Validation, vérification et tests

Validation : vérifier une spécification par rapport aux attentes métier

Vérification : vérifier une spécification par rapport aux attentes techniques

Cas de test : instance détaillé de la condition de test (données d'entrée, préconditions, post conditions, résultat attendu)

Oracle : composant soumit aux mêmes conditions que l'application à tester

(**Verdict** : réponse à une requête d'exécution d'un test)

OK : réussie

KO : échec

NA : non appliqué

NE : non exécuté

NI : non implémenté

Test doit trouver les défauts pas seulement prouver le fonctionnement

Développeur ne sont pas les meilleurs testeurs

Défauts sont générés par le processus de développement dans son ensemble et non pas par un individu

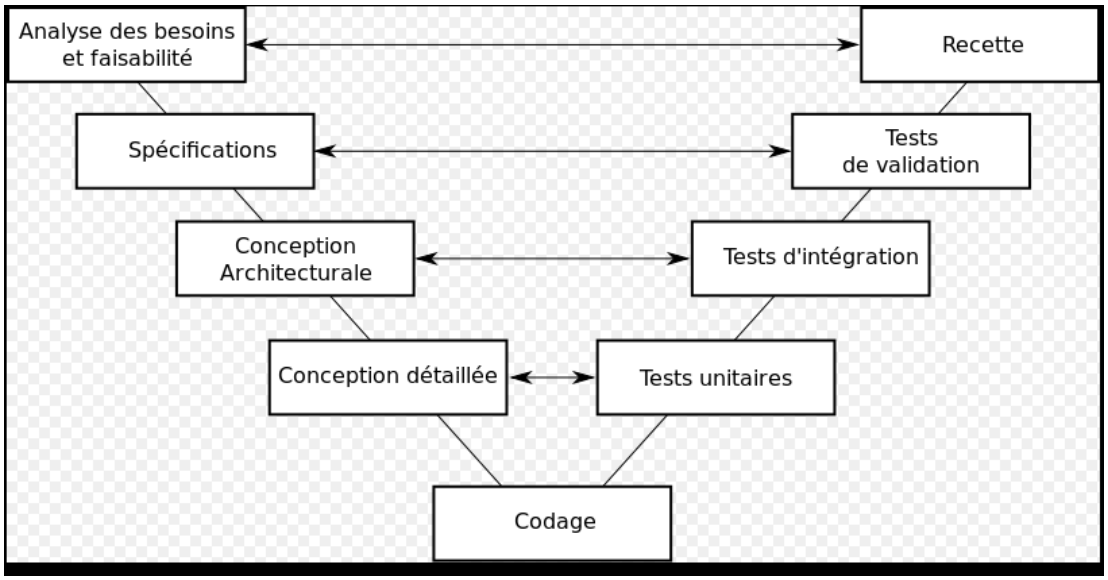
Principe de base du test :

- Indépendance testeurs/développeurs
- Tester au plus tôt
- Automatisation
- Résultats prouvés et reproductibles
- Harmonie avec le développement

Modèle en cascade :

- Exigences
- Conception
- Implémentation
- Vérification
- Maintenance

Cycle en V :



Modèle agile : test continue

Objectif d'un test :

- valider le bon fonctionnement d'un logiciel par rapport aux besoins et aux exigences recueilles après des utilisateurs-
- assurer un niveau de qualité suffisant
- identifier les dysfonctionnements, anomalies et régressions
- économiser du temps et de l'argent
- rendre le développement efficace
- augmenter la satisfaction du client
- identifier les lacunes des développeurs

Plan de test : document définissant la stratégie et la portée des tests

Objectif plan de test :

- définit le déroulement les conditions d'arrêt
- conçu par le responsable des tests et validé par le chef de projet

Prérequis du plan de test :

- réunion de démarrage
- analyse des exigences
- interviews focalisés

Plan de test contient :

- Introduction : informations générales
- Projet à tester : précision (date)
- Environnement de test : outils, bases, données
- Tests à réaliser : objectifs, exigences, analyse des risques
- Stratégie de tests : approche, ordre d'exécution
- Gestion de fiches d'anomalies :
- Gestion des ressources : ressources matérielles et humaines
- Périmètre d'intervention :
- Définition des livrables :

Equipe de testeurs :

- Coordinateur de tests
- Testeurs

Cas de test : ensemble d'entrée de tests, de conditions d'exécution et de résultats attendus pour un objectif particulier

Anatomie d'un cas de test :

ID
Description
Priorité
Préconditions
Scénario
Résultats attendus
Résultats actuels
Remarques

TestLink : outil de gestion de cas de tests

Défaut : imperfection dans un composant ou un système qui peut en perturber le fonctionnement

Tests unitaires : tester individuellement les composants, spécifiés par l'équipe

Tests de recette : spécifiés par le client, mesure l'avancement du projet

Revue du code : vérification du respect d'un ensemble de règles de programmation

Tests d'intégration : valider l'intégration des différents modules entre eux dans leur environnement définitif

Tests d'installation : contrôler la documentation et l'installation

Tests fonctionnels : vérifier la conformité de l'application avec le cahier des charges

Tests IHM : vérifier la charte graphique

Tests de configuration

Tests de performance : valider la capacité des logiciels

Tests d'endurance : vérifier le comportement du système dans le temps

Test de charge : vérifier que le système se comporte comme prévu

Test de stress : vérifier le comportement du système dans des conditions hors limites définis

Test de non-régression : vérifier que les modifications n'ont pas altérées

3 catégories de tests unitaires :

- Cas nominaux (succès)
- Cas d'erreurs
- Cas limites

Génération de classe : nom+ « Test » hérite de TestCase

Méthode de test : public ne renvoie rien

Classe Assert :

- assertEquals(a,b) test l'égalité entre ses deux paramètres
    - a : résultat attendu
    - b : résultat obtenu
- assertFalse et AssertTrue
- assertNotNull et assertNull
- assertNotSame et assertSame : test les références
- fail : fait échouer le test sans condition