ECOLE PRATIQUE DES HAUTES ETUDES COMMERCIALES

INSTITUT D'ENSEIGNEMENT SUPERIEUR DE TYPE COURT

Intégration des technologies

Introduction à Scrum

Marie-Noël Vroman,
Virginie Van den Schrieck

Baccalauréat en :

- *Technologie de l'Informatique*

1. Préambule....3
2. Les méthodes Agile....4
2.1. Le Manifeste Agile....4
Les valeurs Agile :....4
Principes Agile....4
Pratiques Agile....5
3. Scrum : Généralités....6
4. Scrum : Sprints et releases....7
4.1. Les releases....8
Phase de début de release....9
4.2. Les sprints....10
Définition du sprint....10
Chronologie du sprint....10
La réunion de planification de sprint....10
Le scrum quotidien....12
La review de sprint....12
La rétrospective de sprint....13
5. Scrum : L'équipe et les rôles....14
5.1. Le Product Owner....14
5.2. Le Scrum Master....14
6. Le Product Backlog et les Stories....16
6.1. Les stories....16
Construction des stories....16
Tests d'acceptation pour les stories....17
6.2. Le Product Backlog....17
7. Synthèse....19

1. Préambule

L'objectif de ce document est de présenter de manière succincte les principes sous-jacents aux pratiques Agile et à détailler l'application de ces principes dans la méthode Scrum, afin de permettre aux étudiants de les mettre en pratique dans le cadre du projet.

Ce document constitue une synthèse des éléments détaillés dans les ouvrages suivants :

- [1] Claude Aubry, Scrum, le guide de la méthode agile la plus populaire, Ed. Dunod, 2010 (source principale)
- [2] Ken Schwaber et Jeff Sutherland, Le guide Scrum, trad. collective, juillet 2013
- [3] Jeff Sutherland's Scrum Handbook, disponible à http://www.misuratau.edu.ly/uploads/3K75P9W79B5HC4Q.pdf

Wikipedia peut également servir de point de départ au recueil d'informations sur les méthodes Agile et sur Scrum.

Chaque étudiant est invité à lire attentivement ce document, et à y revenir régulièrement au cours du projet en cas de problème ou d'interrogation quant à la marche à suivre pour la réalisation de ce dernier.

En plus des éléments présentés ici, et en fonction du rôle qui lui est attribué, chaque étudiant devra éventuellement approfondir certains aspects de la méthode Scrum. Ce document est donc destiné à servir de point de départ, mais en cas de manque de détails, il ne faut pas hésiter à explorer par soi-même la littérature Scrum disponible en ligne, ou bien à s'adresser à l'équipe enseignante.

En guise d'introduction, ce document commence par présenter les principes Agile. Ce chapitre est à lire une fois pour s'imprégner des valeurs de l'Agilité.

Au chapitre 3 sont exposées les grandes lignes de la méthode Scrum. Ce chapitre gagne à être lu attentivement par tous, afin de se faire une vision globale du processus.

Le document examine ensuite plus en détails les différents éléments de Scrum: Tout d'abord les éléments de temps (sprints et releases) au chapitre 4, puis, les différents rôles des participants à un projet Scrum (spécifiquement, le Product Owner et le Scrum Master) au chapitre 5, et enfin, au chapitre 6, les stories et le Product Backlog.

Le chapitre 7 conclut cette introduction à Scrum en reprenant une figure de Jeff Sutherland, résumant le cycle de vie d'un projet Scrum.

L'étudiant ayant le rôle de Product Owner lira attentivement la section du chapitre 4 correspondante, et s'assurera d'être familier avec les stories et le Product Backlog tels que présentés au chapitre 6. Il pourra d'ailleurs être nécessaire de consulter d'autres sources en complément afin d'assurer au mieux ce rôle.

Enfin, l'étudiant en charge du rôle de Scrum Master se doit de maîtriser le processus Scrum afin de servir de guide et de facilitateur à l'équipe. Il doit donc particulièrement être attentif aux jalons temporels présentés au chapitre 4 et notamment aux différents meetings, à ce qu'ils apportent, et à comment ils doivent se dérouler.

2. Les méthodes Agile

Les méthodes Agile sont un ensemble de méthodes de gestion de projets de développement informatique reposant sur des principes communs, qui sont repris dans le Manifeste Agile, rédigé en 2001.

Une définition possible reprise par Claude Aubry dans [1], adaptée de Scott Amber[1] est la suivante :

« Une méthode agile est une approche itérative et incrémentale pour le développement de logiciel, réalisé de manière très collaborative par des équipes responsabilisées, appliquant un cérémonial minimal, qui produisent un logiciel de grande qualité dans un délai contraint, répondant aux besoins changeants des utilisateurs. »

Les concepts mis en avant dans cette définition sont :

- **Une approche itérative et incrémentale** : Il s'agit de travailler par étapes, chaque étape enrichissant et améliorant la précédente.
- **Réalisation collaborative par des équipes responsabilisées** : Cette méthode requiert une implication et une motivation importante de chaque membre de l'équipe. Il s'agit de mettre en place un esprit de collaboration et de responsabilisation commune, à l'inverse d'une hiérarchie où les développeurs suivent et se reposent sur les directives de leurs responsables.
- **Un cérémonial minimal** : Il s'agit bien d'établir et de suivre des règles et des conventions aussi bien pour le produit que pour la vie en équipe, mais en visant l'efficacité et en minimisant la perte de temps qu'elles peuvent induire (ex : réunions interminables et inefficaces).

2.1. Le Manifeste Agile

Le Manifeste Agile, rédigé en 2001, fédère les pratiques Agile à travers quatre valeurs et douze principes :

Les valeurs Agile :

- Les personnes et leurs interactions (équipe soudée) passent avant les processus et les outils
- Un logiciel opérationnel prime sur une documentation exhaustive
- La collaboration avec le client est plus importante que le suivi à la lettre du contrat
- L'adaptation au changement est plus importante que le respect d'un plan

Ces valeurs se déclinent en douze principes :

Principes Agile

1. Satisfaire le client en livrant rapidement et régulièrement des fonctionnalités à forte valeur ajoutée
2. Accepter les changements même en fin de projet
3. Livraison fréquente d'un produit fonctionnel

[1] http://www.agilemodeling.com/essays/agileSoftwareDevelopment.htm

4. Collaboration régulière voire quotidienne entre le client et les développeurs
5. Implication et motivation forte des personnes impliquées avec mise à disposition de l'environnement et du soutien nécessaire + climat de confiance
6. Communication par conversation directe
7. Mesure de la progression sur base d'un logiciel fonctionnel (les fonctionnalités non terminées ne sont pas comptabilisées)
8. Rythme de travail durable (la fatigue nuit à la qualité !)
9. Accent mis sur l'excellence technique et la qualité de conception
10. Rechercher la simplicité et minimiser les tâches parasites
11. Auto-organisation de l'équipe pour optimiser l'architecture, les spécifications et la conception
12. Prise de recul régulière pour réfléchir à comment augmenter l'efficacité de l'équipe

Pratiques Agile

Il existe de nombreuses pratiques qui peuvent être définies comme Agile :

- **RAD** (Rapid Application Development) : Historiquement, la première méthode Agile. Se définit par opposition au modèle en cascade, et met l'accent sur le développement plus que sur la planification, et sur l'ajustement des spécifications en réponse à l'évolution des connaissances au fur et à mesure du développement.
- **DSDM** (Dynamic Systems Development Method) : Evolution de RAD, met l'accent sur l'implication de l'utilisateur
- **Scrum** : Méthode/cadre de développement basé sur des itérations de taille fixe (Sprints), commençant par une séance de planification et se terminant par une démonstration du produit et une rétrospective visant à améliorer le processus.
- **Feature Driven Development** : Imaginé au départ pour de grandes équipes utilisant des technologies orientées-objet, FDD se caractérise par cinq phases, dont notamment une phase initiale dédiée à la modélisation.
- **Extreme Programming** (XP) : En plus de principes déjà cités ailleurs comme l'implication du client et des cycles courts avec livraisons fréquentes, XP reprend des bonnes pratiques de développement comme le travail en binôme, l'intégration continue, des conventions de nommage, des tests unitaires (TDD) et fonctionnels, le refactoring, etc.

En pratique, une équipe peut combiner plusieurs approches Agile. Par exemple, Scrum définit un cadre pour la gestion du projet au niveau chronologique et au niveau de la production des fonctionnalités, mais ne prescrit rien concernant les techniques de développement. Il est donc fréquent de combiner Scrum avec des pratiques issues de XP (programmation en binôme, intégration continue, TDD, …).

3. Scrum : Généralités

Scrum permet de développer un produit en définissant initialement un **Product Backlog** (amené à évoluer au fil du projet), qui est une liste de fonctionnalités classées par priorité. Le développement est divisé en cycles, appelés **releases**, chaque release étant elle-même découpée en **sprints**. Au début de chaque sprint, développeurs et client se mettent d'accord sur les fonctionnalités à développer, en tenant compte des priorités et de la capacité de l'équipe.

Le mot Scrum vient du rugby, et signifie « mêlée ». L'idée est qu'au cours d'un sprint, l'équipe se réunit pour le « **scrum** » quotidien, courte réunion destinée à faire le point sur l'avancement du projet et faire des ajustements pour assurer le succès du sprint.

A la fin de chaque sprint, l'équipe a produit un incrément, c'est-à-dire un ensemble de fonctionnalités supplémentaires **fonctionnelles** du logiciel. Le logiciel avec le nouvel incrément est utilisable et potentiellement livrable. L'évaluation de cet incrément permet d'ajuster le backlog pour le sprint suivant.

Les principes de Scrum peuvent être organisés en trois axes: les unités de temps (timeboxes), les rôles et les artefacts.

Les unités de temps, fixes, sont d'une part les **releases**, qui aboutissent à une version livrable du logiciel, releases qui sont elles-mêmes composées d'itérations appelées **sprints**. Lors des sprints se dérouleront diverses réunions, de durée limitée : Le **planning**, les **scrums** quotidiens, la **review** et la **retrospective**.

Au niveau des rôles, nous en distinguerons trois :

- Le **Product Owner** : Responsable du produit, il collecte les informations auprès des clients pour savoir ce qui est attendu, et traduit ces inputs en fonctionnalités à ajouter au **Product Backlog**. Son objectif est de maximiser la valeur du logiciel produit.
- Le **Scrum Master** : Il est responsable du bon déroulement du travail en équipe. Il joue le rôle de facilitateur lors des réunions, essaie d'optimiser l'organisation, et s'assure qu'aucune contrainte extérieure inutile ne vienne perturber le bon fonctionnement du projet. C'est l'expert chargé de mettre en place le contexte Scrum.
- **L'équipe de développement** : Les développeurs responsables de la production de l'incrément prévu pour chaque sprint, sur base du backlog produit.

En ce qui concerne les artefacts, nous soulignerons essentiellement le **Product Backlog**: c'est une liste prioritisée des fonctionnalités attendues du logiciel à produire. Cette liste est maintenue par le Product Owner, mais accessibles à tous les participants au projet. Souvent, les fonctionnalités du Product Backlog sont appelées **stories**. Les stories sont des descriptions de haut niveau, **compréhensibles par l'utilisateur**, des fonctionnalités du produit. C'est dans le Product Backlog que seront sélectionnées les stories à produire lors d'un sprint. A cette occasion, les stories seront détaillées par l'équipe de développement en une série de tâches dont la description sera alors plus technique.

Nous allons détailler chacun de ces éléments dans la suite de ce document.

4. Scrum : Sprints et releases

Chaque technique de gestion de projet, et plus spécifiquement chaque projet, a un déroulement temporel spécifique appelé cycle de développement, ou cycle de vie, défini par des phases ou des jalons. Des modèles de cycles de développement bien connus sont le développement en cascade ou encore le développement en V.

Scrum suit un modèle itératif : il est basé sur une phase qui se répète plusieurs fois, l'itération, appelée **sprint**. Il est également incrémental : le produit est construit morceau par morceau, la partie construite au cours d'un sprint venant s'ajouter à l'existant.

L'aspect itératif de ce mode de développement permet de revenir sur ce qui a été fait (feedback à la fin d'un sprint), donnant ainsi l'opportunité à l'équipe non seulement d'enrichir le produit, mais également d'améliorer les aspects dont la qualité n'est pas satisfaisante.

En plus d'être itératif et incrémental, Scrum est classé parmi les pratiques Agile entre autres parce que les itérations sont courtes (maximum un mois), suivent une séquence stricte (pas de chevauchement) et un rythme régulier (durée fixe et constante).

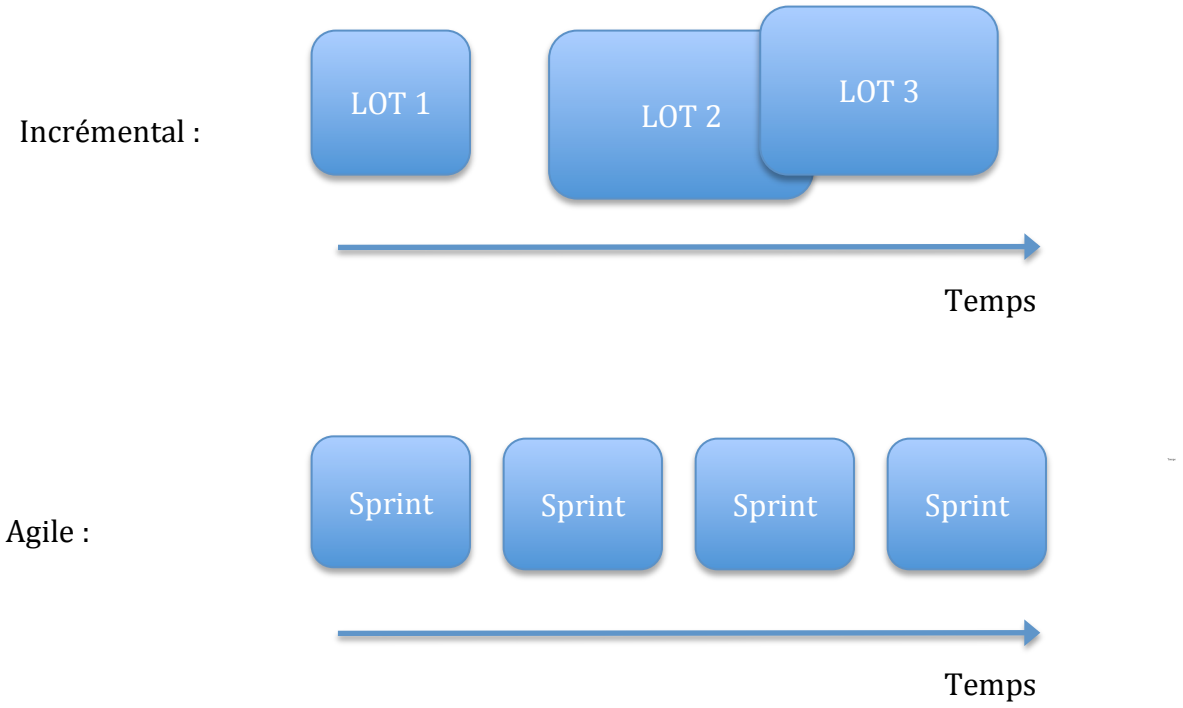

Figure 1 : Incrémental versus Agile. Image tirée de [1]

Le fait que la durée d'un sprint soit fixe et non modifiable est important : Cela permet d'éviter la tentation de reporter la date de fin, par exemple parce qu'une fonctionnalité est presque terminée. A la date de fin, ce qui est terminé est comptabilisé dans le bilan, le reste sera remis dans le Product Backlog et repris en considération dans la planification des sprints suivants. En cas de difficulté pour tenir le planning ou en cas de modification de la taille de l'équipe, il n'y a donc pas lieu de modifier la durée de l'itération, mais au contraire, d'ajuster la quantité de travail à fournir à chaque itération.

4.1. Les releases

Initialement, dans le jargon informatique, une release est une version d'un logiciel fonctionnelle et mise sur le marché. Par extension, on appelle aussi release la période de temps qui permet de la produire, c'est-à-dire son cycle de production. Une release est constituée de plusieurs sprints (typiquement 4 à 6), et se termine lorsque les incréments successifs constituent un produit présentant suffisamment de valeur aux yeux de ses utilisateurs.

Une release sera constitué de trois phases :

1. **La période de début de la release**, qui sera consacrée à la construction du Product Backlog (choix et prioritisation des fonctionnalités et estimation du temps nécessaire) et à une première planification de la release. S'il s'agit de la première release d'un logiciel, il faudra également préciser sa définition et réfléchir à son architecture.
2. **Les sprints** : les sprints s'enchaînent les uns après les autres, après éventuellement une courte pause de récupération.
3. **La fin de la release** : Idéalement, l'état du produit à la fin du dernier sprint doit correspondre à l'état attendu à la fin de la release. Parfois, ce n'est pas possible, et il est nécessaire d'effectuer quelques travaux avant la mise en production (packaging, mise à disposition en ligne, …)

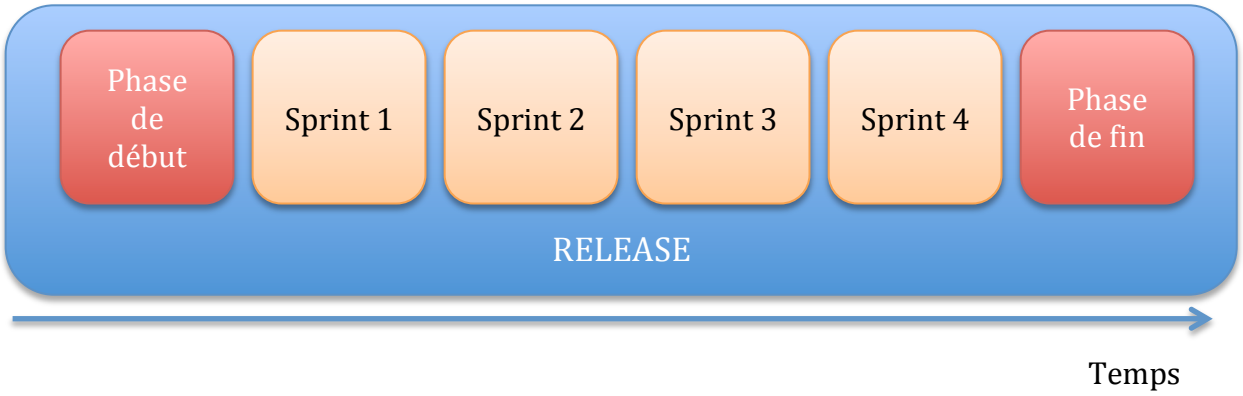

Figure 2 : Les trois périodes d'une release

Au cours d'une release, différentes versions du logiciel seront produites. On peut distinguer trois usages de ces versions intermédiaires :

- Les versions utilisées **en interne par l'équipe** de développement, par exemple pour valider l'intégration d'éléments au fil d'un sprint.
- Les versions utilisées pour **feedback par des utilisateurs** sélectionnés, qui jugent de la facilité d'utilisation des fonctionnalités produites et en proposent éventuellement des nouvelles pour alimenter le backlog produit. De telles versions sont typiquement produites à la fin de chaque sprint pour la démo.
- Les versions mises en production, **utilisables par les utilisateurs finaux**. Généralement, ces versions ne sont pas mises en production avant la fin de la release, à cause de l'effort nécessaire pour la mise en production, la validation du système, la finalisation de la documentation pour les utilisateurs, ou encore la formation de ces derniers.

Dans le cadre du projet, une seule release sera effectuée. Le Product Backlog initial sera construit dans la phase de début de release, et la version de fin de release sera la version finale du produit

Phase de début de release

La phase de début de release est importante, car elle permet l'organisation à moyen terme du projet, et la manière dont les sprints vont s'enchaîner en fonction du Product Backlog. Le **plan de la release** sera ainsi esquissé dans ses grandes lignes. Il n'est pas figé, et sera amené à évoluer au fur et à mesure des sprints.

Les éléments à définir dans cette phase sont les suivantes :

- Le **Product Backlog**, s'il s'agit de la première release, c'est-à-dire la liste des stories composant le produit. Ces stories seront classées par ordre de priorité. C'est le Product Owner qui est responsable de cette tâche. Toutes les stories ne doivent pas être décomposées au même niveau : la décomposition n'a besoin d'être précise que dans le cas des stories qui seront réalisées prochainement, donc dans les premiers sprints. Les autres, moins prioritaires seront raffinées au fur et à mesure de l'avancement du projet. Le Product Backlog sera donc composé d'abord de petites stories précises à haute priorité, puis de stories plus larges, moins précises, à réaliser ultérieurement. De la sorte, les stories qui vont être réalisées seront suffisamment précises pour être comprises sans ambiguïté par l'équipe de développement.
- La **définition de fini :** Il est important que l'équipe se mette d'accord sur les conditions à remplir pour qu'une étape soit considérée comme finie. Il convient de discuter de l'état « fini » pour une story (ex : tests unitaires et d'acceptation passés, documentation rédigée, …), un sprint (ex : déploiement en environnement de test, couverture minimum pour les tests, critères de qualité du code, ..), et une release (ex : déploiement en production).
- **L'estimation en points du temps nécessaire** pour les stories les plus prioritaires (et donc, celles qui ont été le plus détaillées). Un moyen d'effectuer collectivement ces estimations est le **Planning Poker**[2]. Cette estimation doit tenir compte du temps nécessaire aux tests et au contrôle des conditions pour que la story soit considérée comme finie.
- La **capacité de l'équipe**, c'est-à-dire la quantité de travail par unité de temps qu'elle peut effectuer. Initialement, cette capacité est difficile à estimer, mais au fur et à mesure des sprints et des releases, l'équipe garde une trace du nombre de points qu'elle a produit dans chaque sprint. Elle pourra donc alors estimer la capacité de l'équipe sur base de ces valeurs.
- Et enfin, la **planification de haut niveau des sprints**, c'est-à-dire quelles stories seront réalisées durant quel sprint. Cette planification se base bien entendu sur la capacité de l'équipe et sur l'estimation du temps nécessaire à réaliser les stories sélectionnées.

Il est important que tous les membres de l'équipe Scrum soient impliqués dans cette phase : L'équipe de développement, le Product Owner et le Scrum Master.

---

[2] Nous ne détaillerons pas le Planning Poker ici, mais de plus amples informations peuvent être trouvées par exemple sur http://referentiel.institut-agile.fr/poker.html, ou bien entendu sur Wikipedia : http://fr.wikipedia.org/wiki/Planning_poker
Des outils sont aussi disponibles en ligne : http://www.planningpoker.com (outil online) ou http://www.entreprise-agile.com/CartesPlanningPoker.pdf (cartes à imprimer).

4.2. Les sprints

Définition du sprint

Comme mentionné plus haut, un **sprint** est un bloc de temps fixé aboutissant à un **incrément *potentiellement* livrable** du produit.

La durée de ce bloc de temps est donc fixe et constante. Initialement, Scrum définissait la durée d'un sprint à un mois, mais aujourd'hui, on constate une tendance à faire des sprints plus courts [1]. Dans le cadre du projet, la durée d'un sprint sera fixée à deux semaines.

L'incrément produit au cours d'un sprint doit être utilisable par le client. Il sera donc nécessaire, au cours de ce sprint, de se livrer à plusieurs types d'activités :

- Les spécifications fonctionnelles (requirements)
- L'architecture (conception)
- Le codage (avec tests unitaires)
- La validation (tests d'intégration et d'acceptation)
- La production de la documentation pour l'utilisateur

Contrairement au modèle en V, avec Scrum, ces quatre activités se font en parallèle. Les activités de spécification et d'architecture sont continues et évoluent (dans une certaine limite) tout au long du projet. Un sprint n'est pas non plus un « mini-cycle en V », avec des phases bien distinctes. L'équipe travaille à développer des fonctionnalités (stories) dès le début du sprint, et effectue en fait des micro-incréments tout au long de ce sprint. Ces micro-incréments produisent des versions intermédiaires utilisées par l'équipe pour passer les tests fonctionnels.

Chronologie du sprint

Un sprint est composé de différentes étapes :

1. **La réunion de planification de sprint** : Sélection et raffinement en tâches des stories à développer durant le sprint.
2. Le développement des fonctionnalités, avec le **scrum quotidien** d'un quart d'heure
3. La **review de sprint** (avec démo)
4. La **rétrospective** (feedback) de sprint

La réunion de planification de sprint

Au début de chaque sprint, l'équipe complète se réunit. Le but est de revenir sur la partie du plan de release qui concerne le sprint, et de détailler en tâches les stories qu'il faut produire. Le résultat attendu de la réunion est un tableau en trois colonnes (« à faire », « en cours », « fini ») , sur lesquels sont placés des post-it représentant les stories et les tâches (ce tableau peut aussi être sous forme numérique), et qui servira d'outil de travail collaboratif durant le sprint. On rappelle également sur ce tableau les critères adoptées par l'équipe pour considérer finis une tâche, une story et un sprint.

Préalablement à la réunion, le Product Backlog doit avoir été préparé par le Product Owner, c'est-à-dire :

- Les stories ont été rangées par priorité
- Les stories les plus prioritaires ont été estimées (points)
- Les stories associées au sprint qui commence sont suffisamment détaillées et connues.

Si la réunion de début de release a été efficace, ces éléments sont normalement prêts.

La **première partie** de la réunion consiste à définir le contexte et à rappeler le but du sprint (« Quoi ») :

- Dates de début et de fin du sprint, éventuels spécificités (jours fériés, absences de certains membres, …)
- Rappeler les éléments du Product Backlog qui sont prévues dans le plan de release, et éventuellement les rajuster.
- Définir en une phrase l'objectif principal du sprint (ex : authentification des utilisateurs)

La **seconde partie** de la réunion va permettre à l'équipe de s'organiser pour la réalisation des stories sélectionnées. En partant de la liste des stories sélectionnées, l'équipe va **identifier les tâches à effectuer**. Il est donc nécessaire de discuter des solutions techniques à mettre en œuvre, et le cas échéant, l'équipe se tournera vers le Product Owner pour obtenir des précisions sur le comportement attendu.

Toutes les activités liées au développement d'une stories doivent être prises en compte : Lecture de documents, refactoring de code, production de manuel utilisateur, etc. D'autres tâches, non liées aux stories, peuvent s'ajouter à la liste (ex : mise en place d'un serveur de test, participation à une conférence, etc.)

Il faut ensuite estimer le temps de travail nécessaire pour chaque tâche. Idéalement, cela se compte en heures, chaque tâche devant être réalisable en moins d'une journée de travail. Si la tâche semble nécessiter plus de temps, il convient alors de la redécouper en sous-tâches.

Enfin, chaque membre de l'équipe prend en charge une série initiale de tâches. Toutes les tâches ne doivent pas être sélectionnées dès le début, celles qui ne le sont pas le seront au fur et à mesure du sprint.

Pour chaque tâche, on définira donc les éléments suivants :

- Un nom et la description du travail à faire
- La story associée
- Une estimation en temps du reste de travail à faire pour finir la tâche
- Le nom de la personne responsable de la tâche

Le découpage en tâches peut amener l'équipe à revoir à la hausse ou à la baisse le nombre de stories à effectuer durant le sprint. Il est donc important que le Product Owner reste disponible pour participer à cette discussion.

Le scrum quotidien

L'objectif du scrum quotidien est de maximiser la probabilité que l'équipe atteigne les objectifs du sprint. Il faut pour cela :

- Eliminer les obstacles nuisant à la progression de l'équipe
- Garder l'équipe concentrée sur l'objectif du sprint
- Evaluer l'avancement du travail pour le sprint en cours
- Communiquer objectivement sur l'avancement

Toute l'équipe participe au scrum, y compris le Scrum Master, et, idéalement, le Product Owner. Le Scrum se tient devant le tableau des tâches qui aura été mis à jour au fur et à mesure de la réalisation des tâches. Il doit être quotidien, et durer moins d'un quart d'heure. Vu sa brièveté, il est indispensable que tous les membres se soient éloignés de leurs claviers, écrans, tablettes, GSM et smartphones…

Trois questions sont successivement posées à chaque membre de l'équipe :

- « Qu'as-tu fait depuis le dernier scrum ? » (tâches en cours, finies, …)
- « Que prévois-tu de faire d'ici le prochain scrum ? » (décrire chaque tâche et indiquer si elle sera finie dans la journée)
- « Quels sont les obstacles qui te freinent dans ton travail ? » Si quelqu'un possède la solution immédiate au problème, il l'indique, sinon, si une discussion s'engage, elle est reportée ultérieurement au scrum.

Après ces questions vient une **éventuelle adaptation des objectifs du sprint**, après discussion avec le Product Owner sur l'ajout ou du retrait d'une story. Si l'équipe dispose d'un peu de temps, mais pas suffisamment pour prendre une nouvelle story, il peut être utile de consacrer du temps à l'amélioration de la qualité (ex : refactoring de code).

La review de sprint

La review de sprint intervient en fin de sprint, et son objectif est de montrer ce qui a été réalisé afin d'en tirer des enseignements pour la suite du projet. Une démonstration du produit est réalisée, devant l'équipe et toutes les autres personnes impliquées dans le projet, voire éventuellement des invités extérieurs. La durée typique d'une review de sprint pour un sprint de deux semaines est d'une heure ou deux. La démonstration se fait uniquement sur base de la version opérationnelle du produit, il n'y a donc pas lieu de préparer des slides. Les étapes de la review sont les suivantes :

1. Préparer la démonstration (environnement, vidéo-projecteur, connexions réseaux, …)
2. Rappeler les objectifs du sprint (Product Owner), avec la liste des stories prévues/montrées
3. Effectuer la démonstration (uniquement stories terminées), idéalement par un binôme Product Owner – membre de l'équipe de développement. Il est intéressant de recueillir le feedback du public.
4. Calculer la quantité de travail produite par l'équipe durant le sprint (appelée **vélocité**) en additionnant les points de toutes les stories répondant à la définition de « fini » telle que décidée par l'équipe. La moyenne des valeurs obtenues à chaque sprint donnera **l'estimation de la capacité de travail de l'équipe**.

5. Ajuster le plan de release en fonction des modifications éventuelles appliquées au stories (ajout, retrait, modification de la priorité, ...), ou de l'évolution de la capacité de travail de l'équipe.

Le Product Owner ne doit bien sûr pas oublier de mettre à jour le Product Backlog à l'issue de ce sprint, en déplaçant les stories terminées dans la zone prévue à cet effet. En outre, le feedback des participants a pu donner lieu à de nouvelles stories, qui doivent donc être ajoutées au Product Backlog.

La rétrospective de sprint

Le but de la rétrospective est de permettre à l'équipe de faire le bilan sur son mode de fonctionnement et d'y apporter éventuellement des modifications dans le but d'améliorer son efficacité. Cette réunion dure de l'ordre d'une heure, et est généralement consécutive à la review de sprint. La réunion regroupe toute l'équipe Scrum, et est animée par le Scrum Master, ou éventuellement par une personne extérieure. L'objectif est bien l'apprentissage et l'amélioration, et non stigmatisation des erreurs.

Les étapes à suivre sont :

1. Créer un environnement propice à l'expression (climat de confiance)
2. Collecter les informations sur le sprint (qu'est ce qui a bien/mal fonctionné, que pouvons-nous améliorer ?) et sur les obstacles/péripéties rencontrés
3. Identifier des idées d'amélioration
4. Regrouper les idées en catégories
5. Définir la catégorie prioritaire (pour un sprint de deux semaines, une est suffisante)
6. Adapter le mode de fonctionnement

Le but de la réunion est de produire une liste d'actions concrètes que l'équipe s'approprie et met en place pour le prochain sprint.

5. Scrum : L'équipe et les rôles

Une équipe scrum définit trois rôles : Le Product Owner, qui est le responsable du produit, le Scrum Master, qui cherche à faciliter le processus de développement, et les développeurs, qui, en plus d'être parties prenantes des réunions et des décision, s'occupent de la réalisation technique du produit. Nous allons détailler ici les deux premiers, sachant que les développeurs sont en charge de la réalisation des stories, et sont invités à s'impliquer dans l'ensemble des décisions liées au projet (responsabilité collective).

5.1. Le Product Owner

Le Product Owner est la personne de référence pour la vision du produit, et est donc responsable des choix stratégiques liés aux fonctionnalités de ce dernier. Son rôle est triple. Il doit :

1. Fournir une **vision partagée** du produit : Elle consiste à définir l'énoncé du problème que le produit doit résoudre, un positionnement clair du produit, et une liste de ses fonctionnalités essentielles. Le Product Owner doit s'assurer que toutes les parties prenantes du projet comprennent et partagent cette vision.
2. **Définir le contenu** du produit : Le Product Owner identifie les fonctionnalités requises (en partenariat avec l'équipe), les collecte en une liste de stories qui constituent le Product Backlog. Il en est responsable, et le met à jour continuellement en fonction de l'avancement du travail et de la nécessité de préciser les descriptions de ces stories.
3. **Planifier la vie du produit :** Le Product Owner définit l'ordre de réalisation des fonctionnalités du produit d'une part en fixant les priorités des stories, et d'autre part en définissant les objectifs et les échéances des releases. `

Le Product Owner doit posséder une bonne connaissance du domaine métier, être capable de prendre des décisions rapidement, savoir quand détailler les fonctionnalités, être ouvert au changement et apte à la négociation. Il doit être motivé pour ce rôle.

Il est également important que le Product Owner soit la seule personne de référence concernant la vision du produit. C'est à lui d'aller à la rencontre des utilisateurs finaux du produit, mais aussi du client ou des sponsors impliqués dans le projet.

Le Product Owner doit connaître et utiliser quotidiennement le produit afin d'avoir une perception claire de sa qualité, de son ergonomie et de la manière dont les fonctionnalités ultérieures s'y intégreront. Il va aussi **s'impliquer dans les tests d'acceptation**, puisqu'il doit s'assurer que le travail fait par l'équipe est effectivement fini à la fin d'un sprint.

5.2. Le Scrum Master

Le rôle de Scrum Master est différent du rôle traditionnel de chef de projet, dans le sens où il n'y a pas de hiérarchie : Les décisions sont prises conjointement par tous les membres de l'équipe afin de garantir leur implication et leur responsabilisation.

Le rôle essentiel du Scrum Master est d'aider l'équipe à appliquer Scrum et à l'adapter au contexte. Il joue un rôle de facilitateur et influence la façon de travailler. Alors que le

Product Owner est responsable du produit, le Scrum Master est quant à lui responsable du processus. Ses responsabilités sont :

- Veiller à la mise en application de Scrum en organisant les différentes réunions et en s'assurant qu'elles se déroulent dans le respect des règles établies conjointement par l'équipe.
- Encourager l'équipe à apprendre et à progresser
- Faire en sorte d'éliminer ou de prévenir les obstacles qui pourraient freiner l'avancement (ex : un développeur malade, un serveur Git en panne, …)
- Inciter l'équipe à devenir autonome.

Le Scrum Master doit avoir une bonne connaissance de Scrum, être capable de comprendre les aspects techniques du projet, pouvoir communiquer et guider facilement, être un bon médiateur, et apprécier se mettre **au service de l'équipe.**

Dans de petites équipes, le Scrum Master peut également participer aux travaux de développement, mais cela doit rester limité et le rôle de Scrum Master doit primer sur les autres tâches. Le rôle de Scrum Master peut éventuellement tourner tous les quelques sprints.

Dans l'idéal, et contrairement au Product Owner dont l'implication augmente au fur et à mesure du projet, le Scrum Master voit son rôle diminuer au fur et à mesure que l'équipe acquiert de l'expertise Scrum.

6. Le Product Backlog et les Stories

6.1. Les stories

Une story, c'est une fonctionnalité qui doit être fournie par le produit. Les stories sont définies par le Product Owner sur base de la vision du produit partagée par l'équipe. Si cette vision est amenée à évoluer au fur et à mesure de la réalisation du projet, il importe d'en avoir une bonne idée dès le début.

Construction des stories

Pour construire les stories, les étapes suivantes peuvent être suivies :

1. Définir précisément le problème à solutionner (énoncer le problème, les intervenants, l'impact), et expliciter ce qu'apporterait une solution
2. Positionner le produit (public-cible, définition du produit, différences par rapport à la concurrence ou à la pratique actuelle,…)
3. Définir les fonctionnalités générales (entre la vision et les stories) et les prioritiser
4. Décomposer les fonctionnalités en stories à ajouter au Product Backlog

Il est également important de réfléchir aux rôles des utilisateurs : Qui utilisera le produit ? Dans quel cadre ? Y a-t-il différents types d'utilisateurs ? Cette réflexion peut permettre d'identifier des stories supplémentaires, ou bien de raffiner des fonctionnalités en stories différentes en fonction du type d'utilisateur. Il faut également prévoir les rôles nécessaires pour assurer la maintenance et le support du système.

Il peut aussi être nécessaire d'identifier d'autres logiciels avec lesquels le produit sera amené à interagir.

Pour la définition des stories, on peut se baser sur la décomposition suivante :

- Le rôle
- Le but fonctionnel
- La justification (optionnelle si évidente)

Ces trois éléments peuvent être identifiés en utilisant la formulation suivante :

« En tant que <rôle>, je veux <but> afin de <justification> »

Par exemple :

- « En tant qu'étudiant, je veux m'inscrire à une formation afin d'obtenir un diplôme »
- « En tant que voyageur, je veux réserver un billet de train »
- « En tant qu'organisateur, je veux connaître le nombre de personnes inscrites à la conférence afin de connaître la salle adéquate »

En plus de ces trois attributs, il est également utile d'identifier les éléments suivants :

- La fréquence d'utilisation de la story par le rôle (une story utilisée fréquemment devra être testée de manière plus approfondie)
- Des informations complémentaires (texte, schéma, diagramme UML, …)
- Les tests associés à la story

Une story prête à être développée dans un sprint doit être de taille restreinte, et doit en moyenne pouvoir être développée en trois jours, tout compris (définition de fini). Si une story est de trop grande taille, on peut la décomposer en sous-stories, par exemple en distinguant le type d'utilisateur, ou éventuellement différents types de données (ex : compte espèce ou compte titre pour une application bancaire).

Les stories identifiées sur base des fonctionnalités utilisateurs sont un type de story (on peut parler de « **user story** »), mais il faudra également identifier les stories non liées aux fonctionnalités utilisateurs mais liées à des éléments techniques invisibles aux yeux des utilisateurs (« **technical story** ») . Par exemple, une telle story pourrait consister en la mise en place de l'environnement de développement, ou à travailler à l'architecture du projet.

Si vous désirez approfondir le sujet des User Stories, la lecture des articles suivant peut constituer un bon point de départ :

http://blog.xebia.fr/2012/06/20/scrum-master-academy-parlons-des-user-stories/
http://www.aubryconseil.com/post/Les-types-de-story-dans-un-backlog

Tests d'acceptation pour les stories

Pour chaque story, il est donc nécessaire de définir les conditions selon lesquelles elle sera considérée comme finie. La vérification de cette condition se fera sur base de **tests d'acceptation**. Ainsi, le produit sera testé au fur et à mesure de sa réalisation, puisque le premier test intervient dès la fin de la réalisation de la première story.

A la différence des tests unitaires qui permettent de valider des morceaux de code et qui sont effectués par les développeurs, les tests d'acceptations permettent de valider des fonctionnalités et sont orientés client. Les tests d'acceptation sont idéalement écrits avant la réalisation de la stories, et servent à détailler sa spécification. Pour cela, on identifie les comportements possibles pour la stories, et les conditions de satisfaction pour chacune.

Dans l'idéal, les stories devraient être validées à chaque sprint (tests de régression). Il apparaît alors nécessaire de les automatiser.

6.2. Le Product Backlog

Le Product Backlog a déjà été décrit plus haut. Il s'agit d'une liste prioritisée de stories, maintenue par le Product Owner. Les stories ont été identifiées initialement par l'équipe, mais c'est au Product Owner de les décomposer en éléments de taille suffisamment petites, en fonction de leurs priorités et au fur et à mesure de l'avancement du projet, pour que l'équipe puisse facilement les réaliser au cours du sprint pour lequel elles sont prévues.

Le backlog n'est donc pas un document figé, des éléments y sont ajoutés, supprimés, décomposés, ou leur priorité est modifiée, par le Product Owner.

Il importe cependant qu'au cours d'un sprint, les stories qui ont été sélectionnées pour développement soient gelées.

Le Backlog doit fournir des informations suffisantes sur les stories, en plus de leurs priorités. Des attributs possibles sont :

- Le nom
- Un identifiant
- Une description
- Un type (user, technique ou défaut/bug)
- Son état (créé, accepté, estimé, planifié, en cours, fini)
- Sa taille (en « points »)
- Plus éventuellement : créateur/date de création, estimation de la fréquence d'exécution, rôle associé)

Afin de garder le Product Backlog utilisable, il est recommandé au Product Owner de garder la partie active de ce dernier (les stories qui ne sont ni finies, ni en cours) de taille raisonnable (max. une cinquantaine d'éléments). Il réalise cela en prenant soin de décomposer les user stories au fur et à mesure du développement, ou encore en faisant des nettoyages occasionnels pour supprimer par exemple les stories qui restent au fond du backlog (très basse priorité), qui ne seront finalement pas utiles.

7. Synthèse

Le cycle de développement d'un produit est résumé par la figure suivante :

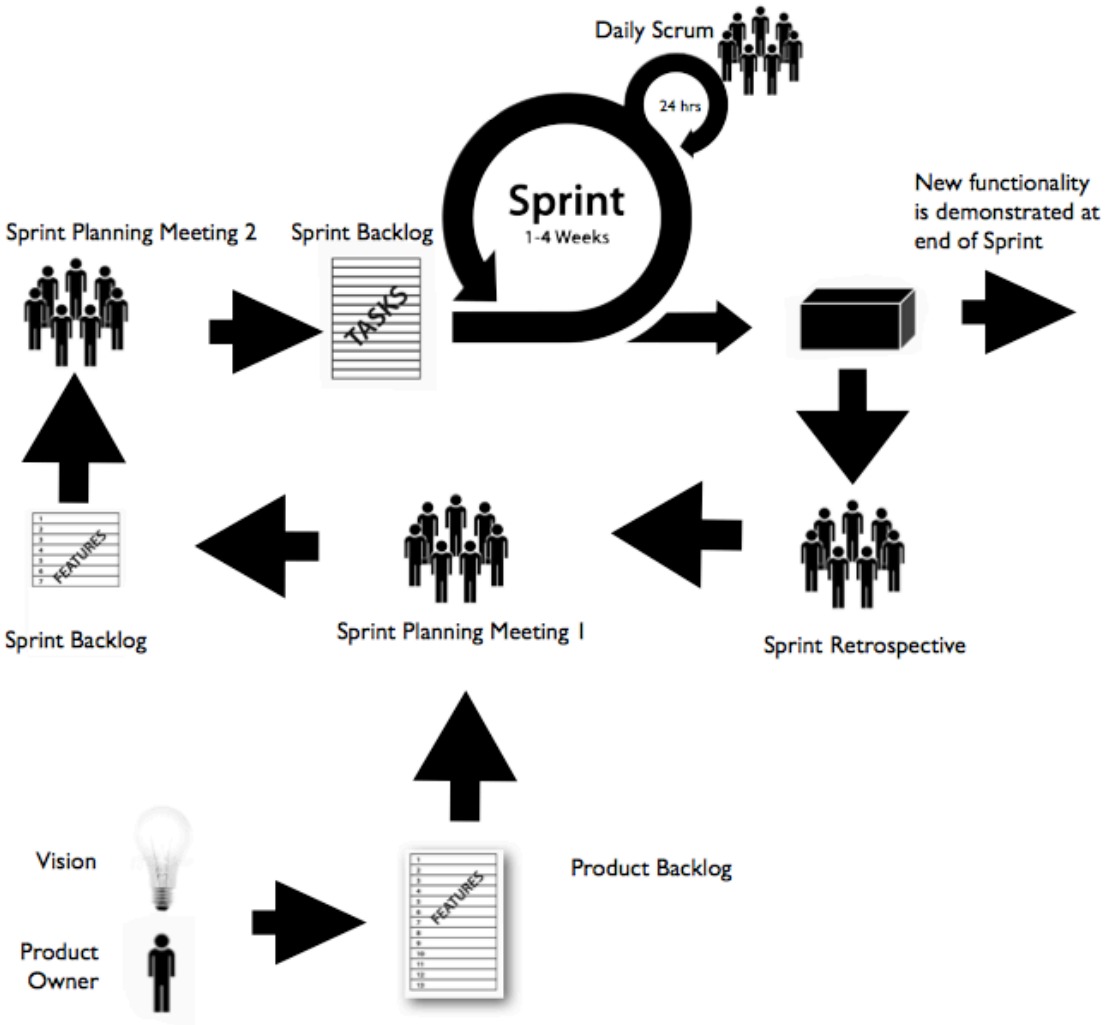

Figure 3 : Synthèse du découlement d'un projet scrum (une release), tiré de [3].