Testing logiciel : tests unitaires, tests d'intégration

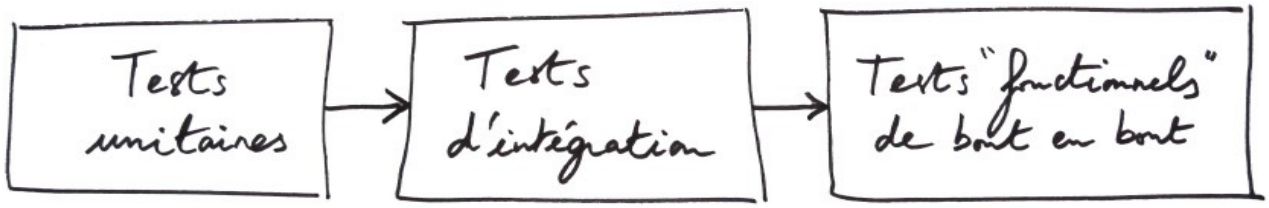
http://www.test-recette.fr/

Comme nous l'avons vu plus tôt une partie importante de la qualité d'un logiciel doit être apportée et contrôlée par les équipes techniques. Ce contrôle est assuré par la mise en œuvre de tests techniques : tests unitaires, tests d'intégration, tests de montée en charge…

Cinématique de test

En matière de testing, beaucoup de monde s'excite avec ce que l'on appelle les « tests fonctionnels » ou encore (souvent à tort) la « recette ». On parle aussi parfois de « tests end to end » ou « tests de bout en bout » pour qualifier les tests de l'ensemble des intégrations nécessaires pour la mise en œuvre d'un service logiciel répondant à une problématique fonctionnelle (commande d'un article sur un site marchand, réservation d'un billet de train, etc.).

En réalité, pour fournir un produit de qualité, un seul chemin est possible, exposé de manière schématique par la figure ci-dessous :



Important : toutes les étapes décrites par la figure ci-dessus sont obligatoires (tests unitaires, tests d'intégration, tests de bout en bout) et l'ordre compte !

Contenu de ce chapitre

Dans ce chapitre, nous allons nous consacrer aux deux premiers étapes du processus décrit plus haut : testing unitaire et testing d'intégration.

- Les bases du testing – Dans cette section, vous découvrirez quelques éléments de base, à connaître absolument avant de vous lancer dans l'écriture de vos premiers tests unitaires, d'intégration, de montée en charge, etc.
- Déployer des tests unitaires – OK, finie la théorie, venons en à la concrétisation de ce qui a été exposé dans la section consacrée aux bases du testing technique : voyons comment implémenter concrètement des tests dans un système informatique : http://www.test-recette.fr/tests-techniques/deployer-tests-unitaires/
- Déployer des tests d'intégration – Après avoir vu comment mettre en place des tests d'intégration, dans différents langages et selon différents paradigmes, nous allons à présent nous concentrer sur les tests d'intégration : http://www.test-recette.fr/tests-techniques/deployer-tests-integration/

Les bases du testing

Tests unitaires

Les plus beaux et complexes ouvrages sont composés de pièces individuellement fiables. Cette page vous permettra de comprendre la nature et l'intérêt des tests unitaires en génie logiciel.

Définition : qu'est-ce qu'un test unitaire ?

Comme beaucoup de concepts en informatique, le concept de test unitaire n'est pas simple à définir. Non pas qu'il soit réellement complexe, mais plutôt parce que beaucoup de développeur se font leur propre idée de ce qu'est un test unitaire…

Définition : un test unitaire est un procédé permettant de s'assurer du bon fonctionnement d'une unité de programme.

OK, cette définition est assez générale, il ne prend pas de risque vous direz-vous peut-être. Certes, mais il est vrai que le terme « unitaire » peut lui-même renvoyer à ces réalités assez différentes. Une unité de programme est-elle une fonction ? Une classe ? Une instruction ?

Au fond, à quoi sert un test unitaire ?

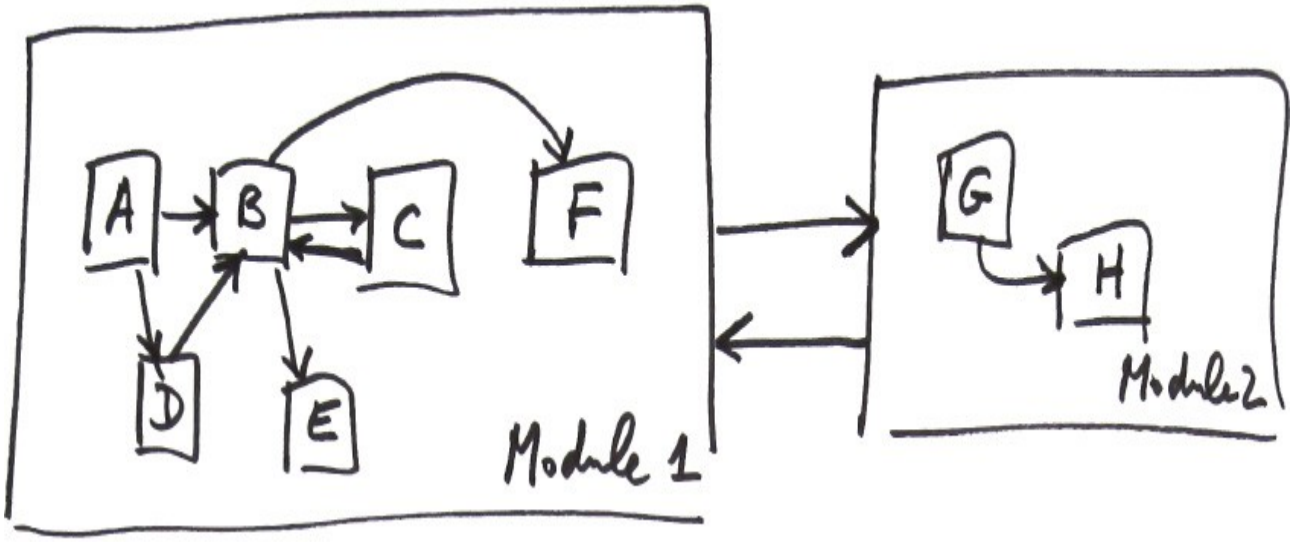
Au delà des définitions, une bonne manière de comprendre le testing unitaire est d'en comprendre le but, de façon pragmatique.

Il s’agit simplement de vérifier, en fonction de certaines données fournies en entrée d’un module de code (on parle parfois d'unité fonctionnelle testée, ou SUT pour system under test), que les données qui en sortent ou les actions qui en découlent sont conformes aux spécifications du module.

En d'autres termes, il s'agit de vérifier le respect du contrat de service du module.

Testing unitaire : un exemple

Considérons un système informatique composé de deux gros modules interdépendants. Dans chacun de ces modules, nous trouvons différents composants, eux-même dépendants entre eux.



Sur notre exemple, le module D accepte en entrée des données qui lui sont fournies par le composant A, et produit des données en résultat qui sont utilisées par le composant B. OK ?

Imaginons maintenant que le développeur Arnaud soit responsable du développement et de la maintenance du composant A, que Brigitte soit responsable du composant B, et que Daniel soit responsable du composant D.

Respecter son contrat…

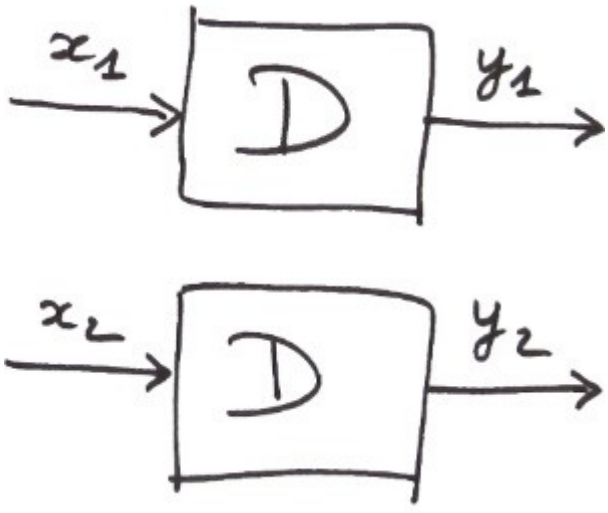
Si Arnaud fait mal son boulot en déployant un composant A bogué qui n'implémente pas correctement la spécification qui lui a été fournie, il y a fort à parier que le composant D du pauvre Daniel plantera lamentablement, même s'il est correctement développé. C'est alors que commencent de longues et pénibles discussions, du genre « Non, mais c'est ton code qui est faux, non c'est le tien, bla, bla, bla… ».

Une solution simple pour s'assurer que chacun fait correctement son travail est tout simplement d'implémenter des tests unitaires sur chacun des composants. Cette tâche revient à chaque développeur : Arnaud écrira les tests unitaires de A, Brigitte écrira les tests unitaires de B, Daniel écrira les tests unitaires de D, etc.

Le cas de Daniel

Dans le cas de Daniel, il s'agit de s'assurer que si le composant D reçoit les données qu'il attend (i.e. conformes à la spec), alors il produit les résultats qu'il est censé produire (i.e. conformes à la spec). Ni plus, ni moins.

Les tests unitaires de D auront donc grosso-modo cette forme :



En d'autres termes : si je fais entrer le jeu de données x1 dans la moulinette D, alors il sort y1. Si je fais entrer le jeu de données x2 dans la moulinette D, alors il sort y2.

Pré-requis pour tester unitairement

Il y a en fait peu de pré-requis pour pouvoir tester son code unitairement. Nous le verrons plus loin, quasiment tous les langages de programmation et tous les frameworks de développement dignes de ce nom proposent des outils de testing unitaire.

Le principal pré-requis au testing est la modularité du code. Une idée simple : diviser pour mieux régner. Un code modulaire est bien plus robuste qu'un imbroglio de règles de gestion et de traitements. Plus il est modulaire, plus un code est facile à tester unitairement…

Conclusion : chacun respecte son contrat et tout le monde est content !

À retenir – Le testing unitaire repose au fond sur un principe très simple : un système aussi gros et complexe qu'il soit est toujours composé différentes parties plus petites que lui. Il est indispensable (mais pas suffisant) que chacune de ces parties remplisse correctement sa fonction pour que l'ensemble soit fonctionnel et fiable.