Démystifier les types de tests

Félix-Antoine Bourbonnais et **Pascal Roy**
23 janvier 2014

L'expérience nous a permis de constater qu'il semble y avoir une confusion assez répandue au sein de l'industrie quant à la terminologie et le rôle des différents tests utilisés dans le cadre du développement logiciel Agile.

La plupart des entreprises semblent avoir leur propre terminologie basée sur leur expérience antérieure à l'agilité. La littérature Web présente de multiples versions souvent très disparates des mêmes concepts (e.g. pyramide des tests), ce qui vient probablement ajouter à cette confusion.

Dans ce premier article, nous proposons d'explorer avec vous quels sont les différents types de tests et leurs objectifs de même que la portée des différents niveaux de tests.

Terminologie simplifiée : type et niveau

La cause principale de la confusion que nous observons résulte probablement du fait qu'il est possible de classifier les tests en fonction d'une multitude de caractéristiques.

Tout comme on peut catégoriser les légumes en fonction de leur couleur ou de leur forme. Il existe donc, dans les faits, une multitude de taxonomies possibles. Ainsi, un même test peut appartenir à plusieurs groupes à l'instar d'une carotte qui peut être à la fois classifiée dans "*orange*" et "*racine*".

Pour cet article, nous nous concentrerons sur deux classifications proposées par l'*International Software Testing Qualifications Board [1]* : soit le type (objectif) et le niveau (portée).

Types et niveaux de tests

Type de tests

Un groupe d'activités de tests qui ont un **objectif commun** (*fonctionnalité, acceptation, convivialité, performance, etc.*). Un type de test peut opérer à un ou plusieurs niveaux.

Niveau (portée) de tests

Groupe de tests ayant une **portée similaire** (*unitaires, composantes, systèmes, bout-en-bout, etc.*). Ils sont souvent organisés et gérés ensemble.

Orthogonalité des types et niveaux

Puisque ces deux regroupements sont indépendants et orthogonaux, il est tout à fait possible qu'un test d'acceptation puisse être implanté par un test bout-en-bout, mais pourrait aussi être implanté par des tests unitaires.

À titre d'exemple, considérons une règle d'affaires, permettant de calculer des taxes, implémentée dans une classe (*e.g. Facture.calculerTaxes*).

Pour tester cela, on peut:

- Faire un test **d'acceptation** (*type*) **bout-en-bout** (*niveau*) qui pilote l'interface utilisateur pour créer une facture, manipule les champs, sauvegarde cela dans la base de données et vérifie le résultat du calcul affiché à l'écran au bas de la facture.

- On pourrait aussi faire un test **d'acceptation** à un *niveau* **unitaire** qui utilise directement la classe Facture pour vérifier que la règle d'affaires est bien implantée. Par exemple, on pourrait vérifier que la valeur retournée par la méthode "calculerTaxes" est bien la bonne selon les articles contenus dans la facture.

Dans les deux cas, l'objectif est de vérifier que le besoin du client (la règle) est respecté. Par contre, on peut vérifier cela à différents niveaux avec une portée différente. Ce choix sera guidé en balançant la *fragilité* par rapport à la *complétude* du test comme nous le verrons dans les articles subséquents.

Types de tests

Type de test	Objectif
Acceptation	Déterminer si le système sera accepté ou non par les parties prenantes (client) en vérifiant qu'il répond bien aux critères d'acceptation définis préalablement. Ces tests visent à prouver au client que l'on a développé "le bon produit".
Fonctionnel	S'assurer qu'une fonctionnalité du système respecte bien sa spécification de manière détaillée. Ces tests visent à vérifier si les fonctionnalités sont complètes, valides et solides (dans le détail).
Exploratoire	Essayer de tester des combinaisons ou une utilisation rare pour parcourir des chemins non spécifiés.
Non fonctionnel	S'assurer que le système rencontre bien certains attributs non fonctionnels tels que la fiabilité, l'efficacité, la convivialité (voir plus bas), la maintenabilité, la portabilité, etc.
Performance	S'assurer que le système respecte un niveau de performance requis.
Convivialité (usability)	S'assurer que le logiciel est compréhensible, simple d'apprentissage, facile et agréable à utiliser.

Cette liste ne se veut pas exhaustive, mais présente les types de tests les plus fréquemment discutés dans la communauté Agile.

Niveaux de tests

Niveau de test	Portée
Bout-en-bout (Voir figure 3)	Les tests bout-en-bout assurent que les composantes ou systèmes intégrés d'une application fonctionnent ensemble tel que prévu. L'application complète est testée par *un scénario d'utilisation complet* allant d'un **bout à l'autre du système**: typiquement de l'interface usager jusqu'à la base de données, réseau, matériel, etc. C'est une *longue tranche très mince* qui permet de passer dans toutes les composantes. Certains y incluent même le processus de déploiement, etc.
Système	Comme les tests bout-en-bout, les tests de système opèrent sur le **système complètement intégré**. Toutefois, ils ne testent généralement pas des scénarios d'utilisation complets (workflow). *Ils visent à déterminer que le système intégré répond à certaines caractéristiques spécifiques*. Ces caractéristiques ne sont pas forcément fonctionnelles. Par exemple, un test de charge (type *performance - non fonctionnel*) sur le système (niveau).
Intégration (Voir figure 2)	Ces tests exposent principalement les problèmes et irrégularités dans les interfaces et dans l'interaction entre les différentes composantes du système qui doivent être intégrées. **L'emphase de ces tests est mise sur la communication entre les composantes.**
Composante	Ces tests valident chaque **composante individuelle en isolation** complète des autres composantes du système.
Unitaire (Voir figure 1)	Test d'une **unité en isolation**. Dans un système orienté-objet, l'unité considérée est généralement une **classe**. Malgré le fait qu'il s'agisse probablement de la définition sur laquelle la communauté s'entend le mieux, c'est aussi une source constante de confusion dans les entreprises.

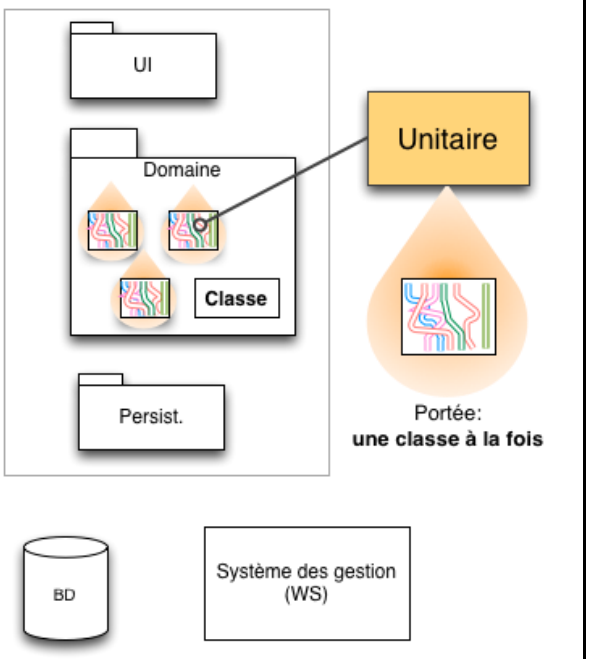

Figure 1 - Portée des tests unitaires

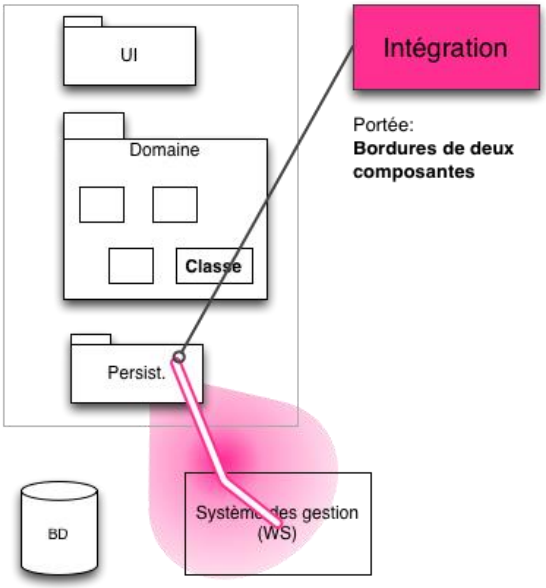

Figure 2 - Portée des tests d'intégration

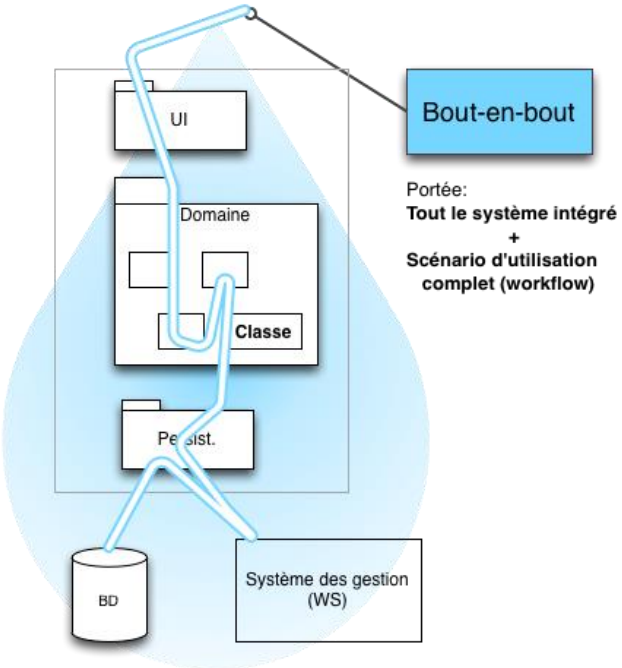

Figure 3 - Portée des tests bout-en-bout

Autres taxonomies

Comme nous l'avons mentionné déjà, il existe maintes autres façons de catégoriser les tests. Voici quelques-unes qui reviennent fréquemment :

Selon la visibilité qu'a le test : Boîte noire/boîte blanche

Cette catégorisation fait référence au fait que les tests utilisent ou non la connaissance du fonctionnement interne du système, de la composante ou même de l'unité testée.

Selon le critère de qualité visé : "-ility tests"

Exemples: performance, convivialité, stabilité ...

Selon le niveau de confiance

Exemple: Alpha, Beta ...

Selon l'auteur : Développeur/Utilisateur

Souvent source de confusion avec le test unitaire qui pourtant, est un niveau (une portée).

Selon la couche de branchement: UI/GUI, Base de données, etc.

Cette taxonomie porte souvent à confusion, car elle peut être interprétée de multiples façons.

Une interprétation commune est que les tests UI sont des tests bout-en-bout qui sont pilotés par l'interface utilisateur graphique. Pourtant, ce n'est pas la seule interprétation possible. Il est tout à fait pensable de créer des tests qui ne testeront que certaines composantes de l'interface utilisateur de façon unitaire (ex.: test unitaire d'un *Presenter* ou d'un *Controller*).

L'utilisation de UI dans le haut de la pyramide de tests de Martin Fowler [9] porte aussi à confusion car on associe naturellement ce terme aux interfaces utilisateurs graphiques (GUI). En réalité, tous les systèmes n'ont pas une telle interface utilisateur graphique. Le UI d'un système pourrait très bien être un service Web par exemple.

Selon le pilote : Automatisé/Manuel

Une autre catégorisation orthogonale à celle du type et du niveau est reliée à l'automatisation ou non des tests. Le fait qu'un test soit automatisé ou non n'est qu'une caractéristique du test. Par exemple, un test d'acceptation peut être automatisé, mais il pourrait également être exécuté de façon manuelle si l'automatisation s'avérait trop difficile ou coûteuse.

Selon l'assurance : Régression (ou non)

Souvent présenté comme un type de test. Les tests de régression ont pour méta-objectif de s'assurer que toutes les modifications ou ajouts de nouvelles fonctionnalités au système ne brisent pas l'existant.

En réalité, il est intéressant de remarquer que tous les types de tests mentionnés dans la section "Type de tests" peuvent servir de régression s'ils sont exécutés lors de tout changement au système.

Selon l'audience des tests : Validation / Vérification

La vérification assure que le système respecte l'ensemble de ses spécifications. Elle est généralement un processus interne à l'équipe de développement.

La validation assure que le système remplit les besoins opérationnels de l'utilisateur (ou des parties prenantes). Elle implique généralement l'acceptation du système par un utilisateur externe. Il ne suffit pas que le système respecte ses spécifications, il doit également être utilisable dans le contexte de l'utilisateur (*"Fit for use"*).

L'exemple de tests de validation typiques sont les tests d'acceptation. En comparaison, les tests unitaires sont généralement utilisés pour la vérification.

Conclusion et prochain article

Nous avons vu qu'un même test peut être classé dans différentes catégories selon l'angle sous lequel on le regarde et qu'il existe dans les faits une multitude de taxonomies possibles. La taxonomie des tests utilisée en développement logiciel est encore très immature. On retrouve en effet énormément de variations aussi bien dans l'industrie que dans la littérature.

Cela ne remet pas en cause la valeur des tests, mais c'est sans nul doute un frein à l'établissement d'un langage et d'une compréhension commune des différents types de tests pour les équipes de développement.

Il est donc important que l'équipe se dote d'une taxonomie de tests commune, afin de faciliter la communication et développer une stratégie cohérente permettant à celle-ci de bénéficier au maximum de son investissement dans les tests.

Dans notre prochain article, nous allons explorer comment les différents tests sont reliés entre eux. Qui est le public visé par les différents types de tests? Qui les écrit? Le tout, afin de profiter pleinement de ses tests à long terme.

Références

[1] International Software Testing Qualifications Board. *Standard glossary of terms used in Software Testing.* http://www.istqb.org/downloads/finish/20/101.html .
[2] IEEE. *IEEE 610.12:1990 Standard Glossary of Software Engineering Terminology.*
[3] W. Hetzel.1988. *The complete guide to software testing – 2nd edition. QED Information Sciences.*
[4] G. Myers.1979. *The Art of Software Testing.* Wiley.
[5] Evans. 2004. *Domain Driven Design: Tackling Complexity in the Heart of Software*. Addison-Wesley.
[6] Overshoe. 2014. *The Art of Unit Testing, Second Edition*. Manning Publications.
[7] Wikipedia. *System Testing*. http://en.wikipedia.org/wiki/System_testing .
[8] Wikiversity. *Software Testing*. http://en.wikiversity.org/wiki/Software_testing .
[9] M. Fowler. *TestPyramid*. http://martinfowler.com/bliki/TestPyramid.html . Thoughtworks.
[10] Wikipedia. *Verification and Validation*. http://en.wikipedia.org/wiki/Verification_and_validation