Correction du QCM

Nous avons proposé dans le dernier chapitre de notre livre quelques questions à choix multiples en relation avec les différents chapitres du livre et couvrant le syllabus de l'ISTQB ; ces questions ont pu être utilisées dans la préparation à la certification ISTQB niveau fondation telle qu'elle était définie lors de la parution de cet ouvrage. La certification, niveau fondation, contient une quarantaine de questions auxquelles il faut répondre en 1 heure. Chaque question ne contient qu'une bonne réponse et il n'y a pas de points négatifs. Pour obtenir la certification, il faut avoir plus de 65 % de réponses valides. Pour le niveau avancé, il faut répondre à un questionnaire de 65 questions à réaliser en 1 h 30 avec un niveau de difficulté associé à chaque question (de 1 à 3). Il faut obtenir plus de 65 % des points totaux.

Nous proposons ici les réponses à ce QCM et apportons quelques éclairages sur ces réponses.

Q1) Pendant les tests, il est nécessaire de : ***(1)***

a) Tester toutes les parties du système avec la même intensité, parce que des anomalies peuvent être détectées n'importe où. ☐

b) Tester l'interface utilisateur en priorité parce que ses anomalies sont les plus dérangeantes pour l'utilisateur. ☐

c) Tester en priorité les composants du système pour lesquels les défauts génèrent les risques les plus importants. ■

d) Tester en priorité les accès aux bases de données de façon à éviter des données erronées et des inconsistances dans la base de données. ☐

e) Tester en priorité les performances de façon à améliorer les temps de réponse. ☐

Q1) Bien que les défauts puissent apparaître potentiellement dans toutes les parties du logiciel, l'effort de test ne doit absolument pas être réparti uniformément mais centré sur les zones du logiciel qui présentent le plus grand risque de défaut car plus complexes ou plus impactant en cas de défaut ; i.e. les 20 % du code qui peuvent contenir les 80 % de défauts. La bonne réponse est donc la réponse **c**.

Q2) Les tests statiques et les tests dynamiques : ***(1)***

a) Sont deux familles de tests complémentaires. ■

b) Sont incompatibles : il faut choisir avant de commencer la phase de tests. □

Q2) Les tests tentent de découvrir les défauts présents dans un logiciel ; pour ce faire, une approche est d'observer le comportement de celui-ci dans différentes situations bien choisies, on parle alors de tests dynamiques. L'autre possibilité est d'analyser le logiciel en étudiant la façon dont il est conçu, sans le faire fonctionner ; on parle alors de tests statiques. Ces deux approches sont bien entendu complémentaires et la bonne réponse est **a**.

Q3) Les tests sont : ***(1)***

a) Inutiles la plupart du temps du fait de l'amélioration continue des techniques de développement et d'intégration. □

b) Indispensables pour réduire les coûts et les délais tout en augmentant la qualité. ■

c) Importants mais impraticables donc à éliminer. □

Q3) Les tests, bien que demandant des efforts non négligeables, permettent, lorsqu'ils sont bien menés, d'éliminer bon nombre de défauts présents dans un logiciel et donc d'en améliorer la qualité. Si les outils, techniques et les langages utilisés pour développer un logiciel permettent « d'encadrer » les phases de développements, ils ne peuvent, et ne pourront sans doute jamais, éliminer la présence de défaut dans un logiciel ; les tests sont et resteront nécessaires ; la bonne réponse est donc la réponse **b**.

Q4) Quelle affirmation est exacte : ***(1)***

a) L'activité de test nécessite d'être curieux c'est pour cela que c'est préférable de la confier à des personnes sans expérience. □

b) L'activité de test nécessite d'être curieux mais demande de l'expérience. ■

c) L'activité de test ne nécessite qu'un bon sens de la communication. □

d) L'activité de test nécessite de prendre position quitte à « grossir » ou extrapoler certaines données. □

Q4) La curiosité est un atout indéniable pour un testeur ; néanmoins, cette qualité à elle seule ne peut se substituer à l'expérience et à la formation, comme c'est le cas pour la plupart des métiers. La bonne réponse est donc la réponse **b**.

Q5) Une fonction calcule un résultat à l'aide quatre paramètres entiers (codés sur 32 bits) ; il y a donc 232*4 = 2 128 ≈ 1 043 possibilités distinctes d'appel de la fonction : ***(2)***

a) Ce nombre est tellement grand qu'il est inutile d'essayer de tester la fonction ; la tâche est impossible ! ☐

b) En y consacrant de l'énergie on pourra tester toutes les possibilités. ☐

c) En procédant avec méthode il suffit de tester la fonction avec quelques valeurs pertinentes pour avoir une très grande assurance sur sa correction. ■

d) Ce cas n'arrive jamais, il est inutile de se poser ce genre de problème ! ☐

Q5) Les méthodes de tests par partition ou de tests par table de décision permettent de mettre en évidence, à partir des spécifications, des ensembles de valeurs conduisant à des comportements identiques ; ces stratégies permettent ainsi de réduire des ensembles de valeurs gigantesques, cas fréquents, à quelques valeurs pertinentes. La réponse juste est donc la réponse **c**.

Q6) Les tests d'intégration : ***(1)***

a) Ne peuvent être fait correctement que si les tests unitaires ont été faits correctement. ■

b) Remplacent les tests unitaires. ☐

c) Sont incompatibles avec les tests unitaires. ☐

d) Se font avant les tests unitaires. ☐

Q6) Les tests d'intégration visent à détecter les défauts d'interaction entre les composants et les défauts présents au niveau des interfaces. Ils n'ont d'intérêt que si les composants fonctionnent correctement en isolation, donc si les tests unitaires ont déjà été menés. La réponse correcte est donc la réponse **a**.

Q7) Les méthodes agiles : ***(1)***

a) Mettent en avant la souplesse des programmeurs car ils doivent programmer dans différentes postures pour maintenir une forte concentration. ☐

b) Mettent en avant les tests qui doivent être réalisés très régulièrement. ■

c) Rendent inutiles les tests car le code ne peut contenir des erreurs avec ces techniques. ☐

d) Déconseillent l'utilisation des tests car ceux-ci risquent de « casser » l'aspect agile des méthodes. ☐

Q7) Bien que les méthodes agiles demandent une certaine jeunesse d'esprit il n'est pas besoin d'être athlétiques pour les mettre en œuvre. Un des points essentiel de ces méthodes est de placer les tests au cœur du processus de développement en réduisant la durée des cycles et en favorisant l'anticipation des tests avant les réalisations. La réponse correcte est donc la réponse **b**.

Q8) Quel terme ne désigne PAS une stratégie de tests d'intégration : ***(1)***

a) Méthode incrémentale ascendante. ☐

b) Méthode incrémentale descendante. ☐

c) Méthode de la régression minimale. ■

d) Méthode du big bang. ☐

Q8) Les tests d'intégration consistent à observer les interactions entre les différents composants du logiciel. Dans le cas où les observations portent sur tous les composants qui sont mis en fonctionnement au même moment on parle de stratégie « big bang », dans le cas où l'on met en fonctionnement les composants dans un ordre lié aux dépendances entre les composants on parle de stratégie ascendante ou descendante. Il fallait donc choisir la réponse **c**.

Q9) La technique des tests aux limites consiste à : ***(1)***

a) Pousser aux limites les équipes de tests en les mettant fortement sous pression. ☐

b) Essayer d'atteindre la limite des fonctions de maturité du logiciel. ☐

c) Faire fonctionner le logiciel aux limites de ses spécifications. ■

d) Faire fonctionner le logiciel le plus longtemps possible. ☐

Q9) Tester un composant aux limites consiste à observer son comportement lorsqu'il est placé près de ses limites de fonctionnement telles que décrites par les spécifications. La bonne réponse est donc la réponse **c**.

Q10) La technique de partitionnement en classes d'équivalence : ***(1)***

a) Consiste à trouver des domaines sur lesquels le logiciel se comporte de façon homogène. ■

b) Consiste à trouver des logiciels équivalant au logiciel à tester et à réutiliser les jeux de tests qui ont été faits pour ce logiciel. ☐

c) Consiste à diviser l'équipe de test en groupe de tailles et d'expériences équivalentes. ☐

Q10) La technique de partitionnement en classes d'équivalence permet par l'analyse des spécifications de réduire de grands ensembles de valeurs en plus petits domaines de telle sorte que chacun de ces domaines définisse un ensemble de valeurs équivalentes vis-à-vis du comportement du logiciel à tester. La bonne réponse est donc la **a**.

Q11) Comment désigne-t-on également les tests « boîtes noires » ? ***(1)***

a) Tests unitaires ☐

b) Tests d'intégration ☐

c) Tests fonctionnels ■

d) Revues de code ☐

Q11) Les tests « boîtes noires » sont souvent appelés également tests fonctionnels car l'on s'intéresse au fonctionnement du logiciel sans essayer d'en connaître les détails de réalisation ; ces tests peuvent porter sur un composant seul ou sur un logiciel complet ; la bonne réponse est donc la réponse **c**.

Q12) Les tests « boîtes noires » utilisent : ***(1)***

a) Le code du logiciel à tester. ☐

b) Les commentaires du logiciel à tester. ☐

c) Les spécifications du logiciel à tester. ■

d) Les commentaires de l'équipe de réalisation. ☐

Q12) Les tests « boîtes noires » portent ce nom car on ne cherche pas à connaître ce qu'il y a dans le logiciel testé ; c'est une boîte opaque, noire. Les seuls éléments que le testeur va utiliser pour composer ses tests sont tout ce qui relève des spécifications et il va s'interdire tout ce qui pourrait être lié aux détails de réalisation. Il fallait donc choisir la réponse **c**.

Q13) Les tests « boîtes blanches» utilisent (plusieurs réponses possibles) : ***(1)***

a) Le code du logiciel à tester. ■

b) Les commentaires du logiciel à tester. ■

c) Les spécifications du logiciel à tester. ■

d) Les commentaires de l'équipe de réalisation. ■

Q13) Au contraire, les tests « boîtes blanche » vont utiliser les détails de réalisation pour définir les jeux de tests ; il peut s'agir du code, des commentaires présents dans le code mais aussi les commentaires faits par l'équipe ainsi que les spécifications ; il fallait donc cocher l'ensemble des réponses.

Q14) La notion de couverture dans les tests : ***(1)***

a) N'est pas utilisée. ☐

b) Permet de « couvrir » une erreur faite lors des tests qui endommage le logiciel à tester. ☐

c) Sert à vérifier qu'un ancien jeu de valeurs peut être réutilisé dans un nouveau contexte. ☐

d) Sert à définir un objectif pour les tests. ■

Q14) La notion de couverture ne fait pas référence au sens de protection, dans le sens se couvrir des conséquences d'une action, mais au sens de parcourir, couvrir un espace, couvrir un ensemble d'instruction ou de condition liées au code à tester dans le but d'analyser comment les tests ont parcourus ou couvert le logiciel lors de leur exécution. C'est une notion ancienne très utilisée. La bonne réponse est donc la **d**.

Q15) La couverture de « toutes les décisions » (dite encore « toutes les branches ») : ***(1)***

a) Recouvre la couverture « tous les chemins ». ☐

b) Recouvre la couverture « toutes les instructions ». ■

c) Recouvre la couverture « toutes les conditions ». ☐

d) Recouvre la couverture « tous les chemins ». ☐

Q15) On peut chercher à exécuter toutes les instructions lors des tests et l'on parle de couverture de toutes les instructions ; on peut aussi chercher à évaluer l'effet des conditions élémentaires présentes dans les branchements ou décision ; on parle alors de toutes les conditions. Lorsque l'on se focalise sur les branchements, on parle alors de toutes les décisions. Par construction, lorsque l'on couvre toutes les décisions, on

couvre tous les branchements et par effet secondaire, toutes les instructions. Par contre on ne couvre pas toutes les conditions et encore moins tous les chemins. Il fallait donc choisir la réponse **b**.

Q16) Combien de cas de tests sont nécessaires pour atteindre le critère « toutes les instructions » (C0) sachant que les deux conditions sont indépendantes : ***(2)***

```
                ...
      if (condition 1)
            then statement 1
            else statement 2      fi
      if (condition 2)
            then statement 3
      fi
                ...
```

a) Un cas de tests ☐

b) Deux cas de tests ■

c) Trois cas de tests ☐

d) Quatre cas de tests ☐

Q16) Dans cette portion de code, les deux conditions étant indépendantes, il suffit de choisir deux cas de tests pour couvrir toutes les instructions (et aussi, ici, toutes les conditions et toutes les décisions) ; un cas où les deux conditions sont fausses et un cas où les deux conditions sont vraies. La bonne réponse est donc la réponse **b**.

Q17) Combien de cas de tests sont nécessaires pour atteindre le critère « toutes les décisions » (C1) sachant que les deux conditions sont indépendantes :

```
                ...
      if (condition 1)
            then statement 1
            else statement 2
      fi
      if (condition 2)
            then statement 3
```

```
fi
...
```

a) Un cas de tests ☐

b) Deux cas de tests ■

c) Trois cas de tests ☐

d) Quatre cas de tests ☐

Q17) De la même façon que précédemment deux cas de test permettent d'obtenir également la couverture de toutes les décisions ; un cas où les deux conditions sont fausses et un cas où les deux conditions sont vraies ; réponse **b**.

Q18) L'effort de tests dans un projet : ***(1)***

a) Est de l'ordre de 20 % de l'effort global, voire moins pour les applications peu critiques. ☐

b) Est proportionnel au nombre de lignes de code livrées. ☐

c) Inclut également le temps passé par le client dans le cadre de cette activité. ☐

d) Doit être évalué en début de projet. ■

Q18) Mesurer l'effort de test pour un projet donné reste une activité pour laquelle peu de résultat théorique existe dans le cas général ; cet effort dépend grandement des conditions particulières liées au projet, des équipes en place, des versions déjà livrées ou en fonctionnement, etc. La réponse exacte est donc **d**.

Q19) Rechercher les défauts : ***(1)***

a) Est très coûteux en début de projet, cela mobilise beaucoup trop de monde pour faire des inspections. ☐

b) Il est plus efficace de rechercher les défauts le plus tôt possible. ■

c) Pour un défaut donné, son coût de recherche est beaucoup plus élevé en fin de projet qu'en début de projet. ☐

d) N'a finalement qu'une bonne efficacité en fin de projet. ☐

Q19) La recherche des défauts est une opération qui nécessite des ressources mais peut être menée efficacement aux différents stades du projet. Il n'est pas nécessairement plus difficile de détecter un défaut en fin qu'en début de projet. Cependant, la découverte d'un défaut entraîne des analyses, des corrections et potentiellement l'annulation d'opérations erronées déjà réalisées avec un logiciel défectueux. Ainsi, plus le défaut est découvert tôt plus les conséquences de ce défaut seront faciles à corriger. En termes d'efficacité, il est donc plus efficace au niveau du projet de rechercher les défauts dans les premiers stades de développement du logiciel ; la réponse juste est donc la réponse **b**.

Q20) Les défauts sont introduits : ***(1)***

a) Tout au long du cycle du projet. ■

b) Uniquement dès que l'on programme, les défauts ne sont essentiellement que les conséquences d'erreurs de programmation. ☐

c) Éventuellement en conséquence d'une spécification ambiguë. ☐

d) Essentiellement volontairement et de manière malveillante par les équipes projet. ☐

Q20) La réalisation de logiciel est une activité complexe faisant intervenir de nombreux acteurs qui peuvent tous involontairement commettre des erreurs de raisonnement ou de réalisation. Ces erreurs peuvent intervenir à tous les stades du projet et toucher les spécifications, le code, ou même les recettes ; si les actes malveillants sont possibles, ils sont relativement rares ne sont en aucun cas la cause essentielle de ces défauts ; la réponse correcte est donc la réponse **a**.

Q21) Que recommanderiez-vous à votre client ? ***(1)***

a) De s'impliquer dans le projet dans les phases de validation. ☐

b) De faire totalement confiance à la MOE qui garantira la qualité de l'application. ☐

c) D'être vigilant et de contrôler régulièrement les livraisons intermédiaires en fixant des jalons avec la MOE en début de projet. ■

d) D'être souple sur les délais de livraison, un retard peut améliorer la qualité du produit livré. ☐

Q21) Dans le cadre de la gestion des projets, si des délais trop courts risquent d'entraîner une augmentation des défauts, tout retard est potentiellement le signe de difficultés et donc, de défauts dans le logiciel livré. L'implication régulière de la maîtrise d'œuvre est essentielle pour garantir la qualité du logiciel et un bon moyen d'y parvenir est de suivre régulièrement l'avancée des travaux en fixant des jalons et

en vérifiant l'avancée du projet vis-à-vis de ces jalons. La bonne réponse est donc la réponse **c**.

Q22) Je suis chef de projet, mon projet est en retard, que dois-je faire ? ***(2)***

a) Livrer tel quel, je négocierai avec mon client ensuite. ☐

b) Essayer de réduire au maximum le nombre de défauts, quitte à renégocier à la baisse les fonctionnalités à livrer. ■

c) Demander au client une prolongation du projet. ☐

d) Adopter une démarche préventive, pour éviter de me trouver dans ce type de situation. ☐

Q22) En tant que chef de projet, et si l'on omet les vœux pieux, l'approche à privilégier lors d'un retard est de viser à restreindre les fonctionnalités sans baisser les objectifs en termes de qualité. Cette approche est naturelle dans les démarches itératives où l'on peut arrêter le projet à une itération lorsque les délais ou les budgets sont dépassés ; on obtient alors un logiciel moins ambitieux mais néanmoins opérationnel et fiable. La bonne réponse est donc la réponse **b**.

Q23) À quel moment dans un projet dois-je penser à l'activité de tests ? ***(1)***

a) Le plus tôt possible, les tests doivent être planifiés. ☐

b) Il est possible d'effectuer les scénarios de tests de recette dès la phase d'expression de besoins. ■

c) J'ai tout mon temps, le cycle de vie que j'utilise est en V, je commencerai l'activité de tests dès lors que mon équipe aura terminé le codage. ☐

d) Je suis tellement sous pression que nous ferons les tests que si nous avons le temps en fin de projet. ☐

Q23) Tester efficacement un logiciel ne peut se faire lorsque la pression du client ou des utilisateurs devient trop forte. Il faut avoir prévu ces tests bien en amont des phases de développement ; en particulier, les scénarios de test peuvent être prévus dès la phase d'expression de besoins à l'aide des cas d'usage utilisés lors de cette phase. La bonne réponse est donc la réponse **b**.

Q24) Quelle affirmation sur les outils de test statiques est exacte : ***(1)***

a) Les outils de test statiques sont issus de technologies proches de celles utilisées par les compilateurs. ■

b) Les outils de test statiques ne peuvent pas donner d'information sur le comportement dynamique du programme. ☐

c) Les outils de test statiques ne peuvent utiliser que lorsque tout le code est finalisé. ☐

d) Les outils de test statiques sont récents et donc encore immatures. ☐

Q24) Les tests statiques s'appuient sur les détails de réalisation et de façon privilégiée sur le code du logiciel. Les outils de tests statiques sont donc des outils anciens et matures, basés sur les mêmes technologies que les compilateurs, analyse lexicale et analyse grammaticale en particulier et peuvent donner une information précieuse sur le comportement du logiciel. La bonne réponse est donc la réponse **a**.

Q25) Lorsque l'on introduit un nouvel outil dans une organisation, il est vrai que : ***(1)***

a) Un outil « Open source » est généralement de moins bonne qualité qu'un outil propriétaire. ☐

b) Le développement de l'aide en ligne rend inutile la formation des usagers. ☐

c) L'impact sur l'organisation peut être important. ■

d) L'outil étant amené à évoluer il n'est pas nécessaire de gérer cette introduction comme un projet normal. ☐

Q25) Tout nouvel outil demande une formation de ses utilisateurs et peut avoir un impact important sur l'organisation des équipes concernées. Cette évolution des pratiques doit donc être gérée comme un projet normal ; il est par ailleurs maintenant vain d'opposer outils propriétaires et outils « libres » qui sont de qualité équivalente. La bonne réponse est donc la réponse **c**.