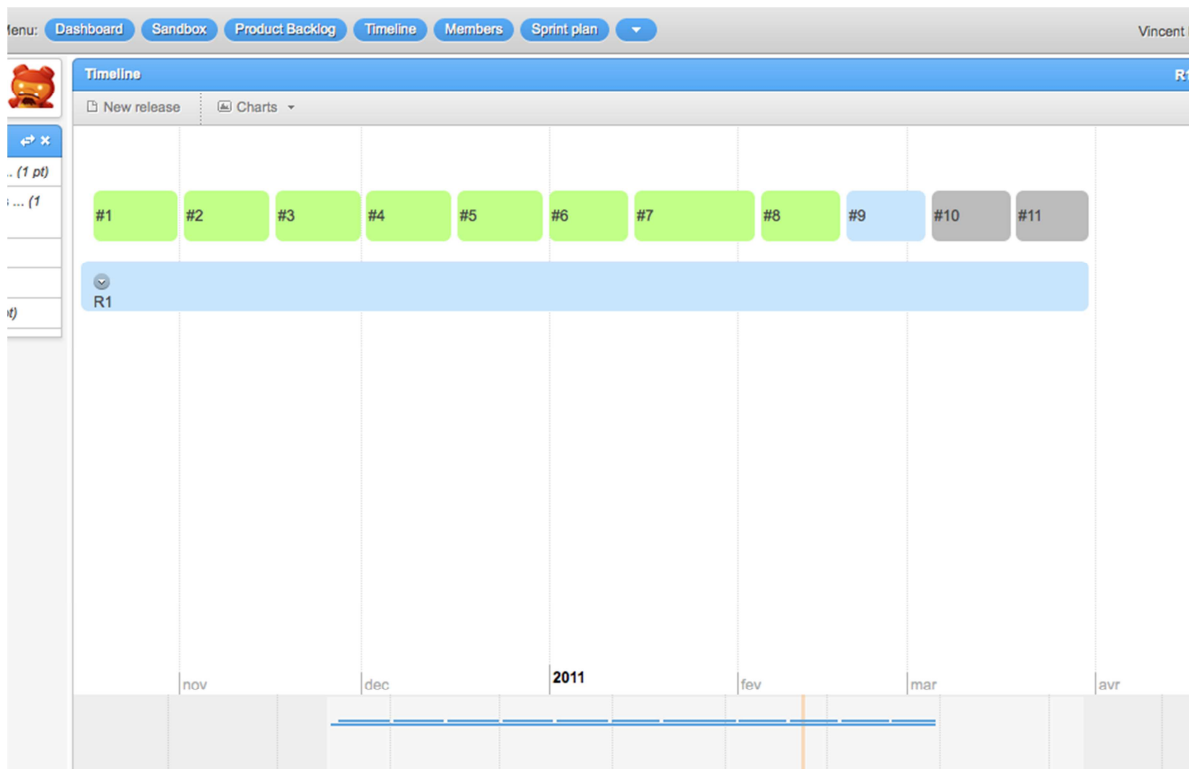
Il offre les fonctionnalités principales :

- ✓ Product Backlog complet accompagné d'un 'bac à sable' (*sandbox*) où les différentes parties prenantes peuvent déposer des souhaits de fonctionnalités à développer ;
- ✓ Plan de Release qui permet de ventiler les Stories sur une Roadmap virtuelle ;
- ✓ Liste des acteurs (Personas) affichant les Stories qui les concernent ;
- ✓ Liste des membres de l'équipe Scrum et droits associés ;
- ✓ Sprint Backlog détaillant les tâches nécessaires pour permettre de réaliser les Stories du Sprint et reproduisant un Scrum Board où les développeurs peuvent déplacer des Post it virtuels ;
- ✓ Tableau de bord complet incluant les Charts Scrum et les indicateurs.

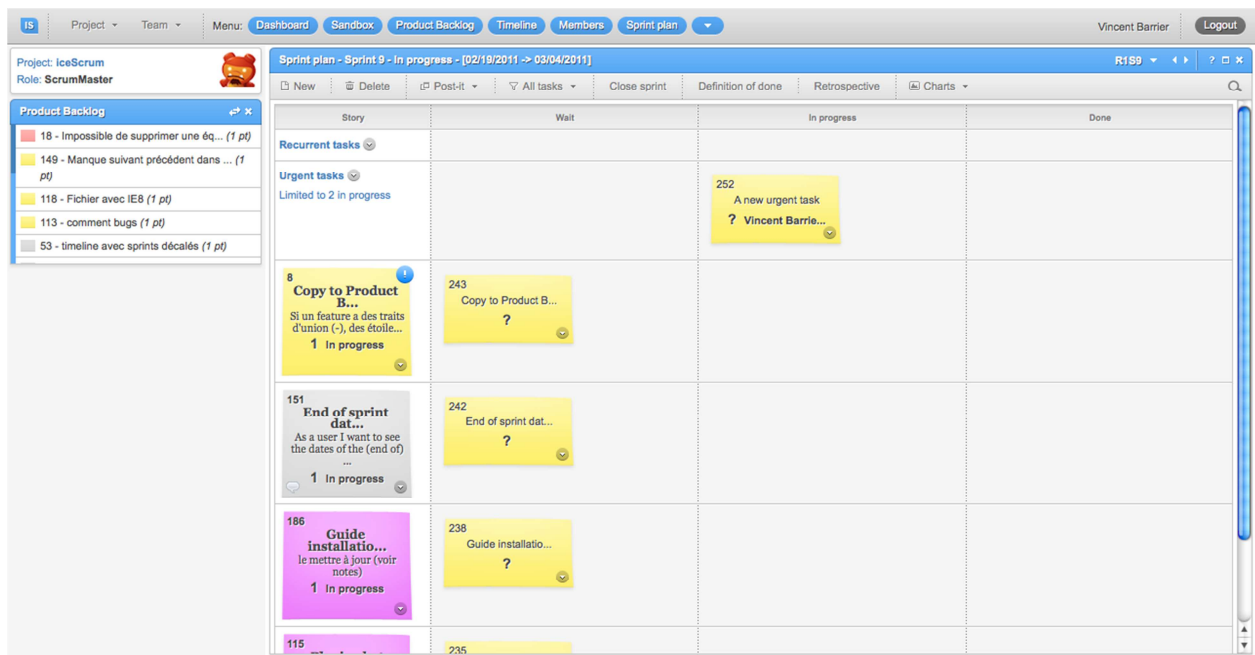
Exemples d'images-écran IceScrum :



Vue générale du projet incluant entre autres les indicateurs choisis, la version de la Release et la 'Definition of Done'.



Plan de Release




Scrum Board virtuel et ses Post it associés au Stories

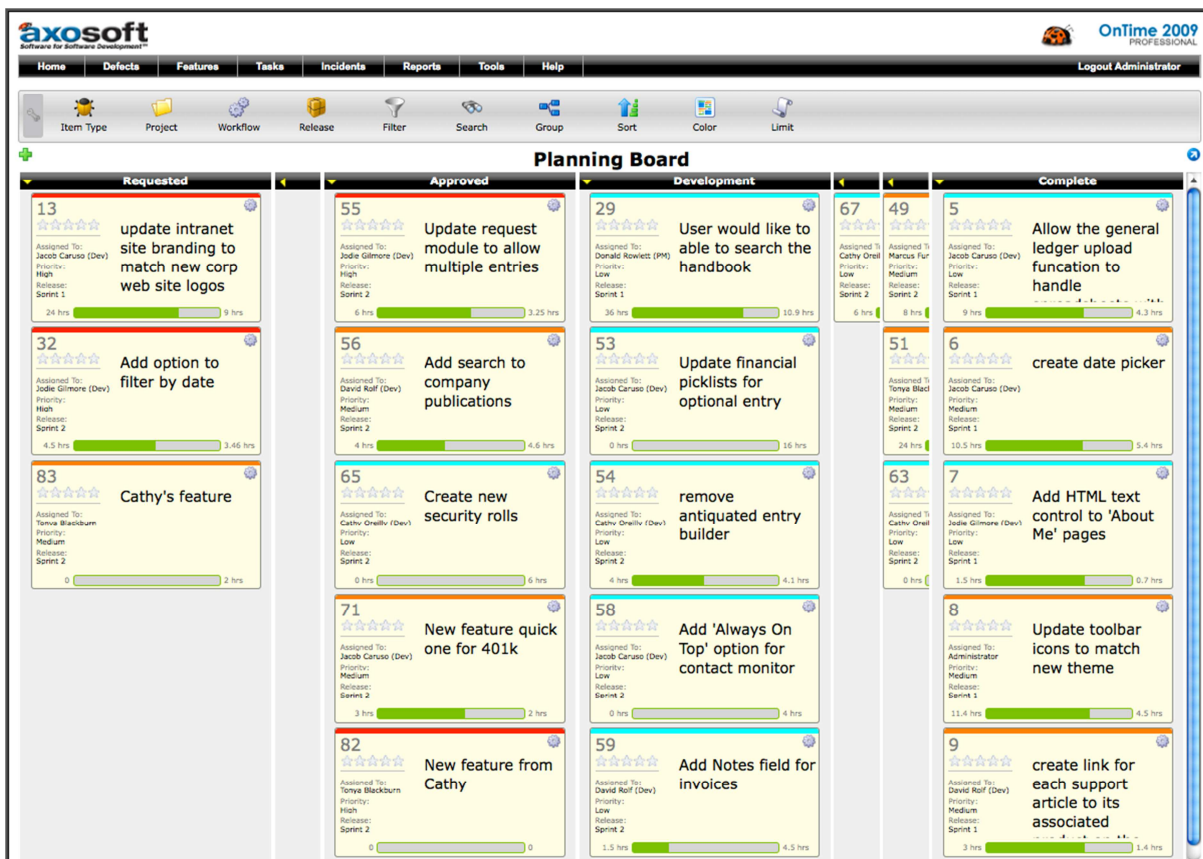
Agilité et Scrum

Afpa © 2015 – Section Tertiaire Informatique – Filière « Etude et développement »

3.8.2 JIRA Agile

Edité par Atlassian ( <https://fr.atlassian.com/software/jira/agile/>), **JIRA est le produit de référence** (payant) pour la gestion de projets Agiles, utilisé par de nombreuses grosses entreprises. Il reprend bien entendu les mêmes fonctionnalités que IceScrum, en offre d'autres, utiles pour de grands projets ou de grandes équipes, et s'intègre aux autres produits de l'éditeur comme Bitbucket (pour le travail de codage collaboratif sur le Cloud ou Confluence pour faciliter le travail collaboratif entre équipes distantes).

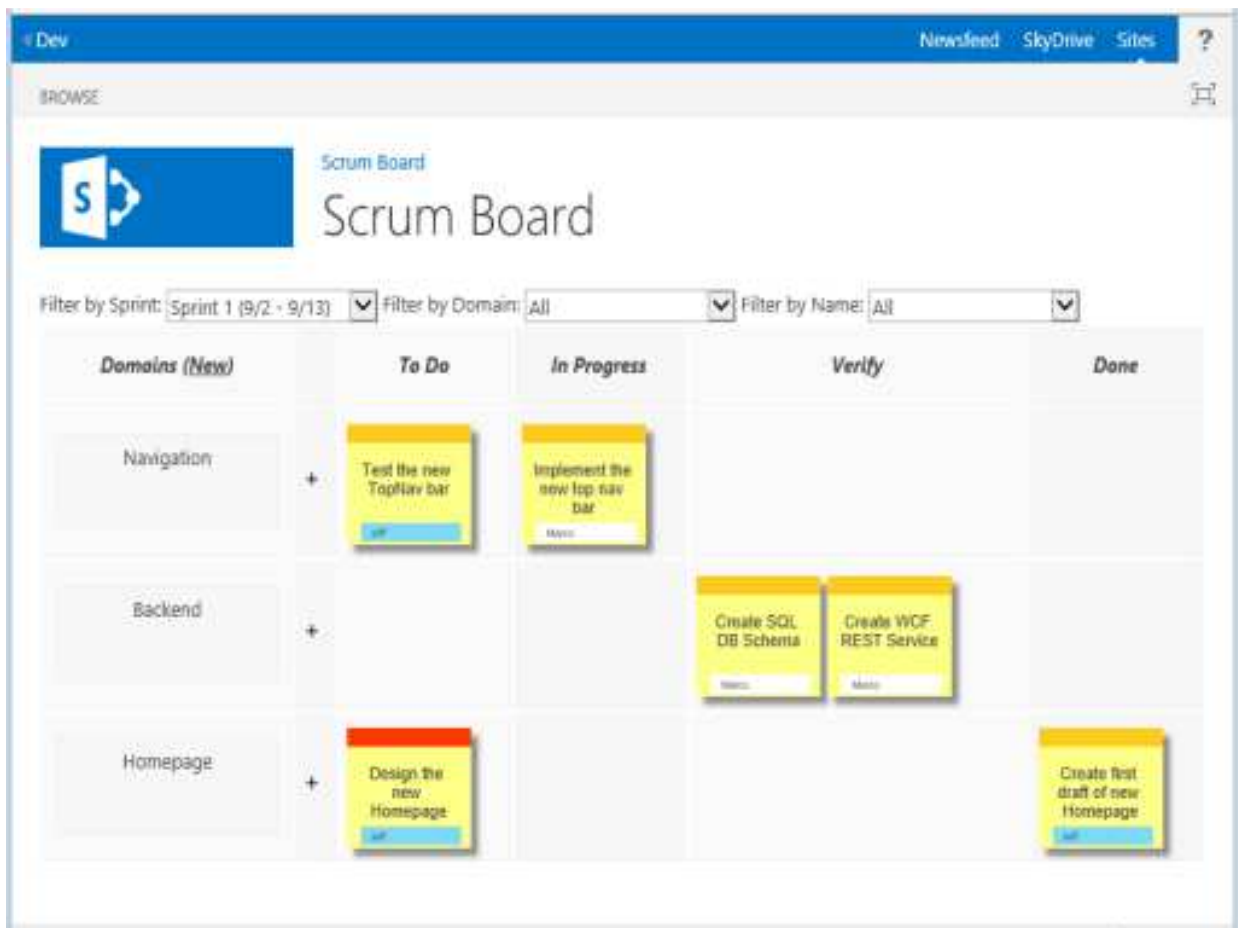
3.8.3 Autres produits



The screenshot displays the 'Planning Board' interface of AxoSoft's OnTime 2009 Professional software. The board is organized into four main columns: 'Requested', 'Approved', 'Development', and 'Complete'. Each column contains several task cards. Each card includes a task number, a star rating (1-5), a title, the assigned person, priority, release date, and a progress bar. The tasks are as follows:

Column	Task Number	Task Title	Assigned To	Priority	Release	Progress
Requested	13	update intranet site branding to match new corp web site logos	Jodie Gilmore (Dev)	High	Sprint 1	24 hrs / 9 hrs
	32	Add option to filter by date	Jodie Gilmore (Dev)	High	Sprint 2	4.5 hrs / 3.46 hrs
	83	Cathy's feature	Tonya Blackburn	Medium	Sprint 2	0 / 2 hrs
Approved	55	Update request module to allow multiple entries	David Rolf (Dev)	High	Sprint 2	6 hrs / 3.25 hrs
	56	Add search to company publications	David Rolf (Dev)	Medium	Sprint 2	4 hrs / 4.6 hrs
	65	Create new security rolls	Cathy Orell (Dev)	Low	Sprint 2	0 hrs / 6 hrs
Development	29	User would like to be able to search the handbook	Donald Rowless (PM)	Low	Sprint 1	36 hrs / 10.9 hrs
	53	Update financial picklists for optional entry	Jacob Caruso (Dev)	Low	Sprint 2	0 hrs / 16 hrs
	54	remove antiquated entry builder	Cathy Orell (Dev)	Low	Sprint 2	4 hrs / 4.1 hrs
Complete	67	Allow the general ledger upload function to handle	Marcus Fur	Medium	Sprint 2	6 hrs / 8 hrs
	49	create date picker	Jacob Caruso (Dev)	Low	Sprint 1	9 hrs / 5.4 hrs
	51	Add HTML text control to 'About Me' pages	Jodie Gilmore (Dev)	Low	Sprint 1	10.5 hrs / 0.7 hrs

Scrum Board virtuel de l'éditeur AxoSoft



Scrum Board virtuel dans Microsoft Project

3.9 SCRUM ET CONTRACTUALISATION

🔥 Le principal obstacle à la mise en œuvre d'une démarche Agile dans le cas d'une relation client/fournisseur entre entreprises différentes reste la contractualisation. En effet, dans une démarche classique, le client rédige le Cahier des Charges qui liste précisément toutes les fonctionnalités attendues. Ce document servira de base contractuelle avec son fournisseur qui s'engage alors sur un résultat en termes de qualité, de coûts et de délai.

En démarche Agile, rien de tout cela ! **Client et fournisseur doivent s'engager sur une vision imprécise et les moyens à mettre en œuvre pour avancer vers sa réalisation.** Toute demande de changement en cours de route est la bienvenue ; on accepte même de remettre en cause des réalisations antérieures si de nouvelles idées apparaissent meilleures. A l'issue de chaque Release (voire à l'issue d'un Sprint), il peut être décidé de ne plus poursuivre les développements. **Impossible dans ces conditions de réaliser un contrat basé sur des critères de qualité, coûts et délai !**

C'est pourquoi Scrum et eXtreme Programming sont plus souvent mis en œuvre pour des développements internes que dans le cas de véritables contrats entre entreprises.

Cependant, au fil des expériences, des juristes se sont penchés sur la question et des modèles de contrats existent qui proposent des solutions contractuelles nouvelles pour les problèmes de clauses de pénalité ou d'engagement financier itératif (📖 voir le document en annexe « z-Contrat_Agile_V1.1.pdf »).

Historiquement (le mouvement Agile a déjà plus de 10 ans), les méthodes Agiles étaient plutôt appliquées à des projets modestes mobilisant une seule équipe très motivée au sein d'une *startup* en pleine expansion. Aujourd'hui on parle **d'Agile à l'échelle** (*Scale Agile*) car tout cela peut s'étendre au travail collaboratif de plusieurs équipes, même réparties sur des zones géographiques éloignées, ce qui pose d'autres problèmes (de communication par exemple) pour lesquels des réponses voient le jour.

Scrum est maintenant adopté par certaines grandes ESN (Atos) ou équipes internes (Orange, Amadeus) pour développer aussi bien de grands que de petits projets ; les retours d'expérience actuels sont plutôt encourageants et permettent d'imaginer une adoption généralisée à moyen terme de ce cadre de travail.

3.10 EN GUISE DE BILAN

Scrum remet en question bien des habitudes de travail entre client et fournisseur de logiciel. On peut même dire que la pratique de l'Agilité induit une '*révolution culturelle*' dans l'entreprise (collaboration, auto-organisation, relations hiérarchiques bousculées, prévisionnel flou, acceptation du changement, communication visuelle, contractualisation...).

On peut oser dégager les principales forces, faiblesses, menaces et opportunités de Scrum :

<i>Points forts</i>	<i>Points faibles</i>
Scrum est axé sur la qualité du livrable (toute la méthode y concourt) et donne à voir tôt dans le projet	Scrum nécessite une forte implication et une présence importante du Product Owner
L'avancement du projet est mesurable et facilement visible (Scrum Board, Charts)	Scrum entraîne une remise en question des méthodes et de l'organisation du travail (maladie de jeunesse ?)
L'équipe est auto-organisée et peut prendre des décisions, ce qui favorise la motivation et l'implication des développeurs	Les cérémoniaux Scrum occupent un temps non négligeable parfois ressenti comme trop lourd (même s'ils sont <i>timeboxés</i>)
Les problèmes sont détectés tôt, avant qu'ils ne deviennent trop importants (Daily Meeting, Sprints courts, présence du PO)	

<i>Opportunités</i>	<i>Menaces</i>
Scrum permet un gain de productivité et de qualité qui séduit et fidélise les clients	Difficulté de contractualisation
Scrum se concentre sur la valeur Métier ce qui permet de livrer du logiciel plus 'rentable' pour le client	Aspect 'nouveau' qui est parfois ressenti comme 'risqué' (malgré 10 ans d'expérience)
Scrum permet d'arrêter un projet à moindre frais en cas de problème grave ou si la rentabilité est suffisante	

CREDITS

ŒUVRE COLLECTIVE DE l'AFPA

Sous le pilotage de la DIIP et du centre d'ingénierie sectoriel Tertiaire-Services

Equipe de conception (IF, formateur, mediatiseur)

Benoit Hézard - Formateur

Claudine Dupont – Formatrice

Ch. Perrachon – Ingénieure de formation

Date de mise à jour : 15/07/2015

Reproduction interdite

Article L 122-4 du code de la propriété intellectuelle.

« Toute représentation ou reproduction intégrale ou partielle faite sans le consentement de l'auteur ou de ses ayants droits ou ayants cause est illicite. Il en est de même pour la traduction, l'adaptation ou la reproduction par un art ou un procédé quelconque. »

Agilité et Scrum

Afpa © 2015 – Section Tertiaire Informatique – Filière « Etude et développement »