IUT Nantes
Pôle Sciences et technologie

Nantes Université

# *R2.03 - Qualité de développement 1*
# *Automatisation des tests*

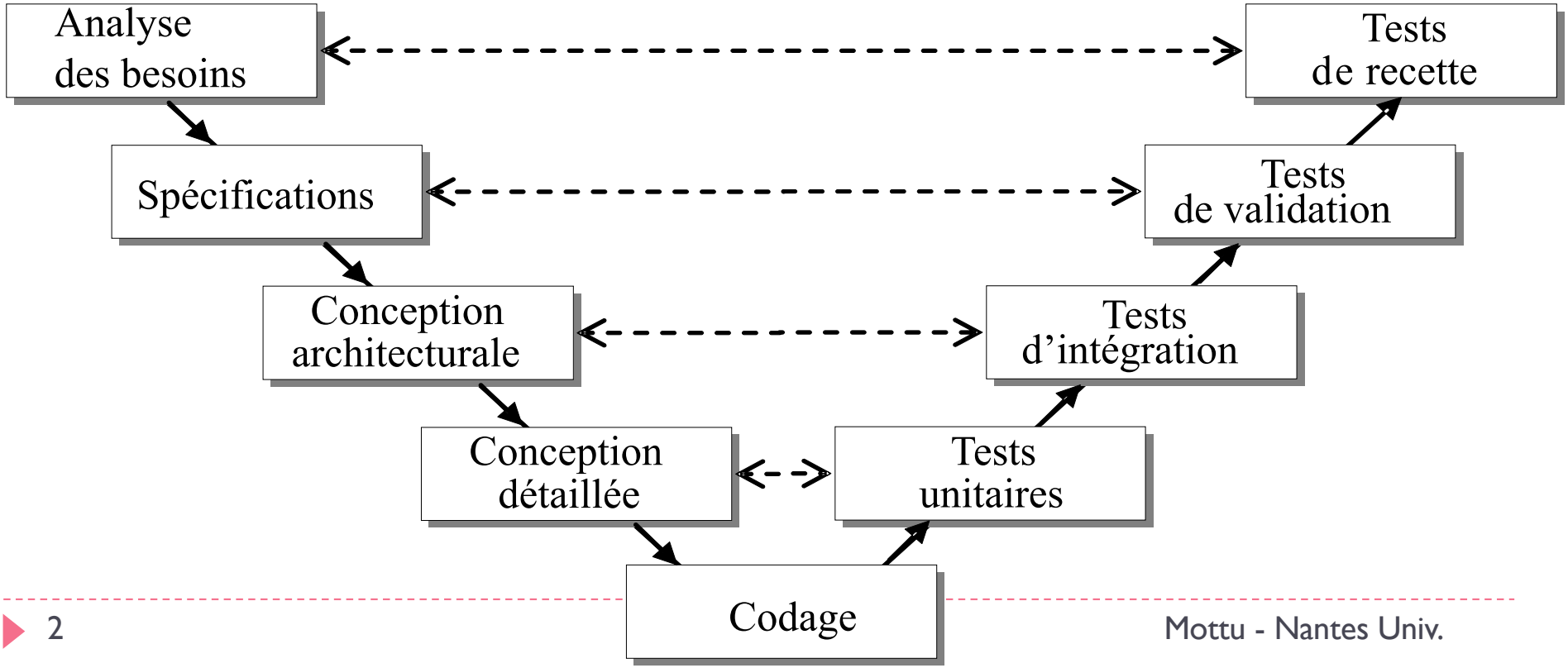
## CM8 Testabilité

Jean-Marie Mottu

# Introduction à la Testabilité

- Il ne suffit pas de décider de tester pour pouvoir tester
  - Concevoir les tests
  - Implémenter les tests
  - Diagnostiquer

} Nécessitant de la testabilité



# 2 principales heuristiques de la Testabilité

- Observabilité
  - Qu'est-ce qu'il est possible d'observer dans le logiciel ?
    - Peut-on observer une seule/toutes les partie(s) du logiciel ?
      - Classes, propriétés, variables
    - Peut-on observer l'environnement ?
      - Le temps qui s'écoule, l'interaction avec d'autres logiciels
- Contrôlabilité
  - Qu'est-ce qu'il est possible de manipuler ?
    - Logiciel sous test, l'environnement
  - Peut-on mettre le système dans l'état voulu ?
    - Faire des tests sur un système en marche ?

# Heuristiques secondaires de la Testabilité

- Disponibilité
  - Logiciel disponible (boite noire et blanche)
  - Le logiciel doit être suffisamment développé pour exécuter les tests
  - La spécification doit être explicite et disponible
- Stabilité
  - Les éléments testés sont modifiés en fournissant un suivi

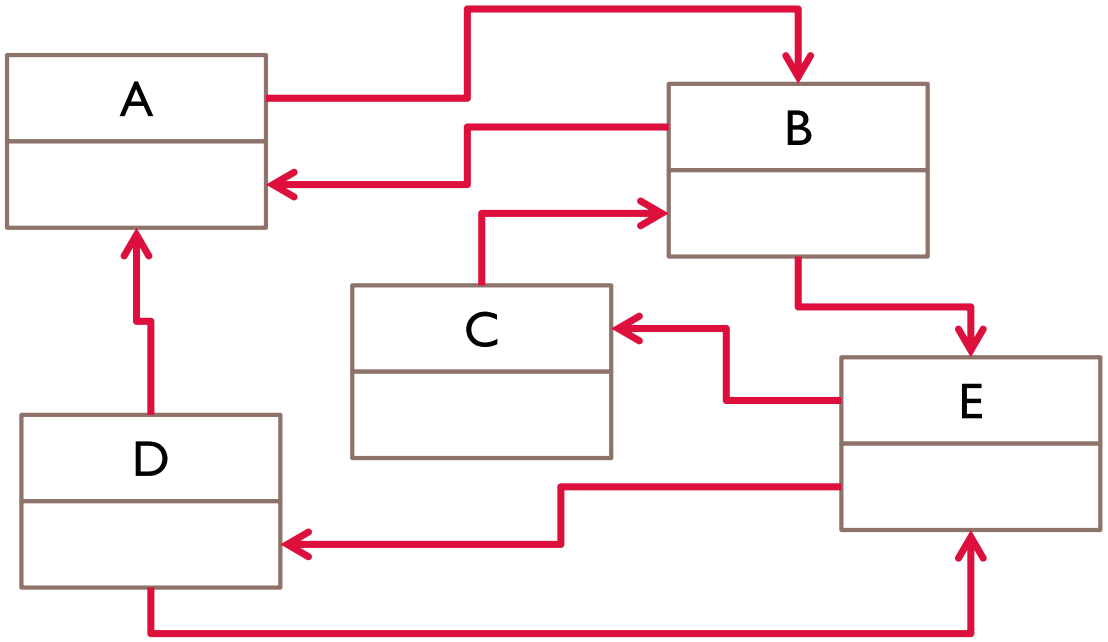
# Testabilité

- La testabilité s'évalue et complète les tests

- La testabilité est l'activité qui permet d'évaluer la capacité à tester un système selon plusieurs points de vue :
  - La capacité d'un logiciel à être testé
    - (observabilité, contrôlabilité, disponibilité, stabilité)
  - La capacité des tests à faire des vérifications pertinentes
    - (observabilité, contrôlabilité, stabilité)
  - La capacité d'une spécification à formuler des exigences vérifiables
    - (disponibilité, stabilité, observabilité)

# Testabilité du point de vue design du logiciel

- Exemples de choix du design pouvant être préjudiciables à la testabilité
  - Les variables privées d'une classe ne sont pas observables
  - Peu de typage, typage dynamique
  - Les interdépendances entre classes provoquent des interblocages



# Comment anticiper/résoudre les problèmes de testabilité ?

- Dès la spécification, conception
  - Considérer les besoins de contrôlabilité, d'observabilité

- Comment faire quand on ne peut pas contrôler certaines parties du code ?
  - Le temps
  - La communication avec l'extérieur du programme
    - D'autres programmes
    - Des liaisons vers le monde physique, nettement moins contrôlable
      - Contrôle impossible
      - Contrôle possible mais avec des coûts/temps non raisonnables

# Améliorer la testabilité dès la conception

- Améliorer la structure des packages
  - Diminuer les interdépendances entre classes.

- Limiter la complexité des classes et des méthodes
  - De trop nombreuses imbrications de boucles et de conditionnelles font exploser la combinatoire pour résoudre la sensibilisation des chemins du graphe de flot de contrôle
    - En diminuant le nombre de chemin d'une méthode, on augmente sa testabilité [Nejmeh 1988]
  - L'analyse du flot de donnée est aussi importante
    - Rapport entre la définition et l'utilisation des variables

# Améliorer la testabilité dès la conception

- Permettre l'observation intermédiaire d'état
  - Un programme dont l'état interne est important est moins testable qu'un autre n'ayant que des entrées et des sorties
  - L'état interne est-il masqué ?
  - L'état interne est-il observable en continue ?
    - On peut ajouter des observateurs
    - Des contraintes embarquées
  - Important pour le test mais aussi la localisation d'erreur

# Evaluer et améliorer la testabilité avant de tester

- Les testeurs conçoivent leurs tests puis les implémentent
  - Les tests qui ne peuvent pas être implémentés ne doivent pas être négligés
  - La testabilité est évaluée progressivement pendant l'implémentation des tests
  - Mais il est plus efficace de le faire avant la conception
    - L'approche structurelle (cf R4.02) nécessite le code pour concevoir les tests
      - Améliorer la testabilité change le code et nécessaite d'ajuster les tests structurels