DANS LA SÉRIE : B.D. AGILE!

IDÉE ET DESSINS PAR
ANIS BEREJEB : WWW.BEREJEB.COM

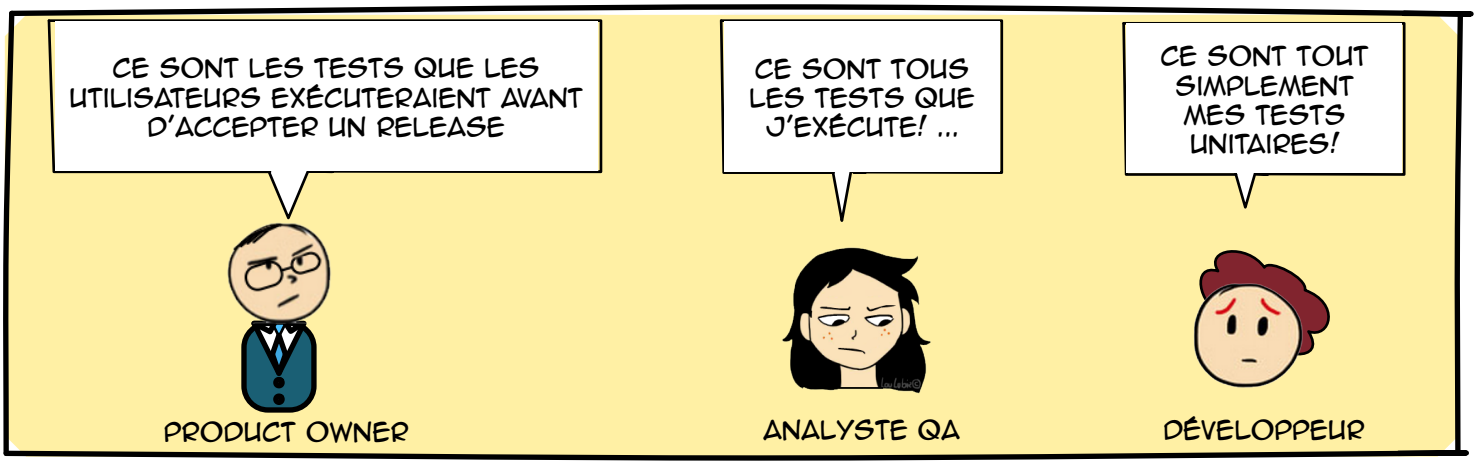
LES TESTS D'ACCEPTATION

REFLEXIONS, EXPERIMENTATIONS ...
RÉUSSITES ET ÉCHECS ...
APPRENTISSAGE ET AMELIORATION.

À PARTAGER AVEC VOS ÉQUIPES SANS PROBLÈME SI VOUS AIMEZ!
À JETER DANS LA CORBEILLE SINON!

LES TESTS D'ACCEPTATION
1. LA DÉFINITION DU DONE

Si je peux donner un nom à l'année 2014 professionnellement parlant, c'est l'année des "Tests d'acceptation"!
Au cours de cette année nous avons beaucoup de discussions, expérimentations, de réussites et d'échecs, et je pense que cela vaut la peine d'en partager quelques unes!
Le terme "Tests d'acceptation" est un terme ambigu et souvent mal utilisé...



La définition à Laquelle j'aborde personnellement est plutôt celle qui dit que :

LES TESTS D'ACCEPTATION DÉFINISSENT QUAND EST CE LE REQUIS EST FAIT (DONE)!

Ah la D.O.D! La Definition de done! ... Heu.. quand un développeur dit que sa story est done, que veut-t-il dire au juste? ...

Combien de fois avez vous entendu l'expression : c'est pas encore "done-done" ? Serieusement! :o) il y'avait la blague, mais derrière la blague il y'avait toujours une situation réelle.

À mon avis, toute définition de done acceptable doit absolument contenir:

TOUT LE CODE EST ÉCRIT + TOUS LES TESTS PASSENT + LE CLIENT (P.O.) A ACCEPTÉ LA FONCTIONNALITÉ

= DONE. FINI. DONE.

Mais comment faire pour arriver a avoir ce niveau tout en faisant du progrès de sprint en sprint?

Ceci veut dire un truc :

VOUS CONSIDÉREZ QUE CES TESTS SONT LES SPÉCIFICATIONS DU REQUIS

CE N'EST PAS PLUS DE TRAVAIL!

... Ce qui implique un niveau de précision assez grand. On le sait pertinemment, nous programmeurs, que nous avons besoin de ce niveau de détails!

La spécificication au niveau de precision que celui des tests est le seul moyen pour nous, les programmeurs, de savoir ce que voudrait dire le mot "Done".

La specification jusqu'a ce niveau de precision est le seul moyen pour le client de savoir que le système qu'il vont payer pour fait ce qu'ils ont besoin.

Specifier jusqu'a ce niveau de detail est le seul moyen pour écrire des tests automatisées avec succès.

Au lieu de voir ces tests comme de l'extra travail, voyez les donc plutôt comme moyen pour sauver du temps et de l'argent.

Ces tests vont vous protéger afin de ne pas écrire le mauvais système et vont vous permettre de savoir quand est ce que votre travail est fini.

LES TESTS D'ACCEPTATION
2. LA COMMUNICATION

L'un des sujets les plus difficiles en termes de communication entre les développeurs et la business sont les requis. Généralement, Les gens d'affaire disent ce qu'ils ont besoin, et par la suite les développeurs codent ce qu'ils pensent que la business a décrit. La réalité est que la communication des requis est extrêmement difficile et que le processus est bordé d'erreurs.

Le rôle de développeur est aussi bien un rôle de communication qu'un rôle de développement. De la meme facon qu'il s'assure que son code est lisible et comprehensible par ses collègues développeurs, quand il construit le logiciel, il s'assure aussi de bien communiquer ce que fait ce dernier à la business et aux autres membres de l'équipe tel que les analystes d'affaires et les analystes QA.

L'objectif des tests d'acceptation est la communication, la clarté et la précision. En s'accordant la dessus, les developpeurs, le client et les testeurs comprennent tous le plan pour le comportement du systeme. Il est de la responsabilité de toutes les parties d'atteindre ce degré de clarté. Chaque developpeur doit considerer que ce c'est de sa responsabilité de travailler avec le client et les QA pour s'assurer que touts les parties sachent ce qui va etre developpé.

LES TESTS D'ACCEPTATION NOUS PERMETTENT À NOUS LES DÉVELOPPEURS, AVEC LES EXPERTS DU DOMAINE, DE COMPRENDRE ET S'ENTENDRE SUR QUOI NOUS ALLONS BÂTIR PAR LA SUITE. NOUS LES UTILISONS AUSSI POUR NOUS ASSURER QUE NOUS N'AVONS PAS CASSÉ D'AUTRES FONCTIONNALITÉS EXISTANTES EN COURS DE DÉVELOPPEMENT.

LES TESTS D'ACCEPTATION

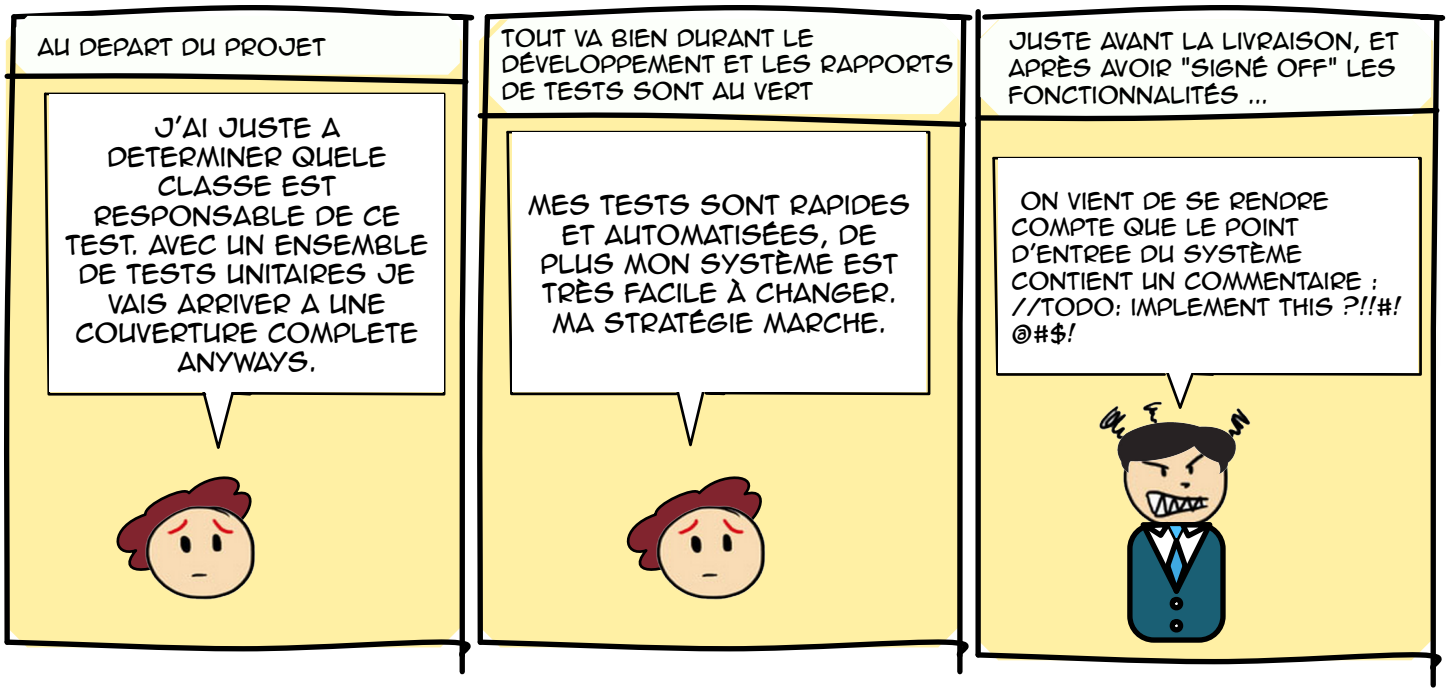
3. LES TESTS D'ACCEPTATION ET LES TESTS UNITAIRES

Les auteurs de l'excellent livre "Growing Oriented Object Software Guided By Tests" (Une référence que tout développeur devrait lire, à mon avis) décrivent les tests d'acceptation comme des tests qui exercent le système de bout-en-bout sans appeler son code interne.

Un test de bout-en-bout interagit avec le système uniquement de l'externe, via l'interface utilisateur, en lui envoyant des messages comme une tierce partie, en appelant un webservice, etc.

Le comportement Global du système inclut son interaction avec l'environnement externe. Ceci est souvent l'aspect le plus risqué, le plus difficile, mais aussi le plus souvent ignoré.

Les auteurs expliquent qu'il faut essayer d'éviter les tests d'acceptation qui exercent uniquement des objets internes du système.



LES NIVEAUX DE TESTS

La question des niveaux de tests est un sujet en soi, et plusieurs auteurs ont plusieurs definitions. En voici une version que je trouve assez simple et claire :

TESTS D'ACCEPTATION : *EST CE QUE LE SYSTÈME MARCHE, DANS SA TOTALITÉ?*
TESTS D'INTEGRATION : *EST CE QUE NOTRE CODE MARCHE AVEC UN CODE QUE NOUS NE POUVONS PAS CHANGER?*
TESTS UNITAIRES : *EST CE QUE NOS OBJETS FONT LA BONNE AFFAIRE? EST CE QUE C'EST FACILE DE TRAVAILLER AVEC?*

QUALITÉ EXTERNE ET QUALITÉ INTERNE

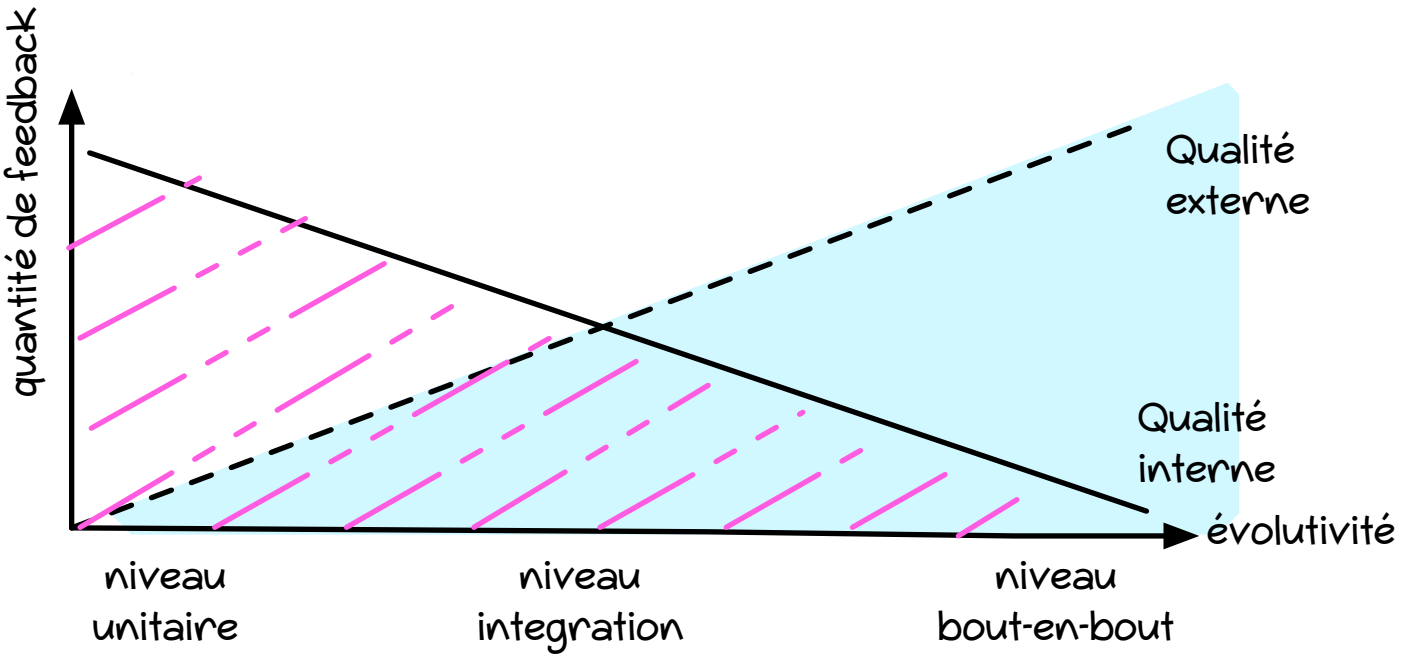
Nous pouvons faire la distinction entre deux types de qualité dans un système

La qualité externe : qui décrit comment le système répond au besoins des clients et utilisateurs. Il fonctionne, il est fiable, disponible etc.
Le cas de la qualité externe est simple à comprendre, ca fait partie du contrat a livrer.

La qualité interne : qui décrit comment le système répond aux besoin de ses développeurs et administrateurs. il est facile à comprendre, facile à changer etc.
Le cas de la qualité interne est plus subtile, il a plutôt rapport avec l'anticipation du continuel changement.

Maintenir la qualité interne nous permet de modifier le comportement du système en toute sécurité. De plus, elle nous permet une sorte de "prédictabilité" puisqu'elle minimise le risque qu'un changement nous oblige à retravailler une majeure partie du système

Le graphique ci-dessous expose l'apport de chaque niveau de tests sur deux axes : la quantité de feedback et l'évolutivité



Les tests de bout-en-bout nous exposent la qualité externe du système. Les écrire nous permet de determiner a quel point nous, en tant qu'équipe, comprenons le domaine. D'autre part, les tests de bout-en-bout ne nous disent rien quand à la qualité de notre code.
Ecrire des tests unitaires nous donne beaucoup de feedback sur la qualité du code, et les rouler nous permet de savoir si nous avons cassé des classes. Ceci ne nous donne pas une confidence complete que le système marche en totalité.
Les tests unitaires se trouvent au milieu.

Ne me comprenez pas mal, je ne veux pas diminuer l'apport des tests unitaires ici. Je veux toutefois porter le point que les tests d'acceptation sont généralement ceux qui nous informent plus sur la qualité externe du système.

C'EST QUOI LE POINT?

Le point est que les tests d'acceptation ne sont pas des tests unitaires.

Les tests unitaires sont écrits par les programmeurs pour les programmeurs.
Ce sont des documents de design qui décrivent la structure de bas niveau du logiciel, et le comportement du code.

Je répète, l'audience des tests unitaires sont le programmeurs et non la business.

Les tests d'acceptation sont écrits par la business (ce qui pourrait vous inclure des fois, cher programmeur). Ce sont des documents formels de requis qui spécifient le comportement du système d'un point de vue business. L'audience de ces tests est la business et les programmeurs.

C'EST LA MÊME CHOSE.. ÉCRITE DEUX FOIS

Il est souvent temptant d'essayer d'éliminer l'extra-travail en assumant que ces deux types de tests sont redondants. Malgré que c'est vrai que les tests unitaires et les tests d'acceptation testent souvent les memes choses, ils ne sont toutefois pas redondants du tout!

Donc non. Non. Ce n'est pas la même chose. Et ce n'est pas du travail en double. Pourquoi?

1. LES CHEMINS SONT DIFFÉRENTS

Premièrement, Malgré qu'ils testent souvent la meme chose, ils le font via des chemins et des mécanismes différents. Les tests unitaires s'enfoncent dans les entrailles du système en appelant des méthodes particulières de classes particulières. Les tests d'acceptation, d'autre part, invoquent le système de plus loin, au niveau de l'API ou encore au niveau du UI. Les chemins d'execution que ces tests prennent sont très différents.

2. LA VRAIE RAISON

Mais la vraie raison qu'ils ne sont pas redondants est que leur première fonction n'est pas de tester! Le fait que ce sont des tests est un incident. Les tests d'acceptation et les tests unitaires sont avant tout des documents, et deuxièmement des tests. Leur objectif principal est de documenter le design, la structure et le comportement du système. Le fait qu'ils vérifient automatiquement le design, le comportement et la structure est très utile, mais la Spécification est leur objectif premier!

Vous aimez cette bande dessinée? Retrouvez en d'autres sur mon blogue : www.berejeb.com.

L'agilité et le développement vous intéresse? Vous avez une question, une idée ou un projet, n'hésitez pas à me laisser un message sur anis.berejeb@gmail.com

Telechargez gratuitement et partagez avec votre équipe la bande dessinée pas très drôle :