



Fondamentaux en test logiciel

Sur la base de la certification ISTQB Foundation

Sommaire

Les fondamentaux

- 1.1 Pourquoi les tests sont-ils nécessaires ?
- 1.2 Que sont les tests ?
- 1.3 Les 7 principes généraux des tests
- 1.4 Processus de test
- 1.5 La psychologie des tests

1.1 Pourquoi les tests sont-ils nécessaires ?

Pourquoi tester



1.1 Pourquoi les tests sont-ils nécessaires ?

Is Knight's \$440 million glitch the costliest computer bug ever?

CNNMoney

By Brian Patrick Eha @CNNTech August 9, 2012: 10:22 AM ET

Recommend 147



Défaillance

Les ordinateurs exécutent en quelques minutes des ordres prévus sur plusieurs jours

Cause de la défaillance

Défaut introduit dans la nouvelle version du logiciel

Cause du défaut

Absence de tests sur les scénarios de décision

Conséquences : pertes financières

- 440 M\$ perdus en 1 jour (près de 4x le bénéfice 2011)
- Chute de l'action de 48%
- Chute des marchés boursiers en Asie, Europe et US

1.1 Pourquoi les tests sont-ils nécessaires ?

Ariane 5



Défaillance

Le calculateur envoie des données de trajectoire erronées

Cause de la défaillance

Incompatibilité des formats de données
Ariane 4 (int 16 bits) et Ariane 5 (float 64 bits)

Cause du défaut

Normes logicielles non respectées
Tests d'intégration insuffisants

Conséquences : financières & réputation

- Destruction de la fusée : coût global 1 Md \$
- Dégradation d'image pour l'ESA

1.1 Pourquoi les tests sont-ils nécessaires ?

Mars Climate Orbiter



Défaillance

La sonde est détruite à cause d'une erreur de navigation, et s'écrase sur Mars

Cause de la défaillance

Le module de correction de trajectoire exploitaient des données codées selon le système métrique international (km, m, cm) alors que le logiciel embarqué dans les capteur le sollicitait avec des données codées dans le système anglo-saxon

Cause du défaut

Tests d'interopérabilité insuffisants
Spécifications incomplètes

Conséquences : financières et réputation

Une perte de plus de 300 millions de dollars

1.1 Pourquoi les tests sont-ils nécessaires ?

Les défauts logiciels ont des conséquences pour :

- Les sociétés qui les utilisent
 - *Résultats financiers*
 - *Réputation*
 - *Respect de la réglementation*
- L'environnement dans lequel nous vivons
 - *Pollution*
 - *Catastrophes environnementales*
- Les personnes qui les utilisent
 - *Simple désagrément pour le client/consommateur*
 - *Atteinte aux droits et libertés individuelles*
 - *Risque d'accident ou décès*

1.2 Que sont les tests ?

Les systèmes logiciels sont partout :

- Applications bancaires et commerciales
- Produits de grande consommation
- Transports
- Loisirs

La plupart d'entre nous a eu l'expérience d'un logiciel qui n'a pas fonctionné comme attendu.



**Les risques de dysfonctionnement existent dans tous
les logiciels**

Ce sont leurs conséquences qui varient

1.2 Que sont les tests ?

Dans un cycle de développement logiciel, à quel moment pouvons-nous commencer les tests ?

- Lorsque la première version du logiciel est livrée
- Dès qu'une fonctionnalité est créée par l'équipe de développement
- Juste avant de livrer le logiciel au client
- Avant de commencer le développement
- Dès le cahier des charges

1.2 Que sont les tests ?

Nous pouvons distinguer deux grandes familles de tests, qui ont le même objectif :

Sans faire tourner le système

- Exigences
- User Stories
- Code source

Tests statiques

En utilisant l'application

- fonctionnels
- unitaires
- intégration
- de charge...

Tests dynamiques

Objectif de test

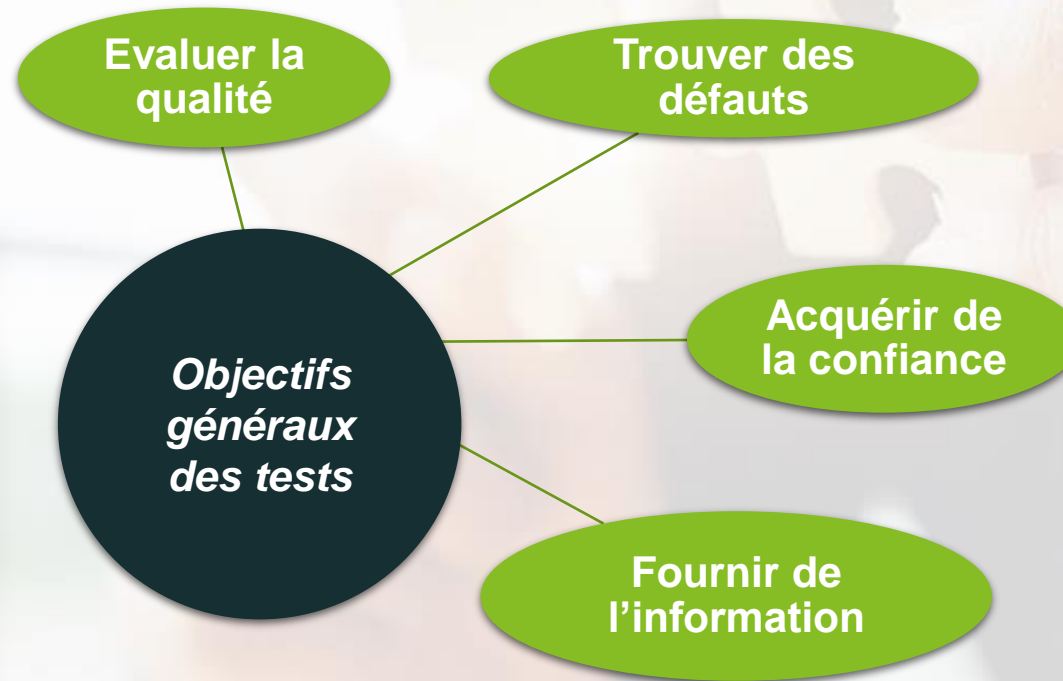
Prévenir l'apparition des pannes et dysfonctionnements

Amélioration

- de la qualité du logiciel
- de son processus de fabrication

1.2.1 Objectifs habituels des tests

Objectifs généraux des tests



Les objectifs de test varient en fonction des niveaux et des phases du cycle de vie

1.2.2 Contribution des tests au succès

Réduire les risques d'occurrence de problèmes dans l'environnement opérationnel

- Réduire la fréquence des livraisons problématiques
- Par l'application de techniques de test appropriées associées au niveau d'expertise des testeurs

Contribuer à la qualité des systèmes logiciels

- Si les défauts découverts sont corrigés avant la livraison du système
- Si les besoins des utilisateurs sont pris en compte

Respecter des exigences légales ou contractuelles

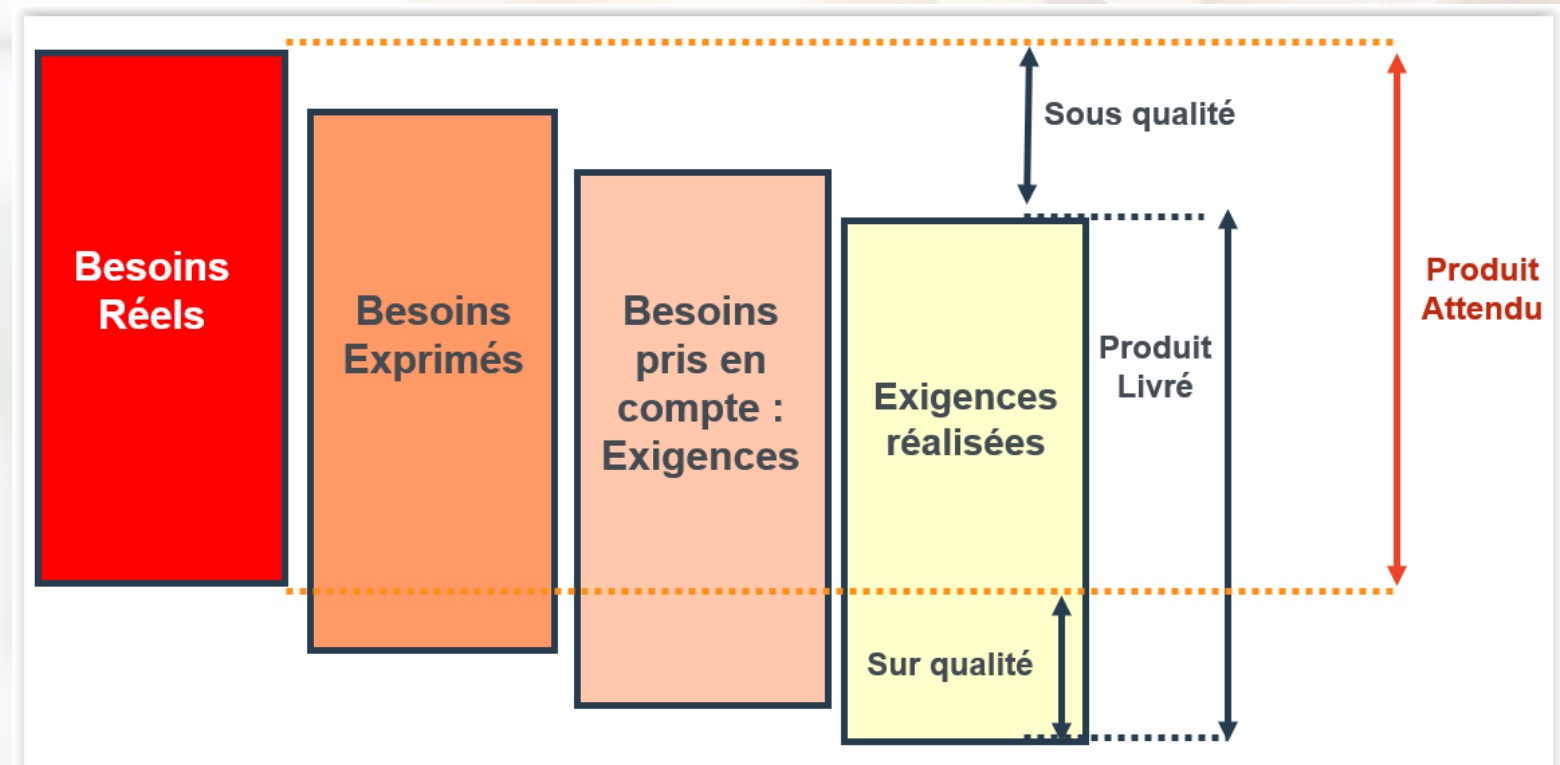
- Calculs de TVA sur les factures, de CSG sur les bulletins de salaire...
- Traçabilité des résultats pour les applications comptables – Loi de Sécurité des Finances (LSF)
- Bonnes Pratiques Pharmaceutiques (BPF)

Atteindre des normes industrielles spécifiques

- Aéronautique (DO-178B/C) ; Industrie (IEC 61508) ; Automobile (ISO 26262) ; Transport ferroviaire (EN 50128) ; Dispositifs médicaux (IEC 62304)

1.2.2 Contribution des tests au succès

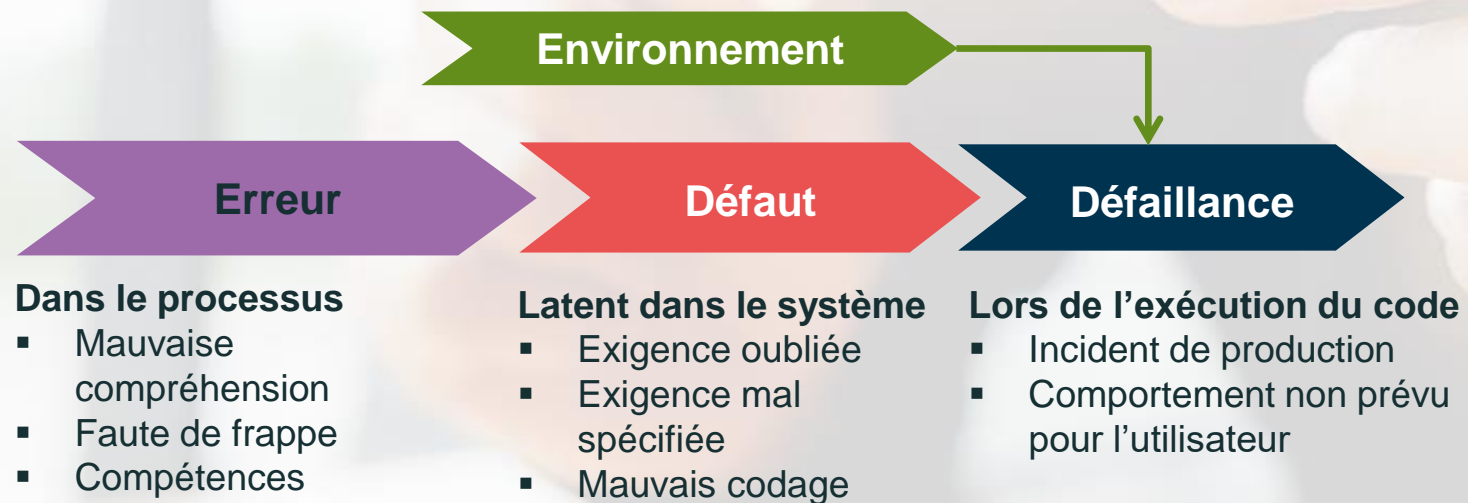
Exemple 2 – La vérification et validation du logiciel par les testeurs avant sa sortie permet de détecter des défaillances qui auraient pu être manquées et augmente la probabilité que le logiciel réponde aux besoins et exigences des parties prenantes



1.2.3 Erreurs, défauts et défaillances

Origine des dysfonctionnements lors de l'utilisation de logiciels

- Défauts ou bugs présents dans
 - *La documentation : informations erronées ou manquantes dans les spécifications*
 - *Le code : erreurs de codage, normes non respectées...*
- Conditions d'environnement
 - *Radiations, magnétisme, champs électroniques*
 - *Infrastructures, systèmes d'exploitation...*



Les défauts n'entraînent pas toujours une défaillance

1.2.3 Erreurs, défauts et défaillances

Les erreurs proviennent de plusieurs sources comme :

Les facteurs humains :

- Erreurs humaines
- Manque d'expérience et des compétences nécessaires
- Mauvaise communication des exigences et de la conception
- Echéanciers serrés, travail sous pression...

Les facteurs techniques :

- Complexité du code, de la conception, des architectures,
- Complexité fonctionnelle du problème à résoudre,
- Méconnaissance des interfaces intra-système et inter-système,
- Technologies nouvelles, peu connues ou complexes

1.2.3 Erreurs, défauts et défaillances

Un résultat inattendu n'est pas forcément une défaillance

- Il doit être analysé pour en déterminer la cause racine



- Des faux positifs peuvent se produire en raison d'erreurs dans la façon dont les tests ont été exécutés : description des étapes, données incorrectes, environnement...
- Si le cas de test est incorrect, il devrait être corrigé et le test réexécuté



Un cas de test incorrect peut mener à un faux-négatif, à savoir que le défaut n'est pas détecté alors qu'il existe bel et bien...

1.2.4 Défauts, causes racines et effets

Analyse des causes racines

- Exemple de questionnement des « 5 pourquoi » (5 whys)
 - *Utilisé pour aller à la base du problème*
 - *Détermine la raison sous-jacente (cause racine) de son introduction*

Problème observé (effet)

Défaillance

Question (why?)	Explication
Pourquoi notre service client a vu une forte augmentation de clients mécontents ?	Parce qu'ils étaient prélevés d'intérêts supérieurs à ceux qu'ils étaient censés payer
Pourquoi le calcul des intérêts était incorrect ?	A cause d'une ligne de code incorrecte
Pourquoi cette erreur s'est glissée dans le code ?	La User Story rédigée par le Product Owner était ambiguë
Pourquoi la User Story était ambiguë ?	1. Le Product Owner avait une mauvaise compréhension de la façon de calculer les intérêts 2. L'équipe n'a pas posé de questions lors de la revue des User Stories

Causes du problème

1.2.4 Défauts, causes racines et effets

Bénéfices de l'analyse des causes racines :

- Déterminer les principaux problèmes qui aboutissent à la création de défauts
- Quantifier le coût des défauts pouvant être rattachés à une cause racine particulière
- Concentrer les corrections sur les causes racines les plus importantes
- Réduire l'apparition de défauts similaires à l'avenir

1.2.5 Tests et débogage

Tester et déboguer sont différents :



- Le test dynamique peut trouver des défaillances
- Le débogage est l'activité de développement qui
 - *Identifie les causes de la défaillance*
 - *Analyse et répare le code*
- Des tests de confirmation sont effectués ultérieurement
 - *Par un testeur*
 - *Pour assurer que la correction résout effectivement la défaillance*



**Les testeurs testent
Les développeurs déboguent**

1.3 Principes généraux des tests

De nombreux principes de tests ont été suggérés au cours des 40 dernières années, ils offrent des indications communes à tous les tests.

- Software Engineering Institute (SEI)
 - *Capability Maturity Model : CMM, CMMI*
 - *Test Maturity Model : TMM*
- SOGETI
 - *Test Process Improvement : TPI / TMAP*

Des principes courants aujourd'hui sont :

- Tests gérés par les risques (Risk Based Testing)
- Tests basés sur les modèles (Model Based Testing)

L'ISTQB a regroupé ces pratiques et défini 7 principes généraux des tests

1.3 Principes généraux des tests

Les 7 principes généraux des test

Principe 1 – Les tests montrent la présence de défauts

Principe 2 – Les tests exhaustifs sont impossibles

Principe 3 – Tester tôt

Principe 4 – Regroupement des défauts

Principe 5 – Paradoxe du pesticide

Principe 6 – Les tests dépendent du contexte

Principe 7 – L'illusion de l'absence d'erreurs

1.3 Principes généraux des tests

Principe 1 - Les tests montrent la présence de défauts

- Les tests peuvent prouver la présence de défauts
- Ils ne peuvent en prouver l'absence
- Ils réduisent la probabilité que des défauts restent cachés dans le logiciel



Est-ce que les tests **couvrent tous les cas d'utilisation du système ? Sans doublons ?**

X Erreurs non détectées



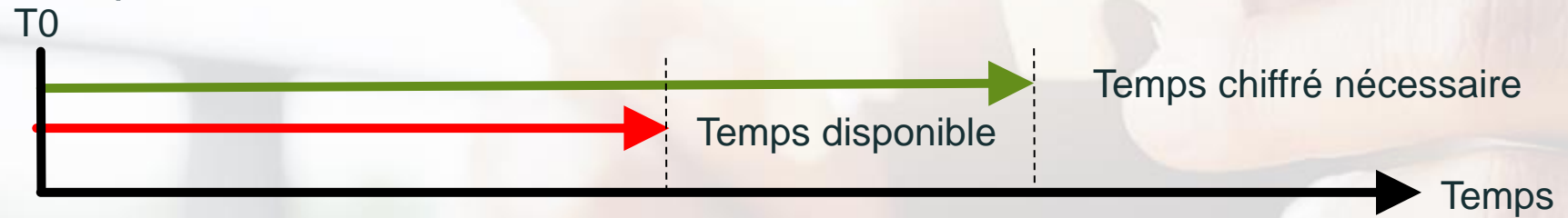
Si aucun défaut n'est découvert, ce n'est pas une preuve d'exactitude

1.3 Principes généraux des tests

Principe 2 - *Les tests exhaustifs sont impossibles*

- Tout tester (toutes les combinaisons d'entrées et de préconditions) n'est pas faisable sauf pour des cas triviaux

Exemple :



- Application de comptabilité
 - Saisie d'un compte
 - Changement des couleurs
 - Recherche multi critères
 - Virement bancaire
 - Envoi de courrier

Quelle fonction n'est pas critique, contournable dans un premier temps ?

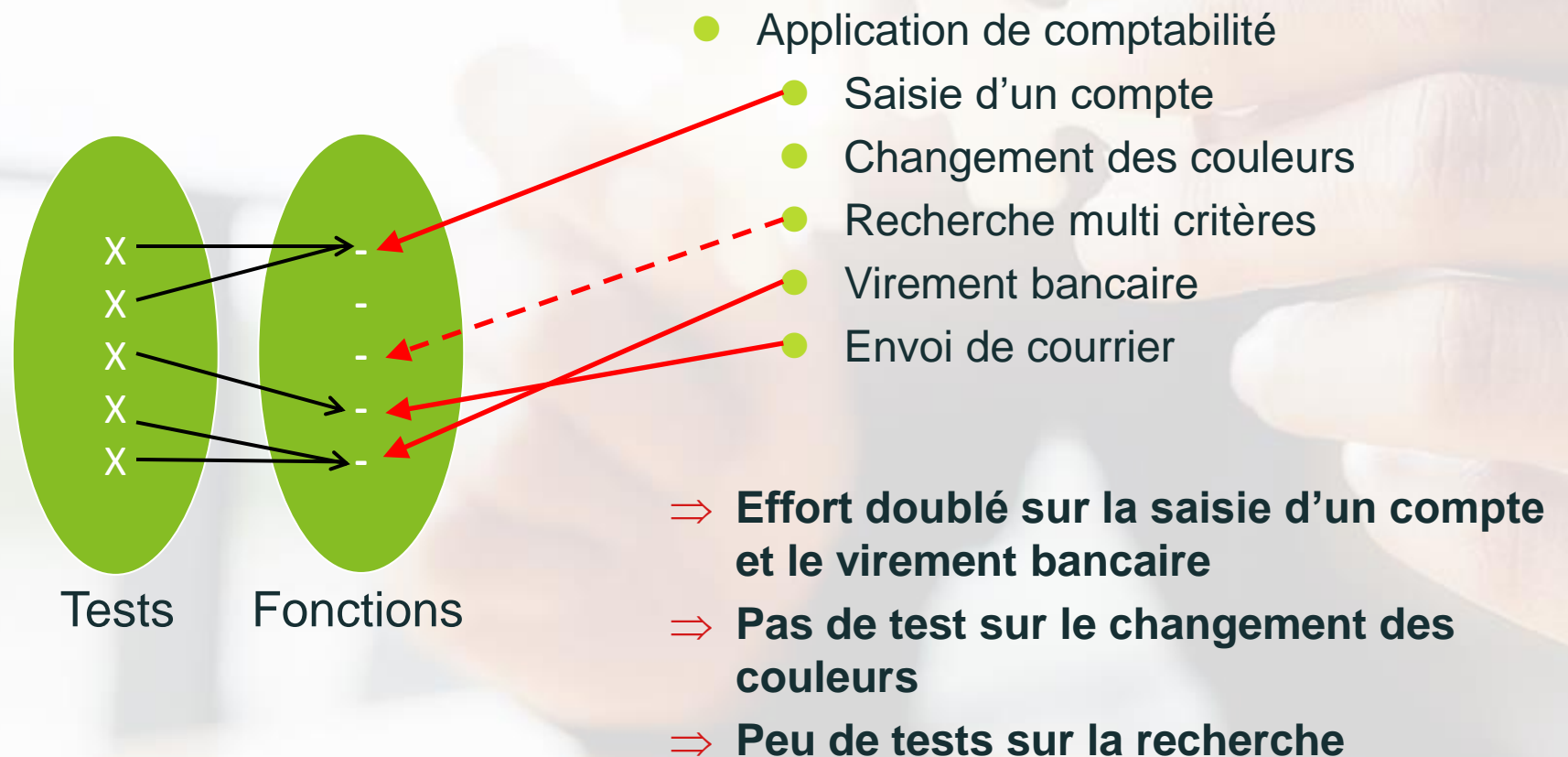


Utiliser les risques et les priorités pour focaliser les efforts de tests

1.3 Principes généraux des tests

Principe 2 - Les tests exhaustifs sont impossibles

- L'effort de test sera porté sur les fonctions critiques et prioritaires



1.3 Principes généraux des tests

Principe 3 – Tester tôt

- Les activités de tests devraient commencer aussi tôt que possible dans le cycle de développement du logiciel
- Projet traditionnel : revue des spécifications
 - *Les tests fonctionnels se préparent d'après les spécifications*
 - *Le testeur fait sans le savoir une revue et demande à éclaircir les points ambigus*
- Projet Agile : revue des User Stories
 - *Rédaction des critères d'acceptation*
 - *Exécution des tests en parallèle du développement (shift-left)*



**Economiser du temps et de l'argent
en identifiant les défauts dès leur introduction
et en éliminant les changements coûteux**

1.3 Principes généraux des tests

Principe 4 – Regroupement des défauts

20 % des modules

- Un petit nombre de modules contient la majorité des défauts détectés
 - *Lors des tests en pré-livraison*
 - *En opération (incidents de production)*

	Modules	Erreurs	%	%cumulé
1	Comptes	88	55	55
2	Personnel	25	16	70
3	Virements	14	9	79
4	Analytique	8	5	84
5	Paramétrage	6	4	88
6	Commercial	4	2	90
7	Editions	3	2	92
8	Connexion	3	2	94
9	Importations	3	2	96
10	Exportations	2	1	97
11	Consolidation	2	1	98
12	Rapprochements	1	1	99
13	Info D	1	1	99
14	Divers	1	1	100
	Total	161		

79 % des erreurs



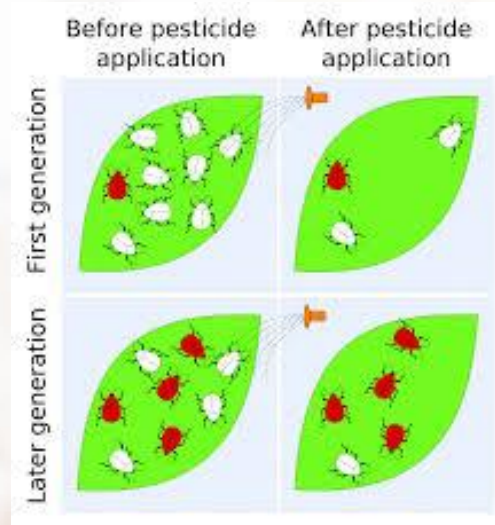
**Faire un Pareto des anomalies et de leur modules (20/80)
Sur les 20% des modules sensibles aux erreurs**

- Approfondir les tests
- Mettre en place des mesures de correction du processus de développement

1.3 Principes généraux des tests

Principe 5 – Paradoxe du pesticide

- Répéter le même ensemble de cas de tests ne trouvera plus de nouveaux défauts.



- Revoir régulièrement les cas de test
- Chercher d'autres types de défaillances
- Prévoir des nouveaux tests
 - Partitions et valeurs différentes
 - Cas aux limites, cas d'erreur...
 - Nouveaux chemins

1.3 Principes généraux des tests

Principe 6 – Les tests dépendent du contexte

- Les logiciels de sécurité critique seront testés différemment d'un site de commerce électronique
- Le test dans un projet Agile est effectué différemment du test dans un projet à cycle de vie séquentiel



Un processus de tests peut convenir à une organisation mais pas à celle de son voisin

Le plan de tests

- identifie les risques
- définit la stratégie et décrit la typologie des tests à faire

Les risques peuvent orienter type et profondeur des tests

- effets sur le système
- nombre et types d'utilisateurs
- fréquence d'utilisation d'une fonction



1.3 Principes généraux des tests

Principe 7 – L'illusion de l'absence d'erreurs

- Trouver et corriger des défauts n'aide pas
 - *Si le système conçu est inutilisable*
 - *S'il ne comble pas les besoins et les attentes des utilisateurs*



- La norme ISO25000 Square détaille les caractéristiques des logiciels, dont le critère d'utilisabilité
- Le testeur peut proposer des améliorations donnant une meilleure ergonomie

1.4.3 Les produits d'activités du test

Produits d'activités de la conception des tests

Pendant les spécifications des cas de test, les cas de test et les données de test sont développés et décrits en détail en utilisant les techniques de conception des tests

Un cas de test couvre des conditions de tests et **est composé** :

- De pré-conditions d'exécution
- De données d'entrée
- D'une action
- De résultats attendus
- Et de post-conditions d'exécution

1.4.3 Les produits d'activités du test

Exemple de cas de test

- La condition de test « facturation particuliers » est couverte par plusieurs cas de tests :

Cas de test	Pré-conditions	Donnée d'entrée / Action	Résultat attendu
TC1	<ul style="list-style-type: none">Se connecterLe client « cli1 » doit exister5 produits « Prod1 » en stock (15 €)	Facturer une commande de 1 produit	<ul style="list-style-type: none">Reste 4 « prod1 » en stockLe compte Cli1 est débiteur de 15€
TC2	<ul style="list-style-type: none">Se connecterLe client « cli1 » doit exister5 produits « Prod1 » en stock (15 €)	Facturer une commande de 10 produits	<ul style="list-style-type: none">Message d'alerte sur la quantitéEnvoi d'une lettre au clientEnvoi message au commercial
TC3	<ul style="list-style-type: none">Se connecterLe client « cli1 » doit exister et est débiteur de 1 000 €4 produits « Prod1 » en stock (15 €)	Facturer une commande de 3 produits	<ul style="list-style-type: none">Demande d'accord au commercialFacture avec rappel des commandes non réglées
...

1.4.3 Les produits d'activités du test

Bonnes pratiques sur la conception des tests

Au niveau du détail des cas de test

- Générer des cas réutilisables sur plusieurs cycles de test avec des données concrètes différentes
- Tout en documentant de manière adéquate la portée du cas de test

Pour la définition des résultats attendus des tests

- Devraient faire partie des spécifications de cas de test
- Si des résultats attendus n'ont pas été définis alors un résultat plausible mais erroné peut être interprété comme un résultat correct
- Les résultats attendus devraient idéalement être définis avant l'exécution des tests

1.4.3 Les produits d'activités du test

Livrables produits par l'implémentation des tests

- Procédures et suites de test
- Cas de test concrets
- Calendrier d'exécution
- Scripts de test automatisés
- Services virtualisés
- Vérification des données
- Vérification des environnements

1.4.3 Les produits d'activités du test

Procédure de test

- Les cas de test sont mis en un ordre exécutable dans une procédure de tests (ou scripts de tests manuels)
- Elle spécifie la séquence des actions pour l'exécution d'un test
- La traçabilité entre les procédures de test et des éléments spécifiques des bases de test permet de mesurer l'atteinte des critères de couverture établis dans le plan de test

Cycle de test

- Exécution du processus de test sur une version unique et identifiable d'un objet de test

1.4.3 Les produits d'activités du test

Exemple :

Req	Exigence	Cas de test
REQ1	Le nom a 10 caractères maxi	TC1
REQ2	Le nom est un champ obligatoire	TC2
REQ3	Le bouton « valider » passe à l'écran suivant	TC3

- **Condition de test** : saisir un client
- Liste des exigences et traçabilité avec les cas de test.

Cas de test	Donnée d'entrée / Action	Résultat attendu
TC1	Entrer « nomduclient » (11 caractères)	Message d'erreur « 10 caractères maximum »
TC2	Valider le formulaire sans saisir le nom du client	Message d'erreur « Le nom est obligatoire »
TC3	Entrer « Client1 » et valider	Passage à l'écran suivant

- **Cas de test**

- **Procédure de tests**

Étape de test ou TC	Donnée d'entrée / Action	Commentaires
1	Se connecter à l'application	
2	Menu client, création	
3	Cas de test TC2	
4	Cas de test TC1	
5	Cas de test TC3	

1.4.3 Les produits d'activités du test

Planning (calendrier) d'exécution des tests

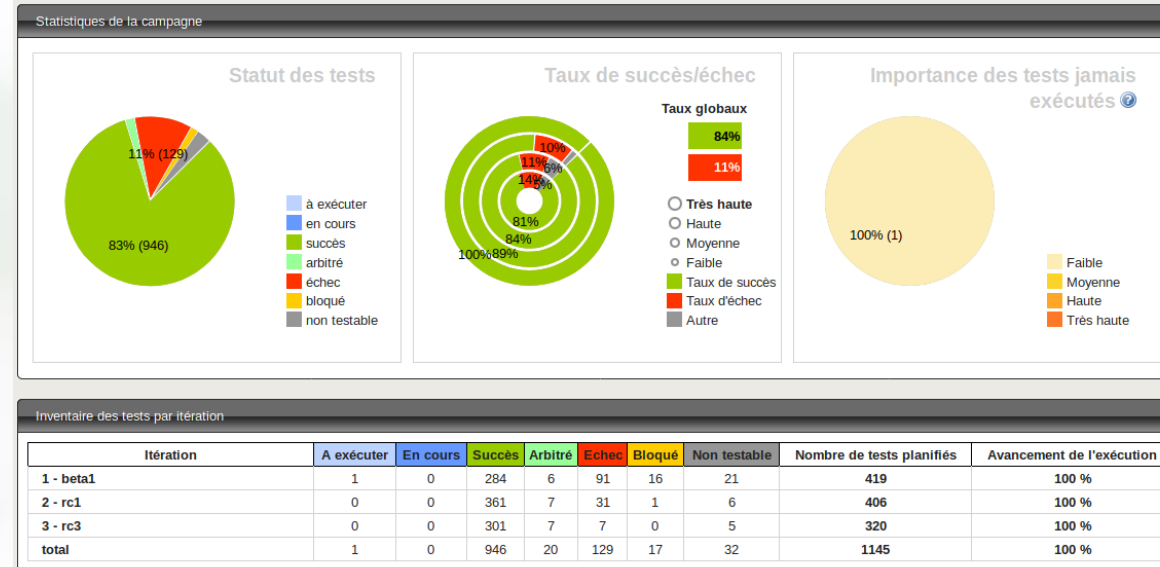
- Le planning d'exécution des tests définit l'ordre dans lequel les diverses procédures, et possiblement les scripts de tests automatisés, sont exécutés, quand et par qui
- Les plannings d'exécution des tests prendront en compte les facteurs tels que :
 - *les tests de régression (vérification de la non régression),*
 - *la priorisation,*
 - *les dépendances techniques et logiques (suites de tests).*

8			CAPACITE TOTALE	364,8	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	5	3	3	5	2	10	10	8	5	7
9																																
10					S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15	S16	S17	S18	S19	S20	S21	S22	S23	S24	S25	S26	S27	S28
11			Dates clés																	L1						L2		L3			MEP	
12			Capacité restante	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	8	5	7
13	Étiquettes de lignes ▼		RAF																													
14	LOT 1																															
15	Préparation	2	0															1	1													
16	Exécution	7	0																1	1	1	4										
17	LOT 2																															
18	Préparation	5	0																				1	3	2							
19	Exécution	9	0																							3	1	5				
20	LOT 3																															
21	Préparation	8	0																					1	1	2	1	4				
22	Exécution	7	0																									1	6			
23	Total Préparation	14,5																														
24	Total Exécution	23,0																														
25																																

1.4.3 Les produits d'activités du test

Exécution des tests

- La documentation du statut de chaque cas de test ou des procédures de test (prêt à être exécuté, réussite, échec, blocage, omission délibérée, etc.)
- Les rapports de défauts
- La documentation précisant quel(s) élément(s) de test, objet(s) de test, outils de test et testware ont été impliqués dans le test



1.4.3 Les produits d'activités du test

Couverture des exigences par les tests

- Une fois l'exécution des tests terminée, le statut de chaque élément des bases de test peut être déterminé et indiqué par traçabilité bidirectionnelle avec la ou les procédures de test associées
 - *Exigences ayant réussi tous les tests planifiés*
 - *Exigences ayant échoué aux tests et/ou ont des défauts associés*
 - *Exigences ayant des tests planifiés en attente d'être exécutés*



Cela permet de vérifier que les critères de couverture ont été satisfaits et de rendre compte des résultats des tests de façon compréhensible par les parties prenantes

1.4.3 Les produits d'activités du test

Exemple – Couverture des exigences par les tests

Espace Exigences

Bibliothèque

Filtre activé

- Sandbox #1
- Sandbox #2
- Squash Démo
 - 02 - Connexion à OpenEcole
 - 2.1.01 - Connexion URL Open Ecole
 - 5.2.0.01 - Accès à la saisie
 - 04 - Saisie des écoles
 - 4.2.0 - Description
 - 4.2.0.01 - Accès à l'écran d'inscription
 - 4.2.0.01 - Accès à l'écran d'inscription
 - 4.2.1 - Ajout d'une école
 - 4.2.1.00 - Description générale
 - 4.2.1.03 - Enregistrement - M
 - 4.2.1.04 - Enregistrement**
 - 4.2.1.05 - Enregistrement validé - M
 - 4.2.1.06 - Onglet Classe
 - 05 - L'inscription des élèves
 - 5.2.0 Accès à l'inscription des élèves
 - Dossier test

Exigence : 4.2.1.04 - Enregistrement - Message de confirmation

Créé le : 11/12/2013 13:45 (admin)
Modifié le : 12/03/2014 12:45 (henix_demo)

Renommer Créer une nouvelle version Imprimer

Informations Pièces jointes

Informations générales

No de version : 1 [\[Consulter l'historique des versions\]](#)
ID : 1492
Référence : 4.2.1.04
Criticité : 1-Majeure
Catégorie : Ergonomique
Statut : 1-En cours de rédaction
Champ personnalisé A1 : (Cliquez pour éditer...)

Description

Au clic sur le bouton [Valider] un message de confirmation <Enregistrement de la table Ecole [1 Enregistrement ajouté]> doit s'afficher.

Cas de test vérifiant cette exigence [Associer des cas de test](#) [Supprimer les associations](#)

#	Projet	Référence	Cas de test	Type
1	Squash Démo	02.2	Connexion BDD - pour fonction Appel de CT	manuel
2	Squash Démo	4.2.02	Ajout d'une Ecole	manuel
3	Squash Démo	4.2.03	Ajout Ecole - Onglets additionnels	manuel

Afficher 50 éléments

Historique des modifications

1.4.3 Les produits d'activités du test

Clôture des tests

- Rapports de synthèse
- Rapports de rétrospectives / plans d'amélioration
- Tickets non résolus : demandes de changement, bugs, user stories...
- Testware finalisés : tests de non régression, scripts d'automate de tests etc...

Rapport de synthèse				
nom du projet : Minimum contributif		Avis de la VN : OUI avec Réserves		
code projet :		date : 02/02/12		
CP MOA : Malika				
CP MOE : Antoine				
CP QUAL : Vittorio				
Périmètre : Ce dossier de synthèse concerne une évolution réglementaire de l'application XXX qui se traduit par la mise en place d'un minimum contributif inter-régimes au 01/01/2012. Le périmètre de ce dossier concerne la première partie des Lots B et C correspondant à la livraison des composants nécessaires pour: <ul style="list-style-type: none">- lot B : programmes de liquidation et de blocage des dossiers en VFU (Versement Forfaitaire Unique)- lot C : programme de calcul de l'échéance La mise en service est suivie dans le dossier AGIS suivant: DS AGIS n°148576 - XXX Livraison v1.12.47 : MICO Lots B et C réduits - Patch				
Synthèse : Les éléments suivants ont été produits pour le projet minimum contributif dans son ensemble: <ul style="list-style-type: none">- Les exigences priorisées par la MOA- Les stratégies de validation et plans de test associés La livraison des lots B et C sur le périmètre réduit a été effectuée le 27/12/2011 et a fait l'objet des vérifications suivantes: <ul style="list-style-type: none">1) Des tests unitaires ont été déroulés par la TMA sur les différentes versions livrées pour vérifier les corrections des anomalies sur l'environnement de développement.2) Une recette fonctionnelle des lots B et C sur ce périmètre s'est déroulée entre le 27/12/2011 et le 01/02/2012. Elle a été effectuée par les équipes MOA et les utilisateurs de la caisse d'Auray en parallèle, sous la responsabilité de la MOA retraite commerçants. Un sous-ensemble de 71 dossiers a été utilisé tout au long				
Eligibilité :		date PV : 11/04/11		
		valeur éligibilité : 5		
		niveau de risque : Elevé		
Exigence / risque / adhérence		Conforme aux attentes VN		
Stratégie et campagne de test		Partiellement conforme aux attentes VN		
Synthèse ACN		Non conforme aux attentes VN		
Documentation		Alerte		
Légende :				
Conforme aux attentes VN				
Partiellement conforme aux attentes VN				
Non conforme aux attentes VN				
Alerte				
Résumés				
Identifiant		Libellé		Date de fin attendue
R1		Anomalies		15/02/2012
R2		Tests opérationnels		25/02/2012
		Description		Date de fin effective
		Fournir un plan de correction des anomalies qui ont bloqué le passage de certains tests		
		Passage de l'échéance à blanc en pré production sur un échantillonnage de cas significatifs à partir de données de production		

2.2 Les Niveaux de tests

Les niveaux de test sont composés de groupes d'activités de test qui sont organisées et gérées ensemble.

Chaque niveau de test est une instance du processus de test :



2.2 Les Niveaux de tests

Les niveaux de tests :

Test de composants

Test d'intégration

Test système

Test d'acceptation

Sont chacun caractérisés par les éléments suivants :

Objectifs spécifiques

Bases de test (références utilisées pour dériver des cas de test)

Objet de test (ce qui est testé)

Défauts et défaillances types

Approches et responsabilités spécifiques



2.2.1 Tests de composants

Les tests de **composants** (de modules ou unitaires) se concentrent sur des composants qui peuvent être testés séparément

Les objectifs des tests de composants sont les suivants :

- Réduire le risque
- Vérifier si les comportements fonctionnels et non-fonctionnels du composant sont tels qu'ils ont été conçus et spécifiés
- Renforcer la confiance dans la qualité du composant
- Trouver des défauts dans le composant
- Empêcher les défauts de passer à des niveaux de tests plus élevés



2.2.1 Tests de composants

Caractéristiques des tests de composants

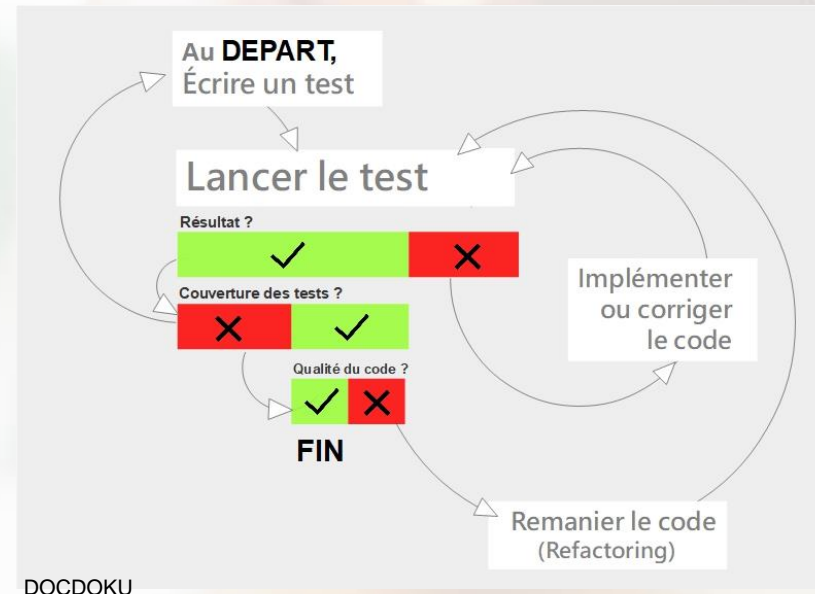
- Souvent effectués indépendamment du reste du système et de manière isolée, ce qui nécessite :
 - *des objets fictifs*
 - *virtualisation des services*
 - *harnais de test, bouchons et pilotes*
- Peuvent porter sur des caractéristiques
 - *fonctionnelles : exactitude des calculs*
 - *non-fonctionnelles : recherche de fuites mémoire*
 - *structurelles : tests de décisions*
- Lorsque les changements de code sont continus, les tests de régression automatisés jouent un rôle clé dans l'établissement de la confiance que les changements n'ont pas endommagés les composants existants



2.2.1 Tests de composants

TDD ou Test Driven Development (**approche Test First**)

Commencer par l'écriture de tests, puis procéder par itérations successives, alternant le lancement des tests avec l'implémentation ou la refonte du code livrable. La règle à suivre est de ne toujours implémenter que le minimum faisant passer le test. Une phase de remaniement (refactoring) peut s'avérer nécessaire pour maximiser la qualité du code.



DOCDOU



2.2.2 Tests d'intégration

Les tests d'intégration se concentrent sur les interactions entre les composants ou les systèmes.

Les objectifs des tests d'intégration comprennent :

- Réduire les risques
- Vérifier si les comportements fonctionnels et non-fonctionnels des interfaces sont tels qu'ils ont été conçus et spécifiés
- Renforcer la confiance dans la qualité des interfaces
- Trouver des défauts (qui peuvent se trouver dans les interfaces elles-mêmes ou dans les composants ou systèmes)
- Empêcher les défauts de passer à des niveaux de test plus élevés

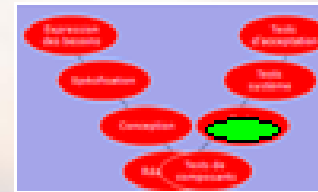
Comme pour les tests de composants, dans certains cas, les tests d'intégration automatisés utilisés en tests de régression donnent l'assurance que les changements n'ont pas altéré les interfaces, composants ou systèmes existants.



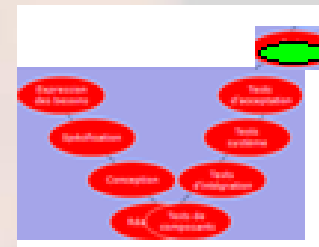
2.2.2 Tests d'intégration

Deux niveaux différents de tests d'intégration, qui peuvent être effectués sur des objets de test de taille variable comme suit :

- **Les tests d'intégration de composants** effectués après les tests de composants



- **Les tests d'intégration de systèmes** entre les différents systèmes





2.2.2 Tests d'intégration

L'approche Top-Down :

- Approche incrémentale où les composants en haut de la hiérarchie sont testés d'abord, avec les composants de niveau inférieur simulés par des bouchons.
- Les composants testés sont ensuite utilisés pour tester des composants de niveaux inférieurs. Le processus est répété jusqu'à ce que les composants de plus bas niveau aient été testés.



Phase 1 : tests des composants de plus haut niveau à l'aide de bouchons remplaçant les niveaux inférieurs.

Phase 2 : Ajout d'un composant inférieur en intégration avec A et utilisation d'un nouveau bouchon.



2.2.2 Tests d'intégration

L'approche Top-Down :

Avantages	Inconvénients
Les fonctions du système restent observables durant les tests	L'utilisation de fonctions importantes du système pourraient ne pas être possibles en raison de l'absence de modules « inférieurs » ou de bouchons
L'isolement d'erreurs au niveau des interfaces des composants est facilitée par la nature incrémentale de la démarche	La conception des cas de tests et des bouchons devient de plus en plus complexe au fur et à mesure que l'on descend dans la hiérarchie
Les tests définis sur un module M serviront directement de tests de non-régression lors de l'intégration de nouveaux modules	Les bouchons peuvent représenter une quantité de code importante à gérer
Les fonctions du système restent observables durant les tests	L'utilisation de fonctions importantes du système pourraient ne pas être possibles en raison de l'absence de modules « inférieurs » ou de bouchons



2.2.2 Tests d'intégration

L'approche Bottom-Up :

- Approche incrémentale où le niveau le plus bas des composants est testé d'abord, et ensuite utilisé pour faciliter les tests des composants de plus haut niveau.
- Ce processus est répété jusqu'au test du composant le plus haut de la hiérarchie.



Phase 1 : tests des composants de plus bas niveau à l'aide de pilotes.

Phase 2 : tests d'un composant supérieur en intégration avec B et C à l'aide d'un nouveau pilote.



2.2.2 Tests d'intégration

L'approche Bottom-Up :

Avantages	Inconvénients
La connaissance des interactions entre composant augmente au fur et à mesure de l'intégration	Le système entier est testé tardivement
Il est possible de tester au fur et à mesure qu'un système est implémenté	La conception des pilotes peut s'avérer plus complexe que celle des bouchons
	Les pilotes peuvent représenter une quantité de code importante à gérer



2.2.2 Tests d'intégration entre systèmes

Particularités des tests d'intégration entre systèmes

- Se concentrent sur la communication entre les systèmes, et non sur la fonctionnalité des systèmes individuels
- Les types de tests fonctionnels, non-fonctionnels et structurels sont concernés
- Relèvent généralement de la responsabilité des testeurs
- Idéalement, les testeurs qui effectuent des tests d'intégration de systèmes devraient comprendre l'architecture du système et devraient avoir contribué à la planification de l'intégration
- Plus la portée de l'intégration est grande, plus il devient difficile de localiser les défauts d'un système spécifique



2.2.3 Tests système

Les tests système se concentrent sur le comportement et les capacités d'un système ou d'un produit entier, en considérant souvent les tâches de bout en bout que le système peut exécuter et les comportements non-fonctionnels qu'il présente pendant l'exécution de ces tâches.

Les objectifs des tests système comprennent :

- Réduire les risques
- Vérifier si les comportements fonctionnels et non-fonctionnels du système sont tels qu'ils ont été conçus et spécifiés
- Valider que le système est complet et fonctionnera comme prévu
- Renforcer la confiance dans la qualité du système dans son ensemble
- Trouver des défauts
- Empêcher les défauts de passer à des niveaux de test plus élevés ou en production



2.2.3 Tests système

Pour certains systèmes, la vérification de la qualité des données peut être un objectif. Comme pour les tests de composants et les tests d'intégration, dans certains cas, les tests de régression automatisés au niveau système donnent l'assurance que les changements n'ont pas altéré les caractéristiques existantes ou les capacités de bout en bout.

Le test système produit souvent de l'information qui est utilisée par les intervenants pour prendre des décisions de livraison.

Les tests systèmes peuvent également satisfaire aux exigences ou aux normes légales ou réglementaires.

L'environnement de test devrait idéalement correspondre à la cible finale ou à l'environnement de production.



2.2.4 Tests d'acceptation

Les tests d'acceptation se concentrent généralement sur le comportement et les capacités d'un système ou d'un produit entier

Les objectifs des tests d'acceptation comprennent :

- Établir la confiance dans la qualité du système dans son ensemble
- Valider que le système est complet et qu'il fonctionnera comme attendu
- Vérifier que les comportements fonctionnels et non-fonctionnels du système sont tels que spécifiés
- Satisfaire aux exigences ou aux normes légales ou réglementaires



- La découverte de défauts ne constitue en général pas un objectif des tests d'acceptation
- Trouver un grand nombre de défauts à ce stade constitue un risque majeur pour la réussite du projet



2.2.4 Tests d'acceptation

Différentes formes des tests d'acceptation

Utilisateur (UAT – User Acceptance Testing)

- Valider l'aptitude à l'usage par les utilisateurs
- Dans un environnement opérationnel réel ou simulé

Opérationnel (OAT – Operational Acceptance Testing)

- Garantir que les opérateurs et administrateurs système peuvent maintenir le système en bon état de fonctionnement pour les utilisateurs
- Dans l'environnement opérationnel

Contractuel et réglementaire

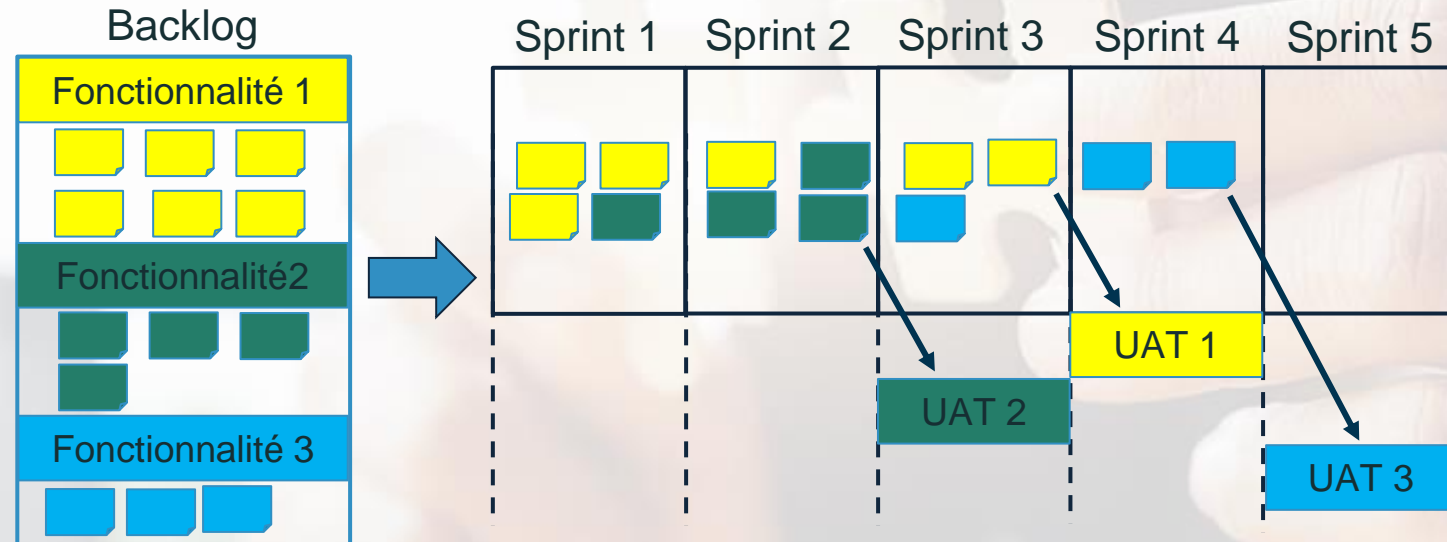
- Valider l'atteinte de critères d'acceptation d'un contrat pour la production de logiciels sur mesure, ou de règlements gouvernementaux, légaux, de sécurité...

Alpha et Beta

- Renforcer la confiance des clients et/ou des opérateurs dans l'utilisation du système dans des conditions normales et quotidiennes et dans l'environnement opérationnel

2.2.4 Tests d'acceptation

Exemple de planification des tests d'acceptation



- Vérification de chaque User Story par rapport à ses critères d'acceptation faite par les testeurs à chaque itération
- Tests d'acceptation sur la fonctionnalité complète effectués par les utilisateurs après une série d'itérations

2.2 Niveaux de tests

Les bases de tests par niveaux de tests (1/2)

Niveau de test	Bases de tests
Test de composants	Conception détaillée Code Modèle de données Spécification des composants
Test d'intégration	Conception du logiciel et du système Diagrammes de séquence Spécifications des protocoles d'interface et de communication Cas d'utilisation Architecture au niveau du composant ou du système Workflows Définitions des interfaces externes
Test système	Spécifications des exigences système et logicielles (fonctionnelles et non-fonctionnelles) Rapports d'analyse des risques Cas d'utilisation Epics et User Stories Modèles de comportement du système Diagrammes d'états Manuels système et manuels d'utilisation

2.3 Types de tests

Un **type de test** est un groupe d'activités de test

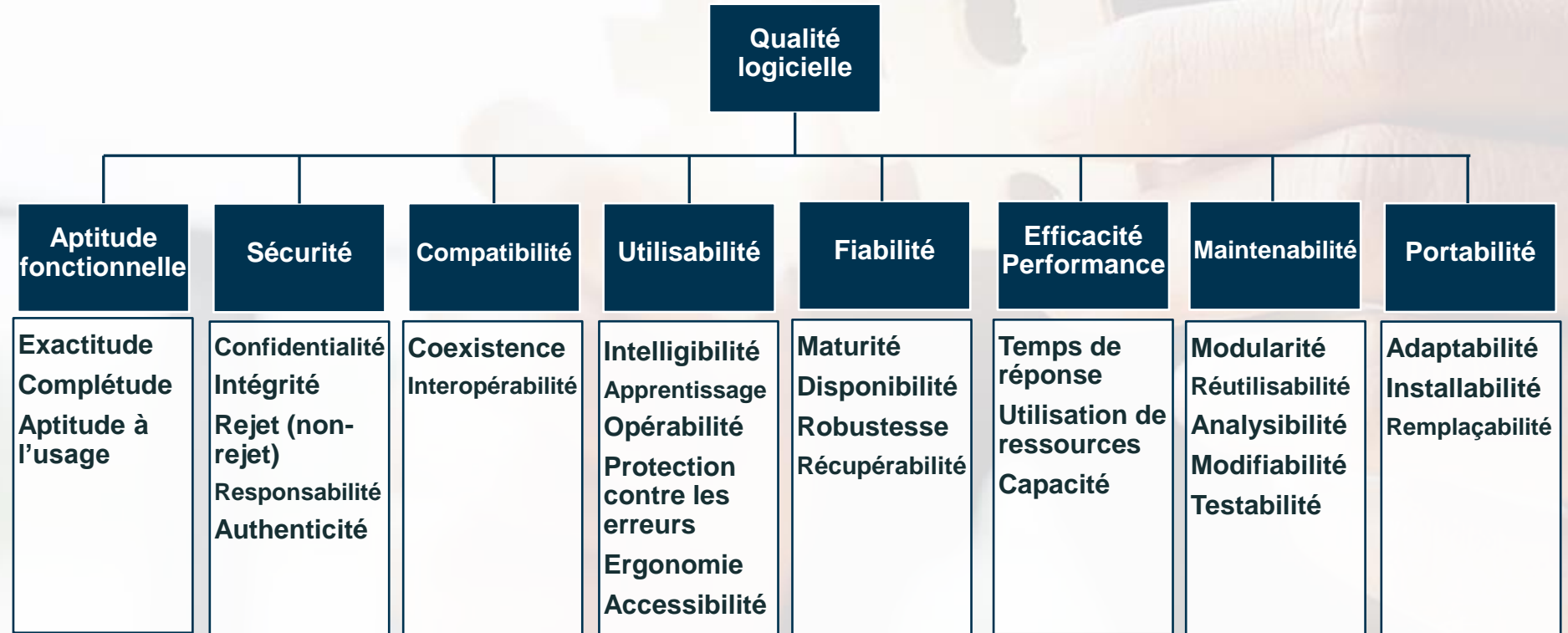
- Visant à tester des caractéristiques spécifiques
- Sur la base d'objectifs de test spécifiques

Les différents types de test servent à évaluer

- Les caractéristiques de qualité fonctionnelle
 - *Complétude, exactitude, pertinence, sécurité ...*
- Les caractéristiques de qualité non-fonctionnelles
 - *Fiabilité, performance, compatibilité, facilité d'utilisation...*
- La structure ou l'architecture du composant ou du système
 - *Complétude et conformité aux spécifications*
- Les effets des changements
 - *Confirmation de la correction de défauts : tests de confirmation*
 - *Recherche de changements non désirés résultant de changements dans le logiciel ou l'environnement : tests de régression*

2.3 Types de tests

Définition des caractéristiques qualité selon la norme ISO 25010



Chacune de ces caractéristiques peut être vérifiée par les types de test qui lui correspondent

2.3.1 Tests fonctionnels

Les tests **fonctionnels** d'un système évaluent les fonctionnalités que le système devrait réaliser

Ces fonctionnalités peuvent être décrites dans :

- Les spécifications des exigences métier
- Les EPICs
- Les User Stories
- Les cas d'utilisation
- Les spécifications fonctionnelles
- Ou elles peuvent ne pas être documentées

Les fonctionnalités décrivent **ce que** le système doit faire

2.3.1 Tests fonctionnels

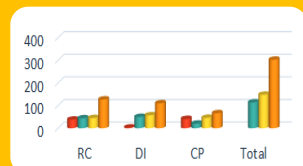
Caractéristiques des tests fonctionnels



S'appliquent à tous les niveaux de test en fonction de la spécification correspondante

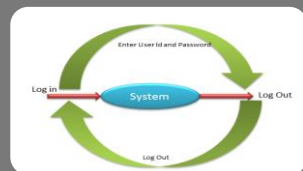


Les techniques boîte-noire peuvent être utilisées pour dériver les conditions et cas de test



Leur complétude est mesurable par la couverture fonctionnelle qui mesure comment un élément fonctionnel a été exercé par des tests

- Exemple : pourcentage d'exigences couvertes par les tests



Leur conception et exécution peuvent nécessiter des compétences ou des connaissances particulières

- Problème métier particulier que le logiciel résout
- Rôle particulier que joue le logiciel

2.3.2 Tests non fonctionnels

Les tests **non-fonctionnels** d'un système évaluent les caractéristiques des systèmes et des logiciels comme l'utilisabilité, la performance ou la portabilité

La norme ISO 25010 propose une classification des caractéristiques de qualité des produits logiciels

Le test non-fonctionnel est le test de **comment** le système se comporte

2.3.2 Tests non fonctionnels

Caractéristiques des tests non fonctionnels



Ils peuvent et devraient souvent être effectués à tous les niveaux de test, et le plus tôt possible

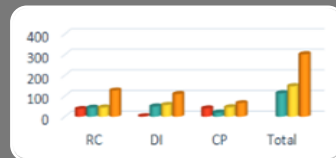


La découverte tardive de défauts non-fonctionnels peut être extrêmement préjudiciable à la réussite d'un projet



Des techniques de boîte-noire peuvent être utilisées pour dériver les conditions et cas de test

- Analyse des valeurs limites pour définir les conditions de sollicitation pour les tests de performance



La complétude des tests est mesurable par la couverture non-fonctionnelle

- Utiliser la traçabilité entre les tests et les appareils pris en charge pour une application mobile
- Calculer le pourcentage d'appareils qui font l'objet de tests de compatibilité
- Elle permet d'identifier potentiellement les lacunes de couverture



Leur conception et exécution peuvent impliquer compétences ou connaissances spécifiques

- Faiblesses inhérentes à une conception ou à une technologie : vulnérabilités de sécurité associées à un langage de programmation
- Spécificité des utilisateurs pour un domaine particulier : utilisateurs des systèmes de gestion des établissements de santé

2.3.2 Tests non fonctionnels

Exemples de tests non-fonctionnels

Tests de performance	Mesurent les temps de réponse du système lors d'une utilisation standard
Tests de charge	Mesurent les temps de réponse du système lors d'une charge standard
Tests de stress	Mesurent la capacité du système à absorber une charge temporairement plus importante que la nominale, afin d'identifier la charge et les débits limites absorbés par le SI
Tests d'endurance	Mesurent la stabilité du système sur une durée élevée (24-48 heures) en simulant une période de production classique
Tests destructifs	Mesurent la réaction du système suite à une modification brutale au niveau de l'architecture afin d'estimer les pertes et valider la bonne reprise du service une fois la situation revenue à la normale

2.5 Tests liés au changement

Test de confirmation ou retest

- Réexécution de tous les cas de test qui ont échoué en raison du défaut sur la nouvelle version du logiciel
- Nouveaux tests si le défaut était une fonctionnalité manquante
- Le but est de confirmer que le défaut d'origine a été réparé avec succès

Tests de régression

- Répéter une suite de test suite à une modification apportée à une partie du code (correction ou autre type de modification)
- Le périmètre peut couvrir le composant modifié, les autres composants du même système ou même d'autres systèmes
- L'objectif est de détecter d'éventuels effets de bord involontaires



Ces tests sont effectués à tous les niveaux de test



Chapitre 3 : Tests statiques

Sommaire

Tests statiques

- 3.1 Bases des tests statiques
- 3.2 Processus de revue

3.1 Bases des tests statiques

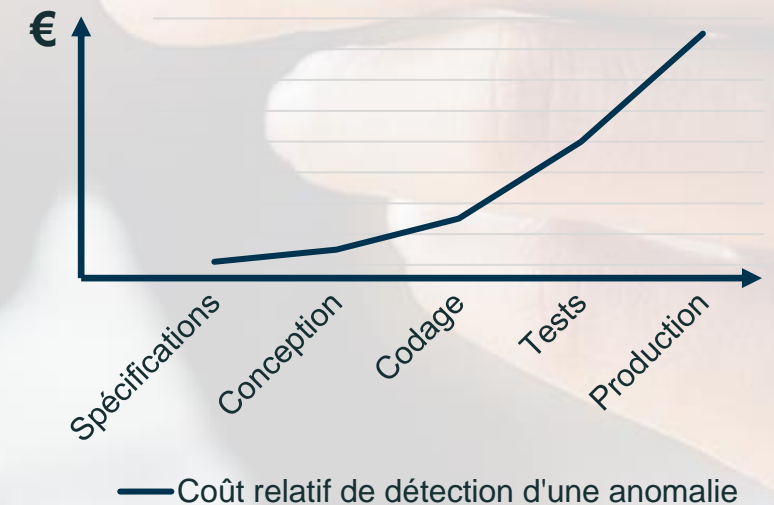
Les techniques de tests **statiques** n'exécutent pas le logiciel qui est testé.

Elles sont :

- Soit manuelles (revues),
- Soit automatisées (*analyse statique*, métrologie de code).

Une revue est un test des produits logiciels

- Des imprécisions dans les spécifications détectées en revue coûtent moins cher à corriger qu'après le développement.



3.1.1 Éléments pouvant être revus

Tout produit logiciel peut être examiné :

- Les spécifications
 - *Exigences métiers, fonctionnelles, de sécurité*
 - *Epics, User Stories, critères d'acceptation*
 - *Spécifications de conception, d'architecture*
- Les modèles
 - *Diagrammes d'activité*
- Le code
- Le testware
 - *Plans de tests*
 - *Spécifications de tests, cas de tests, procédures de tests*
 - *Scripts de tests automatisés*
- Les guides d'utilisateurs ou pages web
- Les contrats, plans, calendriers, budgets

3.1.2 Bénéfices des tests statiques

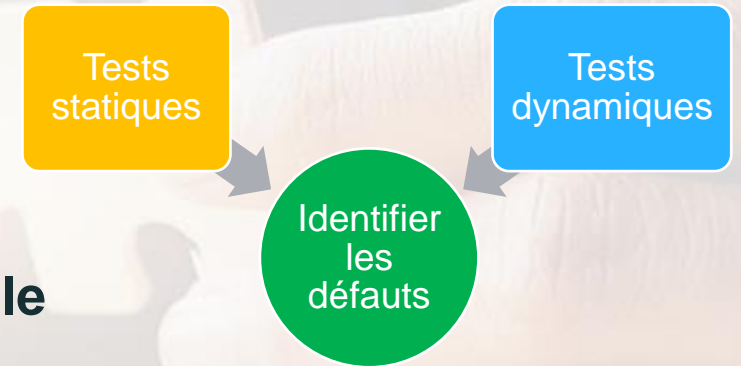
Les bénéfices des tests statiques incluent :

- Une détection anticipée des défauts et leur correction, dont la détection pendant les tests dynamiques est peu probable
- La réduction des défauts de conception ou de codage (levées d'incohérences, d'ambiguïtés, de contradictions, d'omissions, d'inexactitudes et de redondances)
- Des améliorations de productivité dans le développement
- Des durées de développement réduites
- Des durées et des coûts de tests réduits
- Des réductions de coûts tout au long de l'existence du logiciel
- Une meilleure communication grâce à la participation aux revues

3.1.3 Différences entre les tests statiques et dynamiques

Ils sont **complémentaires**

- Les objectifs poursuivis sont identiques
 - **Mesurer** la qualité
 - Identifier les défauts **le plus tôt possible**
- Les tests statiques
 - Trouvent des défauts pouvant être difficiles à détecter avec des tests dynamiques
 - Améliorent la cohérence et la **qualité interne** des produits d'activités
- Les tests dynamiques
 - Identifient les défaillances causées par des défauts lorsque le logiciel est **exécuté**
 - Se concentrent généralement sur les comportements **visibles de l'extérieur**



3.1.3 Différences entre les tests statiques et dynamiques

Les défauts détectés lors de revues sont de différentes sortes :

Domaine	Sous-domaine	Exemples de défauts
Exigences	Défauts de spécification	Incohérences, ambiguïtés, contradictions, omissions, inexactitudes et redondances
Conception	Défauts de conception	Algorithmes ou structures de base de données inefficaces, taux de dépendance élevé, faible cohésion
	Spécification incorrecte d'interfaces	Unités de mesure utilisées par le système appelant différentes de celles utilisées par le système appelé
Code	Ecart par rapport aux normes	Variables avec des valeurs non définies, variables déclarées mais jamais utilisées, code inatteignable, code dupliqué
	Vulnérabilités de sécurité	Sensibilité aux débordements de la mémoire tampon
	Défauts de maintenabilité	Modularité inadéquate Mauvaise réutilisation des composants Code difficile à analyser et à modifier sans introduire de nouveaux défauts
Tests	Couverture	Lacunes ou inexactitudes dans la traçabilité ou la couverture des bases de test (tests manquants pour un critère d'acceptation)

3.1.3 Différences entre les tests statiques et dynamiques

Résumé

	Objectifs	Détections	Bénéfices
Statiques	Découvrir les défauts dans les produits d'activité avant qu'ils soient détectés par des tests dynamiques	Erreurs de codage, conception, compréhension du besoin du client Mauvaise traduction du cahier des charges en spécification	Prévenir l'introduction de défauts dans le code
Dynamiques	Détecter les défaillances à l'exécution du logiciel	Défaillances fonctionnelles ou non fonctionnelles (ex. : trop lent)	Corriger les défauts avant livraison du logiciel au client



Chapitre 4 : Techniques de tests

4.1.1 Choix des techniques de test

L'objectif des techniques des tests est d'identifier :

- Les conditions de tests : article (item ou élément) ou événement d'un composant ou système qui pourrait être vérifié par des cas de test.
- Les cas de tests : ensemble de valeurs d'entrée, de pré-conditions d'exécution, de résultats attendus et de post-conditions d'exécution, développés pour un objectif ou une condition de test particulier, tel qu'exécuter un chemin particulier d'un programme ou vérifier le respect d'une exigence spécifique.
- Les données de test : données créées ou sélectionnées pour satisfaire les préconditions d'exécution et les entrées pour exécuter un ou plusieurs cas de test.

4.1.1 Choix des techniques de test

Le choix des techniques de test à utiliser dépend de plusieurs facteurs :

- Type de composant/système et complexité
- Normes réglementaires
- Exigences client ou contractuelles
- Niveaux et types de risques
- Objectifs du test
- Documentation disponible
- Connaissances et compétences des testeurs
- Outils disponibles
- Temps et budget
- Modèle de cycle de vie du développement logiciel
- Utilisation prévue du logiciel
- Expérience antérieure de l'utilisation des techniques de test sur le composant ou le système à tester
- Types de défauts attendus dans le composant ou le système

4.1.1 Choix des techniques de test

Certaines techniques sont plus applicables à certaines situations et à certains niveaux de test ; d'autres sont applicables à tous les niveaux de test.

Lors de la création de cas de test, les testeurs utilisent généralement une combinaison de techniques de test pour obtenir les meilleurs résultats de l'effort de test.

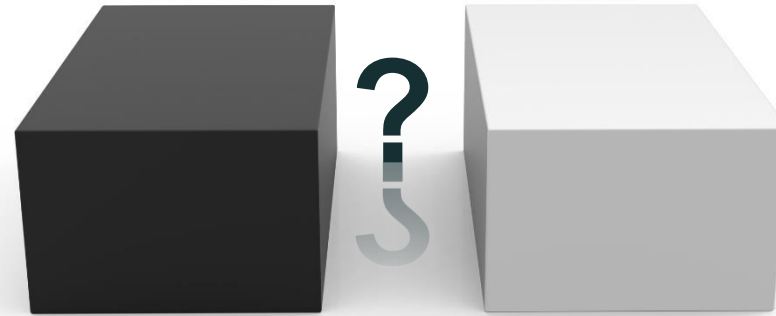
4.1.2 Catégories de techniques de tests et leurs caractéristiques

Techniques boîte noire

- Aucune connaissance de la structure interne du composant
- Concentrées sur les entrées et sorties de l'objet de test
- A partir de l'analyse de la base de test

Techniques boîte blanche

- Architecture/structure du code connues
- Concentrées sur la structure et le traitement à l'intérieur de l'objet de test



Techniques basées sur l'expérience

- Aucune connaissance de la structure interne ou documentation du composant
- Se base sur la connaissance du testeur du fonctionnement et défauts de l'application
- Souvent combinées à des techniques plus formelles

4.4.2 Tests exploratoires

Les **tests exploratoires** sont des tests informels conçus, exécutés, enregistrés et évalués dynamiquement pendant l'exécution des tests

- Basés sur une charte de tests contenant les objectifs de tests
- Effectués dans une session (courte période de temps)
- Parfois en utilisant des fiches de session avec étapes suivies et constatations réalisées

Les résultats des tests sont utilisés pour en apprendre davantage sur le composant ou le système, et pour créer des tests pour les parties qui pourraient avoir besoin de plus de tests.

Ils sont utiles :

- Si peu ou pas de spécifications.
- Si contraintes de temps importants
- En complément d'autres méthodes plus formelles

4.4.2 Tests exploratoires

Exemple de charte de tests :



Charte de tests

- **Explorer** une « story », une fonctionnalité, un système ...
- **Avec** des ressources, des contraintes, des persona, ...
- **Pour découvrir** des informations

*Explorer l'édition de profil
Avec des « scripts injections »
Pour découvrir les vulnérabilités de sécurité*

*Explorer l'édition de profil
Avec différents types d'utilisateurs
Pour découvrir les interactions entre l'édition de profil et les rôles*

4.4.2 Tests exploratoires

Considérations pratiques

- Le test exploratoire est un point d'entrée important pour l'équipe de test qui manque d'expérience sur certaines parties fonctionnelles difficiles à appréhender.
- Le soutien d'utilisateurs finaux ciblés permet de mettre en place de nouveaux scénarios de tests.
- Ce type de test permet aux testeurs de découvrir l'appliquatif de manière libre et d'orienter les tests en fonction de leur expérience et de leurs profils
- Il permet de s'assurer que les chemins négatifs ou inhabituels ont été couverts

4.4.2 Tests exploratoires

Expérience basée sur l'intuition de l'individu :

- Grâce à l'expertise sur des domaines techniques particuliers ou types de tests :
 - Un expert finance aura l'idée de contrôler les chiffres calculés par le système
 - Non-dits
 - Environnement logiciel :
 - Un expert Windows aura l'idée de tester les fuites mémoires ou bien de contrôler les interactions avec d'autres applications.

4.4.3 Test basés sur des checklists

Les testeurs conçoivent, implémentent et exécutent des tests pour couvrir les conditions de test figurant dans une **checklist**

Au cours de l'analyse, les testeurs peuvent :

- Créer une nouvelle checklist
- Utiliser une checklist existante
- Compléter la checklist existante au besoin

4.4.3 Test basés sur des checklists

Les checklists peuvent être construites à partir de :

- La base de l'expérience du produit
- La connaissance de ce qui est important pour l'utilisateur
- La compréhension du pourquoi et du comment des défaillances logicielles

Des checklists peuvent être créées pour prendre en charge différents types de tests, y compris les tests fonctionnels et non-fonctionnels

4.4.3 Test basés sur des checklists

Exemple de checklist de test

Export carton box wet / damaged / torn through inside gift box		X	
Export carton box wet / damaged / torn but not through inside gift box			X
Export carton important marking wrong, incomplete, off position / poor cutting / slanting (brand name, shipping mark...)		X	
Export carton unimportant marking off position / poor cutting / slanting			X
Smudge (dirty) printing on carton / gift box found not readable		X	
Smudge (dirty) printing on carton / gift box but still readable			X
Missing, wrong label or peel off on carton / gift box		X	
Labels off position / slant or not align			X
Wrong barcode or serial number (on unit, gift box, carton)		X	
Wrong color on carton / gift box		X	
Color fade on carton / gift box			X
Security gummed tapes missing or not secured / fully sealed / open end		X	
Length of center security gummed tape less than 2 cm but still secured			X
Loose part, foreign, extra material inside carton		X	
Unusual / instant odor smelled after open the package	X		

Team "Done" List

...With a Story

- All Code (Test and Mainline) Checked in
- All Unit Tests Passing
- All Acceptance Tests Identified, Written & Passing
- Help File Auto Generated
- Functional Tests Passing

...With a Sprint

- All Story Criteria, Plus...
- Product Backup Updated
 - Performance Testing
 - Package, Class & Architecture Diagrams Updated
 - All Bugs Closed or Postponed
 - Code Coverage for all Unit Tests at 80% +

...Release to INT

- All Sprint Criteria, Plus...
- Installation Packages Created
 - MOM Packages Created
 - Operations Guide Updated
 - Troubleshooting Guides Updated
 - Disaster Recovery Plan Updated
 - All Test Suites Passing

...Release to Prod

- All INT Criteria, Plus...
- Stress Testing
 - Performance Tuning
 - Network Diagram Updated
 - Security Pass Validated
 - Threat Modeling Pass Validated
 - Disaster Recovery Plan Tested

60 points à vérifier avant de mettre en ligne votre site web

	Oui/Non
Contenu	
1. Le contenu est cohérent, pertinent et exact	
2. Grammaire, syntaxe et orthographe sont impeccables	
3. Le nombre de paragraphes par page est limité, les phrases sont courtes	
4. Les illustrations sont pertinentes et impactantes (infographies, images)	
5. Les mentions légales sont détaillées	
6. Les droits d'auteur sont respectés et mentionnés	
7. Le copyright est présent dans le footer	
8. Le contrat avec le prestataire garantit une utilisation paisible de leur oeuvre	
9. Déclaration CNIL	
10. Les contenus de test sont masqués ou supprimés	
11. Le contenu est validé par les décideurs	
Mise en page et navigation	
12. L'ensemble est lisible et attrayant	
13. Le contraste coloriel est suffisant	
14. Les styles sont uniformes sur toutes les pages du site	
15. Les contenus sont bien hiérarchisés	
16. La logique de navigation est conforme sur tout le site	
17. Le nombre de liens par page est limité	
18. Les liens ou les espaces cliquables sont facilement identifiables	
19. Un lien sur le logo renvoie vers la page d'accueil	
20. Un fil d'Ariane est présent	
21. Un plan du site et un moteur de recherche sont disponibles sur chaque page	
Fonctionnel	
22. Un moyen de contact est accessible depuis chaque page	
23. Les formulaires de contact sont simples et fonctionnels (vérification anti-spam, réponses automatisées, version texte HTML)	
24. Le formulaire de recherche est efficace et les résultats sont pertinents	
25. Le site reste lisible ou propose une solution pour une navigation sans javascript ou sans plugins (Flash, adobe)	
26. Le temps de chargement des pages est limité	
27. Le site est accessible aux mal-voyants	
28. Le site est conforme aux standards du W3C (validation HTML, javascript, CSS)	
29. Le site est compatible avec les différents navigateurs et systèmes d'exploitation	
30. Le site s'affiche correctement selon les résolutions d'écran	
31. Les pages s'impriment correctement (style CSS spécifique)	
32. La page d'erreur 404 est personnalisée	

4.4.3 Test basés sur des checklists

Conclusion

- En l'absence de cas de tests détaillés, les tests basés sur des checklists peuvent fournir des lignes directrices et un certain degré de cohérence
- Comme il s'agit de listes de haut niveau, il est probable qu'il y ait une certaine variabilité dans les tests réels, ce qui pourrait entraîner une plus grande couverture des tests, mais une reproductibilité plus faible

4.4 Techniques de test basées sur l'expérience

Techniques basées sur l'expérience	Techniques basées sur les spécifications
Méthode peu ou informelle	Méthode formelle
Spécifications rares ou inadéquates	Dépendance de la qualité des spécifications
Forte dépendance des compétences des testeurs	Importance de la qualité des jeux de données
Opérations éclairs	Préparation et planification importantes
Conditions proches des conditions réelles	
Permet aux novices de découvrir l'application	
Permet d'identifier de nouveaux cas de test	