I. Le test

1. Présentation

Le but de ce livre est de vous faire connaître monde du test logiciel. De vous présenter ses contraintes, ses outils et techniques et les personnes qui travaillent dans ce milieu en plein essor.

Pour atteindre cet objectif, le livre se concentrera sur 2 points :

- Découvrir des notions de l'ISTQB pour vous aider à réussir la certification niveau fondation
- Apporter de la culture générale relative à ce métier notamment à travers des retours d'expérience de personnes travaillant dans le test mais aussi en abordant des sujets qui ne font pas forcément nécessaires au passage de la certification.

Ce livre n'a pas vocation à être utilisé uniquement pour passer le niveau fondation de la certification ISTQB. Ce livre est accessible au plus grand nombre et doit être vivant, utilisé régulièrement pour lire des parties de ce dernier, prêté à des amis, utilisé pour se rappeler ou comprendre certains sujets, défis, évolutions du test.

a. But

Le test est en plein essor. On voit se multiplier les offres d'emplois et les postes (assez variés) de testeurs dans les entreprises. Pourquoi le test prend-il autant d'importance, quel est son but ?

Voici les différentes réponses à un sondage mené auprès de professionnels du test à ce sujet, comme vous le voyez les réponses sont diverses et variées :

Réponse	% obtenu
Améliorer la qualité	41,6%
Donner de la visibilité sur la qualité de l'application	25,7%
Vérifier la conformité aux spécifications	16,8%
Trouver les bugs avant les utilisateurs	12,9%
Avoir une application avec 0 défaut	1%
Donner du feedback	1%

Répondre aux exigences explicites ou implicites des usagers du projet	1%

2 réponses se démarquent :

- améliorer la qualité (42%)
- donner de la visibilité sur la qualité de l'application (25%)

Selon ISTQB, le but principal des tests est de donner de la visibilité sur la qualité de l'application, un indice de confiance sur le logiciel testé.

Les problèmes remontés lorsque l'on parle de tests, sont principalement liés à des anomalies (bugs) importantes non détectées ou de non connaissance de la qualité de ce qui était livré en production.

Les tests permettent justement de donner cette visibilité et indiquent un certain niveau de confiance par rapport à la qualité de l'application. L'amélioration de cette dernière est un effet « collatéral » qui se produit lorsque les tests détectent des anomalies qui doivent être corrigées. Dans le cas où les tests ne détectent pas d'anomalies à corriger (et donc que la qualité de l'application est jugée suffisante), cette qualité n'est pas améliorée.

Tout le travail de conception des tests est de construire une batterie de tests permettant d'atteindre la visibilité souhaitée sur la qualité de l'application (et de préférence, atteindre cette visibilité avec le moins de ressources possibles). C'est le résultat de l'ensemble de ces tests qui doit permettre de se faire une idée sur la qualité de l'application et donc de ne pas mettre en production une application en se demandant comment l'application réagira.

b. Pourquoi tester ?

Toutes les entreprises logicielles font du test. Avoir une bonne visibilité sur la qualité de l'application que l'on développe est donc une information particulièrement importante.

Cela paraît évident mais il est toujours bon de rappeler qu'une entreprise ne fait pas de dépenses sans raison. Pour chaque dépense elle attend un retour sur investissement. Les tests (bien gérés) ont donc un bon retour sur investissement et « coûtent » moins qu'ils ne rapportent !

Pourquoi cette importance grandissante des tests ?

Les 2 raisons principales sont :

- **les applications qui sont de plus en plus complexes.** Une « simple » application mail représente des centaines de jours de travail, des milliers de lignes de codes et des dizaines de fonctionnalités.

- **le bouche à oreille**. Lorsqu'un utilisateur juge qu'une application est de mauvaise qualité il le fait savoir à son entourage, ses relations sur les réseaux sociaux et les sites de notation afin de prévenir de cette mauvaise qualité.

La complexification des applications est facile à appréhender. En effet, on ne trouve plus d'application comme un système d'exploitation faite entièrement par un développeur tout seul dans son garage.

Le bouche à oreille prend de plus en plus d'importance et son impact est beaucoup plus important qu'il y a encore 15-20 ans. La raison est simple, notre « entourage » et notre puissance de communication a été multipliée. Multipliée par les réseaux sociaux, multipliée par les forums et enfin, multipliée par les systèmes de notations. C'est simple, si une application sort et qu'elle est de mauvaise qualité, elle sera alors mal notée. S'ensuit alors un cercle vicieux car personne ne veut investir (en temps et/ou argent) sur une application considérée comme mauvaise. Même si l'application peut alors s'améliorer il sera alors très compliqué (voir impossible) de remonter la pente et de devenir une application à succès. On comprend vite le retour sur investissement que peuvent apporter les tests.

c. Stéréotypes/réalités

Le test est encore assez peu/mal connu.

Comme tout ce qui est peu ou mal connu il est sujet aux stéréotypes. Nous allons ici nous attarder sur des stéréotypes fréquents qu'il est bon de combattre.

Les tests ça coûte cher

FAUX, tout du moins, en partie. Mettre en place des processus de test coûte de l'argent. Par contre ces processus permettent souvent de repérer des bugs avant les utilisateurs et donc de les résoudre avant d'impacter les clients. De nos jours, sortir une application inutilisable car trop buggée signifie la mort de cette application, notamment avec le système de note et d'avis sur internet.

Les bonnes procédures de tests ont un bon retour sur Investissement. Les vrais coûts des tests doivent être calculés avec ce qu'ils nous permettent de ne pas dépenser (en réparation d'anomalie en production) et ce qui permettent d'obtenir en adhésion (et donc €) des utilisateurs (par l'atteinte de la juste qualité).

Les tests c'est à la fin du projet

FAUX, plus les tests commencent tôt mieux c'est. Même les spécifications peuvent être testées, l'ISTQB conseille même fortement de le faire. Une erreur trouvée dans les spécifications ne coûte quasiment rien à corriger ! Ce sujet sera développé plus en profondeur dans la partie sur le Shift Left testing et dans la présentation du 3ème principe des tests.

Le projet est en retard à cause des tests

FAUX. On pointe souvent les tests comme cause du retard d'un projet, c'est rarement le cas car si les tests sont en retard cela peut être pour plusieurs raisons comme :

- une livraison tardive
- une qualité en dessous de ce qui est souhaitée
- des problèmes sur la plateforme de tests…

Si un projet est en retard ce n'est pas forcément la faute du dernier maillon de la chaîne. Comme tout acteur du projet, le testeur souhaite une sortie en temps et en heure de l'application.

Plus il y a de tests mieux c'est

FAUX, le nombre de tests ne garantit pas la qualité de ses derniers ni une bonne couverture. Il peut y avoir 50 000 tests pour une application mail s'ils sont tous concentrés sur l'authentification et la lecture des mails, il n'y a alors aucune vision sur la capacité de l'application à envoyer des mails. Le plus important est d'avoir des tests pertinents.

De plus un grand nombre de tests signifie avoir assez de temps pour les maintenir les cas à jour et analyser les cas en échecs. Des tests qui ne sont pas à jours sont inutilisables, des tests non analysés lorsqu'ils sont en échec n'ont pas d'intérêt non plus.

Il faut automatiser tous les tests

FAUX, Il ne faut automatiser les tests qui sont fréquemment exécutés afin d'avoir un retour sur investissement. Il ne faut pas non plus oublier que certains tests sont très difficiles à automatiser et que dans certains cas il est plus intéressant de conserver ces tests en tant que tests manuels. Enfin, il est toujours bon de garder un peu de temps pour l'exécution de tests exploratoires qui sont des manuels par définition (voir chapitre dédié pour plus d'informations).

L'automatisation c'est juste un enregistrement de clics ou de macros

FAUX. C'était vrai au tout début de l'automatisation. Les personnes en charge de l'automatisation enregistraient un parcours (généralement sur une application web) afin d'enregistrer le test. Il n'y avait alors plus qu'à lancer le test pour qu'il se ré-exécute. Depuis nous nous sommes aperçu que ce type d'automatisation était inefficace et totalement impossible à maintenir. Depuis, les tests d'interface graphique sont en fait du code… Le test automatisé, c'est du développement !

L'automatisation est la solution à tous les maux

FAUX, l'automatisation est souvent vue comme la solution miracle. Malheureusement cette solution miracle n'existe pas, les tests automatisés (même bien maintenus, facilement maintenable et exécutable) ont leurs limites. Les tests automatisés ne peuvent pas remplacer les utilisateurs, leurs ressentis, leurs impressions… Dans ce cas la présence de tests manuels (à travers des phases de tests exploratoires est obligatoire).

Une application avec des bugs est une application non ou mal testée

FAUX, on ne peut pas détecter tous les bugs. Et même si on y arrivait, on ne les corrigerait pas forcément (trop coûteux). De plus ce n'est pas le testeur qui a introduit les bugs! Ils viennent majoritairement du développement, le reste étant généralement introduit lors du développement.

Les testeurs n'y connaissent rien en technique et en développement

FAUX, il y a différents profils de testeurs. Certains sont plus techniques que d'autres. Ce n'est pas parce que l'on est testeur que l'on ne comprend rien au développement et à ses problématiques. Un nombre important de testeurs a de bonnes bases voire même une expérience en développement (ou en automatisation de tests). La majorité des jeunes testeurs a d'ailleurs une formation d'ingénieur informatique. De plus les testeurs intégrés aux équipes agiles vivent au contact des développeurs ce qui leur permet de beaucoup mieux comprendre la partie technique du projet, au point de dépanner les développeurs quand l'équipe en a besoin.

Le test c'est le travail des testeurs

FAUX, le test doit être à tous les niveaux du projet, de l'écriture des spécifications (avec des revues), au développement (avec les tests unitaires), jusqu'aux tests d'acceptation (par exemple avec des phases de bêta test) …

Les tests il suffit de les exécuter

FAUX, l'exécution des tests seule ne sert à rien. Un test en échec doit être analysé afin d'avoir une plus-value. De plus avant de les exécuter il faut les définir, les écrire puis les maintenir.

Le métier du test c'est uniquement exécuter et analyser des tests

FAUX, le métier du test c'est aussi beaucoup de communication et de compromis. La communication est en effet nécessaire pour concevoir les cas de test, décrire les anomalies remontées par les tests et faire le bilan des campagnes. Le compromis est également obligatoire. Lorsque l'on est testeur il faut faire des choix, que tester ? Pourquoi le tester ? Dans quelles conditions ? Lorsque l'on teste quelque chose c'est toujours au détriment d'une autre !

N'importe qui peut être testeur

FAUX, pour être testeur comme pour être développeur, business analyste ou commercial il faut avoir certaines qualités. Les principales étant de mon point de vue la curiosité, une bonne communication et une bonne compréhension fonctionnelle du produit.

Le testeur est l'ennemi du développeur

FAUX, le testeur a le même but que le développeur : livrer un logiciel de qualité apprécié par l'utilisateur final. Un testeur ne remonte pas un bug par plaisir, le but du testeur n'est pas non plus de dénigrer le travail du développeur.

Le test c'est uniquement les tests fonctionnels

FAUX, le test c'est l'ensemble des processus visant à améliorer ou assurer la qualité d'un logiciel. Le « testeur » n'est par contre pas affecté à l'ensemble de ces tâches. Le test c'est pour tous les membres du projet ! De plus il y a également tous les aspects non fonctionnels comme les performances, l'adaptabilité, l'ergonomie….

Les tests ne « produisent » rien

FAUX. Certes les tests ne développent pas l'application, n'écrivent pas de spécification et ne rapportent pas d'argent de manière directe. Néanmoins les testeurs « produisent » des tests (automatisés ou non), des campagnes, des bilans… Il y a de nombreux livrables dans le test.

d. Lien test/qualité

Dans l'imaginaire commun une application de mauvaise « qualité », ou même une application avec des anomalies, est une application qui n'a pas (ou mal) été testée.

De même, pour grand nombre de testeurs, le but principal des tests est d'améliorer la qualité de l'application. Pour rappel, les tests en eux-mêmes ne peuvent pas améliorer une application. Néanmoins, l'amélioration de la qualité est une conséquence directe de la visibilité sur la qualité de cette dernière apportée par les tests. En effet, la visibilité sur la qualité de l'application permet de savoir quelles sont les faiblesses (anomalies) de l'application qui doivent être corrigées… Ce qui entraine une amélioration générale de la qualité.

Enfin, le métier de « testeur » est souvent assimilé à « ingénieur QA », QA pour « Quality Assurance » (Assurance Qualité). Le test est ici perçu comme l'assurance d'atteindre la juste qualité pour le logiciel testé.

Ces exemples mettent en lumière le lien étroit entre les tests et la qualité et indique que la qualité de l'application va être au moins en partie liée à la qualité des tests.

Une application est développée pour un public spécifique ciblé. Ce public a des habitudes de consommation, des exigences et un certain pouvoir d'achat. En fonction de tout cela une « juste qualité » est définie. Pour assurer l'atteinte de cette qualité il faut alors concevoir des tests, implémenter des processus et des outils permettant à moindre coût de garantir (ou d'assurer) ce niveau de qualité. Les tests donnent ensuite une visibilité sur l'atteinte ou la non atteinte de cette qualité.

Voici un exemple afin de mieux comprendre le concept de « juste qualité » et de test adaptés :

La problématique est la suivante : « La recherche d'un terrain pour organiser un match de foot »

1er cas :

Un animateur en centre de loisir souhaite proposer un match de football aux enfants de ce centre.

Il recherche donc un terrain :

- assez grand pour que les enfants puissent jouer
- plat
- pas trop dangereux (pas de gros trous ni gros cailloux)
- le plus proche possible du centre
- accès gratuit (aucun budget pour louer un terrain)

Un terrain vague en béton peut donc suffire. Les cages ne sont peut-être même pas nécessaires car on pourrait les « construire » avec des sacs ou des plots disponibles au centre. La phase de « test » sera donc rapide et le premier terrain trouvé fera sûrement l'affaire.

2nd cas :

Un responsable d'un club de foot souhaite organiser un match amical entre son équipe de -13 ans et celle de la ville voisine. Suite à ce match un grand barbecue sera organisé.

Il cherche donc un terrain :

- aux dimensions réglementaires
- avec l'équipement réglementaire
- avec des tribunes
- avec une zone pour faire un barbecue
- avec un parking pour que les parents puis amener et regarder leurs enfants jouer
- rentrant dans le budget alloué à l'évènement

Ici il faudra sûrement louer un terrain municipal et faire de nombreuses vérifications pour s'assurer que tous les critères soient réunis. La distance sera par contre moins importante et il est également fort probable que plusieurs terrains seront étudiés.

On retrouve le 6ème principe du test qui sera abordé dans un chapitre ultérieur : Les tests dépendent du contexte. Même si le besoin primaire est le même, la solution est totalement différente dans les 2 cas présentés.

La qualité est un choix. En effet, le niveau de qualité doit se définir au début de chaque projet et dépend d'un grand nombre de critères comme le temps, le public visé, le budget… Bref, du contexte.

Cette qualité qui est jugée comme « juste qualité » résulte donc de choix et d'analyses. Il est donc parfaitement légitime de lancer une application de « mauvaise qualité » car on souhaite se placer sur du « low cost » ou si l'on souhaite être les premiers sur un marché précis.

e. Les 7 principes

Les 7 principes du test sont des principes qui se sont imposés suite à des années d'expérience avec les tests logiciels. Ces principes sont communs à tous les tests logiciels, et ce, malgré la grande diversité de ces derniers.

Il est bon de noter que ces principes relèvent tous du bon sens.