Chapitre 2-2
Pièges et patterns récurrents

1. Introduction

L'examen ISTQB Foundation Niveau v4.0 repose sur une compréhension approfondie des concepts, mais aussi sur la capacité à **analyser rapidement les questions** et à **éviter les pièges courants**. Ce chapitre vous propose des **stratégies concrètes et activables**, avec des **exemples précis**, pour maximiser vos chances de réussite.

Ces conseils vous permettront de :

- **décrypter la formulation des questions** et repérer les termes clés ;
- **utiliser des stratégies d'élimination** pour affiner vos choix de réponse ;
- **comprendre les erreurs récurrentes** et ne pas tomber dans les pièges typiques de l'examen.

Prenez le temps de lire attentivement cette partie avant de vous lancer dans la préparation des questions. Une bonne méthodologie est aussi importante que la connaissance des concepts pour réussir l'examen ISTQB Foundation Niveau v4.0.

2. Repérer les mots-clés dans la question

Certaines questions sont volontairement formulées de manière à vous **induire en erreur**. Il est donc essentiel de bien identifier les **mots-clés** qui déterminent la bonne réponse.

Exemple 1 - attention aux termes absolus

Quel énoncé est **toujours** exact concernant la portée des tests de régression ?

a) Les tests de régression n'ont **jamais** besoin d'être automatisés.

b) Les tests de régression doivent **exclusivement** couvrir les fonctionnalités récemment modifiées.

c) Les tests de régression sont **obligatoires** après chaque correctif de défaut.

d) Les tests de régression se concentrent généralement sur les zones à risque pour vérifier qu'aucune anomalie n'a été réintroduite.

Analyse

La réponse correcte est : d)

– Les mots-clés absolus (toujours, jamais, exclusivement, obligatoires) exigent une vigilance particulière.
– En pratique, il existe presque toujours des exceptions ou des nuances.
– Les réponses trop catégoriques sont souvent fausses dans le contexte des tests logiciels.

Autres mots piégeux

– « **Jamais** » → trop restrictif, souvent faux.
– « **Exclusivement** », « **uniquement** » → souvent incorrect, car il peut y avoir plusieurs méthodes valides.

« **Doit** », « **obligatoire** » → à vérifier, car il existe souvent plusieurs façons de faire.

Comment éviter ce piège ?

– **Soyez sceptique vis-à-vis des affirmations absolues**.
– **Recherchez des formulations plus nuancées** comme « dans la plupart des cas » ou « généralement ».

3. Lisez attentivement la consigne : certaines questions exigent plusieurs réponses

Lors de l'examen ISTQB, **toutes les questions ne demandent pas une seule réponse**. Certaines vous demandent explicitement de **sélectionner deux options**. Il est **essentiel de lire attentivement la consigne** : il n'y a pas de points partiels. Si vous ne cochez qu'une seule des deux bonnes réponses, ou si vous en cochez une bonne et une fausse, la question est comptée comme fausse et vous perdez la totalité de ses points.

Exemple 2 - attention au nombre de réponses demandées

Quels sont les éléments typiques que l'on trouve dans un plan de test selon le syllabus ISTQB v4.0 ? **Sélectionnez DEUX options**.

a) Le plan de formation annuel de l'équipe RH

© Editions ENI - All rights reserved

b) L'approche de test définissant les niveaux, les techniques et les critères d'entrée

c) Le référentiel des risques associés au produit et au projet

d) L'intégralité du code source de l'application

Analyse

Les bonnes réponses sont b) et c).

a) Faux → le plan de test ne contient pas d'éléments RH génériques, cela n'est pas pertinent.

b) Correct → l'approche de test est **un élément central** du plan de test (types, niveaux, critères…).

c) Correct → le référentiel des risques (produit et projet) fait partie du plan de test.

d) Faux → le code source ne fait pas partie du plan de test, on le trouve plutôt dans la configuration ou la documentation technique.

Comment éviter ce piège ?

- **Lisez toujours la consigne** : si elle précise « **Sélectionnez DEUX options** », vous devez en cocher deux, pas une seule.
- Repérez les mots-clés comme **« TROIS réponses »**, **« DEUX options »**, **« TOUTES celles qui s'appliquent »**, etc.
- **Ne sautez pas d'options** : en cas de doute, éliminez les réponses manifestement absurdes pour augmenter vos chances de succès.

4. Méfiez-vous des formulations négatives

Dans l'examen ISTQB, certaines questions ne vous demandent **pas ce qui est vrai**, mais plutôt **ce qui ne l'est pas**, **ce qui est le moins correct**, ou **ce qui constitue une exception**. Ces formulations **inversent le sens habituel**, et peuvent **entraîner une erreur de compréhension** si on lit trop vite.

Exemple 3 - attention à la formulation « EXCEPTÉ »

Toutes les affirmations suivantes concernant les tests de sécurité sont vraies, **EXCEPTÉ** :

a) Les tests de sécurité évaluent la vulnérabilité d'un système aux attaques.

b) Les tests de sécurité peuvent être réalisés lors des tests système et d'intégration.

c) Les tests de sécurité nécessitent toujours des outils automatisés pour être efficaces.

d) Les tests de sécurité peuvent inclure des tests d'intrusion.

Analyse

a) Faux → c'est une affirmation **vraie** : évaluer les vulnérabilités est un objectif principal des tests de sécurité.

b) Faux → c'est une affirmation **vraie** : les tests de sécurité sont effectivement réalisés à ces niveaux.

c) Correct → c'est l'**exception** recherchée. Les tests de sécurité peuvent être manuels ou automatisés selon les contextes.

d) Faux → c'est une affirmation **vraie** : les tests d'intrusion font partie intégrante des tests de sécurité.

Comment éviter ce piège ?

- **Surveiller les mots pièges** dans la consigne : « **PAS** », « **EXCEPTÉ** », « **LE MOINS** », « **TOUS SAUF** », etc.
- Reformuler la question dans ta tête pour bien la comprendre : « Lequel n'est **pas** un avantage = lequel est **un désavantage** ».
- Souligner mentalement ou sur brouillon le mot **négation** pour ne pas l'oublier en lisant les réponses.

5. Éliminer les réponses manifestement fausses

Dans un **QCM**, il est souvent possible d'éliminer immédiatement une ou deux réponses incorrectes, ce qui **augmente vos chances de trouver la bonne réponse**.

Exemple 4 - élimination des réponses absurdes

Quel est l'objectif principal des tests logiciels ?

a) Vérifier que le logiciel est 100 % exempt de défauts

b) Détecter les défauts et réduire les risques

c) Éviter d'avoir besoin de maintenance après la mise en production

d) Accélérer le développement des logiciels

Analyse

a) Faux → aucune méthode de test ne garantit un logiciel sans défaut.

b) Correct → l'un des principaux objectifs du test est bien **de détecter les défauts et réduire les risques**.

c) Faux → la maintenance est inévitable dans le cycle de vie d'un logiciel.

d) Faux → le test vise à assurer la qualité, pas nécessairement à accélérer le développement.

© Editions ENI - All rights reserved

Comment éviter ce piège ?

- **Repérez les réponses qui contredisent les fondamentaux du test logiciel**.
- **Ne choisissez pas une réponse qui va à l'encontre des principes ISTQB** (ex. : tests exhaustifs possibles, absence totale de défauts, etc.).

6. Analyser les questions à scénario et éviter les conclusions hâtives

Les questions à **scénario** décrivent souvent un contexte précis et demandent d'identifier la meilleure action à entreprendre. Il est crucial de **lire attentivement** l'énoncé avant de répondre.

Exemple 5 - question à scénario

Lors d'un test d'intégration, l'équipe détecte une anomalie liée à l'interface entre deux modules. Quelle est la meilleure action à entreprendre ?

a) Ignorer l'anomalie si l'un des modules fonctionne correctement

b) Signaler immédiatement l'anomalie et documenter les conditions du test

c) Corriger le code sans en informer l'équipe de test

d) Relancer les tests sans analyser les causes

Analyse

a) Faux → un défaut détecté doit toujours être signalé.

b) Correct → la bonne approche est de **documenter et signaler** l'anomalie.

c) Faux → modifier le code sans communication est une mauvaise pratique.

d) Faux → relancer un test sans comprendre la cause d'un défaut est inefficace.

Comment éviter ce piège ?

- **Prenez le temps d'analyser la situation avant de répondre**.
- **Ne vous précipitez pas sur la première réponse qui semble correcte** sans examiner les autres options.

7. Faire attention aux différences subtiles entre les réponses

Parfois, plusieurs réponses semblent correctes, mais **une seule couvre tous les aspects du problème**.

Exemple 6 - nuances dans les réponses

Quelle est la **meilleure** façon de réduire l'impact des défauts détectés tardivement ?

a) Utiliser les tests exploratoires en fin de projet

b) Appliquer des tests statiques et des revues dès les premières phases du développement

c) Exécuter les tests uniquement après le codage complet du logiciel

d) Ne pas exécuter de tests avant la mise en production

Analyse

a) Faux → les tests exploratoires sont utiles, mais **pas suffisants** pour détecter les défauts tôt. Ils arrivent **trop tard** pour prévenir les impacts majeurs.

b) Correct → l'ISTQB recommande les **tests précoces** comme les **revues et les tests statiques** pour réduire les risques. D'après le syllabus, le **principe n° 3 est clair : « Tester tôt économise du temps et de l'argent »**.

c) Faux → tester après le codage complet **augmente le risque de découvrir des erreurs tardivement**.

d) Faux → ne pas tester avant la mise en production est une stratégie risquée.

Comment éviter ce piège ?

- **Comparez chaque réponse avec les bonnes pratiques ISTQB**.
- Choisissez la réponse qui couvre **le plus large éventail de bonnes pratiques**.

8. Se méfier des réponses trop évidentes ou intuitives

Certaines questions sont conçues pour **tester la compréhension approfondie** des concepts. Les réponses les plus intuitives ne sont pas toujours correctes.

Le **piège courant** est de **choisir la première réponse qui semble correcte**sans lire toutes les options. Il peut y avoir une réponse plus nuancée.

Exemple 7 - intuition trompeuse

Les tests de régression sont principalement utilisés pour :

a) Vérifier que les nouvelles fonctionnalités fonctionnent correctement

b) Confirmer qu'une modification n'a pas introduit de nouveaux défauts

c) Tester les performances d'une application après une mise à jour

d) Identifier les besoins des utilisateurs

© Editions ENI - All rights reserved

Analyse

a) Faux → ce sont les **tests fonctionnels** qui valident les nouvelles fonctionnalités. Cette réponse peut vous sembler la bonne, car c'est la première.

b) Correct → le test de régression **s'assure qu'aucune modification n'a introduit de nouveaux défauts**.

c) Faux → ce sont les **tests de performance** qui mesurent l'impact d'une mise à jour.

d) Faux → l'identification des besoins est une tâche d'analyse métier.

Comment éviter ce piège ?

- **Ne choisissez pas la réponse qui « semble évidente » sans réflexion**.
- **Posez-vous la question :** cette réponse correspond-elle exactement à la définition ISTQB du concept testé ?

9. La relation de cause à effet : erreur -> défaut -> défaillance

La base de toute bonne réponse réside dans la compréhension claire de la relation entre ces trois termes. Rappelez-vous que l'**erreur est l'action humaine incorrecte**, le **défaut est la conséquence de cette erreur** dans un produit d'activité, et la **défaillance est la manifestation observable du défaut**.

Le **piège courant** est de **confondre un défaut avec une défaillance**. Une question pourrait par exemple présenter une situation où un code erroné (défaut) n'a pas encore provoqué de problème observable (défaillance). Il faut retenir que tous les défauts ne conduisent pas forcément à des défaillances.

Exemple 8 - erreur ou défaillance ?

Lors d'un projet de développement, un programmeur a mal compris une exigence et a implémenté un calcul incorrect dans le code source. Cependant, ce défaut n'a été détecté que plusieurs mois après la mise en production, lorsque les utilisateurs ont commencé à signaler des résultats erronés. Quelle est la séquence correcte des événements ?

a) Défaillance → défaut → erreur

b) Erreur → défaut → défaillance

c) Défaut → erreur → défaillance

d) Erreur → défaillance → défaut

Analyse

a) Faux → une défaillance ne peut pas précéder un défaut, car elle est sa conséquence.

b) Correct → une erreur humaine (mauvaise interprétation de l'exigence) conduit à un défaut dans le code. Ce défaut ne devient une défaillance qu'une fois le code exécuté en production.

c) Faux → un défaut est créé suite à une erreur humaine, pas l'inverse.

d) Faux → une défaillance ne peut pas apparaître avant que le défaut n'existe.

Comment éviter ce piège ?

- **Se rappeler que l'erreur est toujours une action humaine** (ex. : mauvaise compréhension d'une exigence).
- **Comprendre que le défaut est présent dans un produit (ex. : code, spécification), mais qu'il ne cause une défaillance que lorsqu'il est exécuté**.

10. Identifier le rôle de l'erreur humaine

Il est important de reconnaître que l'**erreur est toujours une action humaine**. Les questions qui mentionnent des « erreurs » feront souvent référence à une action incorrecte, un oubli, ou une mauvaise interprétation humaine.

Le **piège courant** est d'attribuer une « erreur » à la machine ou au système lui-même. Une question pourrait utiliser un vocabulaire trompeur laissant penser que le système a commis une erreur.

Exemple 9 - l'erreur humaine

Un testeur exécute un test et constate que l'application mobile affiche une mauvaise date pour les commandes validées. Quelle est la cause initiale probable de ce problème ?

a) Une erreur humaine lors de la configuration du serveur de base de données

b) Une défaillance dans l'affichage du navigateur

c) Une instabilité temporaire du système

d) Un problème matériel avec le serveur

Analyse

a) Correct → l'erreur humaine (ex. : mauvaise configuration) est la source du défaut.

b) Faux → une défaillance est une conséquence, mais ne peut pas être la cause initiale.

c) Faux → une instabilité temporaire ne crée pas un défaut permanent.

d) Faux → un problème matériel n'est pas une cause directe d'un défaut logiciel.

© Editions ENI - All rights reserved