# Les tests et les plans de tests

# Table des matières

# Table des matières

# Table des matières

# 1. Tester vs Déboguer

- **L'erreur en programmation est normale**

- Tester vs Déboguer
- Tester et déboguer, ce n'est pas la même activité et les responsables sont différents:
  - ✓ Les testeurs testent
  - ✓ Les développeurs déboguent (Déboguer est une activité de développement permettant d'analyser, trouver et supprimer les causes de la défaillance)

# 2. Définition du test informatique

- En informatique, un test désigne une procédure de vérification partielle d'un système.
- Son objectif principal est d'identifier un nombre maximum de comportements problématiques du logiciel.
- Il permet ainsi, dès lors que les problèmes identifiés seront corrigés, d'en augmenter la qualité

➢ https://fr.wikipedia.org/wiki/Test_(informatique)

# 3. ISTQB : International Software Testing Qualifications Board

- L'ISTQB est le Comité international de qualification du test logiciel.
- Cette organisation propose une certification reconnue dans le monde entier.
- Elle a été fondée en novembre 2002 à Édimbourg, comme association à but non lucratif, et est légalement enregistrée en Belgique.
- Sites : https://fr.wikipedia.org/wiki/ISTQB

  https://www.istqb.org/

# 3. ISTQB

- L'ISTQB propose de valider le titre de testeur certifié ISTQB, une qualification standardisée pour le test logiciel.
- Les qualifications validées sont hiérarchisées et suivent des directives d'accréditation et d'examen.
- A ce jour, plus de 500 000 certificats ISTQB ont été délivrés dans le monde et plus de 10 000 en France (cf. RNCP).
- Le comité de l'ISTQB est composé de membres représentant plus de 71 pays.
- L'ISTQB propose trois niveaux de certification : le niveau fondamental, le niveau avancé, le niveau expert

# 4. Les 7 principes généraux des tests

- Principe 1 – Les tests montrent la **présence de défauts**. Les tests peuvent prouver la présence de défauts, mais **ne peuvent pas en prouver l'absence**. Les tests réduisent la probabilité que des défauts restent cachés dans le logiciel mais, même si aucun défaut n'est découvert, ce n'est pas une preuve d'exactitude.

- 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. Plutôt que des tests exhaustifs, nous utilisons **l'analyse des risques et des priorités** pour focaliser les efforts de tests.

# 4. Les 7 principes généraux des tests

- Principe 3 – **Tester tôt**. Pour trouver des défauts tôt, les activités de tests devraient commencer aussi tôt que possible dans le cycle de développement du logiciel ou du système, et devraient être focalisées vers des objectifs définis.

- Principe 4 – **Regroupement des défauts**. L'effort de test devrait être fixé proportionnellement à la densité des défauts prévus et constatés dans les différents modules. **Un petit nombre de modules contiennent généralement la majorité des défauts détectés** lors des tests pré- livraison, ou affichent le plus de défaillances en opération

# 4. Les 7 principes généraux des tests

- Principe 5 – **Paradoxe du pesticide**. Si les mêmes tests sont répétés de nombreuses fois, il arrivera que **le même ensemble de cas de tests ne trouvera plus de nouveaux défauts**. Pour prévenir ce “paradoxe du pesticide”, **les cas de tests doivent être régulièrement revus et révisés**, et de nouveaux tests, différents, doivent être écrits pour couvrir d’autres chemins dans le logiciel ou le système de façon à permettre la découverte de nouveaux défauts

# 4. Les 7 principes généraux des tests

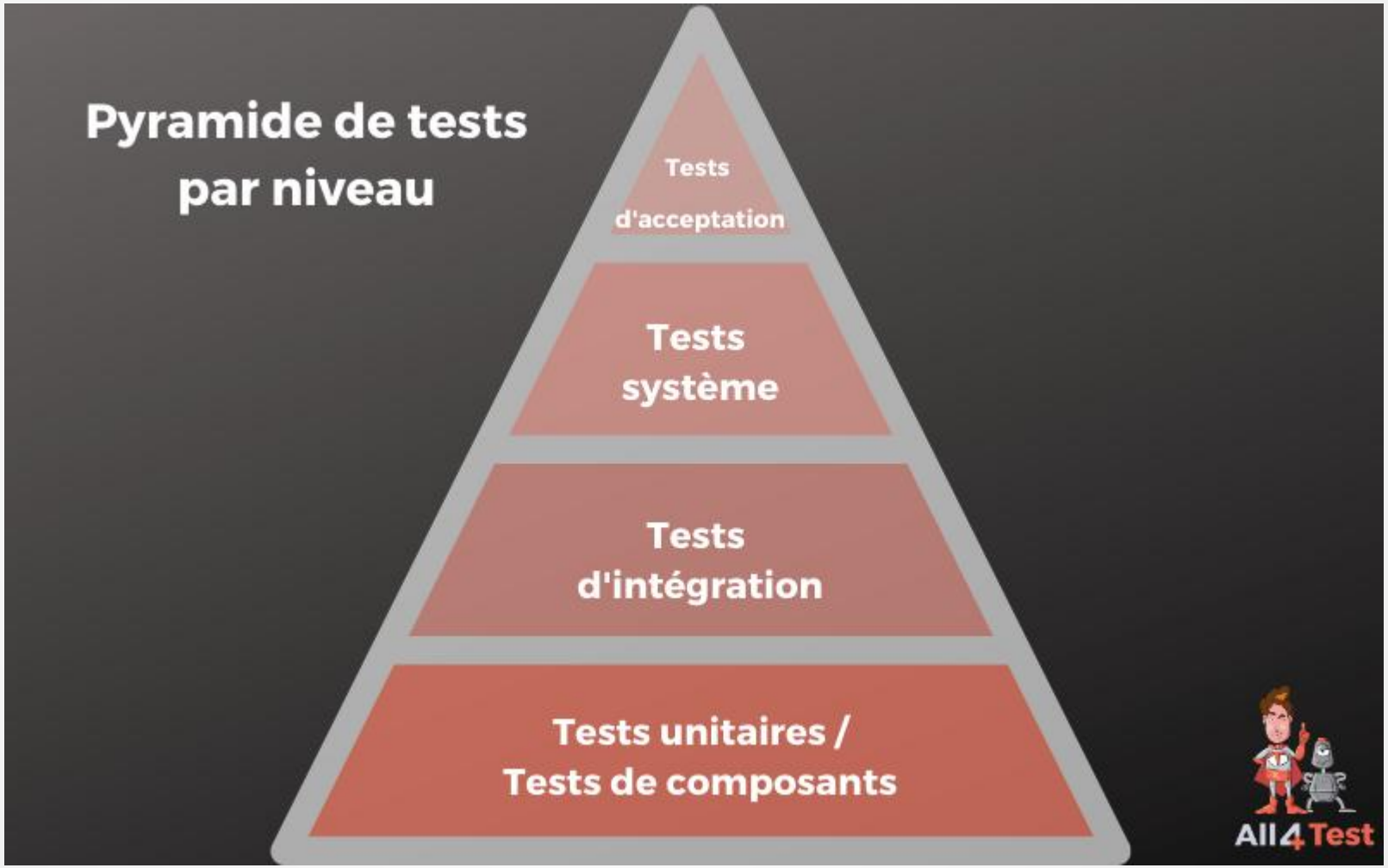
- Principe 6 – **Les tests dépendent du contexte**. **Les tests sont effectués différemment dans des contextes différents.** Par exemple, les logiciels de sécurité critique seront testés différemment d'un site de commerce électronique.

- Principe 7 – **L'illusion de l'absence d'erreurs**. Trouver et corriger des défauts n'aide pas si le système conçu est inutilisable et ne comble pas les besoins et les attentes des utilisateurs

# 5. Quatre niveaux de test

- Le CFTL, Comité Français du Test Logiciel (https://www.cftl.fr) identifie quatre **niveaux** de test :
  1. Tests unitaires
  2. Tests d'intégration
  3. Tests système
  4. Tests d'acceptation

# 5. Quatre niveaux de test



# 5. Quatre niveaux de test

1. **Tests unitaires** (https://fr.wikipedia.org/wiki/Test_unitaire) (ou test de composants).
   - Ils doivent pouvoir être refait si les résultats ne sont pas conformes à ceux attendus
   - Ils permettent de tester un composant, ou un bout de code, isolé de ses dépendances.
   - Ils doivent permettre une couverture de code importante
   - Leur jeu de tests : les cas généraux et tous les cas particuliers.
   - Il doivent être en échec si un seul cas est non conforme

# 5. Quatre niveaux de test

2. **Tests d'intégration** (anciennement test technique ou test d'intégration technique)

- Ils permettent de vérifier que des bouts de code isolés fonctionnent bien ensemble. Mais on est encore sur des fonctionnalités, pas l'application complète
- Exemple : Dans un blog, suite à la saisie d'un formulaire de commentaire, est-ce que le commentaire a bien été crée dans la base de données

# 5. Quatre niveaux de test

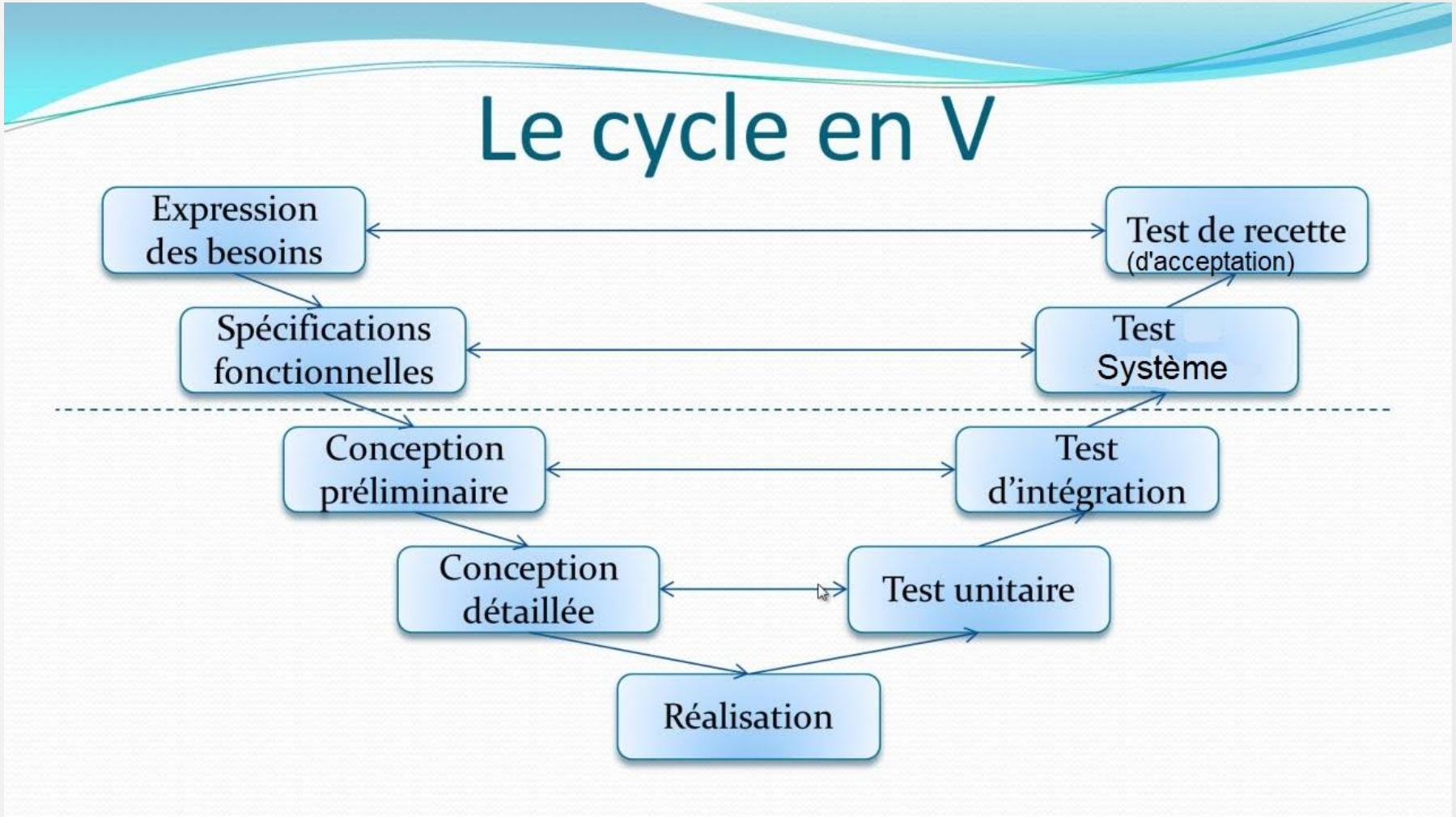
3. **Tests système** (anciennement test fonctionnel ou test d'intégration fonctionnel ou homologation).

- Ils permettent de tester l'application tout entière, avec tous les serveurs concernés, comme les serveurs de BDD
- Ils peuvent utiliser les tests d'intégration, mais avec des serveurs ayant une configuration proche de celle de l'environnement de production

# 5. Quatre niveaux de test

4. **Tests d'acceptation** (anciennement test usine ou recette).
   - Ils permettent de s'assurer que l'application répond bien au besoin fonctionnel. Ils sont **manuels** et ils sont effectués par la MOA sur l'environnement de préproduction (UAT : User Acceptance Test)

# 6. Leur positionnement



# 7. Test amont/test aval

- Par ailleurs, on parle aussi de niveau de *test amont* pour désigner les niveaux unitaire et intégration, et de niveau de *test aval* pour désigner les niveaux système et acceptation.
- De même, en France, le terme « phase de test » est parfois utilisé à la place de « niveau de test », mais il ne respecte pas le vocabulaire du CFTL.

# 8. Pourquoi si peu de tests sont effectués

- Pas motivant
- Prend du temps
- Impression de perdre son temps

# 9. Pourquoi faut t'il faire des tests

- Robustesse de l'application
- Intégrité des données
- Image de marque de l'entreprise
- Eviter les reprises et leurs coût et stress (en cas de problème en production)

# 10. Classifications des tests

- Leur mode d’exécution
- Leurs modalités
- Leurs méthodes
- Les exigences de qualité (iso 25019)
- Leur gravité

# 11. Les modes d'exécution des tests

- Manuels
  - ✓ Exemple :  test d'une UI
- Automatisés
  - ✓ Exemple : tests unitaires avec un logiciel d'automatisation de tests unitaires

# 12. Les modalités de test

- Tests statiques : avant l'exécution
  - Revue de code (simple et croisée sur tout le code ou inspection d'une partie du code précise)

- Tests dynamiques : nécessite l'exécution du code binaire

# 13. Les méthodes de tests

- Structurel (boite blanche)

une boîte blanche (de l'anglais white box), ou boîte transparente, est un module d'un système dont on peut prévoir le fonctionnement interne car **on connaît les caractéristiques de fonctionnement de l'ensemble des éléments qui le composent.**

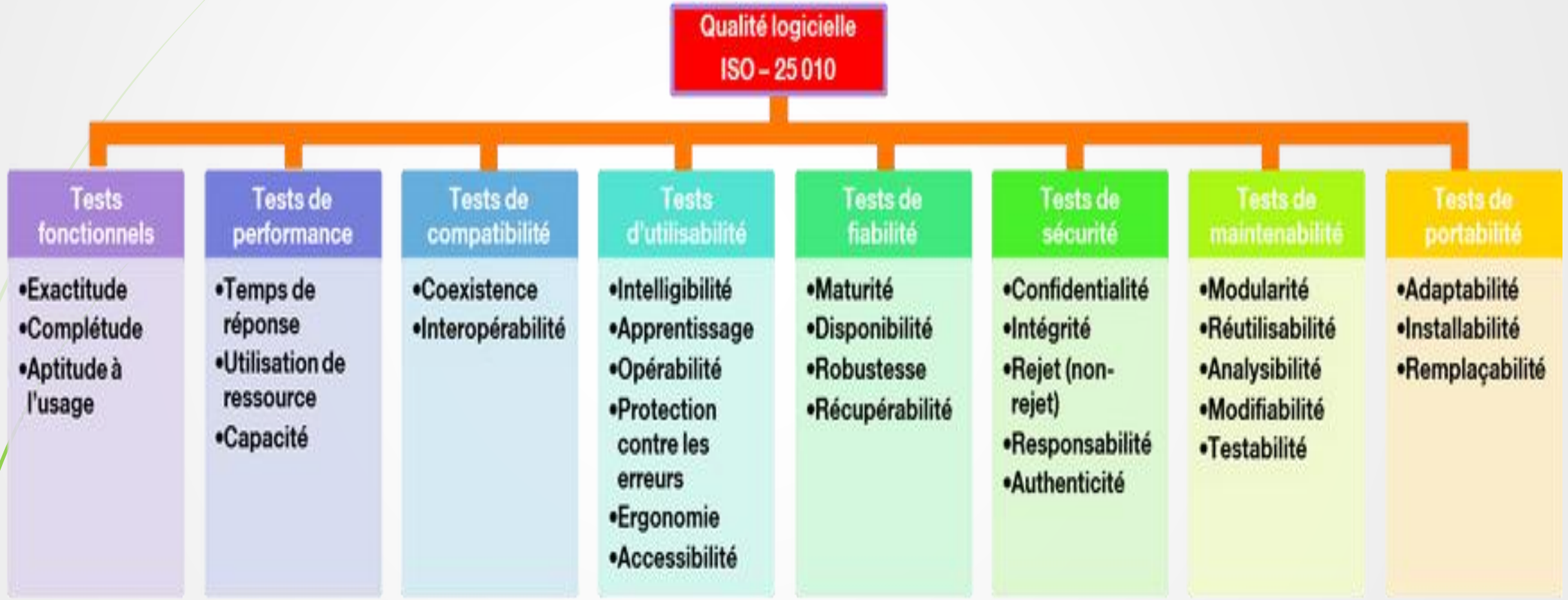
# 13. Les méthodes de tests

- Fonctionnels (boite noire)

Le test de la boîte noire, ou test de la boîte opaque, est utilisé en programmation informatique et en génie logiciel pour **tester un programme en vérifiant que les sorties obtenues sont bien celles prévues pour des entrées données**.

L'expression « boîte noire », ou « boîte opaque », vient du fait que les composants et **les processus du dispositif de traitement ne sont pas visibles** ou transparents et que le programme testé n'est pas étudié.

# 14. Les exigences de qualité (iso 25019)



# 15. Exemples de types de tests

- Test de non régression : s'assurer que les évolutions et les corrections effectuées dans une nouvelle version n'ont pas entraîné de régressions sur les fonctionnalités existantes.

# 15. Exemples de types de tests

- Test d'intrusion ou unitaire de sécurité (pentest) : évaluer la sécurité d'un SI en effectuant des attaques pour rechercher les vulnérabilités

  => audit réalisé par des testeurs spécialisés. On ne recherche pas le bogue, mais on le provoque et on regarde ce que cela implique

# 15. Exemples de types de tests

- Test **maintenabilité**
  - Modularité : logiciel bien **divisé en « composants »** (plus ou moins) indépendants les uns des autres
  - Réutilisabilité : vérifier que le code développé peut être **réutilisé sur d'autres logiciels**.
  - Analysibilité : permettre de savoir **quels sont les impacts d'un changement sur l'application**
  - Modificabilité : dans quelles mesures l'application peut **évoluer sans y introduire de régressions majeures**
  - Testabilité : permettent de savoir s'il est possible d'avoir **une bonne visibilité sur la qualité de l'application** à travers une campagne de tests

# 16. La gravité des anomalies

- 1- **Fatal** Impossible de continuer les tests à cause de la sévérité des défaillances
- 2- **Critique** Les tests peuvent continuer mais l'application ne peut passer en mode production
- 3- **Majeur** L'application peut passer en mode production mais des exigences fonctionnelles importantes ne sont pas satisfaites

# 16. La gravité des anomalies

- 4- **Medium** L'application peut passer en mode production mais des exigences fonctionnelles sans très grand impact ne sont pas satisfaites
- 5- **Mineur** L'application peut passer en mode production, la défaillance doit être corrigée mais elle est sans impact sur les exigences métier
- 6- **Cosmétique** Défaillances mineures relatives à l'interface (couleurs, police, …) mais n'ayant pas une relation avec les exigences du client

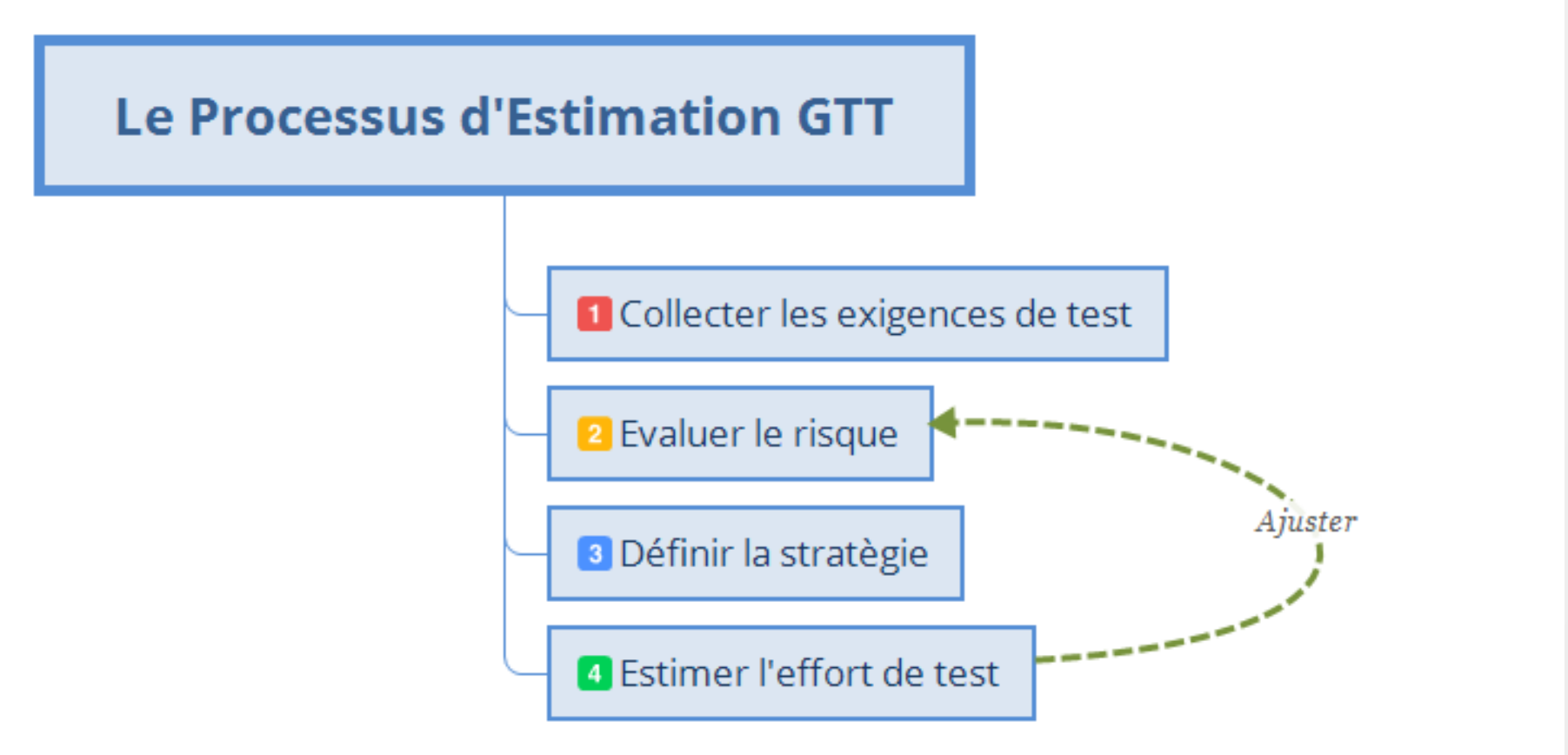
# 17. L' analyse du risque

- La probabilité : inévitable, fréquent, occasionnel, rare
- L'impact : catastrophique, grave, modéré, ennuyeux
- Priorité du risque

| | Inévitable | **Fréquent** | **Occasionnel** | Rare |
|---|---|---|---|---|
| **Catastrophique** | Elevée | Elevée | Elevée | Moyenne |
| **Grave** | Elevée | Elevée | Moyenne | Basse |
| **Modéré** | Moyenne | Moyenne | Basse | Basse |
| **Ennuyeux** | Moyenne | Basse | Basse | Basse |

# 18. Comment estimer un projet de tests

- https://lataverneduteste ur.fr/2018/01/10/comment-estimer-un-projet-de-test/



# 19. Les activités liées aux tests

1. Planifier les tests
2. Spécifier les tests
3. Concevoir les tests
4. Établir les conditions de tests
5. Définir les conditions d’arrêt d’une campagne de tests
6. Contrôler les résultats

# 19. Les activités liées aux tests

1. Planifier les tests
   - Prévoir où et en quelle quantité porter les efforts en personne et en temps
   - Prévoir et réserver l'utilisation de plates-formes de tests et les outils nécessaires

# 19. Les activités liées aux tests

2. Spécifier les tests

- Préciser ce que l'on attend des tests en termes de détection d'erreurs (fonctionnel ou non fonctionnel par exemple)
- Préciser les techniques et les outils à utiliser ou à ne pas utiliser, les parties du logiciel sur lesquelles porteront les tests

# 20. Les activités liées aux tests

3. Concevoir les tests

- Définir les **scénarios de tests**, appelés aussi cas de test, permettant de mettre en évidence des défauts recherchés par les tests, en fonction de leurs spécifications
- Préciser les environnements d'exécution de ces tests

4. Établir les conditions de tests

- Prévoir pour chaque scénario de tests les jeux de valeurs à fournir au logiciel pour réaliser ce scénario

# 20. Les activités liées aux tests

5. Définir les conditions d'arrêt d'une campagne de tests

   - Définir, en relation avec la phase de planification, ce qui permettra de décider l'arrêt ou la poursuite des tests

6. Contrôler les résultats :

   - Comparer les données fournies par l'exécution des tests par rapport aux données attendues ou constatées lors de phases précédentes de tests

# 21. Plan Qualité Projet (PQP)

- L'ensemble de la stratégie de tests est détaillé dans le **Plan Qualité Projet (PQP)**. Le plan qualité projet est très important. Il va notamment :
  - Définir l'**organisation** à mettre en place (équipe de tests)
  - Définir les r**esponsabilités** et relations entre les différents intervenants
  - Définir les **types et les objectifs de tests** pour chacun des niveaux (tests unitaires, tests d'intégration, tests système)
  - Définir les **outils** qui seront utilisés
  - Définir **les moyens et les délais** à investir dans l'activité de tests

# 22. Plan de tests

- Le plan de tests est un **document obligatoire** déterminant le déroulement des tests durant le projet
- C'est ainsi qu'il existe **autant de plan de tests que de phases de qualification** du produit.
- De manière générale, les tests se déroulent **du général au particulier** (détail).
- L'objectif de chaque plan de tests est de fournir un programme pour **vérifier que le logiciel produit satisfait les spécifications et la conception du logiciel**.

# 22. Plan de tests

- Un plan de test doit :

  1. Définir les **éléments à tester et l'ordre** dans lequel ils doivent être testés (planifier les tests)

  2. Décrire **l'environnement de tests**

  3. Définir la façon dont les tests vont être menés (**procédures**) : processus exacts à mener, l'historisation, la traçabilité, le reporting, le suivi, le contrôle...

# 22. Plan de tests

4. Décrire et constituer les **fiches de tests** (définir les actions à réaliser, **les jeux de données** à utiliser, les valeurs et les comportements attendus). L'ensemble des fiches de tests constitue le **dossier de tests**.

5. Fixer les **critères d'arrêt des tests** : selon la couverture définie, selon le nombre d'erreurs trouvés, selon le temps imparti… (Appliquer la stratégie de tests aux tests).

Qu'est qu'un plan de test logiciel ? - All4test

# 23. Plan de tests (exemple)

## Plan de test qualité en gestion de projet

This slide covers the project quality testing table for documenting any quality-compromising issues. It also includes testing information such as expected results, actual results, tester, etc.

| Date | Test Description | Expected Results | Actual Results | Pass/ Fail | Tested by | Tester Comments |
|---|---|---|---|---|---|---|
| 10/01/2022 | Website login | Should land to home screen | User directed to other page | Fail | Henry | Contact web development team for changes |
| 13/01/2022 | Click magnifying glass | Whole page gets zoomed in | Text size increases | Fail | George | Contact web development team for changes |
| 14/01/2022 | Signing up for website | Get confirmation email | Confirmation email received | Pass | Emma | No issues |
| Add text here | Add text here | Add text here | Add text here | Add text here | Add text here | Add text here |
| Add text here | Add text here | Add text here | Add text here | Add text here | Add text here | Add text here |
| Add text here | Add text here | Add text here | Add text here | Add text here | Add text here | Add text here |
| Add text here | Add text here | Add text here | Add text here | Add text here | Add text here | Add text here |

This slide is 100% editable. Adapt it to your need and capture your audience's attention.