ISTQB-Résumé

Chapitre 1 Fondamentaux des tests

Remarques : les éléments détaillés dans les autres chapitres ne sont pas repris

Erreur/Défaut/Défaillance

Un être humain peut faire une erreur (= méprise), qui produit un défaut (= bug) dans le code, dans un logiciel ou un système, ou dans un document.
Si un défaut dans du code est exécuté, le système n'effectuera pas ce qu'il aurait dû faire (ou fera ce qu'il n'aurait pas dû faire), générant une défaillance.

Tous les défauts ne produisent pas des défaillances

Origines des défauts : humaines, environnementales, manque de communication, manque de stabilité du besoin

Vérification / Validation

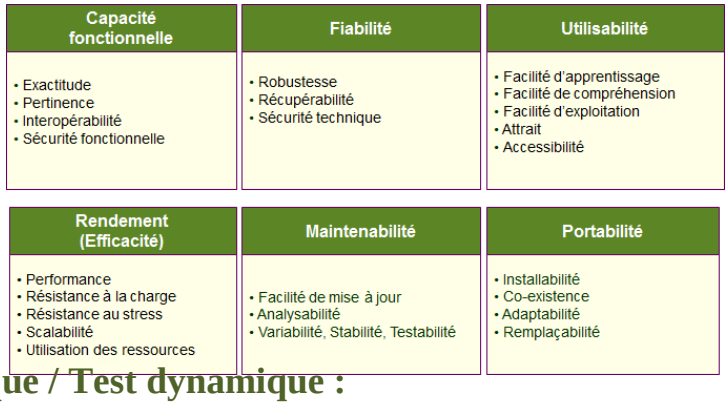
Vérification :
Confirmation par l'examen et la fourniture de preuves objectives que des exigences spécifiées ont été remplies [ISO 9000].
➔ On contrôle par rapport à des spécifications
➔ Objet de la vérification : confirmation qu'on a bien construit le système (sous-entendu par rapport aux spécifications

Validation :
Confirmation par l'examen et la fourniture de preuves objectives que les exigences, pour un usage ou une application voulue, ont été remplies. [ISO 9000]
➔ On contrôle par rapport à l'usage qui sera fait de l'application
➔ Objet de la validation : confirmation qu'on a construit le bon système

Définition de la Qualité

Qualité	Degré par lequel un composant, système ou processus atteint des exigences spécifiées et/ou des besoins ou attentes des clients ou utilisateurs ➔ Ensemble de critères à atteindre
Gestion de la qualité	Activités pour diriger et contrôler une organisation en ce qui concerne la qualité. ➔ norme, processus, formation, contrôles, tests
Assurance qualité	Partie de la gestion de la qualité qui fournit l'assurance que les exigences qualité seront atteintes [ISO 9000] ➔ contrôles, tests

Les 6 caractéristiques qualités



Norme **ISO 9126**

CFURMP :
Cyril FUMERY – Réunion des Musées Parisiens

Test statique / Test dynamique :

- **Test statique :** Test d'un composant ou système au niveau spécification ou implémentation sans exécution de ce logiciel
- **Test dynamique :** Test qui nécessite l'exécution du logiciel d'un composant ou système

Les tests dynamiques et les tests statiques sont complémentaires :

- Ils trouvent des anomalies
- Mais pas nécessairement les mêmes

Les tests dynamiques identifient généralement des défaillances.
Les tests statiques identifient généralement des défauts.

Les 4 objectifs de tests : TAFP

TAFP

1- Trouver des défauts
2- Acquérir de la confiance sur le niveau de qualité
3- Fournir de l'information utile aux prises de décision
4- Prévenir des défauts

Test ad-hoc :

Test effectué de manière informelle

Test ad-hoc <contraire de> Test structuré
Test ad-hoc <> Test exploratoire

Déboguer

Activité de développement : analyser et supprimer les causes de la défaillance

Les 7 Principes généraux des tests

Principe 1 – Les tests montrent la présence de défauts mais ne peuvent pas en prouver l'absence.
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

Processus fondamental de test

Définitions

Politique de tests : Un document de haut niveau décrivant les principes, approches et objectifs majeurs de l'organisation ayant trait aux tests.

Stratégie de test : Un document de haut niveau définissant, pour un programme, les niveaux de tests à exécuter et les tests dans chacun de ces niveaux (pour un ou plusieurs projets).

Condition de test : Un article ou événement d'un composant ou système qui pourrait être vérifié par un ou plusieurs cas de tests; p.ex. une fonction, une transaction, un attribut qualité ou un élément de structure.

Cas de test : Un ensemble de valeurs d'entrée, de pré-conditions d'exécution, de résultats attendus et de post-conditions d'exécution, développées pour un objectif ou une condition de tests particuliers, tel qu'exécuter un chemin particulier d'un programme ou vérifier le respect d'une exigence spécifique [d'après IEEE 610]

Registre de test : un enregistrement chronologique des détails pertinents sur l'exécution des tests➔ « test lab » on recense les scénarios avec le statut d'exécution et la date

Suite de tests : un ensemble de plusieurs cas de tests pour un composant ou système sous test, où les post-conditions d'un test sont souvent utilisées comme pré-conditions du test suivant.

Procédure de test / Spécification de procédure de test : un document spécifiant la séquence d'actions pour l'exécution d'un test. Aussi connu sous le terme script de test ou script de tests manuel [d'après IEEE 829]

Rapport de synthèse de tests : un document synthétisant les activités et résultats de tests. Il contient aussi une évaluation des articles de tests correspondants par rapport aux critères de sortie [d'après IEEE 829].

Testware : Artefact produit pendant le processus de test afin de planifier, concevoir et exécuter les tests, tel que la documentation, les scripts, les entrées, les résultats attendus, les procédures de mise en place et de nettoyage, les fichiers, bases de données, environnements et tout logiciel ou utilitaires supplémentaire utilisé dans les tests➔ Tous les documents produits (tests, reporting...)

Base de tests : Tous les documents à partir desquels les exigences d'un composant ou système peuvent être déduites. La documentation sur laquelle les cas de tests sont basés : spécifications, exigence, …

Article de test : l'élément individuel devant être testé. Il y a généralement un objet de tests et plusieurs articles de test.

Objet de tests : Le composant ou système qui doit être testé. L'application ou le batch à tester.

Les 5 activités de tests

Critères de sortie : l'objectif des critères de sortie est de définir quand arrêter les tests
➔ on arrête de tester quand le risque résiduel est maîtrisé (risque résiduel=anomalies non corrigées et ce qui n'a pas été testé).
Il sont définis dans le plan de test

Remarques

- Planification des tests : organisation, plan de test
- Contrôle des tests : suivi de l'avancement, décision et action corrective, reporting
- Evaluation des critères de sorties :
  - s'ils sont atteint élaboration du rapport de synthèse (compte-rendu de test)

    - s’ils ne sont pas atteint exécution de test complémentaire
- Clôture des tests : retour d’expérience et archivage du testware

Remarque importante :
Le rapport de synthèse (compte-rendu de test) est rédigé à la fin de l‘activité :
    - **Evaluation des critères de sorties**
Et NON dans l’activité : Clôture des tests

Critères de la testabilité d’une exigence

Opérable : on peut définir des étapes pour dérouler un cas de test
Observable : le résultat du test est identifiable et qualifiable
Simple : exprimée d’une manière compréhensible

Mnémotechnique : OOS 117

ISTQB-Résumé

Chapitre 2 Tester pendant le cycle de vie logiciel

Résumé

Les modèles de développement logiciels
- Modèle en V
- Modèle de développement itératif
- Tester au sein d'un modèle de cycle de vie

Niveaux de tests
- Tests de composants
- Tests d'intégration
- Tests système
- Tests d'acceptation

Types de tests
- Tests des fonctions
- Tests des caractéristiques des produits logiciels (Non fonctionnelle)
- Tests de la structure / architecture logicielle
- Tests liés au changement

Tests de maintenance

3 notions structurantes :
- Niveaux de test
- Types de test
- Test de maintenance

En préalable

Deux grands principes :
-Les tests n'existent pas de façon isolée : les activités de tests sont liées aux activités de développement logiciel
-Les différents modèles de développement nécessitent des approches de tests différentes

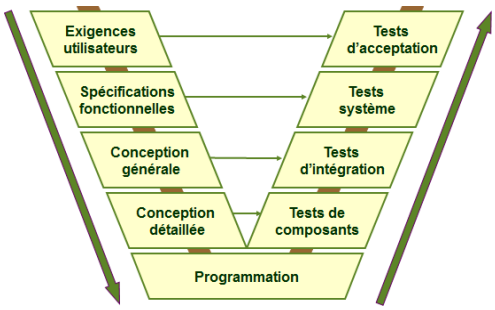
Cycle de vie logiciel
Période entre le Mûrissement du besoin d'un logiciel jusqu'à son retrait.

COTS
Commercial Off-the-shelf Software: Logiciel acheté tout fait et qui peut être intégrée à une solution logicielle maison.

Mnémotechnique : commercial chef software

Modèles de développement logiciels

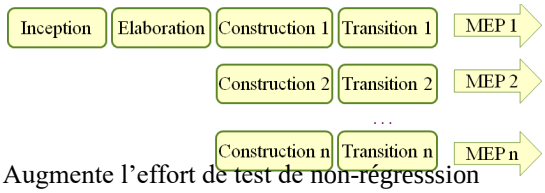
Modèle séquentiel – Exemple : Modèles en V



Les activités sont effectuées de manière séquentielle. Dans le modèle en V, pour chaque activité de conception correspond une activité de test.

Modèles itératif

Exemple du RUP



Augmente l'effort de test de non-régresssion

Autres exemples
- Prototypage
- Rapid Application Development (RAD)
- Modèles Agiles (Scrum, XP)

Bonnes pratiques

- Chaque niveau de test a des objectifs de tests spécifiques pour ce niveau
- A chaque activité de développement, correspond une activité de test
- L'analyse et la conception des tests pour un niveau de test devraient commencer pendant l'activité correspondante de développement
- Les testeurs doivent être impliqués dans la revue des documents aussitôt que des brouillons sont disponibles dans le cycle de développement

Niveaux de tests

Résumé

Niv. de tests	Objectifs	Commentaires
Tests de composants	- Chercher les défauts et vérifier le fonctionnement des composants	Bouchons, conducteurs, simulateurs, harnais de test
Tests d'intégration	- Tester les interfaces entre les composants - Tester les interactions avec l'OS, le matériel, …	Stratégie d'intégration : - Top down - Bottom up - Par tâche fonctionnelle - Big bang
Tests système	- Vérifier le comportement d'un système/produit complet	
Tests d'acceptation	- Prendre confiance dans le système - Evaluer si le système peut être mis en production	Sous niveaux de tests : - Tests d'accep. utilisateur - Tests d'accep. opérationelle - Tests d'accep. contractuelle et réglementaire - Test alpha et test beta

Approche

Les 3 premiers niveaux de test correspondent à la granularité des tests :

- Test des composants : programme par programme
- Test d'intégration : faire fonctionner les programmes de l'application ensemble
- Test systèmes : faire fonctionner l'application avec les autres applications
- Test d'acceptation : recette du client

Remarque très importante :

Tests d'intégration : tests d'assemblage des composants de l'application ou du batch en cours de test

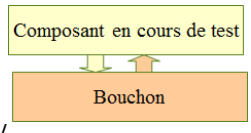
Test de composant

Composant : Un élément logiciel minimal qui peut être testé isolément.
Les tests de composants cherchent des défauts et vérifient le fonctionnement des composants qui sont testables séparément.

Test en isolation : Test des composants individuels indépendamment des autres composants, ces derniers étant simulés par des bouchons ou pilotes si besoin.

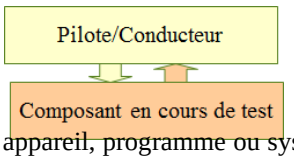
Harnais de tests : Un environnement comprenant des bouchons et des pilotes, nécessaires pour exécuter un test.

Bouchon :



Le bouchon est appelé : « bouchon ! bouchon ! »

Pilote / Conducteur :

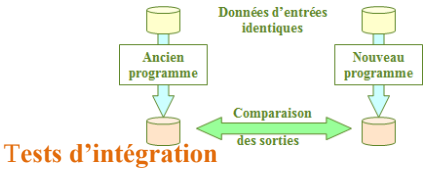


Simulateur : Un appareil, programme ou système utilisé pendant les tests, qui se comporte ou fonctionne comme un système donné à la réception d'entrées contrôlées

Développement piloté par les tests = Tester d'abord
Intérêt de la démarche :
Lors de la conception des tests, le développeur peut identifier des difficultés particulières (règle de gestion complexe, valeurs limites,…). Il pourra prendre en compte ces retours d'expérience lors de la programmation.

Tests dos à dos :

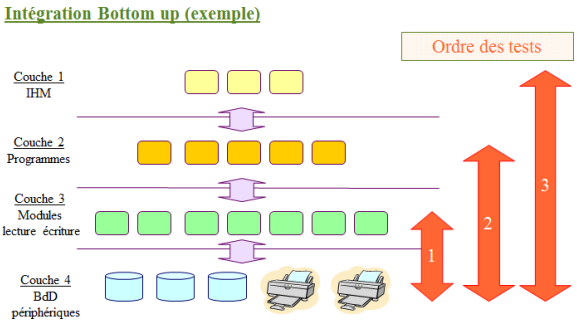
Pour les tests iso fonctionnels, par exemple montée de version d'OS

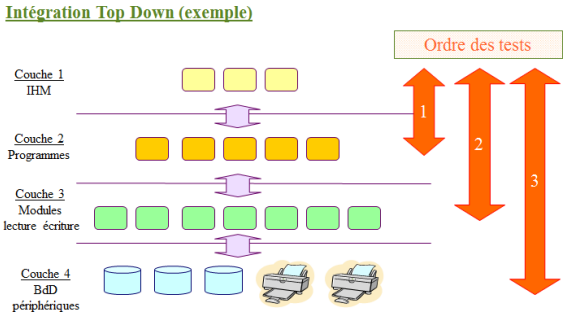


Tests d'intégration

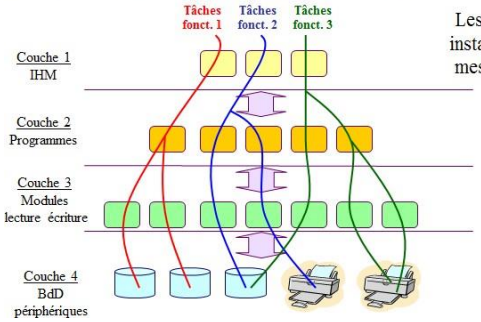
Plus la portée de l'intégration est vaste, plus il devient difficile d'isoler les défauts liés à un composant ou un système particulier.

Stratégie d'intégration :



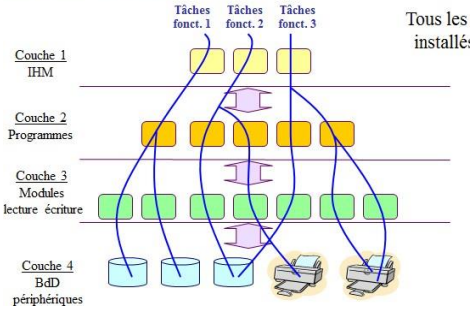


Intégration par Tâches fonctionnelles (exemple)



Les modules sont installés au fur et à mesure du besoin

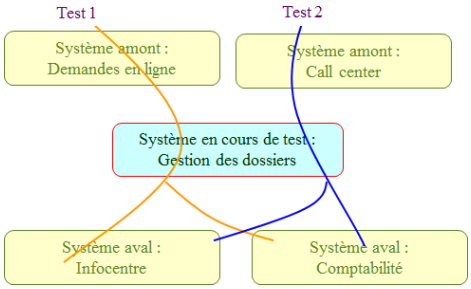
Intégration Big bang



Tous les modules sont installés en une fois

Tests système

Test de bout en bout
Environnement de test : proche de la production
Testeurs : en général, une équipe dédiée



Tests d'acceptation

Objectif principal : prendre confiance dans le système.

La recherche d'anomalies n'est pas l'objectif principal des tests d'acceptation.

Les tests d'acceptation relèvent souvent de la responsabilité :

- des clients
- des utilisateurs d'un système
- d'autres responsables parties prenantes

Scénario d'utilisation = Cas d'emploi = Cas d'utilisation
Cas d'utilisation : une séquence de transactions dans un dialogue entre un utilisateur et le système avec un résultat tangible.

4 sous-niveaux de tests :

- Tests d'acceptation utilisateur ➔ UAT
- Tests d'acceptation opérationnelle ➔ acceptation du système par les exploitants
- Tests d'acceptation contractuelle et réglementaire
- Tests alpha et beta

	Objectif	Testeur	Localisation
Test alpha = tests d'accep. usine	Obtenir du feed-back des clients potentiels de manière anticipée	Clients potentiels	Site de l'organisation effectuant les développ.
Test beta = tests sur le terrain = tests d'accep. sur site			Site des clients potentiels

Types de tests

Un type de tests est focalisé sur un objectif de tests particulier.

Types de tests	Niveaux de tests	Techn. de conception	Commentaires
Tests des fonctions (tests fonctionnels)	Tous niveaux de test	Boîte noire	Comprend aussi les tests de sécurité fonctionnelle et les tests d'interopérabilité
Tests des caractéristiques non fonctionnelles	Tous niveaux de test	Plupart du temps boîte noire	
Test de la structure / architecture logicielle	Tous niveaux mais principalement : Tests de composant et tests d'intégration	Boîte blanche	
Test liés au changement - Test de confirmation = retest - Test de régression			S'appliquent aux tests fonctionnels, non-fonctionnels et structurels

Tests des fonctions

Les fonctions sont ce que « fait » le système.
Test de la capacité fonctionnelle. Le « C » du CFURMP

Tests des caractéristiques non fonctionnelles

Les caractéristiques non fonctionnelles sont le « comment » du système.
Test de la capacité fonctionnelle. Les « FURMP » du CFURMP.

Test de la structure

Boite blanche. Test optimal avec des techniques de couverture de test

Test de confirmation (= retest)

Quand un défaut est détecté et corrigé, le logiciel devrait être re-testé pour confirmer que le défaut original a été correctement ôté. Ceci est appelé test de confirmation ou retest.

Test de régression

Les tests de régression sont exécutés lors d'une modification

- du logiciel
- de son environnement

Les tests de régression sont de bons candidats à l'automatisation.

Test de maintenance

Tests maintenances

Une fois déployé, un système logiciel est souvent en service pendant des années, voire des dizaines d'années. Pendant ce temps, le système, son paramétrage et son environnement sont fréquemment corrigés, modifiés ou étendus.

Analyse d'impacts

Déterminer comment un système existant est affecté par les changements est appelé analyse d'impacts, et est utilisé pour décider de la quantité de tests de régression devant être exécutés.
L'analyse d'impacts peut être utilisée pour déterminer les suites de tests de régression.

Types de maintenance

- Maintenance à chaud
- Maintenance planifiée

Coût de la maintenance

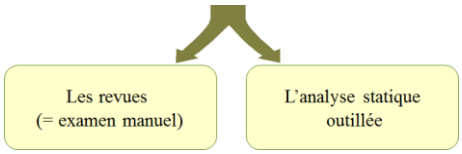
Environ 75 % du coût total

ISTQB-Résumé

Chapitre 3 Techniques statiques

Tests statiques= aux revues de documents ou à l'analyse statique outillée du code ou de la documentation du projet
Les défauts détectés pendant les revues effectuées tôt dans le cycle de vie sont souvent bien moins coûteux à corriger que les défauts détectés lors de l'exécution des tests

Types de techniques statiques



Les revues

Les bénéfices des revues :

- une détection et une correction anticipées des défauts
- 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
- moins de défauts (prévient la multiplication des anomalies)
- une meilleure communication
- la détection des omissions

Par exemple, dans des exigences, dont la détection pendant les tests dynamiques est peu probable.

Les différents types de revues varient d'informel à systématique.

Les 6 phases d'une revue

Planification➔choisir le personnel, définir les critères d'entrée et de sortie
Lancement➔distribuer les docs, expliquer les processus
Préparation individuelle➔ Préparer la réunion de revue en revoyant le(s) document(s)
Réunion de revue➔ Examiner, Evaluer, discuter
Correction des défauts➔ On corrige les défauts
Suivi➔vérifier que les défauts ont bien été traités

Les 5 rôles

Manager	décide l'exécution des revues
Réviseur	vérificateurs ou inspecteurs
Modérateur	d'intermédiaire entre les différents points de vue et est souvent la personne sur qui repose le succès d'une revue
Scribe	documente tous les aspects, problèmes et points ouverts identifiés pendant la réunion
Auteur	'auteur ou la personne à qui incombe la responsabilité principale du ou des document(s) à revoir. En résumé c'est celui qui a écrit le programme ou le document qui est revu.

Les 4 types de revue

	Revue informelle	Relecture technique	Revue technique (Revue de pairs)	Inspection
Formalisation	Pas de processus formel	Quasiment informelle à très formelle	Quasiment informelle à très formelle	Processus formel
Menée par	Auteur	Auteur	Idéalement : modérateur	Modérateur
Audience des revues	Revue entre collègues (buddy review)	Pairs	- En général des pairs - Auteur absent - Présence optionnelle du management	- En général des pairs - Auteur absent
Spécificités	Documentation éventuelle des résultats	- Scénario - Répétition à blanc	- Check list (optionnelle) - Rapport de revue	- Critères d'E/S - Métrique - Rapp. d'inspection
Objectif principal	Manière bon marché d'obtenir des résultats	- Apprendre - Gagner en compréhension - Trouver des défauts	- Discuter, décider - Evaluer des alternatives - Trouver des défauts - Résoudre des pb techniques - Vérifier la conformité aux spéc., règlement., standards	Trouver des défauts

Revue des pairs = Les relectures techniques, les revues techniques et les inspections
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

Analyse statique outillée

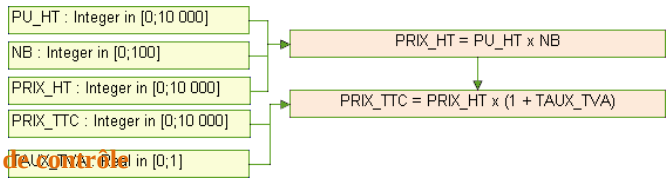
Objectifs

L'objectif de l'analyse statique =revue de code est de trouver des défauts :
- dans le code source
- dans les modèles logiciels

Les outils d'analyse statique analysent : Le code du programme

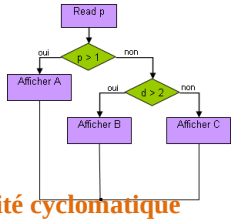
Analyse du flux de données

Une forme d'analyse statique basée sur la définition et l'usage des variables.



Analyse du flux de contrôle

Une forme d'analyse statique basée sur une représentation de séquences d'évènements (chemins) dans l'exécution d'un composant ou système



Complexité cyclomatique

➔ calcule la complexité d'un programme (nombre de if, de while…)
Cc=L-N+2P
Cc = complexité cyclomatique

L = le nombre d'arêtes du graphe
N = le nombre de nœuds du graphe
P = le nombre de composantes connexes du graphe

Complexité lexicale

Indique la complexité du programme en termes de vocabulaire :

- nombre d'opérandes, nombre d'opérateurs

Compilateur :

Un outil logiciel qui traduit un programme exprimé dans un langage de haut niveau dans son équivalent en langage machine.
Le compilateur effectue aussi des contrôles, par exemple de syntaxes

ISTQB-Résumé

Chapitre 4 Techniques de conception de tests

Résumé

Prix Absolu Toute Taxe Comprise PATTC

Introduction

- Spécification de conception de tests : liste des conditions de test
- Spécification de cas de test : objectifs, entrées, actions de tests, résultats attendus et préconditions d'exécution
- Procédure de test / Spécification de procédure de test : = script de tests = scénario de test

- Traçabilité des conditions de test : lien cas de test/conditions de test

- Oracle de tests : toutes sources permettant de définir le résultat attendu (expert, exigence, …)

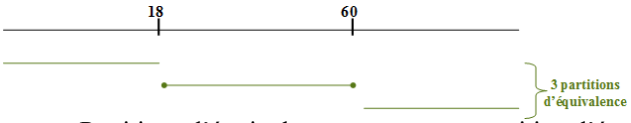
- Résultat attendu : idéalement défini préalablement à l'exécution. Sinon la validation du test repose sur le testeur

- Objet des techniques de conception de test :
  - Identifier les conditions de tests
  - Identifier les cas de tests
  - Identifier les données de tests

Techniques basées sur les spécifications ou boîte noire

Partitions d'équivalence/Analyse des valeurs limites

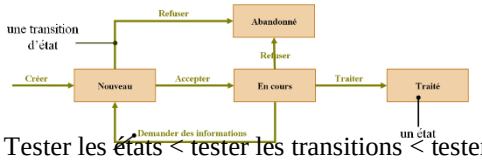
Partition d'équivalence = Classe d'équivalence



- Partitions d'équivalence : un test par partition d'équivalence
- Analyse des valeurs limites : un test à chaque borne des partitions d'équivalence

Partitions d'équivalence non valide => message d'erreur attendu

Transition d'état



Tester les états < tester les transitions < tester les chemins

Test de Chow

- 1-aiguillage couvre toutes les paires de transitions possibles
- 2-aiguillage couvre tous les triplets de transitions possibles

Tables de décisions

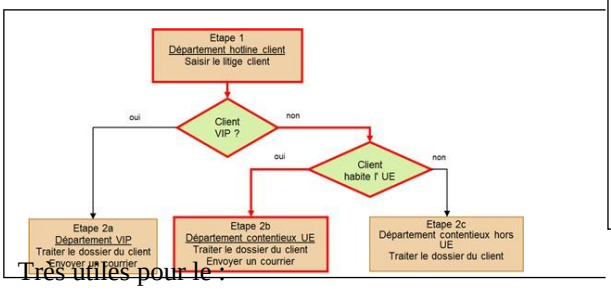
Conditions	Résident en France	FAUX	VRAI	VRAI	VRAI
	Age entre 18 et 60 ans	Pas d'impact	FAUX	VRAI	VRAI
	Fumeur	Pas d'impact	Pas d'impact	FAUX	VRAI
Actions	Assurer le client ?	FAUX	FAUX	VRAI	VRAI
	Faire 7 % de réduction ?	FAUX	FAUX	VRAI	FAUX

Selon ISTQB : Tester une table de décision = tester au moins une fois chaque colonne
Cependant, l'approche peut être complexifiée si système critique

Tests de cas d'emploi

Cas d'emploi = Cas d'utilisation = Processus métier
Par exemple : Création d'un client, relance d'un client, gestion d'un litige

Technique de conception des tests peu développée au niveau Fondation



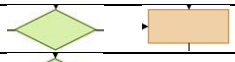
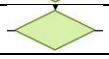
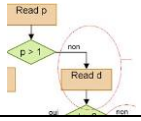
Notion de scénarios dominants : ils représentent la majorité des cas.
Exemple : en rouge dans le workflow ci à gauche

Très utiles pour le :

- niveau de test « acceptation »
- niveau de test « test système » : détection des problèmes inter applications

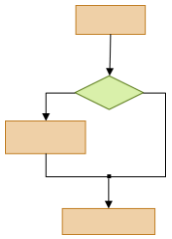
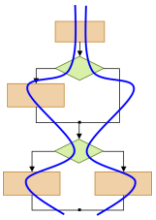
Techniques basées sur la structure ou boîte blanche

Définitions 1

Instruction	
Décision	
Branche (=Arc)	
Chemin	Entre une entrée et une sortie

Puissance des tests

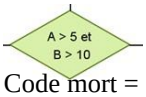
Schéma à avoir en-tête

		Tester les décisions > tester les instructions Tester les décisions = tester les branches Tester les chemins > tester les décisions

Tester les décisions, c'est tester au moins une fois chaque sortie

Définitions 2

Décision à condition multiple



Code mort = Code inatteignable

Notions de « Couverture de test » et de « Conditions de test »

Couverture de test (%) = Ce que j'ai testé/ Sur ce que je dois tester)

Exemple 1: Transition de test
-1- Si j'ai décidé (stratégie) de tester tous les états
Couverture de test (%) = Etats testés/ Nb d'état
Conditions de test : liste des états
-2- Si j'ai décidé (stratégie) de tester les 1-aiguillage
Couverture de test (%) = Paires de transitions / Nombre de paires de transitions d'état possible
Conditions de test : liste des paires de transitions d'état possible

Exemple 2 : Boite blanche

-3- Si j'ai décidé (stratégie) de tester toutes les décisions
Couverture de test (%) : Résultats de décision / Nb total de résultats de décision
Conditions de test : liste des résultats de décisions

Notion de couverture : - liée à l'exécution
- Pas de notion d'anomalies
-

Techniques basées sur l'expérience

Test exploratoire
Le testeur crée les cas de test en explorant l'application < > ad-hoc
Charte de test : le CP test circonscrit le périmètre } Cadrage de l'exploration
Délai : le CP définit un délai puis fait un REX avec le testeur

Peuvent être fait aussi en complément d'autres techniques de conception des tests. Objectif : on a rien louper ?

Estimation d'erreur
Basée sur l'expérience du testeur « je sens qu'ici il va y avoir des anomalies »

Attaque par faute

Par exemple, le testeur va se focaliser sur la justesse des calculs, peut être sur les règles d'arrondi.

ISTQB-Résumé

Chapitre 5 – Gestion des tests

Organisation des tests

Equipe indépendante

- Objectivité
- Trouve d'autre défaut que les développeurs

- Plus de préoccupation qualité dans le développement
- Equipe de test = Goulet d'étranglement

Responsable de test
- Organise
- Contrôle
- Suit

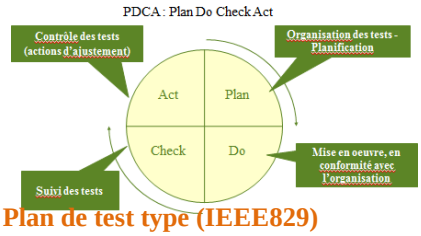
Testeur
- Participe au plan de test
- Conçoit les tests
- Exécute les tests

Estimation et planification des tests

Positionnement

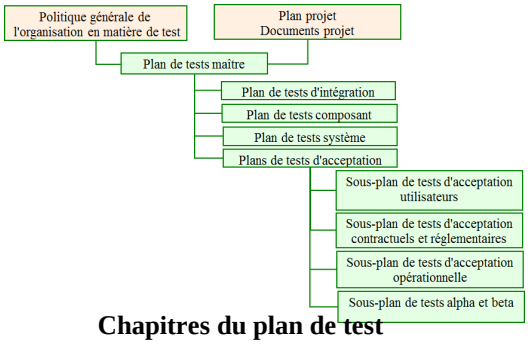
La planification du test est une activité continue.

Roue de Deming =(PDCA)



Plan : organiser/planifier
Do : faire comme l'organiser
Check : suivre l'avancement
Act : prendre des mesures correctives

Plan de test type (IEEE829)



- Politique générale de test : transverse à tous les projets
- Plan de test : lié à un projet
- Plan de test maître : éléments transverses à tous les niveaux de test

Chapitres du plan de test

- Identification du document
- Introduction
- Livrables
- Détail des tâches

- Périmètre des tests
- Caractéristiques à tester
- Caractéristiques non prises en compte
- Approche de test
- Critères d'entrée et de sortie
- Critères de suspension et de reprise
- Ressources matérielles
- Responsabilité
- Ressources humaines
- Planning
- Risques et contingences
- Approbation

NB : les scripts de test ne font pas partie du plan de test

Critères de sortie

- Les risques résiduels
- L'estimation de la densité des anomalies ou des mesures de fiabilité.
- Le coût
- Un calendrier
- Des mesures d'exhaustivité

Estimation des tests

Estimation des tests = estimation des charges

Positionnement : Estimation des tests/Affectation des ressources/Etablissement du planning

Deux méthodes

- Top down : on une enveloppe globale. on la ventile sur les tâches grâce à des coefficients de répartition
- Bottom up : un expert estime chaque tâche. Puis on consolide sur les activités et projet (somme)

Suivi et contrôle du déroulement des tests

Positionnement

- Suivi : Mesurer l'avancement
  = Collecte des informations d'avancement
  - Avancement de la rédaction des cas de test
  - Avancement en exécution des cas de test
  - Bilan des anomalies
  - Point Charge/planning

- Contrôle : Prendre des actions correctives
  Leviers :
  - Planning
  - Ressource humaines
  - Périmètre
  - Organisation
  - Ressource machine
  - Qualité

Rapport de synthèse

Rapport de synthèse = Compte-rendu de test

Chapitres

- Identifiant du document
- Résumé
- Ecarts par rapport aux documents de référence
- Evaluation de l'exécution
- Résumé des résultats
- Evaluation des résultats
- Résumé des activités
- Approbations

Gestion de configuration

Objet :

Gérer les relations entre les différentes versions des documents et logiciels d'un projet.
Exemple :

- le cahier de test v1.2 a été élaboré sur la base de dossier d'exigence v1.5 – traçabilité horizontale
- le cahier de test v1.5 correspond au cahier de test v1.4 avec prise en compte des retours de M. Dupont – traçabilité verticale

Base de référence (Base line) :

Dossier d'exigences		Plan de tests		Spécification de tests
DEX V1.0	⟷	PDT V1.0	⟷	SDT V1.0
DEX V2.0	⟷	PDT V2.0	⟷	SDT V3.0

Test et risques

Risque : événement non encore réalisé qui aurait en cas concrétisation des conséquences négatives

2 attributs primordiaux :

- Probabilité
- Impact

Niveau de risque = Probabilité x Impact

Risque projet : Livraison en retard, manque de formation, organisation défectueuse
Risque produit : Risque pouvant être induit par un dysfonctionnement de l'application – criticité des fonctions

Risque produit permet de savoir où tester en priorité et à quelle profondeur
Le test permet de limiter le risque produit.

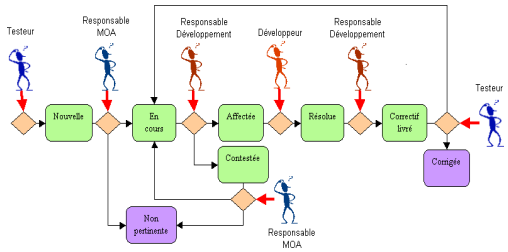
Gestion des incidents

Définitions

Incident : Tout événement arrivant pendant les tests qui requiert une vérification

Une anomalie est un incident.
Dans le cadre des tests, un rapport d'incident s'appelle aussi un rapport d'anomalie

Cycle de vie des incidents (exemple)



Attribut du rapport d'anomalie

- Résultats attendus/effectifs
- Date de détection
- Identification de la configuration
- Etape de détection
- Description de l'anomalie
- Impact
- Référence cas de test et spécifications
- Sévérité
- Urgence de correction
- Etat de l'incident
- Conclusions et Recommandations
- Analyse globale
- Historique des modifications

Sévérité : Impact sur le projet (exemple : test bloqué)

Priorité : Impact sur l'utilisation opérationnelle

ISTQB-Résumé

Chapitre 6 – Outils de test

Définitions préalables

Framework de test

- Bibliothèques réutilisables et extensibles de test qui peuvent être employées pour construire des outils de test =harnais de tests
- Un type de conception d'automatisation des tests (par exemple, pilotés par les données, piloté par mots-clés)

Outil intrusif

Outil dont la mise en place nécessite d'ajouter des lignes de code dans le code de l'application à tester

Effet de sonde

Effet d'un outil intrusif qui fausse les résultats.
Exemple : Ajout de ligne dans le code pour mesurer des temps de réponse. L'ajout de ces lignes fausse qq peu le temps de réponse obtenu.

Classification des outils de test

Aide à la gestion des tests	Aide aux tests statiques	Aide à la spécification des tests
• Outils de gestion des exigences • Outils de gestion des cas de tests • Outils de gestion d'incidents • Outils de gestion de configuration	• Outils de revue • Outils d'analyse statique (D) • Outils de modélisation (D)	• Outils de conception des tests (= outils de génération automatisé des cas de test) • Outils de préparation des données de tests

Aide à l'exécution et à l'enregistrement des tests	Support de performance et de surveillance	Support pour les besoins de tests spécifiques
• Outils d'exécution des tests • Harnais de test (D) • Comparateur de tests • Outils de mesure de couverture (D) • Outils de test de sécurité	• Outils d'analyse dynamique (D) • Outils de tests de performances / de tests de charge/ de tests de stress • Outils de surveillance	• Evaluation de la qualité des données • Test de l'utilisabilité

Outils orientés 100% développeurs/concepteur

- Outil d'analyse statique : de code par exemple
- Outil de modélisation : de base de données par exemple
- Harnais de test : bouchons conducteurs
- Outil de mesure de couverture : de lignes de code couvertes par exemple
- Outil d'analyse dynamique : CPU consommé, hits

Outils d'aide à la gestion des tests

Peuvent s'interfacer avec d'autres outils. Exemple :

- Teslink/Bugzilla

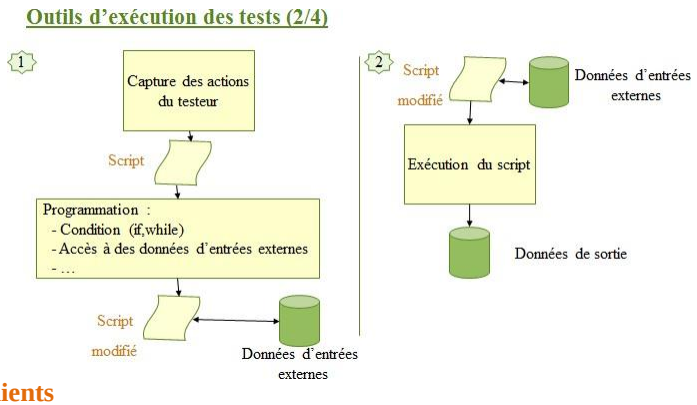
- Quality Center/Quick Test Pro

Outils d'exécution des tests

Nom génériques de ces outils

- CAST : Computer Aided Software Testing
- Outils de capture/rejeu
- Outils d'enregistrement/rejeu

Principes



Inconvénients

- Le script généré comprend uniquement les données exactes d'entrée saisies par le testeur
- Le script peut être instable quand un évènement inattendu intervient
  Par exemple : fenêtre de pub
- Les résultats attendus doivent être ajoutés au script enregistré
- En rejouant un script de test généré, le testeur peut avoir à débugger le script s'il ne fonctionne pas correctement
  Par exemple : suite à un renommage du nom des champs d'une IHM par les développeurs

Coûts/Bénéfices

Bénéfices après la 4e ou 5e exécution

Compétence

Le scripting demande une expertise importante.

Test piloté par les mots clés

Un expert développe des modules correspondant à des fonctionnalités = mot-clé

Par exemple : créer_client

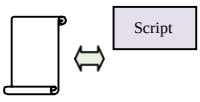
Les testeurs peuvent utiliser ces mots clé pour créer des scripts automatisé

Par exemple : créer_client (M.;Marc;Vlin;14 rue Chax;75017;Paris)

Test piloté par les données

Jeu de données

Le script automatique est joué autant de fois qu'il y a de ligne dans le jeu d'essai

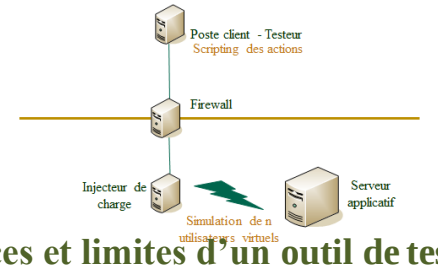


Comparateur de tests

Souvent utilisé avec un outil d'exécution des tests pour valider les résultats.

Outils de test de performance/test de charge/test de stress

Peuvent simuler des milliers d'utilisateurs = utilisateurs virtuels
Utilisent normalement un injecteur de charge



Bénéfices et limites d'un outil de test

- Réduction du travail répétitif
- Répétabilité et cohérence accrues
- Evaluation objective
- Facilité d'accès aux informations

- Attentes irréalistes
- Sous-estimation de l'effort pour la mise en œuvre
- Sous-estimation de l'effort pour obtenir des bénéfices significatifs
- Sous-estimation de l'effort requis pour maintenir les acquis
- Confiance excessive dans l'outil
- Problème de versioning des éléments de test
- Problèmes d'interopérabilité entre les outils
- Risque de faillite de l'éditeur d'outil
- Faible réactivité du support
- Risque de suspension d'un logiciel open-source
- Non gestion d'une nouvelle plate-forme

Mise en place d'un outil de test

Principes

- Evaluation de la maturité de l'organisation
- Evaluation par rapport à des exigences préalablement définies
- Une preuve de concept (POC) pendant l'évaluation (test de l'outil)
- Evaluation du vendeur
- Problématique de support interne
- Evaluation de besoin de formation
- Evaluation du rapport coût/bénéfice

Projet pilote

Objectifs :

- Apprendre l'outil plus en profondeur.
- Evaluation des adaptations du processus de test
- Définition des pratiques standards d'utilisation
- Evaluer précisément le coût/bénéfice

Principes :

- Un sous-ensemble des utilisateurs cible
- Utilisation de manière opérationnelle

Facteurs clés de succès

- Etendre l'outil de façon incrémentale
- Adapter le processus de test à l'outil (et inversement)
- Formation et assistance aux utilisateurs
- Établir des guides d'utilisation

- Implémenter une démarche de REX
- Surveiller l'utilisation de l'outil et les bénéfices recueillis