Département Informatique

Master1 SIGL

Cours: Ingénierie des Besoins

Chapitre 7

Test des besoins

Rappels

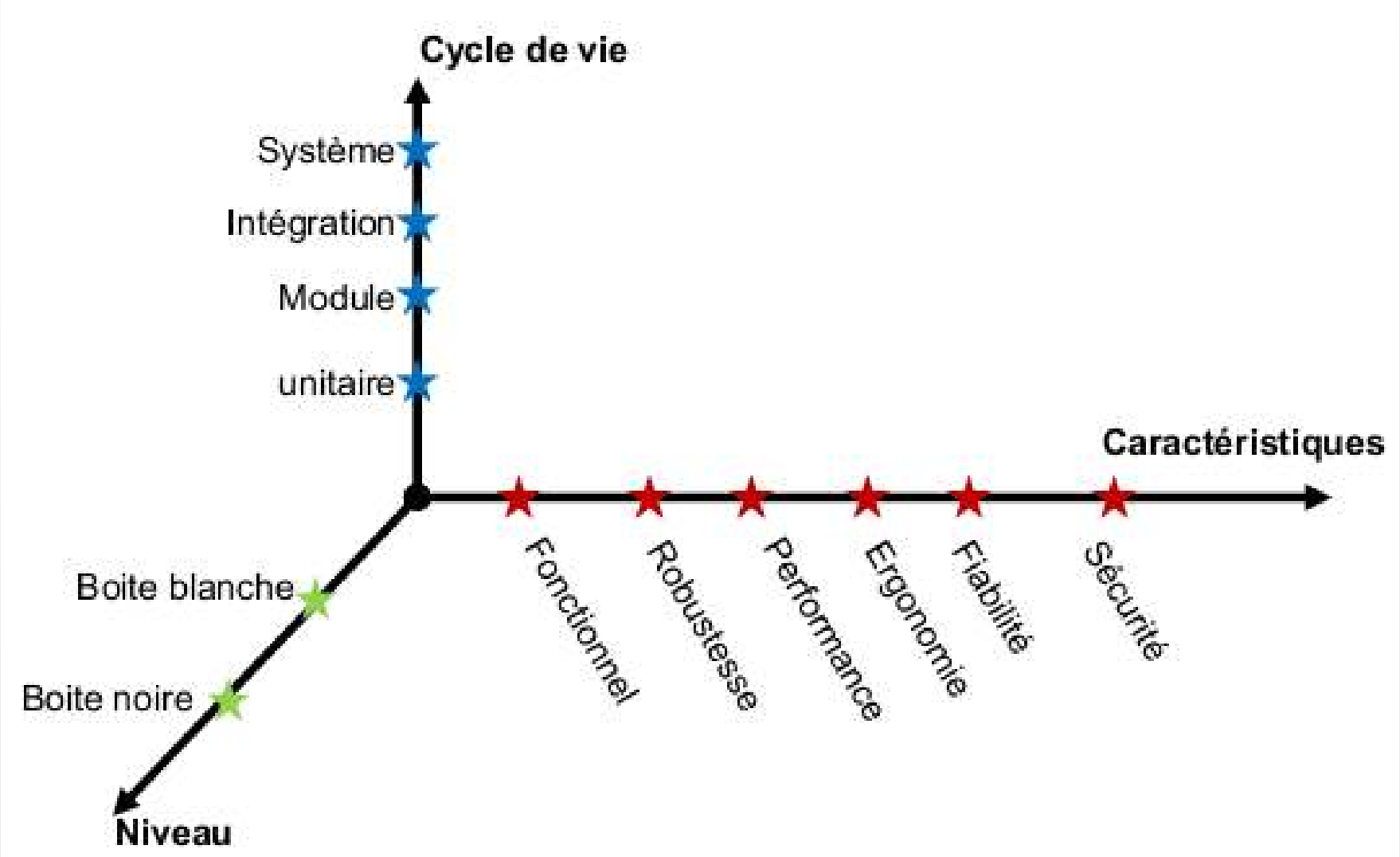
- **Qualité du logiciel** = Conformité aux spécifications+ Fiabilité + Robustesse
- **Test de Validation**: Est-ce que le logiciel réalise les fonctions attendues?
- **Test de Vérification**: Est-ce que le logiciel fonctionne correctement?
- **Test statique**: Relecture du code, spécifications, design, etc;
- **Test dynamique**: Exécution du code pour s'assurer deson bon fonctionnement;

Définition du test

- "Le test est l'exécution ou l'évaluation d'un système ou d'un composant par des moyens automatiques ou manuels, pour vérifier qu'il répond à ses spécifications ou identifier les différences entre les résultats attendus et les résultats obtenus". IEEE (Standard Glossary of Software Engineering Terminology)

- Deux grandes familles de tests:
  - **Test structurel** (ou test boîte blanche): détectent plus facilement les erreurs commises.
  - **Test fonctionnel** (ou test boîte noire): étectent plus facilement les erreurs d'omission et de spécification.

Les trois dimensions pour les tests



Technique: Test-Driven Development (TDD)

- **Approche Test-Driven Development**:
  1. Écrire un test et s'assurer qu'il échoue
  2. Écrire le minimum de code pour faire passer le test
  3. Re-factoriser le code en s'assurant que le test continu de passer
- **Caractéristiques:**
- S'inspire de l'organisation des chaînes de production «Lean-Thinking»
  - L'étape de codage ne produit que ce qui est demandé par l'étape de test,
  - Pas de gâchis causé par des étapes intermédiaires
  - Tout est juste à temps (JIT)
- Les tests unitaires (TU) dérivent directement de la spécification.

Technique: Test-Driven Requirements (TDR)

- Tirer les tests fonctionnels des besoins sans avoir à attendre leur réalisation pour faire les tests.
- Déléguer les tests fonctionnels aux clients.
- Intéressant pour des équipes réparties.
- Possibilité d'avoir dans le même wiki les besoins et les tests qui les vérifient.
- Le codage devient la dernière activité avant la livraison du logiciel.

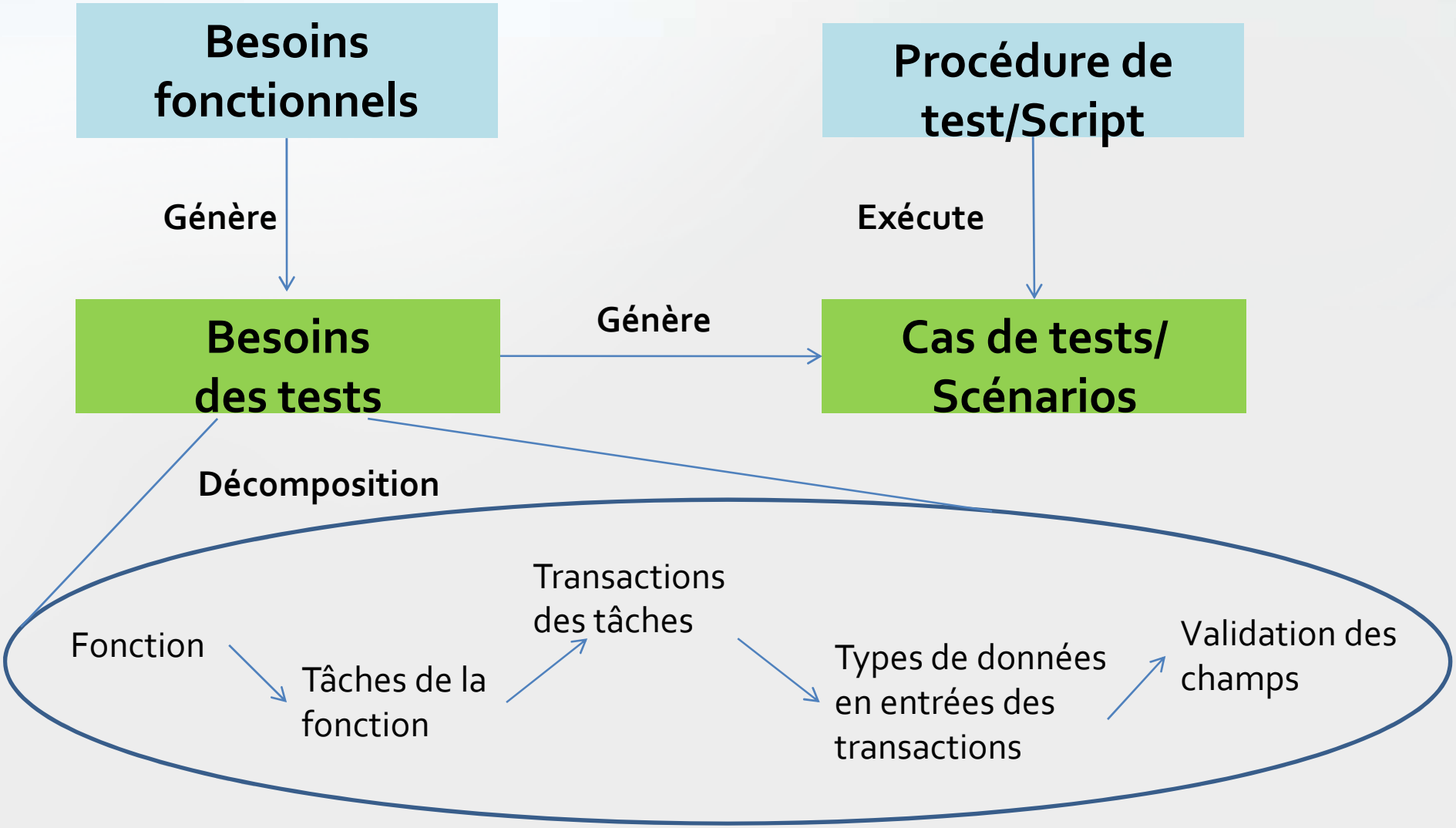
Besoins des tests

- Identifient quoi tester: (traditionnellement le plan de test)
  - qu'est-ce qui doit être testé et
  - qu'est-ce qui doit être validé?
- Comprend à la fois les situations normales et celles avec erreurs
- NE dispose PAS de données pour les assigner aux cas de test.
- Ces données apparaissent dans les cas de test (le «comment» des tests)

Exemple : Tester l'insertion d'un enregistrement dans une table

- Les besoins des tests identifiés (entre autres) :
  - "Valider qu'il est possible d'insérer une nouvelle entrée"
  - "Valider que l'insertion échoue si l'entrée existe déjà"
  - "Valider que l'insertion échoue si la table atteintson maximum"
  - " Valider qu'il est possible d'insérer une entrée dans une table vide (initial)"
  - "Valider que l'insertion échoue si la table est pleine"
- Ce sont des besoins (exigences) de test et NON PAS des tests parce qu'ils ne décrivent pas les données à insérer.
- Les données ne sont pas pertinentes à ce niveau, ils vont apparaitre dans les cas de test.
- Valider que vous pouvez insérer 'Mehenni' est un cas de test et
- non un besoin de test.

Progression des tests des besoins



Exemple: location de voiture

- Valider qu'une location peut se produire.
  - Ouvrez un billet de location

Fonction

Tâche

Transaction

    - Valider qu'une fiche client peut être entrée
      - Valider qu'un nouveau client peut être ajouté à la table des clients
        - Valider que le nom est toujours alpha
        - Valider que l'âge est > 21.
        - Valider que le numéro de téléphone est numérique
        - Valider que le code régional est un numéro existant.
      - Valider qu'on peut modifier la fiche d'un client existant

Distinguer entre les # types de test

- Tests fonctionnels
- Tests des Interfaces Usager
- Tests de sécurité
- Tests de l' installation
- Tests de configuration
- Tests de performance (Temps de Réponse)
- Tests de mémoire et de CPU
- Tests des ressource usager
- Tests de la documentation
- Tests de compatibility
- Tests de récupération
- Tests de maintenabilité
- Tests de charge (utilisateurs simultanés, plusieurs petites transactions)
- Tests de volume (grandes transactions)
- Autres …
- Communication avec les autres systèmes

Méthodes du test fonctionnel

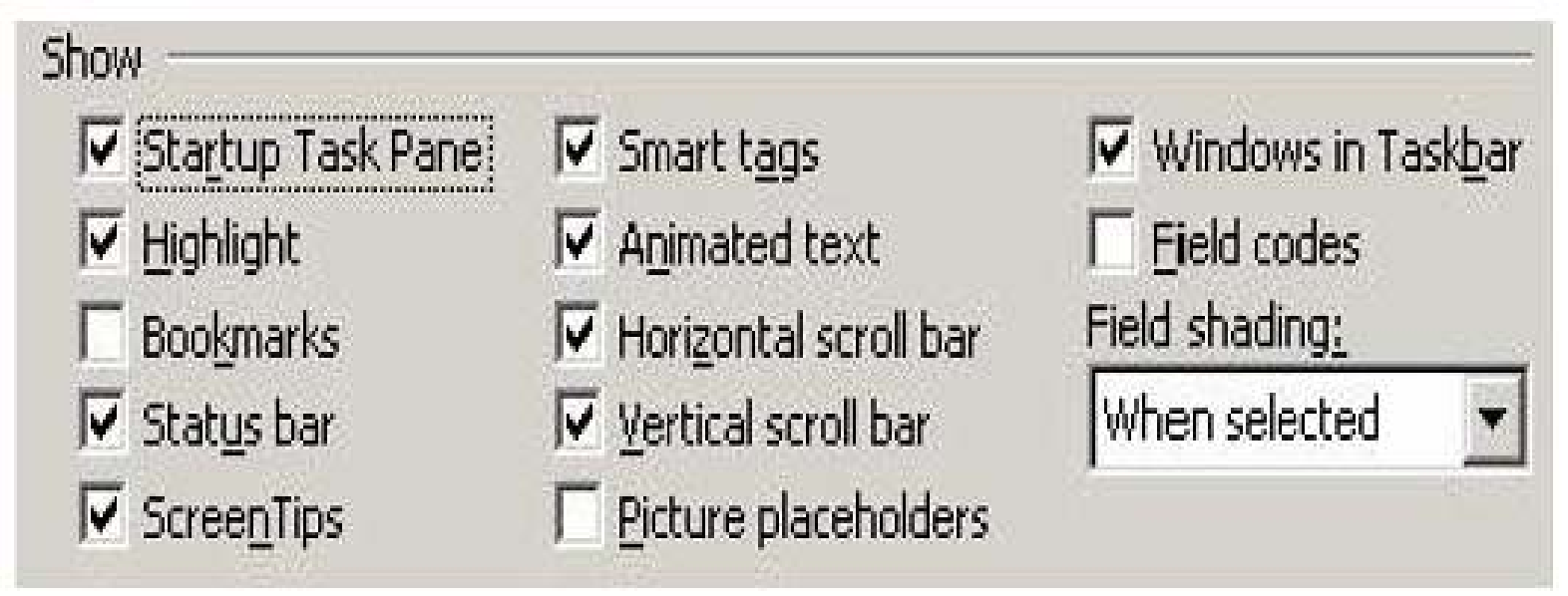
- Le test fonctionnel vise à examiner le comportement fonctionnel du logiciel et sa conformité avec la spécification du logiciel. **(Sélection des Données de Tests (DT))**
- **Méthodes du test fonctionnel**
  1. Test aux limites (test déterministe)
  2. Test combinatoire: Algorithmes Pairwise
  3. Test aléatoire
  4. Génération automatique de tests à partir d'une spécification (TDR).

Test aux limites

- **Principe** : on s'intéresse aux bornes des intervalles partitionnant les domaines des variables d'entrées :
- Pour chaque intervalle, on garde les 2 valeurs correspondant aux 2 limites, et les 4 valeurs correspondant aux valeurs des limites ± le plus petit delta possible
- $n \in [3 .. 15] \Rightarrow v1 = 3, v2 = 15, v3 = 2, v4 = 4, v5 = 14, v6 = 16$
- Si la variable appartient à un ensemble ordonnés de valeurs, on choisit le premier, le second, l'avant dernier et le dernier
- $n \in \{-7, 2, 3, 157, 200\} \Rightarrow v1 = -7, v2 = 2, v3 = 157, v4 = 200$
- Si une condition d'entrée spécifie un nombre de valeurs, définir les cas de test à partir du nombre minimum et maximum de valeurs, et des test pour des nombres de valeurs hors limites invalides.
- Un fichier d'entrée contient de 1 jusqu'à 255 enregistrements, produire un cas de test pour 0, 1, 255 et 256.

Test combinatoire: Approche Pairwise

- Les combinaisons de valeurs de domaines d'entrée donne lieu a une explosion combinatoire
- Exemple : Options d'une boite de dialogue MS Word
- $2^{12} * 3$ (nombre d'items dans le menu déroulant) = 12 288



- **Approche Pairwise:** Tester un fragment des combinaisons de valeurs qui garantissent que chaque combinaison de 2 variables est testé

Exemple: Test combinatoire par l'Approche Pairwise

- **9 cas de test** : chaque combinaison de 2 valeurs est testée
- La majorité des fautes sont détectées par des combinaisons de 2 valeurs de variables²

OS	Réseau	Imprimante	Application
XP	ATM	Canon-EX	Pwpoint
Linux	IP	Canon 900	Word
Mac X	Wifi	HP 35	Excel

Cas	OS	Réseau	Imprimante	Application
1	XP	ATM	Canon-EX	Pwpoint
2	XP	IP	Canon 900	Word
3	XP	Wifi	HP 35	Excel
4	Linux	ATM	HP 35	Word
5	Linux	IP	Canon-EX	Excel
6	Linux	Wifi	Canon 900	Pwpoint
7	Mac X	ATM	Canon 900	Excel
8	Mac X	IP	HP 35	Excel
9	Mac X	Wifi	Canon-EX	Word

Test aléatoire ou statistique

- **Principe :** utilisation d'une fonction de calcul pour sélectionner les données de test :
  - Fonction aléatoire : choix aléatoire dans le domaine de la donnée d'entrée,
  - Utilisation d'une loi statistique sur le domaine.
- **Exemples :**
- Échantillonnage de 5 en 5 pour une donnée d'entrée représentant une distance,
- Utilisation d'une distribution gaussienne pour une donnée représentant la taille des individus.