**Mise en pratique du Framework SCRUM**

Pour la mise en place du Framework SCRUM au sein de l’équipe développement application logiciel, nous allons établir un plan d’action en plusieurs étapes :

Étape 1 :

Attribution des roles

Étape 3 :

Sprint rétrospective

Étape 4 :

Version livrable

Étape 2 :

Conception BackLog Produit

X Sprint

**Étape 1 / Définition des rôles au sein de chaque équipe opérationnelle :**

* Le Scrum Master :

Responsable de l’animation et du suivi de l’Agilité au sein de l’équipe, il est garant de la méthodologie, travail en support avec le PO.

* Le Product Owner :

Responsable des spécifications sous forme d’user Stories et de leur priorisation au sein d’un backlog.

* L’équipe de développeurs et testeurs :

Responsables de l’implémentation et de la qualité de la solution applicative

**Étape 2 / Construction du backlog produit :**

Pour comprendre la construction du backlog produit par le Product Owner il est fondamental d’établir les notions qui le constitue et sa nature propre.

*2.a) Le but du backlog produit ?*

Un backlog Produit est un ensemble ordonné de tâches à réaliser pour délivrer le produit. Il permet non seulement d’avoir une vision globale sur ce qui constituera le produit mais également de préparer le développement de manière incrémentale et itérative.

Souvent présenté sous forme de liste, on retrouve en tête du backlog les éléments apportant une forte valeur ajoutée. Ces éléments sont petits et détaillés pour pouvoir livrer le plus rapidement possible des incréments de valeur. En redescendant dans la liste, la priorité sera moindre et les éléments plus gros, moins détaillés.

Si l’équipe est à l’aise et sénior avec la méthode SCRUM il est possible d’avoir d’autres types de backlog comme :

* un **backlog d’opportunités**. Il contiendra des problèmes à résoudre.
* un **backlog d’expérimentations** vous permettra de gérer les tests à mener. Votre backlog sera alors constitué d’informations bien définies sur les hypothèses retenues, les tests à réaliser pour valider ou non ces hypothèses et les résultats attendus.

Dans le cadre de notre transformation nous allons établir un seul type de backlog, le « produit ».

Évoqué en amont, il est constitué généralement d’incréments :

* User story « US » découpé et détaillées, prêtes pour le développement.
* Epic
* nitiative
* HOTFIX
* Sujet technique
* POC

L’importance sont que les éléments sont transparents pour l’ensemble de l’équipe.

Exemple d’un backlog produit synthétique :



*2.b) L’user story*

Nous allons détailler l’élément principal de notre backlog produit l’US.

Pour cela nous définissons **sa nature,** comment la **construire** et son **objectif**.

2.b.1) Sa nature

Les "User Stories" jouent un rôle fondamental dans les méthodologies agiles en tant que moyens de décrire les fonctionnalités à développer. ***L'objectif premier de ces récits est de préciser les exigences d'une fonctionnalité donnée en indiquant clairement à qui elle s'adresse et en quoi elle apporte une valeur ajoutée.***

La "User Story" se caractérise par une structure qui permet une expression **atomique**, **systématique** et **transparente des besoins de l'utilisateur**, garantissant ainsi une compréhension approfondie de la fonctionnalité à concevoir.

2.b.2) Comment la construire

Typiquement, une "User Story" se compose des éléments suivants :

* Un titre explicite qui résume la finalité de la fonctionnalité (par exemple, "Supprimer un article du panier").
* Une phrase narrative centrée sur l'utilisateur, décrivant le bénéfice souhaité.

Cette phrase suit généralement l'une des deux structures suivantes :

"En tant que... je veux... afin de..." ou "Lorsque... l'utilisateur peut... afin de..."

(par exemple, "En tant qu'utilisateur, je souhaite supprimer un article de mon panier, afin de commander uniquement les produits dont j'ai besoin").

* Un ensemble détaillé d'exigences et de critères d'acceptation « Langage GHERKIN » qui spécifient les tests et les conditions nécessaires à la validation de la fonctionnalité.
* D’un niveau de priorité – Méthode **M O S C O W** :

**M(o**) **— Must have :**  il faut mettre dans cette catégorie uniquement les pré-requis ou les fonctionnalités indispensables au projet, c’est-à-dire les éléments sans lesquels le produit ne peut pas être déployé en production. Ces éléments doivent absolument être développés, sans eux, le produit est un échec.

**S — Should have** :  les éléments importants et qui apportent énormément de valeur au produit final mais sans lesquels le produit peut être déployé en production. Ces fonctionnalités seront développées une fois que celles de la catégorie “Must” sont déployées et si les délais et le budget le permettent.

**C(o) — Could** **have** :  il s’agit ici de user stories dites "**de confort",** à réaliser uniquement si cela n’a pas d’impact sur d’autres tâches et que le temps le permet. Il s’agit de “petits plus” qui contribuent à augmenter la satisfaction des utilisateurs.

**W — Won’t have :** this time but would like in the future :  ces tâches ne seront pas réalisées immédiatement mais restent souhaitables afin d’améliorer le produit dans le futur.

L’écriture d’une US doit respecter un pattern **I N V E S T** pour être optimum :

**I** – Indépendante : elle doit se suffire à elle-même, car les dépendances avec d’autres User Stories induisent des problématiques de testabilité et de planification.

Bon exemple : En tant que client, je veux consulter la liste des factures émises.

Mauvais exemple : En tant que client, je veux créer la liste des factures...

**N** – Négociable : elle doit amener à la discussion et peut-être modifiée jusqu’à son inclusion dans une itération.

Bon exemple : En tant que client, je peux connaître le montant total des factures impayées.

Mauvais exemple En tant que client, je veux, lorsque je clique sur le bouton « calculer » une ligne s’ajoute et on affiche sur cette ligne le montant total des factures impayées.

**V** – Valeur : elle doit apporter de la valeur à l’utilisateur final. La notion de valeur étant toujours difficile à évaluer, la User Story doit être exprimée avec une vision de l’objectif recherché pour l’utilisateur.

Bon exemple : Un bon découpage qui fait apparaitre les scénarios distincts rendus par le service de facturation :

• Consulter la liste des factures émises

• Afficher la liste des factures ordonnées par date

• Afficher la liste des factures par adresse de livraison

• Consulter une facture client

Mauvais exemple : Créer la base de données pour le module de facturation. Créer l’interface graphique pour la facturation

**E** – Estimable : elle doit être suffisamment claire et comprise par l’équipe pour que celle-ci soit en capacité de l’estimer.

Bon exemple : Implémenter les règles métier R1, et R2

R1 : si le nom de l’utilisateur existe déjà, lorsque l’acheteur en ligne essaye de créer son compte, alors le message «ce compte existe déjà doit »être affiché.

R2, un rabais de 5% est octroyé pour tous les clients dont le montant total de la facture dépasse les 1000$. Préparé par Saliha Yacoub Automne 2013 Page 3

Mauvais exemple : Garantir le respect des règles de gestion en vigueur.

**S** – Small : elle doit être de taille assez petite pour prioriser de façon sûre et éviter les effets tunnels. Essayez donc au maximum de découper finement vos User Stories.

Mauvais exemple : réaliser le module de facturation – il faudra découper –

**T** – Testable : la User Story doit raconter une histoire, dont les tests découlent de façon évidente.

Exemple de test d’acceptation : En tant que client, je veux connaître la liste des factures par date. Test d’acceptation : afficher la liste des factures ordonnées par la date de facturation

2.b.3) son objectif

Une image vaut mille mots :



**Étape 3 / Sprint planning :**

3.a) le backLog produit

Le Product-Owner a établi le backlog Produit avec 47 US à faire :

**La colonne Effort ou charge désigne le nombre de jours/homme.**

5jours/homme : a une durée de 5 jours si le nombre de ressources qui travaillent en parallèle et à temps plein est égal à 1. A une durée de 10 jours si le nombre de ressources qui travaillent en parallèle et à mi-temps est égal à 1. A une durée de 1 jours si le nombre de ressources qui travaillent en parallèle et à temps plein est égal à 5

**La colonne Point : indique le niveau de complexité technique (suite finobaci)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Titre | User story | Priorité | Effort | Nombre de point |
| La suppression d'un utilisateur par l'administrateur | En tant qu’administrateur, je dois pouvoir supprimer un usager afin de gérer le nombre d’utilisateurs | M | 1 | 3 |
| Trier les pièces cartographiées par le robot | En tant qu’utilisateur, je veux que mon robot puisse trier les pièces selon leur forme. | M | 2 | 3 |
| Déplacement le plus court du robot | En tant qu’utilisateur, je veux que mon robot se déplace entre le point A et le point B selon le chemin le plus court afin d’avoir le temps le plus court | M | 10 | 13 |
| Afficher les cartographies des différentes pièces | En tant qu’utilisateur je veux avoir une carte de chaque pièce afin de les afficher dans mon application | M | 3 | 1 |
| … | … | … | … | … |

3.b) le backlog de sprint

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| User story | Priorité | | Niveau détail | Maquette | Effort | nbr de point | Critères d’acceptations Gherkin |
| En tant qu’administrateur, je veux voir l’ensemble des utilisateurs sur mon application. Je souhaite afficher les utilisateurs pour les managers | M | | Construire le formulaire avec tous les champs | Lien figma :  Ici | 1 | 3 | **Étant donné que** je suis la page administration  **quand** j’affiche la liste des utilisateurs **Alors** je peux les trier par catégories |
| M | | Indiquer les champs obligatoires | Lien figma :  Ici | 1 |
| M | | Enregistrer correctement le formulaire | Lien figma :  Ici | 2 |
|  |
| S | | Afficher les profils par categories | Lien figma :  Ici | 1 |  | **Étant donné que** je suis la page administration  **Quand** j’affiche la liste des utilisateurs **Alors** je peux les trier par date |
| W | | Trier les profils par date | Lien figma :  Ici | 1 |
| En tant qu’utilisateur, je veux que mon robot se déplace entre le point A et le point B selon le chemin le plus court afin d’avoir le temps le plus court | M | | Afficher les maps enregistrées | Lien figma :  Ici | 1 | 13 | **Étant donné que** je suis la map  **Quand** je clic sur un endroit **Alors** le robot se déplace vers le point spécificié |
| M | | Selectionner deux points et enregistrer les coordonnées | Lien figma :  Ici | 3 |
| M | | Traitement par le back des coordonnées | Lien figma :  Ici | 3 |
| M | | Animation du robot sur la carte | Lien figma :  Ici | 3 |
| En tant qu’utilisateur je veux avoir une carte de chaque pièce afin de les afficher dans mon application |  | | … | … | … | … |  |
| … |  | | … | … | … | … |  |

3.c) L’incrément

Pour construire nos incréments nous allons définir notre Définition of done.

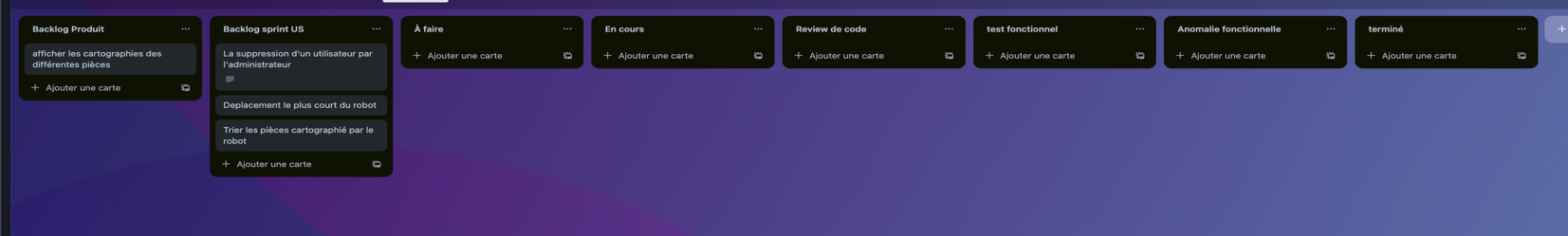
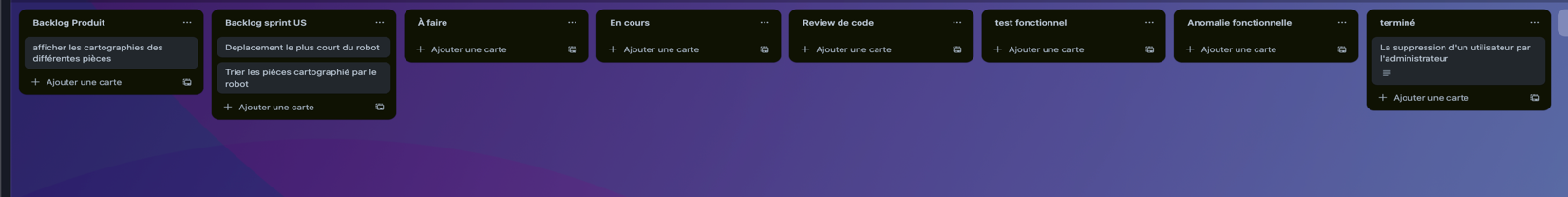
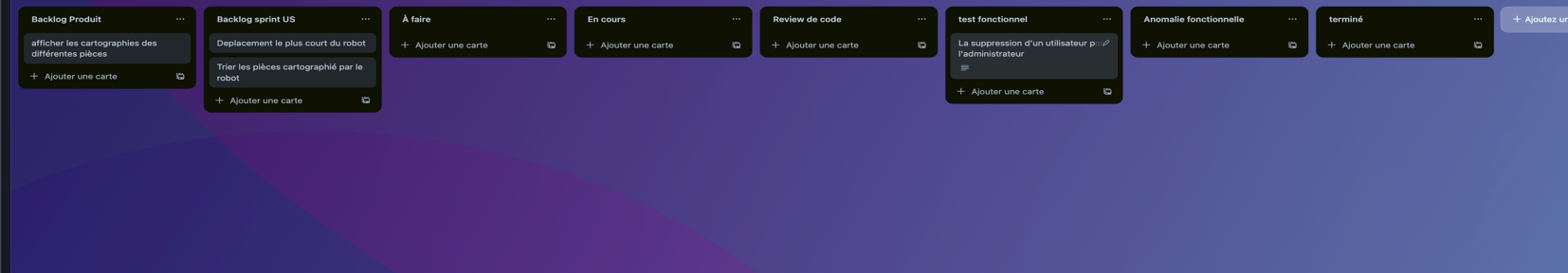
La DOD est composée d’une check-list qui doit être obligatoirement réalisée pour que l’US soit « terminé », par conséquence l’us devient un incrément.

DOD pour notre équipe :

* Test unitaire sur chaque US
* Review de code / refactoring ok
* Test d’acceptation ok

**L’ensemble des incréments US forment l’incrément de sprint en fin de sprint.**

**La DOD peut être représenter dans une organisation KANBAN :**



US en « terminé » DOD OK

3.d) Dashboard Sprint :

Pour calculer le nombre de sprint nécessaire pour terminer le backlog il est important d’avoir trois sprints terminés pour établir la vélocité moyenne.

Vélocité = nbr de points d’efforts effectués à la fin du sprint

Vélocité moyenne = Somme des vélocités / nbr de sprint

Représentation de notre premier sprint :

Sprint retrospective :

Feed back d’amélioration continue de l’équipe

Sprint review :

Présenter l’incrément de sprint et avoir du feedback

Sprint exécution :

**Phase de 2 semaines** de sprint avec daily

Backlog sprint :

12 US

Estimation point d’efforts : 60

US réalisées :

10 US

Points d’efforts : 45

Velocité = 45

Backlog produit :

47 US

Estimation points d’efforts : 376

Réalisé pendant le grooming

Sprint planning - Grooming

Prioriser sur 4h les éléments qui peuvent devenir des incréments

Sprint planning – Poker

Établir les points efforts

Nos trois sprints représentent une vélocité respective :

Sprint 1 : 45

Sprint 2 : 38

Sprint 3 : 41

la vélocité moyenne est donc de : ( 45 + 38 + 41 ) / 3 = 41 points d’efforts

le nombre de sprint totaux est donc de :

376 « nbr points backlog Product » / 41 « vélocité moyenne » = **9 sprint**.