Expert Team

ExperTeam

La Qualité Logicielle, c'est Nous

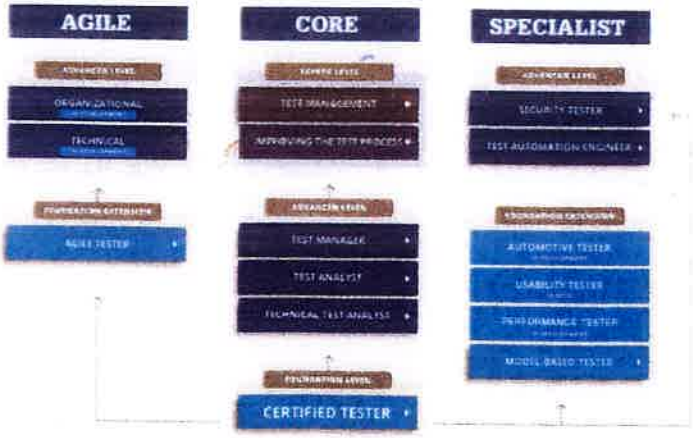
Réussir
La Certification ISTQB

Niveau Fondation

Nous contacter : contact@expert-team.tn Tel : 21 152 281, 00 33 6 68 42 27 66

Organigramme ISTQB

Préparez vous pour un nouveau départ dans votre carrière !



Expert Team

Examen de la certification ISTQB Niveau Fondation

- **Examen :**
  - L'examen c'est 40 QCM (1 seul choix correcte)
  - Chaque QCM vaut 1 point
  - L'examen dure une (1) heure
  - Le score minimum pour réussir est 65 %=26/40
  - 4 Niveaux de QCM : K1 (se souvenir : 20), K2 (comprendre : 12), K3 (utiliser) et K4 (analyser : 8)

	Temps (min)	Temps (%)	K1	K2	K3,K4
Chapitre 1	155	18	4	3	
Chapitre 2	115	13	4	2	
Chapitre 3	60	7	2	1	
Chapitre 4	285	33	4	2	6
Chapitre 5	170	20	3	3	2
Chapitre 6	80	9	3	1	
Total	865	100 %	20	12	8

1. Fondamentaux de test

- Pourquoi les tests sont-ils nécessaires
- Que sont les tests ?
- Les 7 principes généraux des tests
- Processus de test fondamental
- La psychologie des tests

Expert Team

1. Fondamentaux de test

- **Pourquoi les tests sont-ils nécessaires ?**
  - Les logiciels deviennent une part intégrante de notre existence
  - Un être humain peut faire une **erreur (méprise)** qui produit un **défaut (bug)** dans le code, dans un logiciel ou dans, un système, ou dans un document
  - Un **bug** dans un logiciel peut générer de nombreux **défaillances**
  - Contribuer à la **qualité** des systèmes logiciels
  - Réduire les **risques** d'occurrence de problèmes
  - Respecter les **exigences** légales ou contractuelles, et atteindre les normes industrielles spécifiques

Combien de tests sont suffisants?

Minimiser le risque des défauts

1. Fondamentaux de test

- **Quels sont les causes des défaillances ?**
  - L'être humain peut se tromper
  - Des délais serrés
  - La complexité du code et les infrastructures
  - Des modifications de technologies
  - De multiples interactions entre les systèmes.
  - Conditions d'environnement (radiations, magnétisme, champs électroniques et pollution)

Expert Team

1. Fondamentaux de test : 7 bugs connus

Erreur → Défaut → Défaillances → Pertes humaines et financières

7 Bugs informatiques catastrophiques

Erreur — Bug — Défaillances ou pertes

8

Expert Team

1. Fondamentaux de test

- Pourquoi tester ?

1. Trouver des défauts
2. Prévenir des défauts
3. Acquérir de la confiance par rapport au niveau de qualité
4. Fournir de l'information utile aux prises de décision

- Comment et quand tester ?

1. Avant l'exécution du système / logiciel : Tests statiques
2. Au cours de l'exécution du système / logiciel : Tests dynamiques
3. Clôture des activités tests : après l'exécution du système / logiciel

1. Fondamentaux de test : Processus de test

Expert Team

1. Fondamentaux de test: activités de tests

Activités de tester	Exemples	Avant exécution	En cours exécution	Après exécution
Planification des tests	Consolider le plan de test	X	X	
Contrôle des tests	Vérifier les critères d'entrée à chaque phase de test	X	X	X
Analyse et conceptions des tests Identification des conditions de test	Identifier les scénarios d'exception, utiliser des techniques de conception	X		
Implementation	Préparation de l'environnement de test Création des jeux de données	X		
Déroulements des tests	Exécution des tests systèmes Exécution des tests régressions Déclarer une anomalie		X	
Évaluation des critères de sortie	0 bug bloquants, 0 bugs majeurs, 5 normaux sur % couverture de code		X	X
Reporting avancement et état système en cours de test	Taux d'exécutions, cas de test passants échoués, non exécutés Taux de couverture des exigences et risques		X	X
La réalisation et la finalisation des activités de clôture définitive	Archivage des jeux de données Archivage des documents de test Réunion rétrospective			X
Revue de documents projet Revue du code source	Revue de l'architecture Analyse du code Revue d'un algorithme complexe	X		

1.7 Fondamentaux de test : 7 principes de test (K2)

1. Les tests montrent la présence **de défauts mais** ne peuvent pas en prouver l'absence.
2. **Les tests exhaustifs sont impossibles,** utiliser **l'analyse des risques et des** priorités pour focaliser les efforts de tests.
3. Tester tôt : les activités de tests devraient commencer aussitôt que possible dans le cycle de développement du logiciel ou du système
4. Regroupement des défauts, l'effort de test devrait être fixé proportionnellement à la densité des défauts prévus et constatés dans les différents modules.
5. **Paradoxe du pesticide** Si les mêmes tests sont répétés de nombreuses fois, il arrivera que les cas de tests ne trouvera plus de nouveaux défauts les cas de tests doivent être régulièrement revus et révisés
6. **Les tests dépendent du contexte** ; les logiciels de sécurité critique seront testés différemment à un site de commerce électronique.
7. **L'illusion de l'absence d'erreurs** : Trouver et corriger des défauts n'aide pas si le système conçu est inutilisable

Expert Team

1. Exercice : Qu'est ce qui n'est pas un principe de test :

1. Il est impossible de tout tester — un principe de Test
2. Le test est une activité qui dépend du cycle de développement logiciel et du domaine fonctionnel — Un principe (dépend du Con)
3. Le cahier de test doit être révisé et enrichi régulièrement
4. 20 % du code génère 80 % des bugs — regroupent → c'est un principe
5. L'effort de test dépend du nombre de cas de test — × c'est pas un principe
6. Le testeur intervient dès que le développeur livre le logiciel — × c'est pas un pri
7. Tester un système inutilisable est une perte de temps — c'est pas un principe
8. Un logiciel sans défaut n'existe pas — un principe.
9. Le rôle du testeur est de prouver qu'il n'y a plus de défaut dans un logiciel (Non) c'est pas un princip
10. Les bugs sont proportionnels au complexité du code (Oui)

c'est un principe (regroupent de défaut).

Oui
Oui
principe de Test (paradoxe pesticide)
l'effort de Test : les tâche pour

1. Fondamentaux de test : Processus de test

Analyser et concevoir	Implémentation et exécution	Évaluer les critères de sortie et informer	Activités de clôture des tests
• Réviser les bases du test • Évaluer la testabilité des exigences et du système • Identifier et prioriser les conditions de test • Concevoir et prioriser les tests de haut niveau • Identifier les données de test nécessaires • Concevoir l'initialisation de l'environnement • Créer une traçabilité bidirectionnelle entre les bases de test et les cas de test	• Finaliser, développer et prioriser les cas de test • Développer et prioriser les procédures de test • Créer des suites de tests à partir des procédures de test • Vérifier que les environnements de tests sont disponibles • Vérifier et mettre à jour la traçabilité bidirectionnelle • Exécuter les procédures de test soit manuellement ou d'une manière automatique	• Vérifier les registres de tests en fonction des critères de sortie spécifiés dans la planification des tests • Évaluer si des tests supplémentaires sont requis ou si les critères de sortie doivent être changés • Écrire un rapport de synthèse des tests pour les parties prenantes	• Vérifier quels livrables prévus ont été livrés • Clôturer les rapports d'incidents ou créer des demandes d'évolution • Documenter l'acceptation du système • Finaliser et archiver • Fournir les testwares à l'organisation • Analyser les leçons apprises • Utiliser l'information collectée pour améliorer la maturité des tests

Suite

les donne le spec

matrice de Couverture de exigence

Expert Team

1. Fondamentaux de test : psychologie de test

- Tests indépendants :
  - Peuvent être effectués à n'importe quel niveau des tests par des testeurs entraînés et professionnels
  - Tests conçus par une (des) autre(s) développeur (s)
  - Tests conçus par une (des) personne(s) d'un groupe différent au sein de la même organisation
  - Tests conçus par une (des) personne(s) d'une organisation ou d'une société différente
- Spécifier clairement les objectifs des tests.
- Les tests sont souvent vus comme une activité destructrice
- Des problèmes de communication peuvent survenir particulièrement si les testeurs sont vus uniquement comme messagers porteurs de mauvaises nouvelles concernant des défauts

1. Comment améliorer la communication et les relations entre les testeurs et leurs interlocuteurs ?

1. Commencer par une collaboration plutôt que par des conflits
2. Rappeler à chacun l'objectif commun de systèmes de meilleure qualité
3. Communiquer les découvertes sur le produit de façon neutre et factuelle sans critiquer la personne responsable
4. Essayer de comprendre ce que ressent une autre personne et pourquoi elle réagit comme elle le fait
5. Confirmer que l'autre personne a compris ce que l'on a dit et vice versa

Chapitre 2.

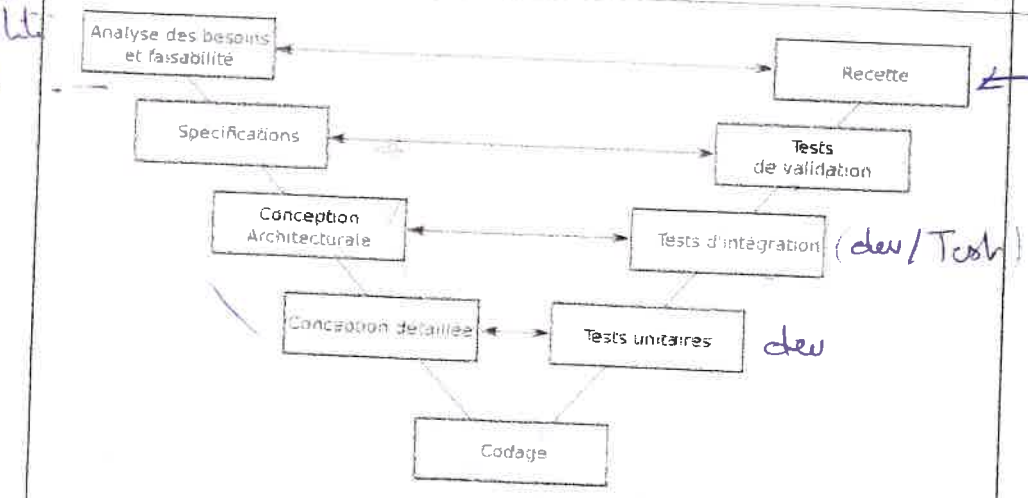
2. Tester Pendant le Cycle de Vie Logiciel

- Les modèles de développement logiciel (K2)
- Niveaux de tests (K2)
- Types de tests: les cibles de tests (K2)
- Tests de maintenance (K2)

17

Temps

2. Tester Pendant le cycle de vie Logiciel : modèle en V



18

Il nous donne la possibilité de Tester tôt

Test d'acceptation

(dev / Test)

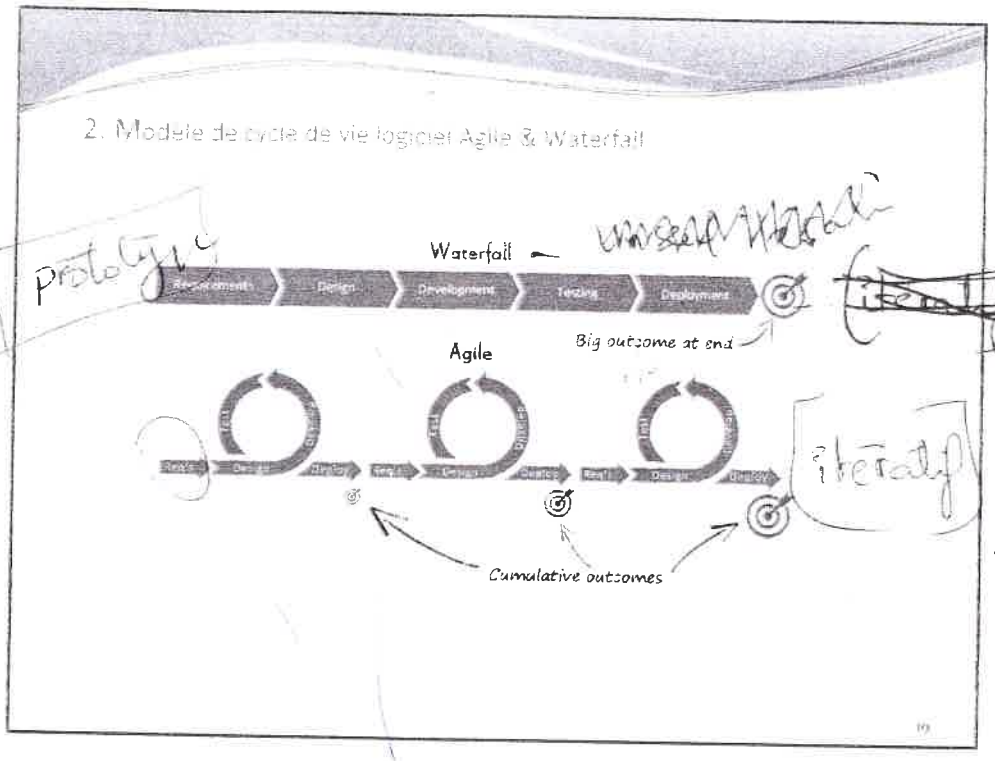
dev

Départ

Expert Team

User story = fonctionnalité ticket.

Expert Team

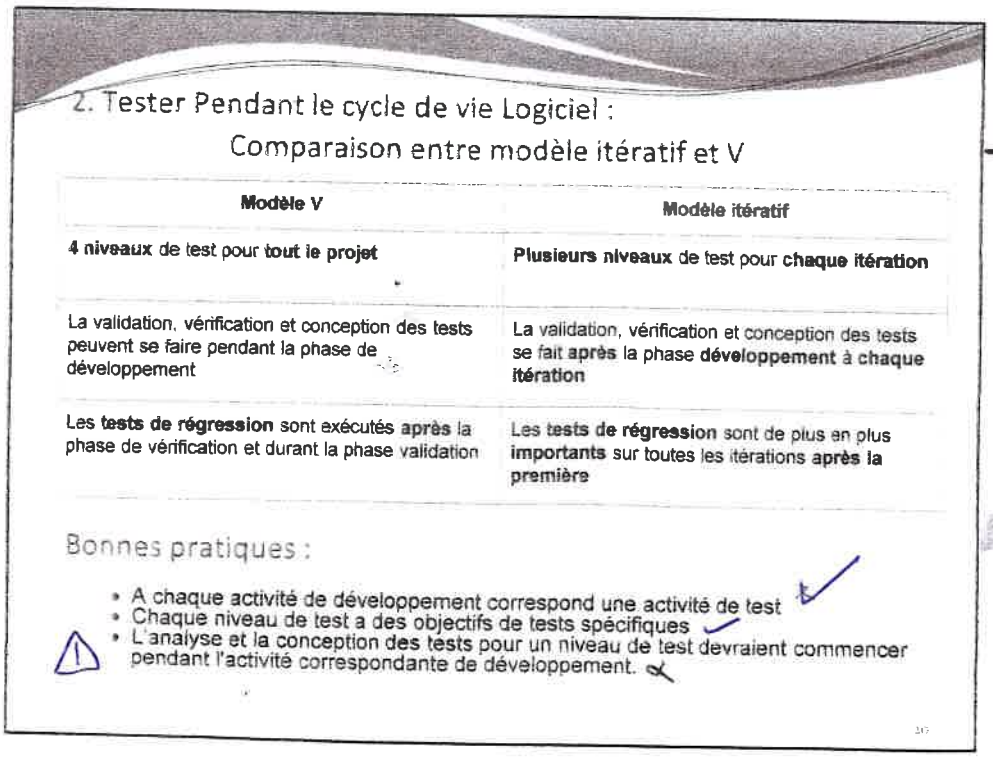


Prototype

itératif

test réel : T. détaillé

T logique : Tests fonctionnels



Modèle V	Modèle itératif
4 niveaux de test pour **tout le projet**	**Plusieurs niveaux** de test pour **chaque itération**
La validation, vérification et conception des tests peuvent se faire pendant la phase de développement	La validation, vérification et conception des tests se fait **après** la phase **développement** à **chaque itération**
Les **tests de régression** sont exécutés **après** la phase de vérification et durant la phase validation	Les **tests de régression** sont de plus en plus **importants** sur toutes les itérations **après la première**

Bonnes pratiques :

- A chaque activité de développement correspond une activité de test
- Chaque niveau de test a des objectifs de tests spécifiques
- L'analyse et la conception des tests pour un niveau de test devraient commencer pendant l'activité correspondante de développement.

Vérification

- Vérifier fonctionnalité par fonctionnalité

Validation

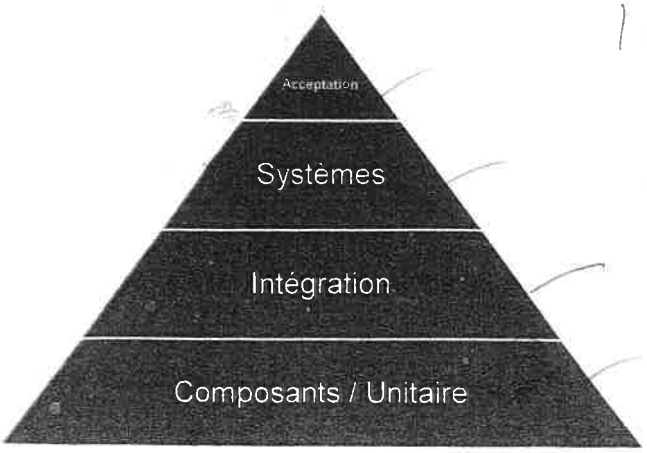
fait par l'Utilisateur pour Avoir Confiance

Expert Team

2. Tester Pendant le cycle de vie Logiciel : Niveaux de test

- Pour chaque niveau de test, on peut identifier :
  - **Les objectifs de test**
  - **Les bases de test (documents ou livrables de référence pour dériver les cas de test)**
  - **Les objets de tests (ce qui est testé)**
  - **Les défauts et les défaillances typiques à trouver**
  - **Les outils** Quel Outil ??
  - **les approches**
  - **Les responsabilités spécifiques aux niveaux.**

2. Tester pendant le cycle de vie Logiciel : Niveaux de test



Pyramide des tests

module = Unitaire = T. Composant : T. fonct. de Composant

IHM = respect de la charte graphique

Expert Team

Developper

T. Unitaire

2.2.1 Niveaux de test : Tests de composants (Test unitaire = Test module)

- Ils peuvent inclure les tests fonctionnels, non fonctionnels(maintenabilité, **robustesse**, et **tests structurels)** Test Structure interne = T. boîte blanche
- ils se font par façon isolée par rapport au reste du système en utilisant des **bouchons**, **simulateurs** et **pilotes**
- Les tests de composants se font par le **développeur** avec l'accès au code et tester sur un environnement de développement soit en **framework TU** ou débogage
- Une approche des tests de composants est de préparer et automatiser des cas de tests avant le codage : BDD, TDD, Test First, Tester d'abord

BDD = behavior Driven development.

TDD = Test Driven

Exercice : Tests fonctionnels et non fonctionnels (selon ISO 9126)

Types de Tests	Tests fonctionnels (Quoi) Je teste	Tests non fonctionnels (Comment) Je teste
Charge		x
Utilisabilité, nb Clic		x
Interopérabilité	x	
Sécurité	x	
Fiabilité	Capacité d'un Système - taux d'erreur accepté	x
Performance	Clic Temps de Réponse	x
Maintenabilité	Capacité d'un Système	x
Charge		x
Portabilité	JVM	x
Accessibilité		x

Donné informatisé

Expert Team

Unit_ Integ - Sys - Acc ...

2.2.1 Niveaux de test : Tests d'intégration

- Ils testent les interfaces entre :
  - les composants (tests d'intégration composants)
  - les systèmes (test intégration système)
  - les différentes parties d'un système (Ex. matériel, logiciel dans les systèmes embarqués)
- L'intégration devrait normalement être **incrémentale** (**top-down ou bottom-up**) plutôt qu'être effectuée en une fois ('**big bang**').
- Les testeurs devraient comprendre l'architecture et influencer le planning d'intégration

- Les tests de caractéristiques non-fonctionnelles particulières (p.ex. performances) peuvent être inclus dans les tests d'intégration

2.2.1 Niveaux de test : Tests système

Test fonctionnalité

- Les tests systèmes traitent le comportement d'un système/produit complet.
- L'environnement de test devrait correspondre à la cible finale
- Les tests système peuvent inclure des tests basés sur les risques et/ou sur les spécifications et les exigences, les processus commerciaux, les cas d'utilisation
- Les tests système devraient examiner à la fois les exigences fonctionnelles et les non-fonctionnelles du système ainsi que la qualité des données.

T. Accep : Je teste l'appareil complet process métier

bases de test

1) ex fon
2) ex Non fon
3) Qualité données

T. Systè

Expert Team

2.2.1 Niveaux de test : Tests d'acceptation

Valider le bon fonctionnement du système

- Les tests d'acceptation relèvent souvent de la responsabilité des clients ou utilisateurs qui veulent avoir confiance dans le système et évaluer si le système est prêt à être déployer et utiliser
- Tests d'acceptation utilisateur : vérifie l'aptitude et l'utilisabilité du système (Process métier).
- Tests (d'acceptation) opérationnelle (la prod).
  - **Tests des backups et restaurations**
  - **Reprise après incidents**
  - **Gestion des utilisateurs;**
  - **Tâches de maintenance;**
  - **Chargements de données et tâches de migration**
  - **Vérification périodique des vulnérabilités de sécurité**

test environnement de production

2.2.1 Niveaux de test : Tests d'acceptation

- Les **tests d'acceptation réglementaires** ou contractuelles sont exécutés par rapport aux règlements et législations qui doivent être respectés, telles que les obligations légales, gouvernementales ou de sécurité.
- **Les Alpha tests (tests d'acceptation usine)** sont exécutés sur le site de l'organisation effectuant le développement mais pas par les équipes de développement.
- **Les Bêta tests (tests d'acceptation sur site)** ou tests sur le terrain sont exécutés par des personnes sur leurs propres sites.

fait par l'Utilisateur Client finale, sur le site de dev.

acceptation	- Rapport analyse des risques • Processus métier • Cas d'utilisation • Exigence sys • Exigence utilisateur	
système	• Rapport analyse des risques. • spec fonctionnelles • cas utilisation • spec exigence.	
intégration	• Cas d'utilisation • workflows • Architecture. • Conception sys	
unitaire	• code. • Conception détaillé • Exigence Comp	
	Base de test	

Expert Team

2. Tester Pendant le cycle de vie Logiciel : Niveaux de tests

Niveaux de test	Bases de test	Objets de test	Types défauts (Bug)
Unitaire	- Exigences des composants - Conception détaillées - Code	- Composants testés - Programmes - Module de base de données	- défauts dans les modules, programmes, objets, classes, etc.) qui sont testables séparément - comportement des ressources
Intégration	- Conception du système - Architecture - Workflows - Cas d'utilisation	- Sous-systèmes Implémentation de bases de données Infrastructure Interfaces - Configuration système et données de configuration	- envoi ou réception des données incorrectes - interaction /communication entre différentes parties du système non fonctionnel - interaction /communication entre deux ou plusieurs systèmes non fonctionnels - Temps de réponse inacceptable - Ordre d'intégration mal planifié

2. Tester Pendant le cycle de vie Logiciel : Comparaison entre Niveaux de tests

Niveaux de test	Base de test	Objets de test	Type défaut
Tests systèmes	- Spécifications d'exigences - Cas d'utilisation - Spécifications fonctionnelles - Rapports d'analyse des risques	- Configuration système et données de configuration - Utilisation du système manuel et opérationnel	- déviation du comportement du système par rapport au base de test - interactions avec le système d'exploitation et les ressources système
Tests d'acceptation	- Exigences utilisateur - Exigences du système - Cas d'utilisation - Processus métier - Rapports d'analyse des risques	Processus métier sur l'intégralité du système Processus opérationnels de maintenance Procédures utilisateur Formulaires Rapports	- évaluer si le système est prêt à être déployé et utilisé - vulnérabilités de sécurité. - données non migrées correctement - déviation par rapport au règlement et législation - problèmes backup ou reprise

Expert Team

2.3 Tester pendant le cycle de vie Logiciel : Types de test

- Les tests **fonctionnels** concernent le comportement extérieur du logiciel, chose que le système **fait**
- Les tests **non-fonctionnels** concernent l'aspect extérieur et évaluent "**comment**" le système fonctionne
- Les tests **structurels** (boîte blanche) peuvent être effectués à tous les niveaux de tests mais spécialement dans les tests dc composants et les tests d'intégration de composants pour augmenter la couverture
- Tests **liés au changement** : tests de confirmation et de régression
- Tests de **maintenance**: sont effectués sur un système opérationnel existant.

2.3 Tester pendant le cycle de vie Logiciel : Types de test

- **Tests liés au changement :**

1. **Tests de confirmation** : Quand un défaut est détecté et corrigé, le logiciel devrait être ré-testé pour confirmer que le défaut original a été correctement ôté.
2. **Tests de régression** : Ré-exécuter à tous les niveaux des tests sur un programme pour identifier tout nouveau défaut dû à un / des changement(s) : logiciel, ou son environnement. ils s'appliquent aux tests fonctionnels, non-fonctionnels et structurels. Ils sont de bons candidats à l'automatisation.

Expert Team

2.3 Tester pendant le cycle de vie Logiciel : Types de test

3. **Tests de maintenance :** sont effectués sur un système opérationnel existant et sont déclenchés par :
   - **des modifications** :
     - évolutions planifiées
     - modifications correctives et d'urgence (hotfix)
     - changements d'environnements (base de données, systèmes d'exploitation, COTS)
     - patchs de correction des vulnérabilités de sécurité potentielles
   - **des migrations (conversion)**
     - les tests opérationnels du nouvel environnement
     - les tests des modifications du logiciel
   - **une suppression de logiciels ou de systèmes**
     - les tests de migration des données ou leur archivage

33

Exercices de révision

1. Différences entre erreur, méprise, défaut, bug, défaillance, échec ?
2. Qui fait le débogage ? developpe
3. Quels sont les 4 objectifs de test ? trouver les défaut, Contribu la Q de logicl
4. Quel est la différence entre test de confirmation et tests régressions ?
5. Quels sont les 7 principes de tests ?
6. Dans un processus de test, qu'est ce qui est une activité transverse ?
7. Qu'est ce qu'un bouchon ?
8. Qu'est ce qu'un pilote ?
9. Quand est ce qu'on exécute les tests fonctionnels ?

ds tous les niveaux.

- Augmter la Confiance
- prise de décisio
- prevenir des défaut

34

Expert Team

Exercices de révision

1. Citez 04 exemples de bonne pratique pour améliorer la communication entre testeurs et leurs interlocuteurs ?
2. Quel relation entre cycle de développement et test ?
3. Quelle est la différence entre cycle itératif et cycle en V ?
4. Citez un exemple de base de test pour chaque niveau de test ?
5. A quoi sert chaque niveau de test selon ISTQB ?
6. Quel est la différence entre tests dynamiques et tests statiques ?
7. Quel est la différence entre vérifier et valider un logiciel ?
8. Que signifie COTS ?

Chapitre 3 : Techniques statiques

- **Techniques statiques et processus de test (K2)**
- **Processus de revue (K2)**
  - **responsabilités**
  - **Formel**
  - **Informel**
  - **Revue Technique**
  - **Relecture Technique**
  - **Inspection**
  - **Facteurs de réussite**
- **Analyse statique outillée (K2)**
- **Exercices**

Expert Team

3. Techniques statiques et processus de test (K2)

- Test **dynamique**, qui exige l'**exécution** du logiciel,
- Tests **statiques** reposent sur **l'examen manuel** (revues) ou l'analyse du code ou de la documentation du projet sans exécution du code
- Les **défauts** détectés pendant les revues effectuées tôt dans le cycle de vie sont souvent bien **moins coûteux**
- Une revue pourrait être effectuée **manuellement** ou **outillée** exp(revue de code)
- Tout produit logiciel peut être revu : **exigences, les spécifications, les spécifications de conception, le code, les plans de test, les spécifications de test, les cas de test, les scripts de test, les guides utilisateur ou pages web**

pas d'execution

Réunion = Revue

Revue = Réunion avec des personne qualifié.
= Apprendre
et trouver des incohérence, et les règles de gestion

3. Processus de revue : formel

1. **Planification :**
   - Définir les critères de revues
   - Choisir le personnel
   - Allouer les rôles
   - Définition des critères d'entrée et de sortie pour des types de revues plus formels (p.ex., inspections)
   - Sélectionner la partie des documents à revoir
   - Vérification des critères d'entrée
2. **Lancement:**
   - Distribuer les documents par mail (préparation de la Revue).
   - Expliquer les objectifs, le processus et les documents aux participants
3. **Préparation individuelle**
   - Préparer la réunion de revue en revoyant le(s) document(s)
   - Ecriture des défauts potentiels, questions et commentaires

dans les Cas des logiciel Aéronautique

38

Expert Team

3.Processus de revue (K2)

4. **Examen/évaluation/enregistrement des résultats (réunion de revue):**
   - Discuter ou enregistrer, avec des résultats ou minutes documentés
   - Noter les défauts, faire des recommandations concernant le traitement des défauts, prendre des décisions à propos des défauts
   - Examen/évaluation et enregistrement pendant toutes les réunions physiques ou enregistrement de toutes les communications électroniques
5. **Ré travail**
   - Correction des défauts détectés (réalisé généralement par l'auteur)
   - Enregistrer le statut modifié des défauts
6. **Suivi:**
   - Vérifier que les défauts ont bien été traités
   - Récolter les métriques
   - Contrôle sur la base des critères de sorties

39

3. Processus de revue : responsabilités

Rôle	Responsabilités
Manager	1. décide l'exécution des revues 2. alloue le temps dans la planification du projet 3. détermine si les objectifs de revue ont été atteints
Modérateur	1. dirige la revue des documents (planification de la revue, l'exécution de la revue, et le suivi post-réunion) 2. intermédiaire entre les différents points de vue et est souvent la personne qui repose le succès d'une revue.
Auteur	1. L'auteur du document 2. la personne à qui incombe la responsabilité principale du ou des document(s) à revoir
Réviseurs : Vérificateurs inspecteurs	1. les individus avec une culture technique ou métier spécifique qui identifient et décrivent les constatations (p.ex. défauts) dans le produit en cours de revue
Scribe (ou greffier)	1. documente tous les aspects, problèmes et points ouverts identifiés pendant la réunion.

40

Expert Team

3. Processus de revue : types de revue

Type de revue	Caractéristiques
Revue informelle	›Pas de processus formel ›Peut inclure la programmation par paires ou une revue de conception et de code par un responsable technique ›Les résultats peuvent être documentés ›Peut varier en utilité selon les réviseurs ›Objectif principal : manière bon marché d'obtenir des résultats
Relecture technique	›Réunion dirigée par l'auteur; ›Peut prendre la forme de scénarios, répétitions à blanc, participation de groupes de pairs; ›Sessions sans limite de durée ›Optionnellement une réunion de préparation de revue par les réviseurs ›Optionnellement préparation d'un rapport de revue incluant une liste de constatations ›Optionnellement un scribe (qui n'est pas l'auteur) ; ›Varie en pratique de quasiment informel à très formel; ›Objectifs principaux: apprendre, gagner en compréhension, trouver des défauts.

3. Processus de revue : types de revue

Type de revue	Caractéristiques
Revue technique	›Documentée, processus de détection de défauts défini incluant des pairs et des experts techniques avec optionnellement la participation de l'encadrement ›Peut être effectuée comme une revue de pairs sans participation de l'encadrement › Idéalement dirigée par un modérateur formé (pas l'auteur) ›Réunion de préparation par les réviseurs ›Peut optionnellement utiliser des check-lists
Inspection	›Dirigée par un modérateur formé (pas l'auteur) ›Généralement menée comme un examen par les pairs ›Rôles définis ›Inclut des métriques ›Processus formel basé sur des règles et des check-lists ›Critères d'entrée et de sortie spécifiés pour l'acceptation du produit logiciel ›Réunion de préparation ›Rapport d'inspection incluant la liste de constatations; ›Processus formel de suivi (inclut le processus facultatif d'amélioration des composants) ›Lecteur (facultatif) ›Objectif principal: trouver des défauts

Expert Team

Processus de revue : facteurs de réussite

- Chaque revue a des objectifs prédéfinis et clairs.
- Les personnes impliquées sont adéquates pour les objectifs de la revue.
- Les testeurs sont des réviseurs de valeur qui contribuent à la revue et ainsi prennent connaissance du produit afin de pouvoir préparer les tests plus tôt.
- Les défauts trouvés sont bien acceptés, et exprimés objectivement.
- Les aspects personnels et psychologiques sont traités (p.ex. en faisant de cela une expérience positive pour l'auteur)
- La revue est menée dans une atmosphère de confiance ; les résultats ne sont pas utilisés pour évaluer les participants
- Les techniques de revue adaptées aux objectifs, adaptées aux types et au niveau de livrable logiciel, et adaptées aux types et niveau des réviseurs, sont appliquées.
- Des check-lists ou des rôles sont utilisés lorsque cela est approprié, afin d'augmenter l'efficacité de détection des défauts.
- Des formations sont données sur les techniques de revue, en particulier celles concernant les techniques plus formelles telles que les inspections.
- L'encadrement supporte un bon processus de revue (p.ex. en incorporant du temps pour les activités de revue dans les plannings des projets).
- L'accent est mis sur l'apprentissage et l'amélioration du processus.

43

Analyse statique avec des outils : valeur ajoutée

- La détection très tôt de défauts avant l'exécution des tests.
- Une information très tôt sur certains aspects suspects du code ou de la conception, par le calcul de métriques, par exemple une mesure de complexité élevée.
- L'identification de défauts difficilement détectables par des tests dynamiques.
- La détection de dépendances et des inconsistances dans les modèles logiciels tels que des liens dans les modèles logiciels
- L'amélioration de la maintenabilité du code et de la conception.
- La prévention des défauts, si les leçons sont prises en compte lors du développement

44

Expert Team

Analyse statique avec des outils : défauts typiques

- Référencement d'une variable avec une valeur indéfinie
- Interface inconsistante entre modules et composants;
- Variables qui ne sont jamais utilisées ou déclarées de façon incorrecte;
- Code inaccessible (code mort)
- Logique absente et erronée (potentiellement des boucles infinies)
- Constructions trop compliquées : complexité de code
- Violation des standards de programmation pb sécurité
- Vulnérabilités de sécurité;
- Violation de syntaxe dans le code et les modèles logiciels

Analyse Statique → sur le code
Avec un outil

revue st → manuel.

Questions de révisions

1. Quels sont les types de revue ? informel, revu tech, relecture, inspect
2. Quels sont les rôles présents dans une revue formelle ?
3. Quels rôles présents dans une revue informelle ?
4. Quel est le rôle du manager dans une revue ?
5. Quel type de défaut l'outil d'analyse statique peut détecter ? Ne peut pas détecter ?
6. Durant quelle phase de test, une revue est actée ?
7. Citez 03 causes d'échec d'une revue ?
8. Quel est la différence entre relecture et revue technique ?

Expert Team

4. Techniques des conceptions de tests

1. Le processus de développement de test
1. Catégories de techniques de conception de tests
1. Techniques basées sur les spécifications ou boîte noire
1. Techniques basées sur la structure ou boîte blanche
1. Techniques basées sur l'expérience
1. Sélectionner les techniques de test

47

4.1 Le processus de développement de test

- Le processus de développement de test dépends de la maturité des tests et des processus de développement, des contraintes de temps, des exigences de sûreté de fonctionnement ou réglementaires et des personnes impliquées :

1. Identifier les conditions de test.
2. Etablir la traçabilité des conditions de tests
3. Sélectionner des techniques de tests
4. Spécifier et créer les cas de test et données de test

Analyse et Conception

Créer Cas de Test.

48

Expert Team

Standard pour la Documentation de Tests Logiciel (IEEE 829-1998)

- Spécifications des cas de test
  1. Identifiant du CT
  2. Documents Références
  3. Contraintes spécifiques
  4. Environnement requis
  5. Dépendance avec d'autres CT
  6. valeurs d'entrée
  7. Préconditions
  8. résultats attendus
  9. post-conditions

Analyse et conception

- Spécifications de conception de tests
  1. Identifiant du CT
  2. Caractéristiques à tester
  3. Approche de tests
  4. Identification des Cas de tests
  5. Critères de succès et échec

- Spécifications de procédure de test
  1. Identifiant du CT
  2. Objectifs
  3. Contraintes spécifiques
  4. Etapes

Implémentation

4.2 Catégories de techniques de conception de tests (K2)

Techniques de conception	Caractéristiques	Exemples
Technique de conception boîte noire (techniques de conception basées sur les spécifications)	• Se basant sur une analyse de la documentation de la base des tests • Les cas de test sont dérivés de façon systématique depuis des modèles	1. Partitions d'équivalence 2 Analyse des valeurs limites 3. Tables de décisions 4. Transition d'états 5. Table de transition d'état 6. Cas d'utilisation
Technique de conception boîte blanche (techniques structurelles, basées sur les structures)	• Les cas de test sont dérivés de la structure interne du système • Le niveau de couverture du logiciel peut être mesuré à partir de cas de tests	1. Tests des décisions 2. Tests des instructions 3. Tests des conditions 4. Tests des conditions multiples
Technique de conception basée sur l'expérience	• Les cas de test sont dérivés de la connaissance et expérience des testeurs, développeurs, utilisateurs et autres parties prenantes • Les cas de test sont dérivés de la connaissance des défauts possibles et de leur distribution	1. L'estimation d'erreur (attaque par faute) 2. Tests exploratoires

Expert Team

4.3 Techniques de conception boîte noire

Techniques	Domaines d'application	Exemples
Partitions d'équivalence → Même Cas de test pour un traitement identique	• tous les niveaux de tests • entrées humaines • aux entrées via le système • paramètres d'interface des tests d'intégration	1. Tranches d'âge pour la présidentielle 2 Saisie valides et invalides 3 Tranches d'âge pour assurance de vie
Analyse des valeurs limites → comportement aux bornes de chaque partition d'équivalence	• Valeur limites des classes d'équivalence • Données d'entrées humaines ou écran	1. Limite de rapidité de transactions 2. numéros de commande sur un système de contrôle des stocks 3. Détection dépassement vitesse
Tests par tables de décisions	• Toutes les situations quand l'action du logiciel dépend de plusieurs décisions logiques.	1 Valider ou non un candidat à la présidentielle selon âge, nationalité, origine, santé, b3 2. Cas d'orientation : score, option demandée, nombre de places prévues

51

4.3 Techniques de conception boîte noire

Technique	Domaines d'application	Exemples
Test de transition d'états (visualiser le logiciel en termes d'états,	• L'industrie du logiciel embarqué L'automatisation technique • Modéliser les objets métier possédant des états spécifiques • Tester les dialogues d'interfaces écran	• État d'un moteur (Arrêt, Allumé) • État de validation d'une commande (créée, en attente d paiement, payé, expédié, livré, annulé..)
Tests de cas d'utilisation (décrit l'interaction entre acteurs)	• les défauts dans le flux de traitement pendant l'utilisation réelle du système • concevoir des tests d'acceptation avec la participation du client/utilisateur • des défauts d'intégration causés par l'interaction et les interférences entre divers composants	• Scénario de commande • Scénario de vente en ligne • Scénario de réservation hôtel

52

Expert Team

4.3.1 Partitions d'équivalences

- Le comportement au **bord de chaque partition** d'équivalence risque plus d'être incorrect qu'à l'intérieur, donc les **limites** sont des zones plus propices pour **découvrir des défauts.**

53

4.3.1 Analyse des valeurs limites

- Le comportement au **bord de chaque partition** d'équivalence risque plus d'être incorrect qu'à l'intérieur, donc les **limites** sont des zones plus propices pour **découvrir des défauts.**

54

Expert Team

Table de décision : exercice

- Conditions de candidature à la présidentielle

condition	1	2	3	4	5	6	7	8
Age>32	T	F	T	F	T	F	T	F
Tunisien	T	T	F	F	T	T	F	F
Vote	T	T	T	T	F	F	F	F
Acceptation	Oui	Non	Non	Non	Non	Non	Non	Non

55

Table de décision : exercice

Avantages accordés à un nouveau client :

- **nouveau client : 15 %**
- **carte fidélité : 10 %**
- **coupon 20 %**

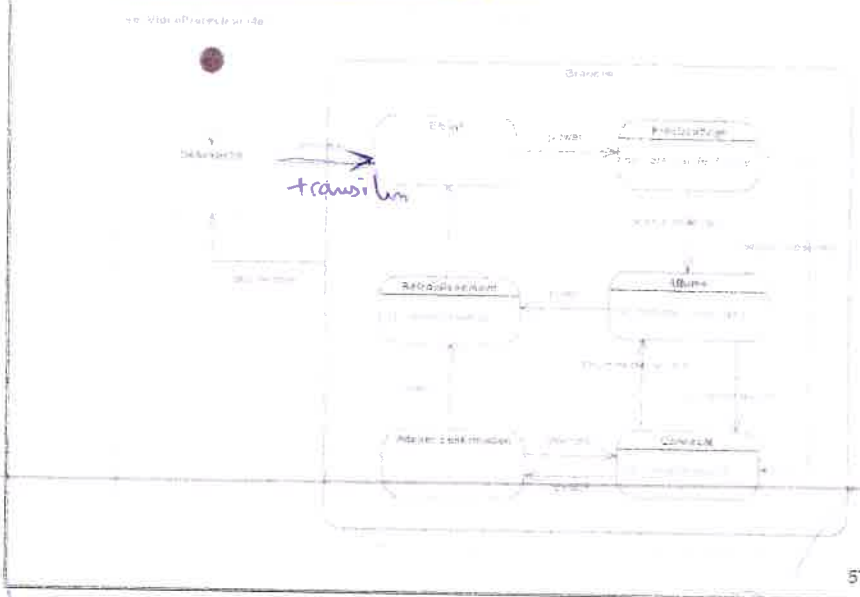
Un seul avantage peut être appliqué à la fois.

Quelle remise est appliquée ?

56

Expert Team

4.1 Diagramme de transition d'états



57

12 transition Valide

12 Cas de Te Valide = 84

(Nb état x nbr transit - nbr trans Valide

84 = nbr Tot de Cas de Test (Valide + invalide)

7 x 12 = 84 - 12 = 72.

4.1 Exemple de Table de transition d'état

	Prochaine état	Transition valide
Débranché	Éteint	brancher
Eteint	Préchauffage	power
Refroidissement	Éteint	Lampe refroidi
Préchauffage	Allumé	Source absente
Allumé	connecté	connecter source

Nombre de cas de test total = *nombre d'états* * *nombre de transition (incluant transitions valides et invalides)*

Pour notre exemple :

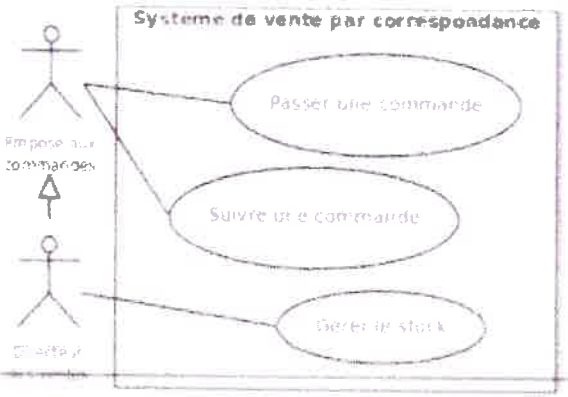
- **Nb Transitions : 12**
- Nombre des états : 7
- Cas de tests (transitions valides et invalides) : 84

58

Switch (n) ^ nbr de transition

Expert Team

Diagramme de cas d'utilisation



59

4. Techniques des conceptions de tests : boîte blanche

Niveau de composant: la structure d'un composant logiciel c'est à dire instructions, décisions, branches ou même des chemins distincts

Niveau d'intégration: la structure peut être un arbre (ou graphe) d'appel (un diagramme où des modules appellent d'autres modules).

Niveau système: la structure peut être une structure de menus, des processus métier ou la structure d'une page web.

60

Expert Team

4.1 Techniques de conception structurels

- Test des instructions et couverture
  - exécuter des instructions spécifiques pour accroître la couverture des instructions
- Test des décisions et couverture
  - liées aux tests de branches
  - une couverture de 100% des décisions garantit une couverture à 100% des instructions, mais l'inverse n'est pas vrai
- Autres techniques basées sur les structures
  - couvertures de conditions
  - les couvertures de conditions multiples

61

4. Techniques basées sur l'expérience

- Les tests sont conçus à partir des compétences des testeurs, de leur intuition et de leur expérience
- **L'estimation d'erreur** : les testeurs anticipent les défauts basés sur l'expérience (attaque par faute)
- **Les tests exploratoires** comprennent la conception et l'exécution des tests, l'écriture des résultats de tests et l'apprentissage
- Les tests **exploratoires** sont généralement utilisés pour enrichir le cahier de test

62

tech 1

tech Conception De Test

- enrichir le cahier de Test

exploratoire = enrichir + extension

Obj

Smoke Test = relever des Anomalies

Expert Team

Sélectionner les techniques de tests (K2)

- Le choix des techniques de tests à utiliser dépend de différents facteurs :
  1. Type de système
  2. Les standards réglementaires
  3. Les exigences client ou contractuelles
  4. Le niveau et le type de risque
  5. Les objectifs de test
  6. La documentation disponible
  7. Les connaissances des testeurs
  8. Le temps disponible et le budget
  9. Le cycle de vie de développement utilisé
  10. Les modèles de cas d'utilisation
  11. L'expérience sur les défauts découverts précédemment

63

Questions de révisions

- Quelle est la différence entre test boite blanche et noire ?
- Quels sont les techniques de conception boite noire ?
- Quel type de défaut chaque technique peut trouver ?
- Dans quel cas, le table de décision est utilisé ?
- Quelle est la différence entre table de décision et couverture des décisions ?
- A quoi sert les tests exploratoires ?
- Quelle est la différence entre cas de test logique et réel ?
- Que trouve-t-on dans une spécification de cas de test ?
- Que trouve-t-on dans une procédure de cas de test ?