RETROUVEZ L'INFORMATICIEN

CHAQUE MOIS

CHEZ VOTRE MARCHAND DE JOURNAUX

(Retrouvez les sommaires et les conditions
d'abonnement sur www.linformaticien.com)

Tester n'est pas jouer !

Trop longtemps, le test du code a été mal, voire pas du tout, considéré par les équipes de développement. Il est pourtant un élément clé d'une application de qualité et permet de gagner temps et argent dans la maintenance d'un logiciel. Tour d'horizon par Jean-Baptiste Marcé, fondateur de nLiive, des étapes à mener et des principaux outils disponibles pour automatiser ce processus et garantir une productivité durable.

Le 4 juin 1996, la première fusée Ariane-V décolle de la base de Kourou, en Guyane. Quarante secondes plus tard, elle explose en vol. Après des mois d'investigation, les équipes d'Ariane Espace découvrent que le système de guidage inertiel, pourtant fiable pour la génération précédente, est en cause. Le Cnes a demandé de ne pas effectuer de simulations de vol pour ces appareils et ainsi pouvoir économiser 100 000 dollars sur le coût des préparatifs. Cependant, les accélérations d'Ariane-V étant cinq fois plus fortes que celles d'Ariane-IV, les valeurs mesurées par les accéléromètres provoquent un dépassement de capacité lors du calcul de la position de la fusée. Ariane-V, d'une valeur de 370 millions de dollars, explose, ce qui lui vaut un triste record : le bug informatique le plus coûteux de l'histoire, à cause d'un logiciel non testé !

Une discipline trentenaire

En 1979, soit près de vingt ans avant cette catastrophe, *The Art of software testing* avait pourtant déjà mis en évidence l'importance, et surtout la complexité, des tests logiciels. L'exercice proposé était fort simple : un programme informatique prend en entrée trois entiers. Ces trois entiers sont interprétés comme représentant les longueurs des côtés d'un triangle. Le programme doit déterminer s'il s'agit d'un triangle irrégulier, isocèle ou équilatéral. Il a alors été demandé à des programmeurs expérimentés de réaliser l'ensemble des tests pour valider tous les cas possibles. La moyenne des résultats obtenus a été de 7,8 tests. En réalité, il en fallait… 14, soit près du double ! Cet exemple illustre parfaitement la difficulté à prendre en compte de façon exhaustive l'ensemble des possibilités de comportement, y compris pour un programme simple, sinon simpliste.

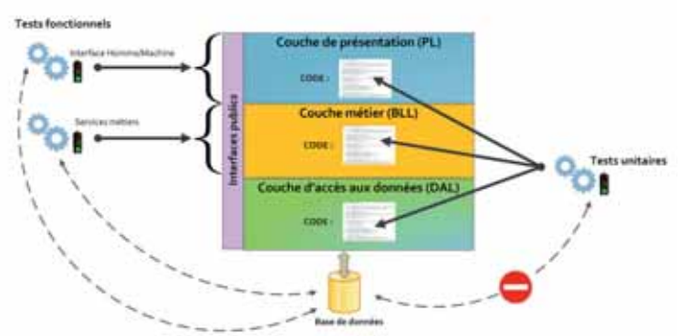
Unitaires et fonctionnels

Unitaires ou fonctionnels, on distingue deux sortes de tests. Les premiers sont réalisés au niveau du code et doivent être écrits par les développeurs. Ces tests unitaires doivent être lancés le plus souvent possible, à chaque « build » et doivent être indépendants de l'environnement dans lequel ils s'exécutent. Deux possibilités existent : écrire le test après avoir développé une fonctionnalité ou avant l'écriture de code – ce qu'on appelle le *Test Driven Development*.

La deuxième catégorie est constituée des tests fonctionnels. Ils se situent au niveau des composants métier de l'application et doivent donc être écrits par des personnes proches du métier de l'application. Ces tests sont destinés à s'assurer que l'application répond aux spécifications fonctionnelles et qu'elle se comporte convenablement en toutes circonstances. Ces tests sont réalisés grâce à des outils non techniques et participent au processus d'intégration continue que nous avons décrit dans le numéro 84 de *L'Informaticien*. Ils peuvent être réalisés de manière très simple, sous forme de wiki avec un logiciel comme FitNesse.

Quels sont les gains ?

Pour savoir quels coûts et quels gains il était possible d'attendre d'une procédure sophistiquée de tests, Microsoft a réalisé une étude à partir d'un cas réel de grande envergure. Le projet comportait en effet 1,2 million de li-



Les tests unitaires testent le code de l'application dans son ensemble. Les tests fonctionnels testent l'application via l'extérieur.

Qu'est-ce qu'un TEST ?

Un test logiciel désigne une procédure de vérification partielle d'un système informatique. Le but est de trouver un nombre maximal de comportements problématiques du logiciel, sachant qu'il est pratiquement impossible de prouver qu'un logiciel fonctionne dans tous les cas. Un test est un ensemble de cas à tester (état de l'objet à tester avant exécution du test, actions ou données en entrée, valeurs ou observations attendues et état de l'objet après exécution), éventuellement accompagné d'une procédure d'exécution (séquence d'actions à exécuter). Il est lié à un objectif.

Les principaux outils

Tests unitaires :

Xunit (jUnit, Nunit, …), MSTest.

Tests fonctionnels :

FitNesse, Greenpepper, Twist, Concordion, Cucumber, jBehave, RobotFramework, Selenium, Watir, Canoo Webtest, Webrat, Test Director.

gnes de code en C# et mobilisait 32 développeurs et 15 testeurs pendant les deux années de développement. La version 1 a été écrite sans test automatisé alors que la version 2 l'a été avec des tests écrits par les testeurs post-développement. Le nombre de bugs remontés par l'équipe de tests a diminué de 20,9 %.

Un évident retour sur investissement

Les développeurs ont également estimé que la mise en place des tests aboutissait à une durée de développement de 30 % supplémentaire. Cependant, les erreurs grossières sont moins fréquentes et les développeurs ont le sentiment de produire un code plus robuste. En parallèle, Microsoft a mené une étude autour d'autres projets réalisés avec la technique du TDD. Le nombre de bugs diminue dans une proportion de 62 à 91 % avec le même coût en termes de temps investi (20 à 30 %). Le retour sur investissement est donc évident.

Pour les équipes, le challenge est donc d'automatiser les tests unitaires, d'intégration et fonctionnels. Pour ce faire, il convient d'orchestrer les divers outils de tests automatiquement dans un processus d'intégration continue. Il devient donc impératif d'instaurer une véritable culture projet pour toute l'équipe autour de la pratique de tests et de l'intégration continue. C'est ainsi qu'il devient possible d'industrialiser le développement logiciel. ■

À propos de l'auteur

Jean-Baptiste Marcé est co-fondateur et directeur technique de la société nLiive. Il totalise plus de vingt d'expérience dans la conception et l'industrialisation du développement logiciel. Le cabinet de conseil nLiive est spécialisé sur les technologies Microsoft .NET et sur la plate-forme de Cloud-computing de Microsoft : Azure. nLiive est également éditeur de solutions distribuées en mode SaaS (Software as a Service). Plus d'infos sur *www.nliive.com*

Bien différencier les tests unitaires des tests fonctionnels.