User Stories, ATDD et BDD

User Story:

Une user story est une façon de « spécifier » un besoin fonctionnel.

C'est au-delà de l'aspect écrit d'une spécification, surtout un moyen de communiquer entre le Product Owner (ou analyste métier) et l'équipe de développement. Je vous renvoie à l'article de Martin Fowler à ce sujet.

Une user story est exprimée en utilisant la phrase suivante : En tant que « rôle », je veux « faire une action », afin d' « atteindre un objectif »

Il est important de définir le rôle, l'action, mais aussi l'objectif à atteindre. Le fait de réfléchir à ces trois éléments permet de préciser son besoin mais aussi son pourquoi : Un point important pour l'estimation de la valeur ajoutée de la user story.

Granularité.
Plus la granularité d'une user story est fine, plus l'implémentation en sera simplifiée. Voir même, influencera l'architecture de la solution et améliorera l'aspect unitaire et autonome du code produit.

Une user story a une granularité en fonction de sa maturité.

Je prends pour exemple la user story suivante :

« En tant qu'utilisateur, je veux me connecter à google afin d'accéder à tous mes services en lignes »

A elle seule, cette user story représente (presque) tout ce que google m'apporte.

Bill Wake propose une méthode d'estimation du bien fondé d'une user story : la méthode INVEST

ATDD:

Mais comment passer d'un besoin exprimé en langage métier en un produit exprimé en langage de programmation. (Il y a eu UML, un temps…)

L'approche **ATDD** ou Acceptance Test Driven Development

C'est d'une certaine manière la faculté de renvoyer l'envoyeur à sa responsabilité de clairement préciser son besoin par la définition des critères de contrôles qui lui serviront à vérifier l'adéquation du produit au besoin.

Il s'agit de compléter la user story par des critères d'acceptance.

Reprenons l'exemple :

« En tant qu'utilisateur, je veux me connecter à google afin d'accéder à tous mes services en lignes »

Imaginons les critères suivants :

- L'utilisateur peut se connecter
- La barre de menu google présente les services disponibles
- L'utilisateur peut accéder à tous ces services
- Google trace la connexion de l'utilisateur
- Google présente la liste des nouvelles fonctionnalités disponibles

Voilà un premier « jet » de critères, mais comment évaluer leur précision, leur utilité.

Premier élément de réflexion :

Un critère d'acceptance doit être :

- Une vision utilisateur
- Ne pas proposer de solution
- Ne pas être interne à la fonction

Dans ce cas, le critère suivant doit être supprimé :

- Google trace la connexion de l'utilisateur. C'est un traitement interne à la fonction, l'utilisateur n'est pas informé. Cela pourrait faire l'objet d'une autre user story du type En tant qu'administrateur google, je veux être informé de toute connexion, afin d'analyser le trafic sur le site

Seconde analyse: Un critère d'acceptance doit être SMART

J'utilise cette méthode d'habitude allouée aux tâches pour les critères d'acceptance.
Un critère d'acceptance doit avoir les caractéristiques suivantes :

- Specific = défini et explicite
- Measurable = quantifiable, observable
- Achievable = peut exister, peut être fait
- Relevant = approprié à cette user story
- Time bound = quand cela doit-il intervenir

Criteria	Specific	Measurable	Achievable	Relevant	Time Bound
L'utilisateur peut se connecter	Acceder à la page de connexion Saisir son identifiant et son mot de passe et demander la connexion	L'utilisateur voit son identifiant à droite dans le bandeau google	OUI	OUI	<30s
La barre de menu google présente les services disponibles	On a la barre de manu même non connecté			N	
L'utilisateur peut accéder à tous ces services	C'est spécifique par service			N	
Google présente la liste des nouvelles fonctionnalités disponibles	Message « Découvrez nos nouvelles fonctionnalités »	Message affiché en haut à droite	OUI	N	ce n'est pas le fait de se connecter qui déclenche ce message

Voilà, j'ai fait le tri.

C'est un exemple bien sur, et je n'ai pas le moyen de communiquer avec les équipes de développement Google pour qu'ils me fassent part de leurs remarques afin d'améliorer l'exhaustivité et raffiner cette liste de critères.

BDD :

J’ai une user story et des critères d’accetance. Maintenant, comment spécifier le comportement de l’application attendu pour répondre au besoin.

A ce stade, j’utilise le langage Gherkin défini pour le **BDD** ou **Behavior Driven Developement** : Given / When / Then (And)

Ce langage permet de spécifier les scénarios utilisateurs, le comportement de l’application.

La user story s'exprime ainsi au final : « En tant qu’utilisateur, je veux me connecter à google afin d’accéder à tous mes services en lignes »

Scénario 1 : Accéder à la page de connexion
Given L’utilisateur n’est pas connecté
And L’utilisateur est sur la page d’accueil google
When L’utilisateur demande à se connecter
Then La page de connexion est affichée

Scénario 2 : Se connecter
Given L’utilisateur n’est pas connecté
And L’utilisateur est sur la page de connexion
When L’utilisateur saisi son identifiant
And L’utilisateur saisi son mot de passe
And L’utilisateur clique sur « Connexion »
Then La page d’accueil google s’affiche en <30s
And L’identifiant de l’utilisateur est affiché à droite dans le bandeau

Je n'ai volontairement pas traduit Given / When / Then (And).
Le langage gherkin est interprété par de plus en plus d’outils de test (robotframework, cucumber, fitnesse, …).
Il suffit, dans ce cas, de simplement copier/coller les scénarios dans votre outil préféré de test. Ne reste plus qu’à implémenter les keywords ou autres fixtures. Robotframework pour les tests IHM, cucumber pour les tests unitaires par exemple.

N.B.: Cet article m’a été fortement inspiré par les travaux de Raj Mudhar au sujet de l’ATDD et aussi du tutoriel BDD sur robotframework proposé par Yannick Ameur