Guide rapide du niveau Fondation ISTQB

Chapitre 1 Principes fondamentaux des tests

1.1 Qu'est-ce que le test logiciel ?

Les tests sont le processus d'évaluation d'un système ou de ses composants dans le but de déterminer s'il satisfait ou non aux exigences spécifiées.

Les tests sont une enquête qui peut fournir aux parties prenantes des informations sur la qualité du produit ou service testé. Les tests sont une activité qui aide à identifier les défauts, les erreurs ou les exigences manquantes.

1.1.1 Objectifs typiques des tests

Il existe plusieurs objectifs typiques des tests :

- Pour identifier les défauts et les erreurs dans le produit logiciel ou le système testé.
- Pour garantir que le produit ou le système logiciel répond aux exigences spécifiées et fonctionne comme attendu.
- Pour vérifier que le produit logiciel ou le système fonctionne correctement, efficacement et de manière fiable.
- Améliorer la qualité du produit ou du système logiciel.
- Augmenter la confiance des parties prenantes dans le produit ou le système logiciel.
- Réduire les risques associés à l'utilisation du produit ou du système logiciel.

1.1.2 Test et débogage :

Les tests et le débogage sont deux activités différentes qui se complètent.

Les tests impliquent l'exécution du produit logiciel ou du système pour détecter les défauts, tandis que le débogage implique identifier et corriger les défauts constatés lors des tests.

Les tests se concentrent sur l'évaluation du produit ou du système logiciel, tandis que le débogage se concentre sur l'identification et la résolution des problèmes découverts lors des tests.

Les tests sont nécessaires pour plusieurs raisons :

- Pour garantir que le produit logiciel ou le système répond aux exigences spécifiées et fonctionne comme prévu
- Identifier et corriger les défauts avant que le produit logiciel ou le système ne soit mis à la disposition des utilisateurs.
- Améliorer la qualité du produit ou du système logiciel.
- Réduire les risques associés à l'utilisation du produit ou du système logiciel.
- Augmenter la confiance des parties prenantes dans le produit ou le système logiciel.

1.2.1 Contributions des tests au succès

Les tests contribuent au succès d'un produit ou d'un système logiciel de plusieurs manières :

- Il permet d'identifier et de corriger les défauts et les erreurs avant que le produit logiciel ou le système ne soit mis à la disposition des utilisateurs.
- Il améliore la qualité du produit ou du système logiciel, conduisant à une satisfaction accrue des utilisateurs.
- Cela réduit les risques associés à l'utilisation du produit ou du système logiciel, conduisant à une confiance accrue des utilisateurs.
- Cela permet de garantir que le produit ou le système logiciel répond aux exigences spécifiées et fonctionne comme prévu.
- Il permet d'identifier les opportunités d'amélioration dans le processus de développement logiciel.

1.2.2 Assurance qualité et tests

L'assurance qualité est le processus permettant de garantir que le produit ou le système logiciel répond aux normes de qualité spécifiées.
Les tests font partie de l'assurance qualité qui implique l'évaluation du produit logiciel ou système pour trouver les défauts et les erreurs.
L'assurance qualité comprend d'autres activités telles que l'amélioration des processus, les revues de code et la gestion des exigences.

1.2.3 Erreurs, défauts et pannes

Une erreur est une action humaine qui produit un résultat incorrect ou inattendu.
Un défaut est une faille ou une imperfection du produit logiciel ou du système qui peut entraîner son échec. pour répondre à ses exigences.
Une panne est l'incapacité du produit logiciel ou du système à remplir la fonction prévue.

- Par exemple, une erreur pourrait être un développeur codant une fonctionnalité de manière incorrecte, entraînant un défaut dans le produit logiciel. Si ce défaut n'est pas identifié et corrigé lors des tests, cela pourrait entraîner une défaillance lorsque le produit logiciel est utilisé par les utilisateurs finaux.

1.2.4 Défauts, causes profondes et effets

Un défaut a une cause profonde, qui est la raison sous-jacente pour laquelle le défaut s'est produit. L'effet est le résultat du défaut.

Identifier la cause première d'un défaut aide à empêcher que des défauts similaires ne se produisent dans l'avenir.

- Par exemple, si un produit logiciel plante lorsqu'un utilisateur saisit des données non valides, la cause première du défaut peut être une routine de validation des données inadéquate. L'effet du défaut est le crash du produit logiciel, ce qui entraîne une frustration des utilisateurs et une perte potentielle de revenus.

1.3 Sept principes de test

Les sept principes de test constituent un ensemble de concepts fondamentaux qui guident le processus de test. Ces principes comprennent :

1. Les tests montrent la présence de défauts
2. Des tests exhaustifs sont impossibles
3. Les tests précoces permettent d'économiser du temps et de l'argent
4. Le regroupement des défauts suggère qu'un petit nombre de modules contiennent la plupart des défauts
5. Le principe de Pareto s'applique aux tests logiciels, ce qui signifie que 80 % des défauts sont détectés dans 20 % des tests. les modules
6. Les tests dépendent du contexte
7. L'erreur d'absence d'erreurs ne doit pas être utilisée pour mesurer le succès des tests

1.4 Processus de test

Le processus de test est un ensemble d'activités et de tâches effectuées pour garantir que le logiciel répond aux exigences et est exempt de défauts.

Le processus de test comprend les phases suivantes :

1. Planification et contrôle
2. Analyse et conception
3. Mise en œuvre et exécution
4. Évaluation des critères de sortie et reporting
5. Tester les activités de clôture

1.4.1 Processus de test en contexte

Le processus de test doit être adapté au contexte spécifique du projet. Les facteurs contextuels qui influencent le processus de test sont les suivants :

Objectifs des tests :

1. Modèle de cycle de vie de développement logiciel 2. Risques associés au logiciel 3. Contraintes commerciales et techniques 4. Exigences réglementaires et légales

1.4.2 Activités et tâches de test

Les activités de test sont un ensemble d'actions et de tâches effectuées pendant le processus de test. Les activités de test peuvent être divisées dans les catégories suivantes :

1. Planification et contrôle 2. Analyse et conception 3. Mise en œuvre et exécution 4. Évaluation des critères de sortie et reporting 5. Test des activités de clôture

- : Les tâches de test sont des activités spécifiques qui sont effectuées au cours de chaque activité de test.

Les tâches de test peuvent inclure :

1. Examiner la base de test 2. Identifier les conditions de test 3. Créer des cas de test 4. Exécuter des cas de test 5. Signaler les défauts

1.4.3 Produits des travaux de test

Les produits de travail de test sont les livrables créés au cours du processus de test. Les produits de test peuvent inclure : 1. Plan de test 2. Spécification de conception de test 3. Spécification de scénario de test 4. Spécification de procédure de test 5. Journal de test 6. Rapport de synthèse de test

1.4.4 Traçabilité entre la base de test et les produits des travaux de test

La traçabilité est la capacité de suivre une exigence ou un autre produit de travail tout au long du développement processus jusqu'à sa mise en œuvre finale.

La traçabilité entre la base de test et les produits de test garantit que toutes les exigences sont couvertes par cas de test et que tous les cas de test peuvent être rattachés aux exigences.

Cette traçabilité permet de garantir que toutes les exigences sont testées et que les tests sont complets.

1.5 La psychologie des tests

La psychologie des tests est l'étude des facteurs cognitifs et émotionnels qui influencent les tests. processus.

Comprendre la psychologie des tests peut aider les testeurs à améliorer l'efficacité de leurs tests.

Les facteurs qui peuvent influencer la psychologie des tests comprennent :

1. Attention et concentration
2. Perception et interprétation
3. Mémoire et rappel
4. Motivation et moral
5. Biais et heuristiques

1.5.1 Psychologie humaine et tests

La psychologie humaine et les tests sont étroitement liés car les tests sont une activité humaine qui nécessite des compétences cognitives et émotionnelles.

Les testeurs doivent comprendre les facteurs humains qui peuvent influencer les tests afin d'améliorer l'efficacité de leurs tests.

1.5.2 Etat d'esprit des testeurs et des développeurs

Les testeurs et les développeurs ont des mentalités différentes qui peuvent influencer leur approche des tests.

Les testeurs ont tendance à se concentrer sur la recherche de défauts, tandis que les développeurs ont tendance à se concentrer sur la mise en œuvre des fonctionnalités.

Les testeurs doivent comprendre l'état d'esprit du développeur afin de communiquer efficacement les défauts et assurez-vous qu'ils sont fixés.

Les développeurs doivent comprendre l'état d'esprit du testeur afin de prioriser et de corriger efficacement les défauts.

Chapitre 2 Tests tout au long du cycle de vie du développement logiciel

2.1 Modèles de cycle de vie de développement logiciel :

Les modèles de cycle de vie du développement logiciel décrivent les phases du processus de développement logiciel, et chaque phase peut nécessiter différents types de tests.

Des exemples de modèles de cycle de vie de développement logiciel incluent le modèle Waterfall, le modèle V, le modèle Agile et Modèle en spirale.

Chacun de ces modèles a ses propres forces et faiblesses, et les organisations peuvent choisir un modèle en fonction de leurs besoins et circonstances spécifiques.

2.1.1 Développement de logiciels et tests de logiciels :

Le développement de logiciels est le processus de création de logiciels à partir de zéro. Les tests logiciels sont le processus de vérification et de validation que le logiciel fonctionne correctement et répond aux attentes de l'utilisateur. exigences et est exempt de défauts.

2.1.2 Modèles de cycle de vie de développement logiciel en contexte :

Les modèles de cycle de vie de développement logiciel fournissent un contexte pour les tests.

Des tests doivent être effectués tout au long du processus de développement pour garantir que le logiciel est livré conformément aux exigences des utilisateurs et qu'il est de haute qualité.

Les activités de test doivent être alignées sur le modèle de cycle de vie de développement logiciel.

2.2 Niveaux de tests :

Les niveaux de test sont les étapes auxquelles les tests sont effectués.

Chaque niveau a des objectifs et des types de tests spécifiques.

Les quatre niveaux de test sont :

1. Tests de composants,
2. Tests d'intégration,
3. Tests du système,
4. Tests d'acceptation.

2.2.1 Test des composants :

Les tests de composants sont également connus sous le nom de tests unitaires ou de tests de modules.

Il s'agit du test de composants logiciels individuels, tels que des fonctions ou des procédures, indépendamment du reste du système.

2.2.2 Tests d'intégration :

Les tests d'intégration sont le processus de combinaison des composants individuels et de test des interactions entre eux.

Il garantit que les composants logiciels fonctionnent ensemble comme prévu et que tout défaut ou problème résultant de leurs interactions sont identifiés et résolus.

2.2.3 Tests du système :

Les tests système sont le processus de test de l'ensemble du système dans son ensemble. Il vérifie que le système répond aux exigences spécifiées et fonctionne comme prévu.

Les tests système peuvent inclure des tests fonctionnels, des tests non fonctionnels et d'autres types de tests.

2.2.4 Tests d'acceptation :

Les tests d'acceptation sont le processus permettant de vérifier que le logiciel répond aux exigences de l'utilisateur et qu'il est acceptable pour la livraison.

Il s'agit généralement de l'étape finale des tests et peut inclure des types de tests fonctionnels et non fonctionnels.

2.3 Types d'essais :

Les types de tests sont les techniques utilisées pour vérifier et valider le logiciel.

Différents types de tests sont requis à différents niveaux de test pour garantir un test complet du logiciel.

Voici des exemples de types de tests :

1. Tests fonctionnels,
2. Tests non fonctionnels,
3. Tests en boîte blanche,
4. Tests liés au changement et autres.

2.3.1 Tests fonctionnels :

Les tests fonctionnels sont le processus permettant de vérifier que le logiciel répond aux exigences fonctionnelles.

Elle est généralement effectuée aux niveaux des tests système et d'acceptation.

2.3.2 Tests non fonctionnels :

Les tests non fonctionnels sont le processus de vérification que le logiciel répond aux exigences non fonctionnelles, telles que :

- performances,
- la sécurité,
- convivialité

- : Il est généralement effectué aux niveaux des tests système et d'acceptation.

2.3.3 Tests en boîte blanche :

Les tests en boîte blanche, également appelés tests structurels, sont une approche de test dans laquelle le testeur a accès au fonctionnement interne du système testé, y compris son code, ses structures de données et ses algorithmes.

L'objectif des tests en boîte blanche est de garantir que tous les chemins de code sont testés et que tous les résultats et résultats possibles sont vérifiés.

- : Ce type de tests est généralement effectué par des développeurs ou des testeurs ayant une formation en programmation.

Voici des exemples de techniques de test en boîte blanche :

- couverture du relevé,
- couverture des succursales,
- couverture du chemin.

2.3.4 Tests liés aux changements :

Des tests liés aux modifications sont effectués lorsque des modifications sont apportées au logiciel.

L'objectif des tests liés aux modifications est de garantir que les modifications n'ont introduit aucun nouveau défaut ou problème et que les fonctionnalités existantes n'ont pas été affectées.

Les tests liés aux changements peuvent être effectués à différents niveaux, tels que :

- composant,
- intégration,
- tests du système

peut utiliser différents types de tests, tels que :

- tests de régression,
- tests fonctionnels,
- tests non fonctionnels.

2.3.5 Types de tests et niveaux de tests :

Les types de tests et les niveaux de tests sont liés les uns aux autres, car différents types de tests sont généralement effectués à différents niveaux de test.

Par exemple, les tests fonctionnels sont généralement effectués au niveau du système et des tests d'acceptation, tandis que les tests non fonctionnels, tels que les tests de performances et les tests de sécurité, sont généralement effectués au niveau du système et des tests d'intégration.

Les tests en boîte blanche sont généralement effectués au niveau des tests de composants et d'intégration.

Il est important de choisir les types et niveaux de tests appropriés en fonction des objectifs, des exigences et des risques du projet.

2.4 Tests d'entretien :

Les tests de maintenance sont un type de test effectué après la publication du logiciel et son utilisation par les utilisateurs finaux.

Le but des tests de maintenance est d'identifier et de corriger les défauts et les problèmes qui surviennent pendant la phase de maintenance du logiciel.

Les tests de maintenance peuvent être déclenchés par différents facteurs, tels que les commentaires des utilisateurs, les modifications de l'environnement logiciel et l'introduction de nouveaux composants matériels ou logiciels.

Les tests de maintenance peuvent utiliser différentes techniques, telles que les tests de régression, les tests correctifs et les tests préventifs.

2.4.1 Déclencheurs de maintenance :

Les tests de maintenance peuvent être déclenchés par divers facteurs, tels que :

- retours des utilisateurs,
- les changements dans l'environnement logiciel,
- l'introduction de nouveaux composants matériels ou logiciels.

- : Les commentaires des utilisateurs peuvent inclure des rapports de bogues, des demandes de fonctionnalités et des problèmes d'utilisation.

- : les modifications apportées à l'environnement logiciel peuvent inclure des modifications dans le système d'exploitation, le navigateur Web ou le système de gestion de base de données.

- : L'introduction de nouveaux composants matériels ou logiciels peut inclure l'ajout de nouveaux périphériques ou l'intégration de bibliothèques tierces.

2.4.2 Analyse d'impact pour l'entretien :

- : L'analyse d'impact pour la maintenance est le processus d'évaluation de l'impact d'un changement sur le logiciel et ses composants.

- : L'objectif de l'analyse d'impact est de déterminer l'étendue du changement et d'identifier les domaines du logiciel qui peuvent être affectés.

L'analyse d'impact peut utiliser différentes techniques, telles que :

- analyse statique,
- analyse dynamique,
- analyse de traçabilité.

L'analyse statique consiste à examiner le code source et d'autres artefacts logiciels sans exécuter le logiciel.

L'analyse dynamique consiste à exécuter le logiciel avec des cas de test et à analyser son comportement.

L'analyse de traçabilité implique le traçage des exigences, de la conception et des artefacts de test pour identifier l'impact d'un changement.

Chapitre 3 Tests statiques

Les tests statiques sont une technique qui examine un produit logiciel ou une documentation associée sans exécuter le code.

Cette technique peut être appliquée tout au long du cycle de vie du développement logiciel et présente plusieurs avantages, notamment la détection précoce des défauts et l'amélioration de la qualité des logiciels.

3.1 Bases des tests statiques :

- : Les tests statiques sont une technique de test qui vérifie les défauts ou les problèmes d'un produit logiciel sans exécuter le logiciel.

- : Il se concentre sur les attributs statiques du produit tels que le code, les exigences, les documents de conception et les spécifications.

- : Les tests statiques impliquent l'examen et l'analyse de documents, de codes ou d'autres produits de travail de manière structurée. manière.

3.1.1 Produits de travail pouvant être examinés par des tests statiques :

Il existe plusieurs produits de travail qui peuvent être examinés à l'aide de tests statiques, tels que :

- les documents d'exigences,
- documents de conception,
- plans de tests,
- cas de tests,
- coder ,
- manuels d'utilisation et autres documents connexes.

3.1.2 Avantages des tests statiques :

Les tests statiques présentent plusieurs avantages, notamment la détection précoce des défauts, l'amélioration de la qualité des logiciels, la rentabilité et une efficacité accrue.

Cela aide également à identifier les défauts qui peuvent être difficiles à trouver lors des tests dynamiques, tels que l'analyse des valeurs limites et le partitionnement d'équivalence.

3.1.3 Différences entre les tests statiques et dynamiques :

Les tests statiques diffèrent des tests dynamiques en termes d'approche et d'objectifs.

Les tests dynamiques se concentrent sur l'exécution du code et l'identification des défauts pendant l'exécution, tandis que les tests statiques se concentrent sur la vérification des attributs statiques d'un produit logiciel sans exécuter le code.

3.2 Processus d'examen :

Le processus d'examen implique un groupe d'individus examinant un produit logiciel ou une documentation pour identifier les défauts et améliorer la qualité.

Ce processus peut être formel ou informel et avoir lieu à différentes étapes du cycle de vie du développement logiciel.

3.2.1 Processus d'examen des produits de travail :

Le processus d'examen des produits de travail comprend les étapes suivantes :

1. Planification de l'examen
2. Réunion de lancement
3. Examen du produit du travail 4.
   Identification et documentation des problèmes 5.
   Retravail et suivi

3.2.2 Rôles et responsabilités dans le cadre d'un examen formel :

Différents rôles sont impliqués dans un processus d'examen formel, tels que :

- le modérateur, •
  l'auteur, • le
  réviseur, • le
  scribe.

Chaque rôle a des responsabilités spécifiques, comme le modérateur qui veille à ce que le processus de révision soit mené de manière structurée et l'auteur qui présente le produit du travail en cours de révision.

3.2.3 Types d'examen :

Il existe différents types d'avis, notamment :

- revues informelles, •
  revues formelles, •
  revues techniques, •
  revues pas à pas.

Chaque type d'examen a ses objectifs et est mené à différentes étapes du cycle de vie du développement logiciel.

3.2.4 Application des techniques d'examen :

Les techniques de révision comprennent :

- procédures pas à pas,
- contrôles,
- évaluations par les pairs.

Ces techniques peuvent être appliquées à différents types de produits de travail et avoir différents objectifs, tels que l'identification de défauts ou l'amélioration de la qualité des logiciels.

3.2.5 Facteurs de réussite des examens :

Les facteurs de réussite des examens incluent des objectifs clairs, un processus d'examen structuré, une préparation adéquate et la garantie que les bonnes personnes sont impliquées.

D'autres facteurs de réussite incluent une attitude positive, des commentaires constructifs et une volonté d'apprendre et de s'améliorer.

Chapitre 4 Techniques de test

4.1 Catégories de techniques de test :

Les techniques de test peuvent être divisées en trois catégories :

1. techniques de boîte noire,
2. techniques de boîte blanche,
3. techniques basées sur l'expérience.

Les testeurs choisissent les techniques de test appropriées en fonction du logiciel testé, des exigences et d'autres facteurs tels que le temps et les ressources.

4.1.1 Choix des techniques de test :

La décision quant à la technique de test à utiliser dépend de plusieurs facteurs, notamment

1. la complexité du système, 2.
les risques associés au système,
3. disponibilité de la documentation, 4.
objectifs des tests, 5. niveau
de compétence de l'équipe de tests.

Les testeurs peuvent utiliser une combinaison de différentes techniques pour obtenir une meilleure couverture de test.

4.1.2 Catégories de techniques de test et leurs caractéristiques :

Les techniques de boîte noire se concentrent sur le test du système sans connaissance de son fonctionnement interne, tandis que les techniques de boîte blanche impliquent de tester le système avec connaissance de son fonctionnement interne.

Les techniques basées sur l'expérience impliquent des tests basés sur l'expérience, l'intuition et la créativité du testeur.

4.2 Techniques de test en boîte noire :

Les techniques de test en boîte noire sont conçues pour tester le système sans connaître son fonctionnement interne.

Ces techniques sont utiles pour tester la fonctionnalité du système et garantir qu'il répond aux exigences.

Des exemples de techniques de test en boîte noire incluent le partitionnement d'équivalence, l'analyse des valeurs limites, les tests de table de décision, les tests de transition d'état et les tests de cas d'utilisation.

4.2.1 Partitionnement d'équivalence :

Le partitionnement d'équivalence est une technique de test en boîte noire qui consiste à diviser le domaine d'entrée d'un système en groupes d'entrées équivalentes.

L'idée est de sélectionner un cas de test représentatif dans chaque groupe, qui devrait fournir des résultats de test similaires.

Par exemple, si un système accepte des entrées comprises entre 1 et 100, le domaine d'entrée peut être divisé en trois classes d'équivalence : les entrées inférieures à 1, les entrées comprises entre 1 et 100 et les entrées supérieures à 100.

4.2.2 Analyse de la valeur limite :

L'analyse des valeurs limites est une technique de test en boîte noire qui consiste à tester les limites d'un domaine d'entrée.

L'idée est de tester les valeurs aux limites du domaine d'entrée, car celles-ci sont susceptibles d'être plus sujettes à les erreurs.

Par exemple, si un système accepte des entrées comprises entre 1 et 100, les valeurs limites à tester seraient alors 1, 100 et les valeurs juste en dessous ou au-dessus de ces limites.

4.2.3 Test de la table de décision :

Le test de table de décision est une technique de test en boîte noire qui consiste à créer un tableau pour représenter les combinaisons d'entrées et de sorties attendues pour un ensemble donné de règles métier.

Le testeur peut ensuite utiliser la table de décision pour dériver des cas de test qui couvrent toutes les combinaisons possibles d'entrées et de sorties attendues.

4.2.4 Tests de transition d'état :

Les tests de transition d'état sont une technique de test en boîte noire qui consiste à tester le comportement d'un système lorsqu'il passe d'un état à un autre.

Le testeur identifie les différents états du système et les événements qui le font passer d'un état à un autre.

Le testeur crée ensuite des cas de tests pour s'assurer que le système se comporte correctement lors de ces transitions.

4.2.5 Tests de cas d'utilisation :

Les tests de cas d'utilisation sont une technique de test en boîte noire qui consiste à tester le comportement du système du point de vue de l'utilisateur.

Le testeur identifie les différents cas d'utilisation du système et crée des cas de test pour garantir que le système se comporte correctement pour chaque cas d'utilisation.

4.3 Techniques de test en boîte blanche :

Les techniques de test en boîte blanche, également connues sous le nom de techniques structurelles ou basées sur le code, se concentrent sur le fonctionnement interne du logiciel testé.

Ces techniques sont principalement utilisées pour tester les logiciels aux niveaux de l'unité, de l'intégration et du système, et reposent sur la connaissance du code interne et de la structure du logiciel.

Certaines des techniques de boîte blanche couramment utilisées sont la couverture des déclarations, la couverture des décisions et la couverture des conditions.

4.3.1 Tests et couverture des déclarations :

La couverture des instructions est une technique de boîte blanche qui garantit que chaque instruction du code est exécutée au moins une fois pendant le test.

Cela implique de créer des cas de test qui exécutent chaque instruction du code, en garantissant que toutes les instructions sont couvertes.

La couverture des instructions est une technique utile pour détecter les problèmes liés au flux de contrôle dans le code.

4.3.2 Tests de décision et couverture :

La couverture des décisions est une technique de boîte blanche qui garantit que toutes les décisions possibles dans le code ont été testées.

Une décision est un point du code où le programme a le choix entre deux ou plusieurs chemins possibles.

La couverture des décisions implique la création de cas de test qui garantissent que toutes les décisions du code ont été appliquées, qu'elles soient vraies ou fausses.

4.3.3 La valeur des tests de déclarations et de décisions :

Les tests d'instructions et de décisions sont des techniques importantes pour vérifier l'exactitude du code.

Ces techniques peuvent aider à identifier les problèmes liés au flux de contrôle dans le code et à garantir que tous les chemins possibles dans le code ont été testés.

4.4 Techniques de test basées sur l'expérience :

Les techniques de test basées sur l'expérience s'appuient sur les connaissances, les compétences et l'intuition des testeurs pour identifier les problèmes potentiels avec le logiciel testé.

Ces techniques sont généralement utilisées dans des situations où les exigences ne sont pas claires ou incomplètes, ou lorsque le logiciel est complexe ou difficile à tester.

4.4.1 Deviner les erreurs :

La recherche d'erreurs est une technique de test basée sur l'expérience qui consiste à utiliser les connaissances et l'expérience du testeur pour deviner quels types d'erreurs pourraient survenir dans le logiciel testé.

Le testeur crée ensuite des scénarios de test qui ciblent spécifiquement ces types d'erreurs. Cette technique peut être particulièrement efficace pour identifier les problèmes qui peuvent ne pas ressortir des exigences ou de la documentation de conception.

4.4.2 Tests exploratoires :

Les tests exploratoires sont une technique de test basée sur l'expérience qui implique des tests ad hoc du logiciel testé, basés sur l'intuition et les connaissances du testeur.

Cette technique est particulièrement utile dans les situations où le logiciel est complexe ou difficile à tester, ou lorsque les exigences sont floues ou incomplètes.

4.4.3 Tests basés sur une liste de contrôle :

Les tests basés sur une liste de contrôle sont une technique de test basée sur l'expérience qui implique l'utilisation d'une liste de contrôle prédéfinie d'éléments à tester.

La liste de contrôle peut inclure des éléments tels que des erreurs courantes, des problèmes de performances ou des problèmes d'utilisabilité.

Le testeur crée ensuite des cas de test qui ciblent spécifiquement les éléments de la liste de contrôle.

Cette technique peut être particulièrement utile pour garantir que des zones importantes du logiciel ne sont pas négligées lors des tests.

Chapitre 5 Gestion des tests

5.1 Organisation des tests :

Les tests indépendants font référence aux tests effectués par un groupe distinct de l'équipe de développement.

Cette séparation peut contribuer à accroître l'objectivité, à améliorer la qualité et à réduire les risques de conflits d'intérêts.

Les testeurs indépendants peuvent faire partie d'une équipe ou d'une organisation distincte, ou ils peuvent travailler au sein de la même organisation mais relever d'un responsable différent.

Les tâches d'un gestionnaire de tests peuvent inclure la définition de la stratégie de test, la planification et le contrôle de l'effort de test, l'allocation des ressources et la communication avec les parties prenantes.

Les tâches d'un testeur peuvent inclure la conception de scénarios de test, l'exécution de tests, le signalement des défauts et l'analyse des résultats des tests.

5.2 Planification et estimation des tests :

Un plan de test est un document qui décrit les objectifs, la portée, l'approche et le calendrier de l'effort de test.

Il doit inclure des détails sur la stratégie de test, l'approche de test, les critères d'entrée, les critères de sortie et le calendrier d'exécution des tests.

Le plan de test doit être examiné et approuvé par toutes les parties prenantes concernées.

L'estimation des tests implique de prédire la quantité de temps, d'efforts et de ressources nécessaires pour mener à bien l'effort de test.

Cela peut s'avérer difficile en raison de l'incertitude liée au développement de logiciels et de la complexité des tests.

Les techniques d'estimation des tests comprennent

1. jugement d'expert, 2.
données historiques,
3. approches basées sur des métriques.

5.3 Surveillance et contrôle des tests :

La surveillance des tests implique la collecte et l'analyse de données sur l'effort de test pour déterminer son statut et sa progression.

Le contrôle des tests implique de prendre des mesures correctives pour résoudre les problèmes survenus lors des tests.

Les métriques de test peuvent être utilisées pour surveiller et contrôler l'effort de test, telles que les métriques de défauts, les métriques de couverture de test et les métriques de progression des tests.

Les rapports de test fournissent des informations sur les efforts de test aux parties prenantes, y compris les résultats des tests, les défauts détectés et les progrès réalisés.

Le contenu d'un rapport de test peut varier en fonction du public cible, mais doit être précis, pertinent et opportun.

5.4 Gestion des configurations :

La gestion de la configuration est le processus d'identification, d'organisation et de contrôle des modifications apportées au logiciel testé, y compris sa documentation, ses exigences, sa conception, son code et ses artefacts de test.

Les testeurs doivent s'assurer que les modifications apportées au logiciel n'affectent pas l'effort de test et que toute modification apportée au cours du processus de test est contrôlée et traçable.

Les outils et processus de gestion de configuration sont utilisés pour gérer et maintenir les différentes versions et artefacts du logiciel, et pour garantir que seules les personnes autorisées y apportent des modifications.

5.5 Risques et tests :

Dans le contexte des tests, le risque fait référence à la probabilité qu'une panne se produise et à l'impact de cette panne sur le logiciel, ses utilisateurs et l'organisation dans son ensemble.

Les tests basés sur les risques impliquent d'identifier les risques qui pourraient affecter le logiciel et d'utiliser ces informations pour déterminer les types de tests à effectuer.

Les testeurs doivent comprendre les différents types de risques et leur impact sur les tests, et ils doivent travailler avec l'équipe de projet pour développer un plan de gestion des risques.

Ce plan devrait comprendre :

1. identification des risques,
2. analyse, 3.
stratégies d'atténuation.

5.6 Gestion des défauts :

La gestion des défauts implique le processus d'identification, de documentation, de suivi et de résolution des défauts détectés au cours du processus de test.

Les testeurs doivent utiliser un outil de gestion des défauts pour enregistrer les défauts, suivre leur état et communiquer avec l'équipe de développement au sujet de leur résolution.

Les défauts doivent être classés en fonction de leur gravité et des priorités doivent être attribuées en fonction de leur impact sur le logiciel.

Les testeurs doivent également effectuer une analyse des causes profondes pour déterminer la cause sous-jacente du défaut et éviter que des défauts similaires ne se reproduisent à l'avenir.

Les mesures de défauts peuvent être utilisées pour suivre et rendre compte de l'efficacité du processus de gestion des défauts.

Chapitre 6 Prise en charge des outils pour les tests

6.1.1 Classification des outils de test :

Les outils de test peuvent être globalement classés en quatre catégories :

1. Outils de gestion des tests : ces outils aident à planifier, organiser et gérer les tests activités.
2. Outils de tests statiques : ces outils aident à exécuter des techniques de tests statiques telles que des critiques et contrôles.
3. Outils de conception de tests : ces outils aident à concevoir des cas de test et des scénarios de test.
4. Outils d'exécution de tests : ces outils aident à exécuter les cas de test, à capturer les résultats et générer des rapports de tests.

6.1.2 Avantages et risques de l'automatisation des tests :

Les avantages de l'automatisation des tests incluent :

1. Couverture de tests accrue : les tests automatisés peuvent couvrir plus de scénarios que les tests manuels.
2. Précision des tests améliorée : les tests automatisés sont moins sujets aux erreurs que les tests manuels.
3. Réutilisabilité : les tests automatisés peuvent être réutilisés plusieurs fois.
4. Réduction des coûts : les tests automatisés peuvent réduire les coûts à long terme.

Les risques de l'automatisation des tests incluent :

1. Coût initial élevé : les outils d'automatisation peuvent être coûteux et la configuration de l'environnement d'automatisation peut être prend du temps.
2. Frais généraux de maintenance : les tests automatisés nécessitent une maintenance et toute modification apportée à l'application peut casser les tests automatisés.
3. Portée limitée : l'automatisation ne peut pas remplacer complètement les tests manuels, car certains types de tests, comme les tests exploratoires, nécessitent l'intuition humaine.

6.2.1 Principes principaux de sélection des outils :

Les grands principes de sélection d'un outil sont :

1. Identifiez le problème à résoudre : l'outil doit être sélectionné en fonction du problème spécifique qu'il résout.
2. Évaluez l'outil par rapport aux exigences : l'outil doit répondre aux exigences fonctionnelles et non fonctionnelles. exigences.
3. Assurer la compatibilité avec l'environnement existant : l'outil doit être compatible avec les outils et technologies existants.
4. Évaluer le coût et les avantages : l'outil doit offrir des avantages significatifs par rapport au coût encouru.

6.2.2 Projets pilotes pour l'introduction d'un outil dans une organisation :

Les projets pilotes sont utiles pour introduire un nouvel outil dans une organisation. Le projet pilote implique la sélection d'un petit projet ou d'un sous-ensemble du projet plus vaste et l'utilisation de l'outil pour Essaye-le.

Le projet pilote offre la possibilité d'évaluer l'efficacité de l'outil, d'identifier les problèmes éventuels et apporter les modifications nécessaires.

6.2.3 Facteurs de réussite des outils :

Le succès d'un outil dépend de plusieurs facteurs :

• Sélection d'outils : sélectionner le bon outil pour le bon problème. • Personnalisation de l'outil : personnalisation de l'outil pour répondre aux besoins spécifiques de l'organisation. • Formation : fournir une formation adéquate aux utilisateurs pour utiliser efficacement l'outil. • Support de l'outil : garantir que l'outil est supporté et entretenu correctement. • Intégration de l'outil : intégrer l'outil avec d'autres outils et technologies utilisés dans l'organisation.

Question #1 (1 Point)

Which one of the following is the BEST description of a test condition?

a) An attribute of a component or system specified or implied by requirements documentation.
b) An aspect of the test basis that is relevant to achieve specific test objectives.
c) The capability of the software product to provide functions which meet stated and implied needs when the software is used under specified conditions.
d) The percentage of all single condition outcomes that independently affect a decision outcome that have been exercised by a test case suite.

Select one option.

FL-1.x (K1) Keywords Chapter 1

Justification

a) Not correct – Definition of feature according to Glossary.
b) **Correct** – From Glossary.
c) Not correct – Definition of functionality according to Glossary.
d) Not correct – Definition of modified condition decision coverage according to Glossary.

Question #2 (1 Point)

Which of the following statements is a valid objective for testing?

a) To determine whether enough component tests were executed within system testing.
b) To find as many failures as possible so that defects can be identified and corrected.
c) To prove that all possible defects are identified.
d) To prove that any remaining defects will not cause any failures.

Select one option.

FL-1.1.1 (K1) Identify typical objectives of testing

Justification

a) Not correct – Component testing is not part of System testing.
b) **Correct** – Syllabus 1.1.1
c) Not correct – Principle #1 states that exhaustive testing is impossible, so one can never prove that all defects were identified.
d) Not correct – To make an assessment whether a defect will cause a failure or not, one has to detect the defect first. Saying that no remaining defect will cause a failure, implicitly means that all defects were found. This contradicts Principle #1.

Question #3 (1 Point)

Which of the following statements correctly describes the difference between testing and debugging?

a) Testing identifies the source of defects; debugging analyzes the defects and proposes prevention activities.
b) Testing shows failures caused by defects; debugging finds, analyzes, and removes the causes of failures in the software.
c) Testing removes faults; debugging identifies the causes of failures.
d) Testing prevents the causes of failures; debugging removes the failures.

Select one option.

FL-1.1.2 (K2) Differentiate testing from debugging

Justification

a) Not correct. Testing does not identify the source of defects.
b) **Correct**. Syllabus 1.1.2: Executing tests can show failures that are caused by defects in the 8 software. Debugging is the development activity that finds, analyzes, and fixes such defects.
c) Not correct. Testing does not remove faults.
d) Not correct. Testing does not directly prevent the causes of failures. Debugging does not removes the failures, only the causes of failures

Question #4 (1 Point)

Which one of the statements below describes a failure discovered during testing or in production?

a) The product crashed when the user selected an option in a dialog box.
b) The wrong version of one source code file was included in the build.
c) The computation algorithm used the wrong input variables.
d) The developer misinterpreted the requirement for the algorithm.

Select one option.

FL-1.2.3 (K2) Distinguish between error, defect and failure

Justification

a) **Correct** – A failure is an external manifestation of a defect. A crash is clearly noticeable by the user.
b) Not correct – This is a defect, not a failure, since there is something wrong in the code. It may not result in a failure, for example if the changes in the source code file are only in comments.
c) Not correct – This is a defect, not a failure, as there is a flaw in the code implementing the algorithm. If this computation is not used in a test or in production, a failure will not occur.
d) Not correct – This is an error, not a failure. The misinterpretation of the requirement may or may not lead to a defect in the impementation of the algorithm, which in turn may or may not lead to a failure.

Question #5 (1 Point)

Which of the following statements CORRECTLY describes one of the seven key principles of software testing?

a) By using automated testing it is possible to test everything.
b) With sufficient effort and tool support, exhaustive testing is feasible for all software.
c) It is impossible to test all input and precondition combinations in a system.
d) The purpose of testing is to prove the absence of defects.

Select one option.

FL-1.3.1 (K2) Explain the seven testing principles

Justification

a) Not Correct – Exhaustive testing is impossible, regardless of it being manual or automated.
b) Not Correct– Exhaustive testing is impossible, regardless of the amount of effort put into testing.
c) **Correct** – Syllabus 1.3: Principle #2 says “Testing everything (all combinations of inputs and preconditions) is not feasible except for trivial cases”.
d) Not Correct– This statement is contradicting Principle #1 says “Testing shows the presence of defects: Testing can show that defects are present, but cannot prove that there are no defects”.

Question #6 (1 Point)

In what way can testing be part of Quality assurance?

a) It ensures that requirements are detailed enough.
b) It reduces the level of risk to the quality of the system.
c) It ensures that standards in the organization are followed.
d) It measures the quality of software in terms of number of executed test cases.

Select one option.

FL-1.2.2 (K2) Describe the relationship between testing and quality assurance and give examples of how testing contributes to higher quality

Justification

a) Not correct – This is Quality assurance but not testing.
b) **Correct** – Syllabus 1.2.2. Testing contributes to the achievement of quality in a variety of ways.
c) Not correct – This is Quality assurance but not testing.
d) Not correct – The quality can not be measured by counting the number of executed test cases without knowing the outcome.

Question #7 (1 Point)

Which of the below tasks is performed during the test analysis activity of the test process?

a) Identifying any required infrastructure and tools.
b) Creating test suites from test scripts.
c) Analyzing lessons learned for process improvement.
d) Evaluating the test basis for testability.

Select one option.

FL-1.4.2 (K2) Describe the test activities and respective tasks within the test process

Justification

a) Not correct – this activity is performed during the Test design activity.
b) Not correct – this activity is performed during the Test implementation activity.
c) Not correct – this activity is performed during the Test completion activity.
d) **Correct** – this activity is performed during the Test analysis activity. Syllabus 1.4.2.

Question #8 (1 Point)

Differentiate the following test work products,1-4, by mapping them to the right description, A-D.

1. Test suite.
2. Test case.
3. Test script.
4. Test charter.

A. A group of test scripts or test execution schedule.
B. A set of instructions for the automated execution of test procedures.
C. Contains expected results.
D. An event that could be verified.

a) 1A, 2C, 3B, 4D.
b) 1D, 2B, 3A, 4C.
c) 1A, 2C, 3D, 4B.
d) 1D, 2C, 3B, 4A.

Select one option.

FL-1.4.3 (K2) Differentiate the work products that support the test process

Justification

Test suite: Syllabus 1.4.3 for Test implementation:
Test implementation work products also include test suites, which are groups of test scripts, as well as a test execution schedule. (1A).
Test case: Glossary, A set of input values, execution preconditions, **expected results** and execution postconditions…. (2C).
Test script: Glossary test script, A set of instructions for the automated execution of test procedures (3B).
Test charter: Glossary, A statement of test objectives, and possibly test ideas about how to test. Test charters are used in exploratory testing. (4D).

Thus:
a) **Correct**
b) Not correct
c) Not correct
d) Not correct

Question #9 (1 Point)

How can white-box testing be applied during acceptance testing?

a) To check if large volumes of data can be transferred between integrated systems.
b) To check if all code statements and code decision paths have been executed.
c) To check if all work process flows have been covered.
d) To cover all web page navigations.

Select one option.

FL-2.3.2 (K1) Recognize that functional, non-functional and white-box tests occur at any test level

Justification

a) Not correct – Relevant for integration testing.
b) Not correct – Relevant for component testing.
c) **Correct** – Syllabus 2.3.5: For acceptance testing, tests are designed to cover all supported financial data file structures and value ranges for bank-to-bank transfers.
d) Not correct – Relevant for system testing.

Question #10 (1 Point)

Which of the following statements comparing component testing and system testing is TRUE?

a) Component testing verifies the functionality of software modules, program objects, and classes that are separately testable, whereas system testing verifies interfaces between components and interactions between different parts of the system.
b) Test cases for component testing are usually derived from component specifications, design specifications, or data models, whereas test cases for system testing are usually derived from requirement specifications, or use cases.
c) Component testing only focuses on functional characteristics, whereas system testing focuses on functional and non-functional characteristics.
d) Component testing is the responsibility of the testers, whereas system testing typically is the responsibility of the users of the system.

Select one option.

FL-2.2.1 (K2) Compare the different test levels from the perspective of objectives, test basis, test objects, typical defects and failures, and approaches and responsibilities

Justification

a) Not correct – System testing does not test interfaces between components and interactions between different parts of the system; this is a target of integration tests.
b) **Correct** – Syllabus 2.2.1: Examples of work products that can be used as a test basis for component testing include: detailed design, code, data model, component specifications.
Syllabus 2.2.3: Examples of work products for system testing include: System and software requirement specifications (functional and non-functional), …,.. use cases.
c) Not correct – Component testing does not ONLY focus on functional characteristics.
d) Not correct – Component testing typically is the responsibility of the developers, whereas system testing typically is the responsibility of testers.

Question #11 (1 Point)

Which one of the following is TRUE?

a) The purpose of regression testing is to check if the correction has been successfully implemented, while the purpose of confirmation testing is to confirm that the correction has no side effects.
b) The purpose of regression testing is to detect unintended side effects, while the purpose of confirmation testing is to check if the system is still working in a new environment.
c) The purpose of regression testing is to detect unintended side effects, while the purpose of confirmation testing is to check if the original defect has been fixed.
d) The purpose of regression testing is to check if the new functionality is working, while the purpose of confirmation testing is to check if the originally defect has been fixed.

Select one option.

FL-2.3.3 (K2) Compare the purposes of confirmation testing and regression testing

Justification

a) Not correct – Confirmation testing does not check successful implementation and confirmation testing does not check for side effects.
b) Not correct– The statement about confirmation testing should be about regression testing.
c) **Correct** – Syllabus 2.3.4
d) Not correct – Testing new functionality is not regression testing

Question #12 (1 Point)

Which one of the following is the BEST definition of an incremental development model?

a) Defining requirements, designing software and testing are done in a series with added pieces.
b) A phase in the development process should begin when the previous phase is complete.
c) Testing is viewed as a separate phase which takes place after development has been completed.
d) Testing is added to development as an increment.

Select one option.

FL-2.1.1 (K2) Explain the relationship between software development activities and test activities in the software life cycle

Justification

a) **Correct**– Syllabus 2.1.1: Incremental development involves establishing requirements, designing, building, and testing a system in pieces.
b) Not correct – This is a sequential model.
c) Not correct – This describes the Waterfall model.
d) Not correct – Testing alone is not an increment in the development.

Question #13 (1 Point)

Which of the following should **NOT** be a trigger for maintenance testing?

a) Decision to test the maintainability of the software.
b) Decision to test the system after migration to a new operating platform.
c) Decision to test if archived data is possible to be retrieved.
d) Decision to test after “hot fixes”.

Select one option.

FL-2.4.1 (K2) Summarize triggers for maintenance testing

Justification

a) **Correct** – this is maintainability testing, not maintenance testing.
b) Not correct – this is a trigger for maintenance testing, see the syllabus chapter 2.4.1: Operational tests of the new environment as well as of the changed software.
c) Not correct – this a the trigger for maintenance testing, see the syllabus chapter 2.4.1: Testing restore/retrieve procedures after archiving for long retention periods.
d) Not correct – this a the trigger for maintenance testing, see the syllabus chapter 2.4.1: Reactive modification of a delivered software product to correct emergency defects that have caused actual failures.

Question #14 (1 Point)

Which of the following options are roles in a formal review?

a) Developer, Moderator, Review leader, Reviewer, Tester.
b) Author, Moderator, Manager, Reviewer, Developer.
c) Author, Manager, Review leader, Reviewer, Designer.
d) Author, Moderator, Review leader, Reviewer, Scribe.

Select one option.

FL-3.2.2 (K1) Recognize the different roles and responsibilities in a formal review

Justification

a) Not correct – Tester and developer are NOT roles as per Syllabus, section 3.2.2.
b) Not correct – Developer is NOT a role as per Syllabus, section 3.2.2.
c) Not correct – Designer is NOT a role as per Syllabus, section 3.2.2.
d) **Correct** –see Syllabus, section 3.2.2.

Question #15 (1 Point)

Which of the following describes the main activities of a formal review?

a) Initiation, backtracking, individual review, issue communication and analysis rework, follow-up.
b) Planning, individual review, issue communication and analysis, rework, closure, follow-up.
c) Planning, initiate review, individual review, issue communication and analysis, fixing and reporting.
d) Individual review, issue communication and analysis, rework, closure, follow-up, root cause analysis.

Select one option.

FL-3.2.1 (K2) Summarize the activities of the work product review process

Justification

a) Not correct – See c) for the activities in the review process.
b) Not correct – See c) for the activities in the review process.
c) **Correct** – According to Syllabus chapter 3.2.1: planning, initiate review, individual review, issue communication and analysis, fixing defects and report.
d) Not correct – See c) for the activities in the review process.

Question #16 (1 Point)

Which of the review types below is the BEST option to choose when the review must follow a formal process based on rules and checklists?

a) Informal Review.
b) Technical Review.
c) Inspection.
d) Walkthrough.

Select one option.

FL-3.2.3 (K2) Explain the differences between different review types: informal review, walkthrough, technical review and inspection

Justification

a) Not correct – Informal review does not use a formal process.
b) Not correct – Use of checklists are optional.
c) **Correct** – As per Syllabus 3.2.3: Formal process based on rules and checklists.
d) Not correct – Does not explicitly require a formal process.

Question #17 (1 Point)

Which TWO of the following statements about static testing are MOST true?

a) A cheap way to detect and remove defects.
b) It makes dynamic testing less challenging.
c) Early validation of user requirements.
d) It makes it possible to find run-time problems early in the lifecycle.
e) When testing safety-critical system, static testing has less value because dynamic testing finds the defects better.

Select two options.

FL-3.1.2 (K2) Use examples to describe the value of static testing

Justification

a) Correct – Syllabus 3.1.2: Defects found early are often much cheaper to remove than defects detected later in the lifecycle.
b) Not correct – Dynamic testing still has its challenging objectives
c) **Correct** – Syllabus 3.1.2: Preventing defects in design or coding by uncovering omissions, inaccuracies, inconsistencies, ambiguities, and redundancies in requirements.
d) Not correct – This is dynamic testing.
e) Not correct – Static analysis is important for safety-critical computer systems. Syllabus 3.1.

Question #18 (1 Point)

The design of a newspaper subscriptions system is being reviewed. The expected system users are:

- Subscribers
- Technical support team
- Billing department
- Database administrator

Each type of user logs into the system through a different login interface (e.g. subscribers login via a web page; technical support via an application).

Different reviewers were requested to review the system's login flow from the perspective of the above user categories.

Which of the following review comments is MOST LIKELY to have been made by all reviewers?

a) The login page on the web is cluttered with too much advertisement space. As a result, it is hard to find the "forgot password?" link.
b) The login to access the billing information should also allow access to subscribers' information and not force a second login session.
c) After logging-in to the database application, there is no log-out function.
d) The log in flow is un-intuitive since it requires entering the password first, before the user name can be keyed-in.

Select one option.

FL-3.2.4 (K3) Apply a review technique to a work product to find defects

Justification

a) Not correct – this impacts only the subscribers; possibly others but for sure not the technical support since they don't access the data via a web page.
b) Not correct – this comment would come from the review that took the perspective of the billing department, but not from other reviewers.
c) Not correct – this comment would come from the review that took the perspective of the database administrator, but not from other reviewers.
d) **Correct** – Every type of user must be authenticated before accessing to the system, so all users of the system would note (and suffer) an un-intuitive login flow.

Question #19 (1 Point)

What is checklist-based testing?

a) A test technique in which tests are derived based on the tester's knowledge of past failures, or general knowledge of failure modes.
b) Procedure to derive and/or select test cases based on an analysis of the specification, either functional or non-functional, of a component or system without reference to its internal structure.
c) An experience-based test technique whereby the experienced tester uses a high-level list of items to be noted, checked, or remembered, or a set of rules or criteria against which a product has to be verified.
d) An approach to testing where the tester dynamically designs and executes tests based on their knowledge, exploration of the test item and the results of previous tests.

Select one option.

FL-4.x (K1) Keywords

Justification

a) Not correct – This is error giessing,defined in Glossary.
b) Not correct – This is black-box test technique, defined in Glossary.
c) **Correct** – Defined in Glossary.
d) Not correct – This is exploratory testing, defined in Glossary.

Question #20 (1 Point)

Which one of the following options is categorized as a black-box test technique?

a) Techniques based on analysis of the architecture.
b) Techniques checking that the test object is working according to the technical design.
c) Techniques based on the expected use of the software.
d) Techniques based on formal requirements.

Select one option.

FL-4.1.1 (K2) Explain the characteristics, commonalities, and differences between black-box test techniques, white-box test techniques and experience-based test techniques

Justification

a) Not correct – This is a white-box test technique.
b) Not correct – This is a white-box test technique.
c) Not correct – This is a experience-based test technique.
d) **Correct** – Syllabus 4.1.2: Black-box test techniques (also called behavioral or behavior-based techniques) are based on an analysis of the appropriate test basis (e.g. formal requirements documents, specifications, use cases, user stories).

Question #21 (1 Point)

The following statement refers to decision coverage:
"When the code contains only a single 'if' statement and no loops or CASE statements, any single test case we run will result in 50% decision coverage."
Which of the following sentences is correct?

a) The sentence is true. Any single test case provides 100% statement coverage and therefore 50% decision coverage.
b) The sentence is true. Any single test case would cause the outcome of the "if" statement to be either true or false.
c) The sentence is false. A single test case can only guarantee 25% decision coverage in this case.
d) The sentence is false. The statement is too broad. It may be correct or not, depending on the tested software.

Select one option.

FL-4.3.2 (K2) Explain decision coverage

a) Not correct – While the given statement is true, the explanation is not.
b) **Correct** – Since any test will cause the outcome of the "if" statement to be either TRUE or FALSE, by defintion we achieved 50% decision coverage.
c) Not correct – A single test can give more than 25% decision coverage.
d) Not correct – The statement is specific and always true.

Question #22 (1 Point)

Which one of the following is the BEST description of statement coverage?

a) It is a metric which is used to calculate and measure the percentage of test cases that have been executed.
b) It is a metric, which is used to calculate and measure the percentage of statements in the source code which have been executed.
c) It is a metric, which is used to calculate and measure the number of statements in the source code which have been executed by test cases that are passed.
d) It is a metric that give a true/false confirmation if all statements are covered or not.

Select one option.

FL-4.3.1 (K2) Explain statement coverage

Justification

a) Not correct – Statement coverage measures the percentage of statements exercised by test cases.
b) **Correct** – Syllabus 4.3.1: Statement testing exercises the executable statements in the code. Coverage is measured as the number of statements executed by the tests divided by the total number of executable statements in the test object, normally expressed as a percentage.
c) Not correct – The coverage does not measure pass/fail.
d) Not correct – It is a metric, not a true/false.

Question #23 (1 Point)

Which TWO of the following statements about the relationship between statement coverage and decision coverage are true?

a) Decision coverage is stronger than statement coverage.
b) Statement coverage is stronger than decision coverage.
c) 100% statement coverage guarantees 100% decision coverage.
d) 100% decision coverage guarantees 100% statement coverage.
e) Decision coverage can never reach 100%.

Select two options.

FL-4.3.3 (K2) Explain the value of statement and decision coverage

Justification

See syllabus chapter 4.3.3: Achieving 100% decision coverage guarantees 100% statement coverage (but not vice versa).

Thus

a) **Correct –** The statement is true.
b) Not correct **–** The statement is false.
c) Not correct**–** The statement is false.
d) **Correct –** The statement is true.
e) Not correct **–** The statement is false.

Question #24 (1 Point)

Which of the following situations is NOT suited for using exploratory testing?

a) When there is time pressure, and/or the requirements are incomplete or inapplicable
b) When the system is developed and tested incrementally.
c) When only new and inexperienced testers are available.
d) When the main part of the application can be tested only at the customer's site.

Select one option.

FL-4.4.2 (K2) Explain exploratory testing

Justification

a) Not correct – Syllabus 4.4.2: Exploratory testing is most useful when there are few or significant time pressure on testing.
b) Not correct – exploratory testing can be used here.
c) **Correct** – exploratory tests should be performed by experienced testers with knowledge of similar applications and technologies. The tester needs constantly to make decisions during exploratory testing, e.g.what to test next.
d) Not correct – exploratory testing can be used at any location.

Question #25 (1 Point)

An employee's bonus is to be calculated. It cannot be negative, but it can be calculated down to zero. The bonus is based on the length of employment.

The categories are: less than or equal to 2 years, more than 2 years but less than 5 years, 5 or more years, but less than 10 years, 10 years or longer.

What is the minimum number of test cases required to cover all valid equivalence partitions for calculating the bonus?

a) 3.
b) 5.
c) 2.
d) 4.

Select one option.

FL-4.2.1 (K3) Apply equivalence partitioning to derive test cases from given requirements

Justification

a) Not correct – see the correct partitions in d).
b) Not correct – see the correct partitions in d).
c) Not correct – see the correct partitions in d).
d) **Correct** – Partions as below:
    1. equivalence partition: 0 < employment time ≤ 2.
    2. equivalence partition: 2 < employment time < 5.
    3. equivalence partition: 5 ≤ employment time < 10.
    4. equivalence partition: 10 ≤ employment time.

Question #26 (1 Point)

A speed control and reporting system has the following characteristics:
If you drive 50 km/h or less, nothing will happen.
If you drive faster than 50 km/h, but 55 km/h or less, you will be warned.
If you drive faster than 55 km/h but not more than 60 km/h, you will be fined.
If you drive faster than 60 km/h, your driving license will be suspended.

Which would be the most likely set of values (km/h) identified by two-point boundary value analysis?

a) 0, 49, 50, 54, 59, 60.
b) 50, 55, 60.
c) 49, 50, 54, 55, 60, 62.
d) 50, 51, 55, 56, 60, 61.

Select one option.

FL-4.2.2 (K3) Apply boundary value analysis to derive test cases from given requirements

Justification

The following partitions can be identified:

1. – 50	Two-point boundaries 50, 51
2. 51 – 55	Two-point boundaries 50, 51, 55, 56
3. 56 – 60	Two-point boundaries 55, 56, 60, 61
4. 61 –	Two-point boundaries 60, 61

Thus:

a) Not correct – Does not include all two-point boundary values. Also includes values not necessary for two-point boundary value analysis.
b) Not correct – Does not include all two-point boundary values.
c) Not correct – Does not include all two-point boundary values. Also includes values not necessary for two-point boundary value analysis.
d) **Correct** – Includes all two-point boundary values

Question #27 (1 Point)

A company's employees are paid bonuses if they work more than a year in the company and achieve individually agreed targets.

The following decision table has been designed to test the logic for paying bonuses:

		T1	T2	T3	T4	T5	T6	T7	T8
Conditions									
Cond1	Employment for more than 1 year?	YES	NO	YES	NO	YES	NO	YES	NO
Cond2	Agreed target?	NO	NO	YES	YES	NO	NO	YES	YES
Cond3	Achieved target?	NO	NO	NO	NO	YES	YES	YES	YES
Action									
	Bonus payment?	NO	NO	NO	NO	NO	NO	YES	NO

Which test cases could be eliminated in the above decision table because the test case wouldn't occur in a real situation?

a) T1 and T2.
b) T3 and T4.
c) T7 and T8.
d) T5 and T6.

Select one option.

FL-4.2.3 (K3) Apply decision table testing to derive test cases from given requirements

Justification

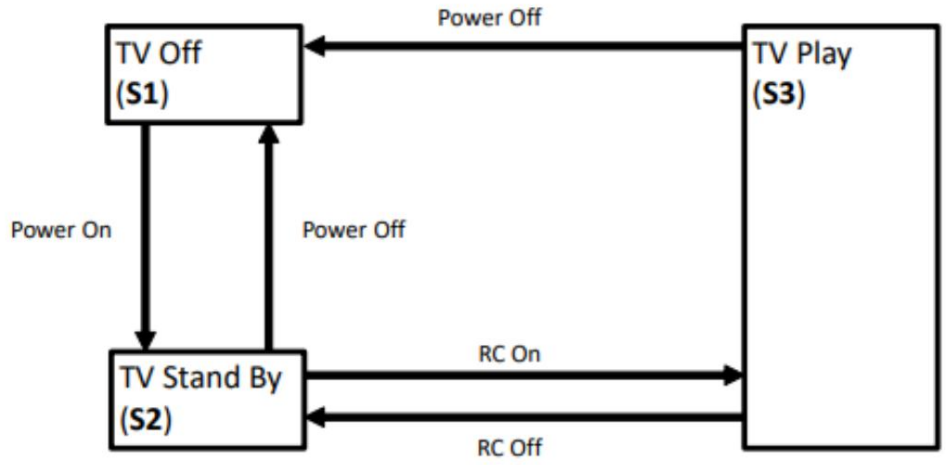
In the test cases T5 and T6 the situation described is logically impossible. If there was no agreement on targets, it's impossible to claim that the targets were reached. Since this situation can´t occur, we therefore can eliminate the corresponding test cases.

Hence

a) Not correct.
b) Not correct.
c) Not correct.
d) **Correct.**

Question #28 (1 Point)

Which of the following statements about the given state transition diagram and table of test cases is TRUE?



Test Case	1	2	3	4	5
Start State	S1	S2	S2	S3	S3
Input	Power On	Power Off	RC On	RC Off	Power Off
Expected Final State	S2	S1	S3	S2	S1

a) The given test cases can be used to cover both valid and invalid transitions in the state transition diagram.
b) The given test cases represent all possible valid transitions in the state transition diagram.
c) The given test cases represent only some of the valid transitions in the state transition diagram.
d) The given test cases represent sequential pairs of transitions in the state transition diagram.

Select one option.

FL-4.2.4 (K3) Apply state transition testing to derive test cases from given requirements

Justification

Proposed test case cover all five possible single valid transitions in the given state diagram

(S1->S2, S2->S1, S2->S3, S3->S2, S3->S1).

a) Not correct – because no invalid transitions are covered.
b) **Correct** – because all valid transitions are covered.
c) Not correct – because all valid transitions are covered.
d) Not correct – because the order in which the test cases are run has not been specified, we don't know what pairs of transitions will occur.

Question #29 (1 Point)

A video application has the following requirement:
The application shall allow playing a video on the following display sizes:

1. 640x480.
2. 1280x720.
3. 1600x1200.
4. 1920x1080.

Which of the following list of test cases is a result of applying the Equivalence Partitioning test technique to test this requirement?

a) Verify that the application can play a video on a display of size 1920x1080 (1 test).
b) Verify that the application can play a video on a display of size 640x480 and 1920x1080 (2 tests).
c) Verify that the application can play a video on each of the display sizes in the requirement (4 tests).
d) Verify that the application can play a video on any one of the display sizes in the requirement (1 test).

Select one option.

FL-4.2.1 (K3) Apply Equivalence Partitioning technique to derive test cases from given requirements

Justification

a) Not correct – See c).
b) Not correct – See c).
c) **Correct** – This is a case where the requirement gives an enumeration of discrete values. Each enumeration value is an Equivalence Class by itself, therefore each will be tested when using Equivalent Partitioning test technique.
d) Not correct – See c).

Question #30 (1 Point)

Which of the following BEST describes how tasks are divided between the test manager and the tester?

a) The test manager plans testing activities and chooses the standards to be followed, while the tester chooses the tools and controls to be used.
b) The test manager plans, organizes, and controls the testing activities, while the tester specifies and executes tests.
c) The test manager plans, monitors, and controls the testing activities, while the tester designs tests and decides about automation frameworks.
d) The test manager plans and organizes the testing and specifies the test cases, while the tester prioritizes and executes the tests.

Select one option.

FL-5.1.2 (K1) Identify the tasks of a test manager and tester

Justification

a) Not correct – Syllabus 5.1.2: The tester uses the tools.
b) **Correct** – See Syllabus 5.1.2.
c) Not correct – Deciding about automation frameworks is not a tester's task.
d) Not correct – Test manager does not specify the test cases.

Question #31 (1 Point)

Which of the following metrics would be MOST useful to monitor during test execution?

a) Percentage of executed test cases.
b) Percentage of work done in test environment preparation.
c) Percentage of planned test cases prepared.
d) Percentage of work done in test case preparation.

Select one option.

FL-5.3.1 (K1) Recall metrics used for testing

Justification

a) **Correct** Syllabus 5.3.1: Test case execution (e.g. number of test cases run/not run, and test cases passed/failed).
b) Not correct – Should be monitored during test preparation.
c) Not correct – Should be monitored during test preparation.
d) Not correct – Should be monitored during test preparation.

Question #32 (1 Point)

Which TWO of the following can affect and be part of test planning?

a) Budget limitations.
b) Test objectives.
c) Test log.
d) Failure rate.
e) Use cases.

Select two options.

FL-5.2.1 (K2) Summarize the purpose and content of a test plan

Justification

a) **Correct** – When you are planning the test and there are budget limitations, prioritizing is needed; what should be tested and what should be omitted.
b) **Correct** – See syllabus 5.2.1.
c) Not correct – it is a part of test monitoring and control.
d) Not correct – it is a part of test monitoring and control.
e) Not correct – it is a part of test design.

Question #33 (1 Point)

Which of the following are typical exit criteria from testing?

a) Reliability measures, degree of tester's independence, and product completeness.
b) Reliability measures, test cost, availability of testable code, time to market, and product completeness.
c) Reliability measures, test cost, schedule and unresolved defects.
d) Time to market, residual defects, tester qualification, degree of tester independence and test cost.

Select one option.

FL-5.2.3 (K2) Give examples of potential entry and exit criteria

Justification

a) Not correct – Degree of tester's independence does not play a role in exit criteria.
b) Not correct – "Availability of testable code" is an entry criteria.
c) **Correct** – See Syllabus 5.2.3.
d) Not correct – Degree of tester's independence as well as tester qualification do not play a role in exit criteria.

Question #34 (1 Point)

Which one of the following is **NOT** included in a test summary report?

a) Defining pass/fail criteria and objectives of testing.
b) Deviations from the test approach.
c) Measurements of actual progress against exit criteria.
d) Evaluation of the quality of the test item.

Select one option.

FL-5.3.2 (K2) Summarize the purposes, content, and audiences for test reports

Justification

a) **Correct** – This information has been defined earlier in the test project.
b) Not correct – This information is included in a test report; see the Syllabus chapter 5.3.2: Information on what occurred during a test period.
c) Not correct – This information is included in a test report; see Syllabus 5.3.2: Information and metrics to support recommendations and decisions about future actions, such as an assessment of defects remaining, the economic benefit of continued testing, outstanding risks, and the level of confidence in the tested software.
d) Not correct – This information is included in a test report; see Syllabus 5.3.2: Information and metrics to support recommendations and decisions about future actions, such as an assessment of defects remaining, the economic benefit of continued testing, outstanding risks, and the level of confidence in the tested software.

Question #35 (1 Point)

There are several test strategies. Which strategy (1-4) is characterized by which description (A-D) below?

1. Analytical.
2. Methodical.
3. Model-based.
4. Consultative.

A. Tests are based on a state diagram of a required aspect of the product
B. Tests are designed and prioritized based on the level of risk.
C. Systematic use of some predefined set of test conditions.
D. Tests are chosen based on the views of business domain experts.

a) 1D, 2B, 3A, 4C.
b) 1A, 2C, 3D, 4B.
c) 1D, 2C, 3B, 4A.
d) 1B, 2C, 3A, 4D.

Select one option.

FL-5.2.2 (K2) Differentiate between various test strategies

Justification

Analytical: Syllabus 5.2.2, This type of test strategy is based on an analysis of some factor (e.g. requirement or risk). (1B).

Methodical: Syllabus 5.2.2, In this type of test strategy relies on making systematic use of some predefined set of tests or test conditions, (2C).

Model-based: Syllabus 5.2.2, In this test strategy, tests are designed based on some model of some required aspect of the product, … (3A).

Consultative (or Directed): Syllabus 5.2.2, This type of test strategy is driven primarily by the advice, guidance, or instructions of stakeholders, business domain experts, or technology experts, who may beoutside the test team or outside the organization itself. (4D).

Thus:

a) Not correct.
b) Not correct.
c) Not correct.
d) **Correct.**

Question #36 (1 Point)

Which one of the following is the characteristic of a metrics-based approach for test estimation?

a) Budget which was used by a previous similar test project.
b) Overall experience collected in interviews with test managers.
c) Overall estimate agreed with the developers.
d) Average of calculations collected from business experts.

Select one option.

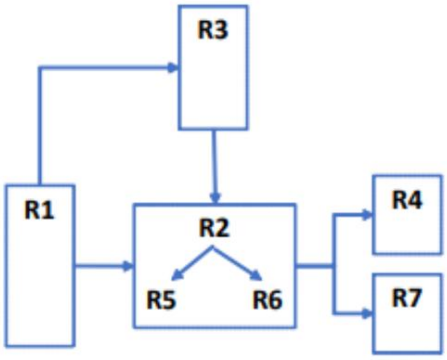
FL-5.2.6 (K2) Explain the difference between two estimation techniques: the metrics-based technique and the expert-based technique

Justification

a) **Correct** – From Syllabus chapter 5.2.6: The metrics-based approach: estimating the testing effort based on metrics of former similar projects or based on typical values .
b) Not correct – This is expert-based approach: estimating the tasks based on estimates made by the owners of the tasks or by experts.
c) Not correct – This is expert-based approach: estimating the tasks based on estimates made by the owners of the tasks or by experts.
d) Not correct – This is expert-based approach: estimating the tasks based on estimates made by the owners of the tasks or by experts.

Question #37 (1 Point)

The following diagram shows the logical dependencies between a set of seven requirements, where a dependency is shown by an arrow. For example, "R1 -> R3" means that R3 depends on R1.



Which one of the following options structures the test execution schedule according to the requirement dependencies?

a) R1 → R3 → R1 → R2 → R5 → R6 → R4 → R7.
b) R1 → R3 → R2 → R5 → R2 → R6 → R4 → R7.
c) R1 → R3 → R2 → R5 → R6 → R4 → R7.
d) R1 → R2 → R5 → R6 → R3 → R4 → R7.

Select one option.

FL-5.2.4 (K3) Apply knowledge of prioritization, technical and logical dependencies to schedule test execution for a given set of test cases

Justification

a) Not correct – everything is dependent on R1, so any test flow that does not start with R1 is FALSE.
b) Not correct – everything is dependent on R1, so any test flow that does not start with R1 is FALSE.
c) **Correct** – the tests are specified in a sequence that takes the dependencies into account.
d) Not correct – R2 is dependent on R3, so R3 should be tested before R2.

Question #38 (1 Point)

You are testing a new version of software for a coffee machine. The machine can prepare different types of coffee based on four categories. i.e. coffee size, sugar, milk and syrup. The criteria are as follows:

- Coffee size (small, medium, large),
- Sugar (none, 1 unit, 2 units, 3 units, 4 units),
- Milk (yes or no),
- Coffee flavor syrup (no syrup, caramel, hazelnut, vanilla).

Now you are writing a defect report with the following information:

Title: Low coffee temperature.
Short summary: When you select coffee with milk, the time for preparing coffee is too long and the temperature of the beverage is too low (less than 40 °C)
Expected result: The temperature of coffee should be standard (about 75 °C).
Degree of risk: Medium
Priority: Normal

What valuable information is MOST likely to be omitted in the above defect report?

a) The actual test result.
b) Data identifying the tested coffee machine.
c) Status of the defect.
d) Ideas for improving the test case.

Select one option.

FL-5.6.1 (K3) Write a defect report, covering defects found during testing.

Justification

a) Not correct – the test result is given in the short summary.
b) **Correct** – when testing different versions of software, identifying information is necessary. Syllabus 5.6: Identification of the test item (configuration item being tested) and environment.
c) Not correct – You are just writing the defect report, hence the status is automatically open.
d) Not correct – This information is useful for the tester, but does not need to be included in the defect report.

Question #39 (1 Point)

Which one of the following is MOST likely to be a benefit of using test execution tools?

a) It is easy to create regression tests.
b) It is easy to maintain version control of test assets.
c) It is easy to design tests for security testing.
d) It is easy to run regression tests.

Select one option.

FL-6.1.2 (K1) Identify benefits and risks of test automation

Justification

a) Not correct – The benefits are not when creating regressions tests, more in executing them.
b) Not correct – This is done by configuration Management tools.
c) Not correct – This needs specialized tools.
d) **Correct** – Syllabus 6.1.2: Reduction in repetitive manual work (e.g. running regression tests, environment set up/tear down tasks, re-entering the same test data, and checking against coding standards), thus saving time.

Question #40 (1 Point)

Which test tool is characterized by the classification below?

1. Tool support for management of testing and testware.
2. Tool support for static testing.
3. Tool support for test execution and logging.
4. Tool support for performance measurement and dynamic analysis.

A. Coverage tools.
B. Configuration management tools.
C. Review tools.
D. Monitoring tools.

a) 1A, 2B, 3D, 4C.
b) 1B, 2C, 3D, 4A.
c) 1A, 2C, 3D, 4B.
d) 1B, 2C, 3A, 4D.

Select one option.

FL-6.1.1 (K2) Classify test tools according to their purpose and the test activities they support

Justification

Support for management of testing and testware: Syllabus 6.1.1, Configuration management tools, (1B).

Support for static testing: Syllabus 6.1.1, Tools that support reviews, (2C).

Support for test execution and logging: Syllabus 6.1.1, Coverage tools, (3A).

Support for performance measurement and dynamic analysis: Syllabus 6.1.1, Performance testing tools/monitoring tools/dynamic analysis tools, (4D).

Thus:

a) Not correct
b) Not correct
c) Not correct
d) **Correct**