

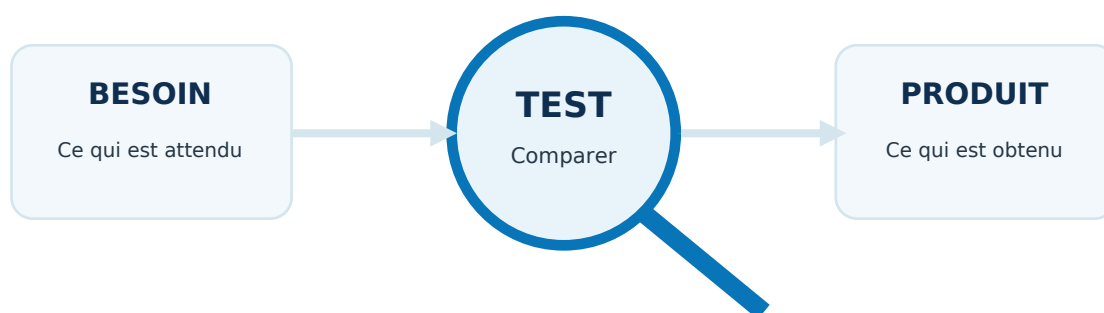


LES PRINCIPES DU TEST LOGICIEL

Guide visuel basé sur les fondamentaux ISTQB®

Des principes, des schémas et des exemples concrets pour comprendre rapidement comment organiser des tests efficaces dans un projet logiciel.

Tester = comparer l'attendu au résultat réel



Ce guide reprend et reformule les notions du programme ISTQB Certified Tester Foundation Level (CTFL) v4.0.1. Il ne remplace pas le syllabus officiel.

Les 7 principes en un coup d'œil

Ces principes sont des repères à garder en tête pendant toute la vie du projet. Ils ne constituent pas une suite d'étapes.

Les 7 principes : une lecture simple



1. Les tests montrent des défauts	Un test peut révéler un problème, mais ne prouve jamais l'absence totale de défaut.
2. Tout tester est impossible	Il faut sélectionner les tests les plus utiles selon le risque et l'usage.
3. Tester tôt	Une ambiguïté corrigée avant développement coûte moins cher qu'un défaut découvert à la fin.
4. Les défauts se concentrent	Certaines zones du produit sont beaucoup plus fragiles que d'autres.
5. Renouveler les tests	Rejouer toujours les mêmes scénarios finit par découvrir moins de nouveaux problèmes.
6. Adapter au contexte	On ne teste pas de la même façon un site vitrine, un e-commerce ou un système critique.
7. Répondre au besoin	Un logiciel peut être techniquement correct mais inutilisable ou inadapté au client.

Comprendre les 7 principes

Chaque principe tient dans un bloc compact : idée clé et exemple d'application.

PRINCIPE 1

Les tests montrent la présence de défauts, pas leur absence

Ne rien trouver ne signifie pas que le logiciel est parfait.

Exemple : Une campagne peut réussir à 100 % tout en laissant subsister un cas jamais testé.

PRINCIPE 2

Les tests exhaustifs sont impossibles

Toutes les combinaisons ne peuvent généralement pas être essayées.

Exemple : On choisit les cas importants, fréquents, sensibles et risqués.

PRINCIPE 3

Tester tôt permet d'économiser

Plus un défaut est détecté tôt, plus il est généralement simple à corriger.

Exemple : Relire le besoin et préparer les tests avant la livraison évite des corrections tardives.

PRINCIPE 4

Les défauts ont tendance à se concentrer

Une petite partie du produit contient souvent une grande partie des problèmes.

Exemple : Renforcer les tests sur les zones complexes, souvent modifiées ou déjà fragiles.

PRINCIPE 5

Répéter toujours les mêmes tests est moins efficace

Une campagne doit évoluer avec le produit.

Exemple : Ajouter de nouvelles données, de nouveaux parcours et de nouveaux scénarios de non-régression.

PRINCIPE 6

Les tests dépendent du contexte

La stratégie dépend du produit et de ses risques.

Exemple : Un site vitrine, un e-commerce et un logiciel médical n'ont pas les mêmes exigences.

PRINCIPE 7

L'absence de défaut ne garantit pas un bon produit

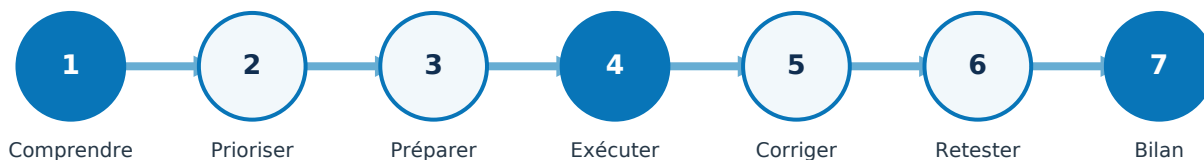
Un produit peut respecter sa spécification mais ne pas répondre au besoin réel.

Exemple : Tester aussi l'utilité, la facilité d'utilisation et le parcours utilisateur.

Le cycle de test

Le test est un processus continu. Les résultats servent à corriger le produit et à améliorer les campagnes suivantes.

Une boucle continue : apprendre → corriger → sécuriser



Comprendre	Que doit faire le produit ?
Prioriser	Qu'est-ce qui est le plus risqué ?
Préparer	Quels scénarios et quelles données ?
Exécuter	Comparer obtenu et attendu.
Corriger	Traiter les anomalies.
Retester	Vérifier correction et non-régression.
Bilan	Présenter résultats et risques restants.

Choisir des tests pertinents

Les techniques de conception permettent de réduire le nombre de cas tout en couvrant intelligemment les situations importantes.

Valeurs limites

Les erreurs apparaissent souvent juste avant, sur ou juste après une limite.

Exemple : âge minimum autorisé = 18 ans



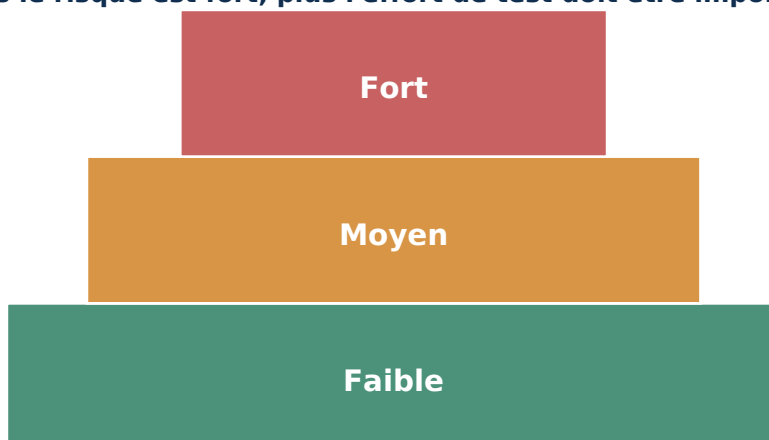
Autres techniques utiles

Partitions d'équivalence	Regrouper les valeurs qui devraient se comporter de la même manière.
Tables de décision	Représenter les résultats attendus lorsque plusieurs règles se combinent.
Transitions d'état	Vérifier les passages entre états : créée → payée → expédiée → livrée.
Tests basés sur l'expérience	Utiliser les défauts passés et l'expérience du testeur pour imaginer des cas supplémentaires.

Mettre l'effort au bon endroit

Quand le temps est limité, tous les éléments ne doivent pas recevoir le même effort de test.

Plus le risque est fort, plus l'effort de test doit être important



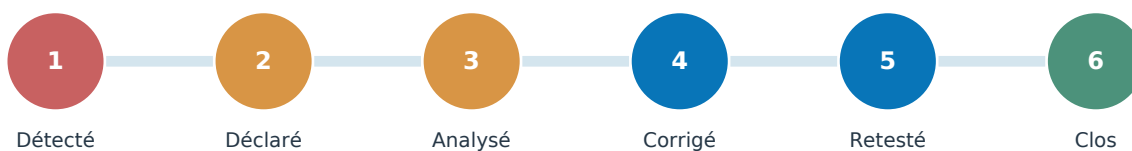
À tester en priorité

- Fonctions indispensables au métier.
- Paiement, données sensibles et parcours clients critiques.
- Zones nouvelles ou fortement modifiées.
- Fonctions ayant déjà connu beaucoup d'anomalies.
- Fonctions utilisées très fréquemment.

Le cycle d'une anomalie

Une anomalie doit pouvoir être comprise, reproduite, corrigée puis vérifiée.

Cycle simple d'une anomalie



Titre	Fonction concernée + problème observé.
Contexte	Version, environnement, navigateur ou appareil si utile.
Étapes	Actions réalisées dans l'ordre.
Attendu	Ce qui aurait dû se produire.
Obtenu	Ce qui s'est réellement produit.
Preuve	Capture, vidéo ou message d'erreur si nécessaire.
Impact	Conséquence du problème pour l'utilisateur ou le métier.

Les réflexes essentiels

Un aide-mémoire à conserver pendant la préparation et l'exécution des tests.

01 Tester le plus tôt possible

02 Prioriser selon les risques

03 Définir l'attendu avant le test

04 Tester aussi les limites et erreurs

05 Faire évoluer les scénarios

06 Décrire précisément les anomalies

07 Retester chaque correction

08 Faire de la non-régression

09 Adapter les tests au contexte

10 Toujours penser à l'utilisateur

À propos de ce guide

Document ICEQ de synthèse basé sur les concepts du syllabus ISTQB® Certified Tester Foundation Level v4.0.1. Ce guide est une reformulation pédagogique indépendante et ne remplace pas le syllabus officiel.