Résumé - Partie II

Qualité Logicielle - ISTQB Foundation Level

Tests Logiciels

---

1 Introduction à ISTQB

1.1 Présentation

ISTQB : International Software Testing Qualifications Board - Qualification internationale en tests de logiciels

Niveau Fondation : Certification de base en tests logiciels reconnue internationalement

1.2 Public cible

— Testeurs, analystes de tests, ingénieurs de tests
— Consultants en test, tests managers
— Développeurs de logiciel, membres de l'équipe de développement
— Chefs de projets, responsables qualité, Product Owners
— Responsables du développement logiciel
— Analystes métier, directeurs informatiques
— Consultants en management

1.3 Examen ISTQB Foundation

Critère	Valeur
Durée	60 minutes
Nombre de questions	40
Note de passage	26/40 (65%)
Réponses par question	1 seule correcte
Pénalité	Pas de marque négative

Répartition par chapitre :

— Ch. 1 : 8 questions
— Ch. 2 : 5 questions
— Ch. 3 : 5 questions
— Ch. 4 : 11 questions
— Ch. 5 : 9 questions
— Ch. 6 : 2 questions

Niveaux de connaissances (K-Level) :

— K1 (Se souvenir) : 20 questions
— K2 (Comprendre) : 12 questions
— K3 (Utiliser) : 8 questions

2 Fondamentaux des Tests Logiciels

2.1 Définition et objectifs

Tests logiciels : Moyen d'évaluer la qualité du logiciel et de réduire le risque de défaillance en cours de fonctionnement.
Garantissent : Fiabilité, Performance, Gain en temps, Gain en argent

2.2 Vérification vs Validation

Vérification "Est-ce que nous construisons le produit correctement ?"
— Confirmation que le produit respecte les spécifications prédéfinies

Validation "Est-ce que nous construisons le bon produit ?"
— Confirmation que le produit répond aux besoins de l'utilisateur

2.3 Objectifs des tests

— Évaluer les produits d'activités
— Vérifier si toutes les exigences spécifiées ont été satisfaites
— Valider si l'objet de test fonctionne comme attendu
— Construire la confiance dans le niveau de qualité
— Trouver et prévenir des défaillances et défauts
— Fournir suffisamment d'information pour la prise de décision
— Réduire le niveau de risque
— Se conformer aux exigences contractuelles, légales ou réglementaires

2.4 Origine des défauts

Erreur (humaine) → **Défaut** (bug dans le code) → **Défaillance** (comportement incorrect)
Important : Tous les défauts ne génèrent pas de défaillances

Types de défauts :

Faux positifs Test signale un défaut qui n'existe pas réellement
— Causés par erreurs dans les données de test, l'environnement ou l'implémentation des tests

Faux négatifs Tests ne détectent pas les défauts existants
— Manque de couverture ou scénarios de test incomplets

2.5 Causes des défauts

— Nature humaine (les humains se trompent)
— Échéanciers serrés
— Complexité du code et des infrastructures
— Modifications de technologies
— Multiples interactions entre systèmes
— Conditions d'environnement (radiations, magnétisme, pollution)

3 Tests Statiques vs Tests Dynamiques

3.1 Tests statiques

Définition : Examen manuel (revues) ou analyse (analyse statique) du code ou de la documentation **sans exécution du code**
Objectif : Identification précoce des erreurs, prévention des problèmes

Exemples :

— Revue de code (inspection manuelle)

— Analyse statique (outils comme SonarQube)

— Vérification de la documentation (exigences, diagrammes)

3.2 Tests dynamiques

Définition : Exécution du code testé pour observer son comportement dans différents scénarios
Objectif : Valider les fonctionnalités, mesurer la performance, identifier les bugs

Exemples :

— Tests fonctionnels (vérification des exigences)

— Tests de performance (rapidité, capacité)

— Tests de sécurité (identification des vulnérabilités)

3.3 Complémentarité

Tests statiques (préventif) + **Tests dynamiques** (curatif) = **Qualité logicielle**

3.4 Test vs Débogage

Tester Activité du testeur : test initial et test de confirmation final

Déboguer Activité du développeur : trouver, analyser et supprimer les causes de défaillance

4 7 Principes Généraux des Tests

1. **Les tests montrent la présence de défauts, pas leur absence**
   — Les tests prouvent la présence mais pas l'absence de défauts
2. **Les tests exhaustifs sont impossibles**
   — Tout tester n'est pas faisable sauf pour des cas triviaux
   — Focaliser les efforts selon les risques et priorités
3. **Tester tôt économise du temps et de l'argent**
   — La prévention est le meilleur investissement
   — Plus un défaut est détecté tard, plus son impact est élevé
4. **Regroupement des défauts**
   — Une petite partie des modules contient la majorité des défauts (Pareto 80/20)
   — Concentrer les tests sur les modules critiques
5. **Paradoxe du pesticide**
   — Si les mêmes tests sont répétés, les nouveaux défauts ne seront pas découverts
   — Le système de tests doit évoluer
6. **Les tests dépendent du contexte**
   — Les tests sont effectués différemment selon les contextes
   — Exemple : Application bancaire vs application de jeu
7. **L'absence d'erreurs est une illusion**
   — Aucun défaut détecté ne garantit pas que le logiciel répond aux besoins
   — Un logiciel peut être fonctionnel mais manquer d'ergonomie

5 Assurance Qualité vs Contrôle Qualité

5.1 Assurance qualité (QA)

Approche préventive : Prévenir les défauts avant qu'ils ne se produisent
Objectif : S'assurer que les processus de développement sont suivis correctement

Activités :

— Respect des processus adéquats

— Analyse des causes racines

— Application des résultats des rétrospectives

Exemple : Checklist obligatoire pour chaque étape de développement

5.2 Contrôle qualité (QC - Test)

Approche corrective : Identifier les défauts avant la livraison
Objectif : Garantir que le produit répond aux attentes

Activités :

— Tests pour vérifier les exigences

— Identification des défauts

— Validation avant livraison

Exemple : Tests de régression après une mise à jour

6 Processus de Test

6.1 Vue d'ensemble

1. Planification, pilotage et contrôle des tests
2. Analyse et conception des tests
3. Implémentation et exécution des tests
4. Évaluer les critères de sortie et informer
5. Activités de clôture des tests

6.2 1. Planification, pilotage et contrôle

Planification :

— Définir les objectifs du test

— Spécifier les activités de test

— Créer le plan de test

Pilotage :

— Comparaison régulière de l'avancement réel au plan

— Utilisation de métriques de pilotage

Contrôle :

— Prendre les mesures nécessaires pour satisfaire aux objectifs

— Mise à jour du plan si nécessaire

6.3 2. Analyse et conception des tests

Analyse (Quoi tester) :

— Réviser les bases du test (exigences, architecture, conception)
— Évaluer la testabilité des exigences
— Identifier et prioriser les conditions de test

Conception (Comment tester) :

— Concevoir et prioriser les tests de haut niveau
— Identifier les données de test nécessaires
— Concevoir l'environnement de test
— Créer une traçabilité bidirectionnelle

6.4 3. Implémentation et exécution

Implémentation :

— Finaliser et développer les cas de test
— Développer les procédures de test
— Créer les suites de tests
— Vérifier l'environnement de test

Exécution :

— Exécuter les procédures de test
— Consigner les résultats
— Comparer résultats actuels et attendus
— Signaler les divergences comme incidents
— Répéter les activités (test de confirmation, régression)

6.5 4. Évaluation des critères de sortie

— Vérifier les registres de tests
— Évaluer si des tests supplémentaires sont requis
— Écrire un rapport de synthèse pour les parties prenantes

6.6 5. Activités de clôture

— Vérifier quels livrables ont été livrés
— Clôturer les rapports d'incidents
— Documenter l'acceptation du système
— Finaliser et archiver les testwares
— Analyser les leçons apprises
— Améliorer la maturité des tests

7 Terminologie des Tests

7.1 Concepts clés

Condition de test Article ou événement qui pourrait être vérifié par un ou plusieurs cas de tests

— Exemple : Vérifier la fonctionnalité "Connexion"

Cas de test Ensemble de valeurs d'entrée, préconditions, résultats attendus et post-conditions

— Exemple : Tester avec identifiant valide, invalide, champ vide

Procédure de test Séquence d'actions pour l'exécution d'un test

— Manuel : Instructions pour le testeur

— Automatisé : Script de test

Testware Artefacts produits pendant le processus de test

— Documentation, scripts, données, environnements, outils

8 Niveaux de Tests

8.1 1. Test de composant (Test unitaire)

Objectif : Rechercher des défauts et vérifier le bon fonctionnement des modules testables séparément

Responsable : Généralement les développeurs

Bases de test :

— Conception détaillée, Code source

— Modèle de données, Spécifications des composants

Objets de test :

— Composants, unités, modules

— Code et structures de données

— Classes, Modules de bases de données

Défauts courants :

— Fonctionnalité incorrecte

— Problèmes de flux de données

— Code et logique incorrects

Outils de support :

Pilote (driver) Remplace un composant qui contrôle ou appelle le composant testé

Bouchon (stub) Remplace un composant appelé par le composant testé

Simulateur Se comporte comme un système donné

8.2 2. Test d'intégration

Objectif : Tester les interfaces entre composants et les interactions entre différentes parties du système

Niveaux : Tests d'intégration des composants, Tests d'intégration système

Bases de test :

— Conception du logiciel et du système

— Architecture, Workflows, Cas d'utilisation

Objets de test :

— Sous-systèmes, Bases de données

— Interfaces, APIs, Micro-services

— Configuration système

Stratégies d'intégration :

Big Bang Intégration en une seule fois

— Avantage : Peu de contraintes de planification

— Inconvénient : Consolidation non maîtrisable, diagnostic difficile

Incrémentale Top-Down Du haut vers le bas de la hiérarchie

— Avantage : Cartographie rapide, visibilité précoce
— Inconvénient : Interfaces matérielles testées tardivement

Incrémentale Bottom-Up Du bas vers le haut

— Avantage : Interfaces matérielles testées tôt
— Inconvénient : Détection tardive des erreurs de conception

8.3 3. Test système

Objectif : Traiter le comportement d'un système/produit fini
Responsable : Équipe indépendante généralement

Bases de test :

— Spécifications des exigences système et logicielles
— Rapports d'analyse des risques
— Cas d'utilisation, Epics, User Stories
— Modèles de comportement, Diagrammes d'états
— Manuels système et d'utilisation

Objets de test :

— Applications, Systèmes Matériel/Logiciel
— Systèmes d'exploitation
— Système sous test (SUT)
— Configuration système

Défauts courants :

— Calculs incorrects
— Comportement incorrect ou inattendu
— Flux de contrôle et/ou de données incorrects
— Incapacité à fonctionner dans l'environnement de production

8.4 4. Tests d'acceptation

Objectif : Évaluer si le système est prêt à être déployé et conforme aux exigences
Responsables : Clients, utilisateurs métier, Product Owners, parties prenantes

Formes de tests d'acceptation :

Tests d'acceptation utilisateurs Vérifier l'aptitude et l'utilisabilité par les utilisateurs

Tests d'acceptation opérationnelle — Tests des backups et restaurations
— Reprise après sinistre
— Gestion des utilisateurs
— Tâches de maintenance

Tests contractuels et réglementaires — Basés sur critères d'acceptation contractuels
— Conformité aux règlements et législations

Tests alpha et beta — Alpha : Tests internes avant la release
— Beta : Tests par utilisateurs sur leurs propres sites

9 Types de Tests

9.1 Tests fonctionnels vs non fonctionnels

Tests fonctionnels "Que fait le système ?"
- Évaluent les fonctionnalités que le système devrait réaliser
- Basés sur les spécifications fonctionnelles
- Couverture fonctionnelle mesurable

Tests non fonctionnels "Comment le système fonctionne ?"
- Évaluent les caractéristiques du système
- Performance, charge, stress, utilisabilité
- Maintenabilité, fiabilité, portabilité

9.2 Tests structurels (Boîte blanche)

Définition : Tests basés sur l'analyse de la structure ou l'implémentation interne
Nécessitent : Connaissance détaillée du code

Exemples :
- Test de couverture des instructions
- Test de couverture des décisions
- Test de couverture des branches
- Test de couverture des conditions

9.3 Tests liés aux changements

Tests de confirmation Vérifier que la correction a résolu le défaut
- Exécution des cas de test qui ont échoué précédemment

Tests de régression S'assurer qu'aucun défaut n'a été introduit dans les parties non modifiées
- Tests d'un programme après modification

9.4 Tests de maintenance

Une fois déployés, les logiciels nécessitent des changements pour :
- Corriger des défauts découverts en production
- Ajouter de nouvelles fonctionnalités
- Préserver ou améliorer les caractéristiques de qualité

10 Cycle de Vie du Développement

10.1 Modèle en V

Principe : À chaque phase de développement correspond une phase de test
- Étude des besoins → Tests d'acceptation
- Spécification → Tests système
- Conception générale → Tests d'intégration
- Conception détaillée → Tests unitaires
- Implémentation

Bonnes pratiques :
- Conception des tests précoce (en amont)
- Exécution des tests en aval
- Plus efficace que la conception tardive

10.2 Modèle en cascade

Les activités se réalisent l'une après l'autre de manière séquentielle :

— Spécifications → Conception → Codage → Tests → Production → Maintenance

10.3 Modèles itératifs

Développement incrémental :

— Définition par morceaux (fragments)

— Fonctionnalités augmentent de façon incrémentale

Développement itératif :

— Groupes de caractéristiques dans des cycles fixes

— Chaque itération délivre une fonctionnalité opérationnelle

Exemples : Scrum (itérations de 1-4 semaines), Spiral, XP
Importance : Les tests de régression deviennent cruciaux

11 Techniques de Test Boîte Noire

11.1 1. Partition d'équivalence

Principe : Groupe d'entrées montrant un comportement similaire
Méthode : Il suffit de prendre une donnée du groupe pour valider toute la partition

Exemple : Remboursement de frais de repas (limite 20 DH)

— Partition 1 : Montant < 0 (ex : -1)

— Partition 2 : Montant $= 0$

— Partition 3 : $0 <$ Montant < 20 (ex : 10)

— Partition 4 : Montant $= 20$

— Partition 5 : Montant > 20 (ex : 25)

11.2 2. Analyse des valeurs limites

Valeur limite : Valeur au bord d'une partition ou à la distance minimale de chaque côté
Types : Valeurs valides, invalides, aux limites

Exemple : Mois de février

— Année non bissextile : 27 (valide), 28 (limite), 29 (invalide)

— Année bissextile : 28 (valide), 29 (limite), 30 (invalide)

11.3 3. Table de décision

Principe : Combinaison des entrées (causes) et de leurs sorties (effets)
Méthode :

1. Identifier les conditions influant sur la décision
2. Identifier toutes les actions possibles
3. Identifier toutes les combinaisons (2^n)
4. Construire la table
5. Simplifier en éliminant les règles impossibles

11.4 4. Tests de transition d'état

Principe : Un système peut prendre différents états et réagir différemment aux événements
Méthode : Diagramme ou tableau montrant les états et transitions possibles

Couverture possible :

— Séquence de plusieurs états
— Tous les états
— Toutes les transitions
— Transitions invalides

11.5 5. Tests de cas d'utilisation

— Tests spécifiés à partir des cas d'utilisation
— Utiles pour découvrir les défauts dans le flux de traitement
— Utiles pour concevoir des tests d'acceptation
— Découvrir des défauts d'intégration causés par l'interaction

12 Techniques de Test Boîte Blanche

12.1 Test et couverture des instructions

Principe : Exécuter chaque ligne du code
Couverture : Nombre d'instructions exécutées / Total d'instructions exécutables

12.2 Test et couverture des décisions

Principe : Tester chaque branche (if et else)
Couverture : Nombre de résultats de décision exécutés / Total de résultats de décision

12.3 Autres techniques

Couverture de condition Tester chaque condition séparément pour vrai et faux

Couverture des conditions multiples Tester toutes les combinaisons possibles

13 Techniques Basées sur l'Expérience

13.1 Estimation d'erreur

Anticiper les erreurs, défauts et défaillances basé sur :

— Analyse du comportement précédent de l'application
— Types d'erreurs fréquents des développeurs
— Défauts observés dans d'autres applications

13.2 Tests exploratoires

Principe : Tests informels conçus, exécutés et évalués dynamiquement pendant l'exécution
Utilisation : Peu de spécifications ou contraintes de temps importantes

13.3 Tests basés sur des checklists

Couvrir les conditions de test figurant dans une checklist

— Utilisation : En l'absence de cas de tests détaillés

14 Gestion des Tests

14.1 Organisation des tests

14.1.1 Indépendance des tests

Niveaux d'indépendance :

1. Pas de testeurs indépendants (développeurs testent leur code)
2. Testeurs indépendants dans l'équipe de développement
3. Équipe de test indépendante dans l'organisation
4. Testeurs de la communauté des utilisateurs
5. Testeurs indépendants externes

Avantages :

— Plus de crédibilité
— Détection de défauts différents
— Vérification objective

Inconvénients :

— Déconnexion de l'équipe de développement
— Perte du sens de responsabilité des développeurs
— Risque de goulet d'étranglement

14.2 Rôles dans les tests

14.2.1 Responsable de test (Test Manager)

— Planifier les tests
— Surveiller et contrôler l'exécution
— Adapter le planning selon les résultats
— Rédiger et mettre à jour les plans de test
— Introduire des mesures appropriées
— Sélectionner les outils
— Décider de l'environnement de test
— Établir des rapports de synthèse
— Promouvoir et soutenir les testeurs

14.2.2 Testeur

— Passer en revue les plans de test et y contribuer
— Analyser et évaluer la testabilité des exigences
— Créer des spécifications de test
— Mettre en place l'environnement de test
— Préparer et obtenir les données de test
— Implémenter, exécuter et documenter les tests
— Utiliser les outils de test
— Mesurer les performances
— Passer en revue les tests développés par d'autres

14.3 Planification des tests

Activités :

— Définir le périmètre et les objectifs
— Définir la stratégie de test
— Définir les critères d'entrée et de sortie
— Intégrer les activités de test dans le cycle de développement
— Planifier l'analyse, la conception et l'exécution
— Assigner les ressources
— Définir le niveau de détail pour la documentation

14.4 Stratégies de test

Analytique Basée sur les risques
Basée sur des modèles À partir d'un modèle du produit
Méthodique Ensemble prédéfini de tests
Conforme à un processus Basée sur des règles et normes externes
Dirigée Dictée par les recommandations des parties prenantes
Anti-régressions Éviter la régression des fonctionnalités
Réactive Réaction aux événements pendant l'exécution

14.5 Critères d'entrée et de sortie

Critères d'entrée :

— Disponibilité de l'environnement de test
— Préparation des outils de tests
— Disponibilité du code testable
— Disponibilité des jeux de données

Critères de sortie :

— Mesures d'exhaustivité (couverture)
— Estimation de la densité des anomalies
— Coût et budget
— Risques résiduels

14.6 Métriques de test

— Pourcentage du travail prévu réalisé
— Exécution des cas de test (ok, ko, bloqués)
— Informations sur les défauts (densité, trouvés, corrigés)
— Couverture des exigences, risques ou code
— Dates des jalons
— Coût des tests vs bénéfice

15 Gestion des Risques

15.1 Définition

Risque = Probabilité d'occurrence × Impact
Risque : Possibilité d'un événement futur ayant des conséquences négatives

15.2 Types de risques

Risques liés au projet Situations affectant la capacité du projet à atteindre ses objectifs

- Problèmes organisationnels (retards de décision)
- Problèmes de personnel (manque de compétences)
- Environnement de test indisponible

Risques liés au produit Le produit ne satisfait pas les besoins

- Fonctionnalité manquante
- Caractéristique non fonctionnelle de moindre qualité
- Possibilité de dommages (logiciel médical)

15.3 Approche basée sur les risques

Les résultats de l'analyse sont utilisés pour :

- Déterminer les techniques de test
- Déterminer les niveaux et types de tests
- Déterminer le volume des tests
- Prioriser les tests (trouver les défauts critiques tôt)
- Déterminer des activités complémentaires pour réduire les risques

16 Gestion des Incidents

16.1 Définition

Incident : Tout événement arrivant pendant les tests qui requiert une vérification
Causes : Bugs dans l'application, tests mal exécutés, résultats attendus incorrects

16.2 Contenu d'un rapport d'incidents

- Identifiant unique
- Titre et bref résumé du défaut
- Date du rapport, organisation émettrice, auteur
- Identification de l'élément de test et de l'environnement
- Phase(s) du cycle de développement où observé
- Description détaillée du défaut
- Historique des modifications
- Portée ou degré d'impact (sévérité)
- Urgence/priorité de la correction
- État du rapport de défaut
- Conclusions, recommandations et approbations
- Enjeux généraux (autres domaines affectés)
- Références (cas de test révélateur)

17 Outils de Support aux Tests

17.1 Pourquoi utiliser des outils ?

Objectifs :

— Améliorer l'efficacité (automatiser tâches répétitives)
— Améliorer la qualité (tests plus cohérents)
— Automatiser les activités impossibles manuellement
— Augmenter la fiabilité des tests

17.2 Classification des outils

17.2.1 Outils de gestion

— Outils de gestion des tests (exécuter, dépister, gérer)
— Outils de gestion des exigences (DOORS, QARespository)
— Outils de gestion d'incidents (suivi de défauts)
— Outils de gestion de configuration (stockage et versioning)

17.2.2 Outils de tests statiques

— Outils de revue (processus, checklists, rapports)
— Outils d'analyse statique (normes de codage, analyse de structures)
— Outils de modélisation (validation de modèles)

17.2.3 Outils de spécification

— Outils de conception de tests (génération de tests)
— Outils de préparation de données de test

17.2.4 Outils d'exécution

— Outils d'exécution des tests
— Harnais de tests / Framework de tests unitaires
— Comparateurs de tests
— Outils de mesure de couverture
— Outils de test de sécurité

17.2.5 Outils de performance

— Outils d'analyse dynamique (fuites mémoire)
— Outils de test de performance/charge/stress
— Outils de surveillance (monitoring)

17.3 Bénéfices et risques

Bénéfices :

— Amélioration de la gestion de l'information
— Réduction du travail répétitif
— Cohérence accrue
— Évaluation objective

— Facilité d'accès aux informations

Risques :

— Attentes irréalistes
— Sous-estimation du temps et coût d'introduction
— Sous-estimation de l'effort de maintenance
— Confiance excessive dans l'outil
— Négligence du contrôle de version
— Problèmes d'interopérabilité entre outils
— Risque de faillite de l'éditeur

17.4 Introduction d'un outil

Étapes :

1. Évaluation de la maturité de l'organisation
2. Évaluation selon des critères objectifs
3. Preuve de concept
4. Évaluation du vendeur
5. Identifier et planifier l'implémentation

Facteurs de succès :

— Étendre l'outil de façon incrémentale
— Adapter les processus (balance processus/outil)
— Fournir formation et assistance
— Surveiller l'utilisation et les bénéfices

Fin de la Partie II - ISTQB Foundation Level