Les User Stories

Syntaxe générale d'une story:

- En tant que <*rôle d'utilisateur*>, je veux <*un but*> afin de <*une justification*>)

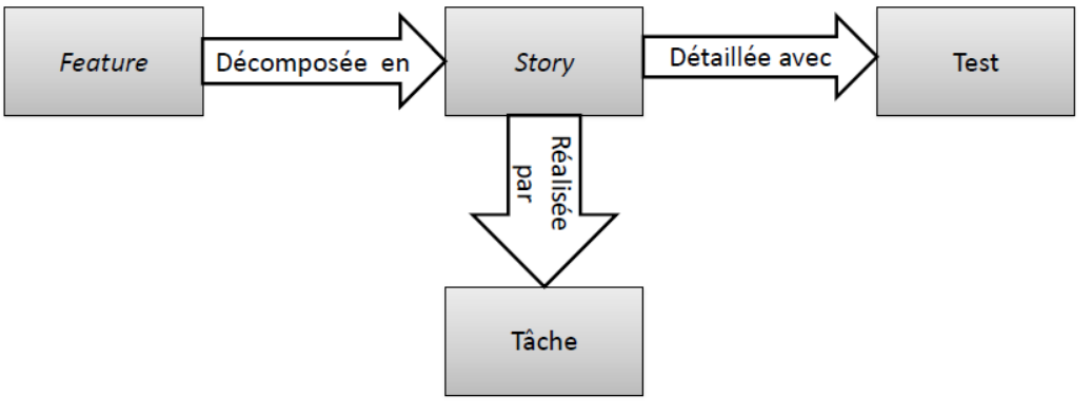
La partie justification est optionnelle, parce qu'elle est parfois évidente.

Exemples:

> En tant *qu'étudiant*, je veux *m'inscrire à une formation* afin *d'obtenir le diplôme*.
>
> En tant que *voyageur* je veux *réserver un billet de train*.
>
> En tant *qu'opérateur* je veux *créer un compte pour un client* afin *de recevoir son argent*.
>
> En tant *qu'organisateur*, je veux *connaître le nombre de personnes inscrites à la conférence* afin *de choisir la salle adéquate*.

Si vous regardez l'exemple de product backlog que je vous ai donné avec le travail vous verrez des "Features" et dans le backlog de sprint, des tâches.

La façon de résumer tout ceci:



Une "feature" se décompose en "stories" qui, elle-même se détaille en "test" dans le "backlog produit". Finalement, une "story" se divise en "tâches" dans le "backlog du sprint". (voir exemple à la page suivante:

Exemple de backlog produit:

ID	Features	Nom	Story	Priorité	Comment confirmer	Estimation
1	Administrer le site	Login	En tant qu'administrateur ou étudiant je veux me connecter au site afin d'y apporter des modifications	90	Vérifier que la connexion au site fonctionne bien	4
2	Gérer profils étudiants	Consulter profil étudiant	En tant qu'internaute nous voulons pouvoir consulter les profils des étudiants	90	Entrer des profils étudiants dans la base de données et vérifier que l'on peut les consulter	5
3	Gérer profils étudiants	Contacter étudiant	En tant qu'internautes nous voulons pouvoir contacter l'étudiant dont nous avons choisi le profil afin de lui transmettre un message	50	Entrer les coordonnées d'étudiants dans la base de données et vérifier que l'envoi de e-mail fonctionne bien	8
4	Afficher les témoignages	Entrer un témoignage	En tant qu'administrateur je veux pouvoir entrer les témoignages des différents intervenants de l'entreprise afin de pouvoir les faire afficher sur le site	30	Remplir le formulaire de témoignage et consulter le témoignage dans la Base de données	3
5	Afficher les témoignages	modifier un témoignage	En tant qu'administrateur je veux pouvoir modifier les témoignages des différents intervenants de l'entreprise afin de pouvoir maintenir l'information à jour	20	Modifier le formulaire de témoignage et consulter le témoignage dans la Base de données pour vérifier que les modifications ont bien été appliquées	3

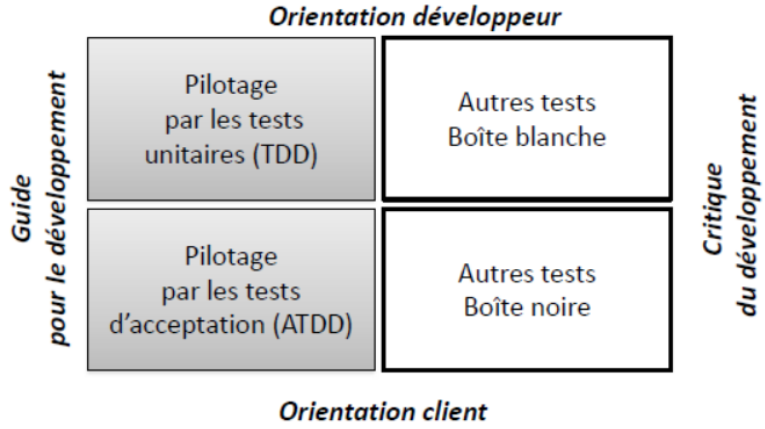
Exemple de backlog de sprint

Id Story	Stories	Tâches	Heures
1	En tant qu'administrateur ou étudiant je veux me connecter au site afin d'y apporter des modifications	1.1 Déterminer les informations nécessaires à conserver sur un utilisateur	3
		1.2 Créer la BD des utilisateurs	5
		1.3 Déterminer le format de l'écran de saisie des utilisateurs	3
		1.4 Créer le formulaire	2
		1.5 Créer le CSS	3
		1.6 Ajouter le lien au menu	1
2	En tant qu'internaute nous voulons pouvoir consulter les profils des étudiants	2.1 Entrer des profils étudiants dans la base de données	2
		2.2 Créer le formulaire de recherche d'étudiants	3
		2.3 Créer l'écran d'affichage du profil	1
		2.4 Créer le CSS	1
		2.5 Ajouter le lien à la BD	1

Les tests d'acceptation

Quelle que soit la méthode de développement utilisée, il existe des tests de nature différente. Les méthodes agiles apportent une nouveauté dans la façon de percevoir certains types de tests. En effet, la vue classique du test est la détection d'erreurs a posteriori, après le travail de développement : le testeur, en cherchant des erreurs, vise à critiquer. Or, le test agile a un objectif différent, celui de guider le développeur dans son travail.

Cette nouvelle vision des tests peut se résumer dans la matrice à quatre quadrants ci-dessous:

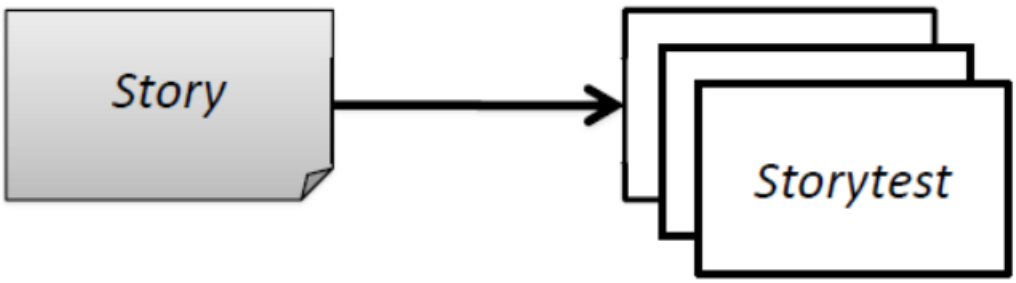


Cette idée de guide pour le développement transparaît dans le pilotage par les tests (TDD pour Test Driven Development), centré sur les tests unitaires. C'est aussi le but pour les tests d'acceptation qui sont des tests orientés client (ATDD, Acceptance Test Driven Development).

Le test d'acceptation est le processus qui permet d'accepter une story à la fin d'un sprint. Il consiste en plusieurs étapes, appliquées à une story:

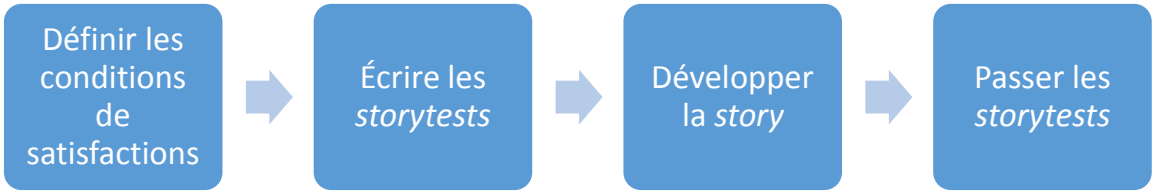
- Décrire le comportement attendu avec les conditions de satisfaction.
- Transformer ces conditions en cas de test, appelés storytests.
- Écrire le code applicatif qui répond au comportement attendu.
- Passer les storytests sur le code applicatif. En cas d'échec, corriger les tests ou le code.

Ce processus est appelé **pilotage par les tests d'acceptation**.



Une user story devrait posséder au moins deux storytests associées : un cas de succès et un cas d'échec. Il peut y avoir d'autres cas de tests pour une story, mais un nombre trop important (disons au-delà de huit) est le signe d'une trop grande complexité de la story, qu'il conviendrait de décomposer.

Étapes du processus:



Définir les conditions de satisfactions:

Le principe, pour toutes les méthodes agiles, est qu'une story soit réalisée en une itération. Mais comment savoir si elle est vraiment finie à la fin de l'itération ?

C'est la responsabilité du Product Owner d'accepter (ou non) une story. Pour cela, le moins qu'il puisse faire est de définir ses conditions de satisfaction. Si toutes les conditions sont satisfaites, la story est acceptée, sinon elle n'est pas considérée comme finie.

Exemple avec la story « Inscription à une conférence ».

> En tant *qu'organisateur*, je veux *connaître le nombre de personnes inscrites à la conférence* afin *de choisir la salle adéquate*.

On peut identifier trois comportements:

- **Inscription acceptée** – C'est le cas de succès, l'inscription d'une personne à une conférence est confirmée.
- **Inscription différée** – L'inscription n'est pas confirmée faute de place et placée en liste d'attente.
- **Inscription refusée** – L'inscription est refusée, la liste d'attente ayant atteint sa limite.

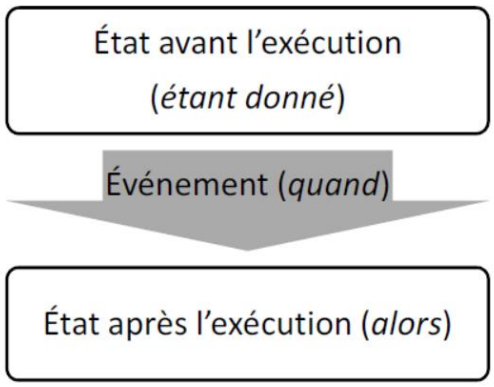
Pour une user story, une condition de satisfaction peut être formalisée par un test.

Écrire les storytests

Les différents tests associés à une *story* correspondent à des comportements différents du logiciel. Les comportements diffèrent parce que, en fonction de l'état du logiciel au moment où la *story* est exécutée, les résultats obtenus seront différents. La technique du BDD (*Behaviour Driven Development*) permet de décrire ces comportements.

Chaque test est formalisé avec trois rubriques:

- l'état du logiciel avant l'exécution du test (on parle aussi de précondition ou de contexte du test);
- l'événement qui déclenche l'exécution;
- l'état du logiciel après l'exécution (on parle aussi de postcondition ou de résultat attendu).



Plus formellement, on pourrait résumer l'écriture des tests comme ceci:

> **Étant donné** le contexte **et** la suite du contexte
> **Quand** l'événement
> **Alors** résultat **et** autre résultat

Cette façon de faire est particulièrement adaptée à des applications interactives. Elle pousse à avoir des tests courts, puisqu'on y décrit la réponse à un seul événement.

Exemple avec la story « Inscription à une conférence »

Chaque condition de satisfaction est transformée en test.

> ***Exemple pour inscription acceptée***
>
> Étant donné l'utilisateur JoeBloe connecté et la conférence AgileMontréal avec le nombre d'inscrits à 134 et la salle A4 d'une capacité de 200 associée à AgileMontréal.
>
> Quand JoeBloe s'inscrit à AgileMontréal
>
> Alors l'inscription de JoeBloe est acceptée et le message Vous êtes bien inscrit à AgileMontréal est envoyé à JoeBloe et le nombre des inscrits passe à 135.

> ***Exemple pour inscription différée***
>
> Étant donné l'utilisateur Toto connecté et la conférence AgileMontréal avec le nombre d'inscrits à 200 et la salle A4 d'une capacité de 200 associée à AgileMontréal et 3 personnes dans la liste d'attente.
>
> Quand Toto s'inscrit à AgileMontréal
>
> Alors l'inscription de Toto est refusée et le message Vous êtes en liste d'attente est envoyé à Toto et le nombre des inscrits reste à 200 et le nombre de personnes en liste d'attente passe à 4.

> ***Exemple pour inscription refusée***
>
> Étant donné l'utilisateur Badot connecté et la conférence AgileMontréal avec le nombre d'inscrits à 200 et la salle A4 d'une capacité de 200 associée à AgileMontréal et 20 personnes dans la liste d'attente.
>
> Quand Badot s'inscrit à AgileMontréal
>
> Alors l'inscription de Badot est refusée et le message Il n'y a plus de places disponibles est envoyé à Badot et le nombre des inscrits reste à 200 et le nombre de personnes en liste d'attente reste à 20.

Où stocker les tests?

Chaque test étant associé à une *story*, il est considéré comme un attribut de la *story* et placé avec elle dans le *backlog* de produit.

Quand écrire les tests ?

Si la *story* est réalisée dans l'itération *n*, cela implique que les étapes du processus de test d'acceptation s'y déroulent (pour quelques *stories*, il faut même faire l'étape d'écriture des tests dans l'itération *n* – 1).

> Une recommandation est, au moins pour une *story* du *sprint*, que ses tests soient prêts avant le début du *sprint* et que tous les tests soient prêts à la moitié du *sprint*.

Les étapes ne sont pas nécessairement séquentielles, l'ajout de nouveaux tests peut se faire en parallèle avec le développement de la *story*.

Qui écrit les tests?

Scrum met l'accent sur l'équipe sans spécialiser les rôles. Il n'y a pas de rôle de testeur, mais cela ne veut pas dire que l'équipe ne teste pas ! Il existe parfois l'idée que c'est le client qui teste, le client étant représenté par le Product Owner, ce qui peut conduire les développeurs à déléguer au Product Owner tout l'effort de test.

Ce n'est pas une bonne idée : pour des raisons de quantité de travail et de compétences, le Product Owner n'est généralement pas en situation pour s'occuper seul du test d'acceptation et surtout cela doit être un travail collectif.

En fait, peu importe qui rédige les tests, ce qui compte c'est que cela soit fait au bon moment.

Développer la story

Le développement de la *story* est mené rapidement pendant le *sprint* ; il dure, en moyenne, trois jours, à plusieurs personnes.

Le pilotage par les tests signifie que l'équipe part des tests d'acceptation pour concevoir et coder la story. Pendant le développement, il est fréquent que des *storytests* soient complétés, voire que de nouveaux soient ajoutés.

Passer les storytests

Pour vérifier que la *story* est bien finie, il faut exécuter ses tests sur la dernière version du logiciel, la version courante. Si des tests ne passent pas, la correction est faite aussitôt, l'objectif étant que tous passent avant la fin du *sprint*.

À chaque nouvelle version, pour éviter les régressions, il conviendrait de repasser tous les tests. C'est une raison pour laquelle il est vite nécessaire de s'intéresser à leur automatisation.