Nantes Université

R2.03 - Qualité de développement 1
Automatisation des tests

CM2 – Typologie

Jean-Marie Mottu (Lanoix, Le Traon, Baudry, Sunye)

Qualité logicielle

- Software Engineering: A Practitioner's Approach. Roger Pressman. 1st (1982), 8th edition (2014)
  - «Conformité aux besoins explicites de fonctionnalité et de performance, aux normes de développement explicitement documentées et aux caractéristiques implicites qui sont attendues de tous les logiciels développés de façon professionnelle» (5ème édition)

- [IEEE]
  - Le degré pour lequel un système, un composant ou un processus répond aux besoins spécifiés.
  - Le degré pour lequel un système, un composant ou un processus répond aux attentes des utilisateurs.

Vérification et Validation par le Test

- Vérification
  - _____
- Validation
  - _____
- Le test a pour but de _____

dans un programme conformément à sa spécification

  - Prouver l'absence de fautes dans un programme est un problème indécidable dans le cas général
- Tester c'est _____ pour _____ si ça _____ ___.
- Différent de la preuve qui consiste à formaliser le système pour appliquer des vérifications de niveau mathématique
  - Difficulté de la formalisation
  - Preuve et test sont complémentaires

Cas de test

- Tester c'est *effectuer* un ensemble de cas de test
  - Principe général : faire des essais
    - Combien ? De quel type ? A quel point ?
- Cas de test
  - Description du cas de test
  - Initialisation
  - Donnée de test
  - Oracle
    - L'Oracle contrôle l'exécution du SUT initialisé, avec la DT, retournant le verdict
- Verdict : passe, échoue

Exemples de cas de test pour l'addition

- Test de l'addition x + y = z
- Cas de test 1
  - Description : vérifier que l'addition de 2 nombres opposés donne 0
  - Initialisation : allumer le programme, etc.
  - Donnée de test : (x = 5 , y = -5)
  - Oracle : (res == 0)
- Cas de test 2
  - Description : vérifier que l'addition de 2 fois le même nombre donne son double
  - Initialisation : allumer le programme, etc.
  - Donnée de test : (x = 2 , y = 2)
  - Oracle : (res == 4) ou (res == 2* x)

- etc. mais avec quelle intensité, jusqu'à quand, avec quels objectifs ?

Problème d'exhaustivité

Objectif :
obtenir un ensemble **fini** de cas de test

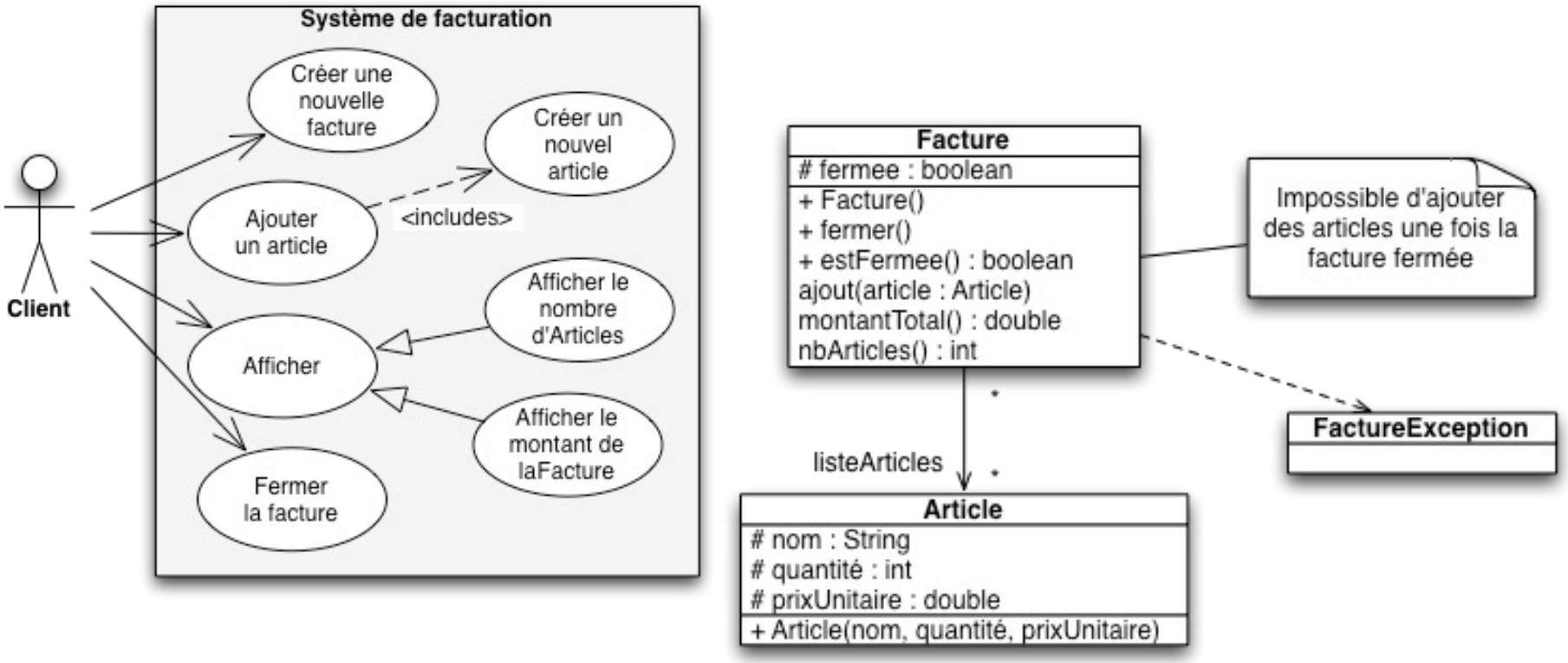
- Potentiellement il y en a une infinité
  - a = b + c sur $\mathbb{N}$
  - Combien de combinaisons :
    - $2^{32}*2^{32} = 2^{64}$ = un nombre en base décimale avec plus de 18 chiffres !
- Il faut choisir des données
  - Pour atteindre les objectifs :
    - Suffisamment
    - Suffisamment réparties
    - Suffisamment efficaces
  - tout en respectant les moyens et les délais
- Sans technique ni outil, le test serait extrêmement laborieux
  - Ne pas savoir où chercher les bugs potentiels
  - Passer du temps sans savoir quand s'arrêter
  - Gaspiller du temps à faire de nouveaux tests sans intérêt

Sur quoi baser le test ?

- Une spécification : exprime ce qu'on attend du système
  - un cahier des charges (en langue naturelle)
  - une documentation
  - des échanges avec le client
  - commentaires dans le code
  - contrats sur les opérations (à la Eiffel)
  - un modèle UML
  - une spécification formelle (automate, modèle B...)

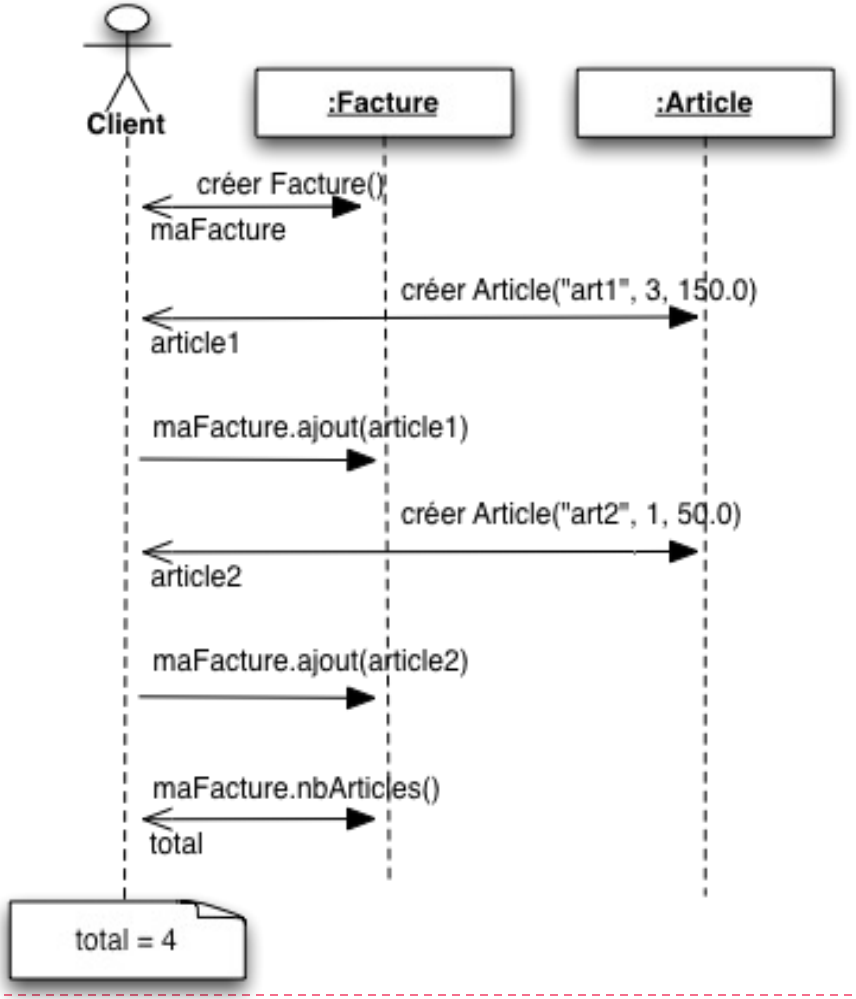
- La référence c'est la spécification, on teste
  - la conformité du système aux exigences de la spécification
  - et pas l'inverse
    - le code potentiellement faux (cf commutativité du TD1)

Spécification
e.g. Gestionnaire « simplifié » de factures

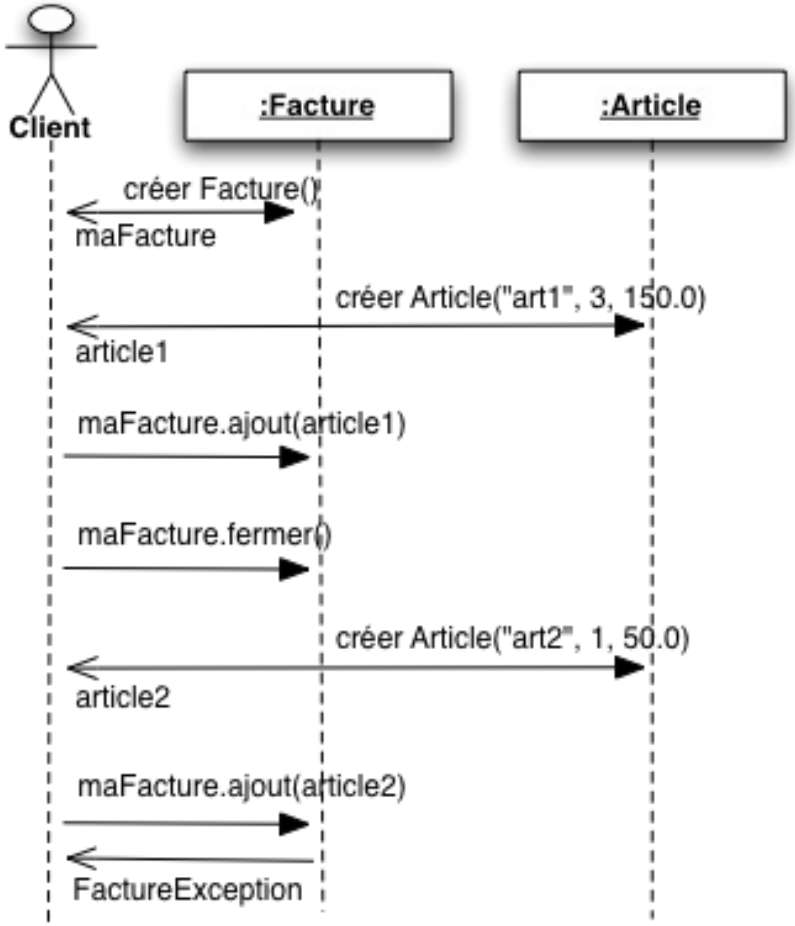


Spécification

- Scénarios d'usage / diagramme de séquences :
  - Nominal :



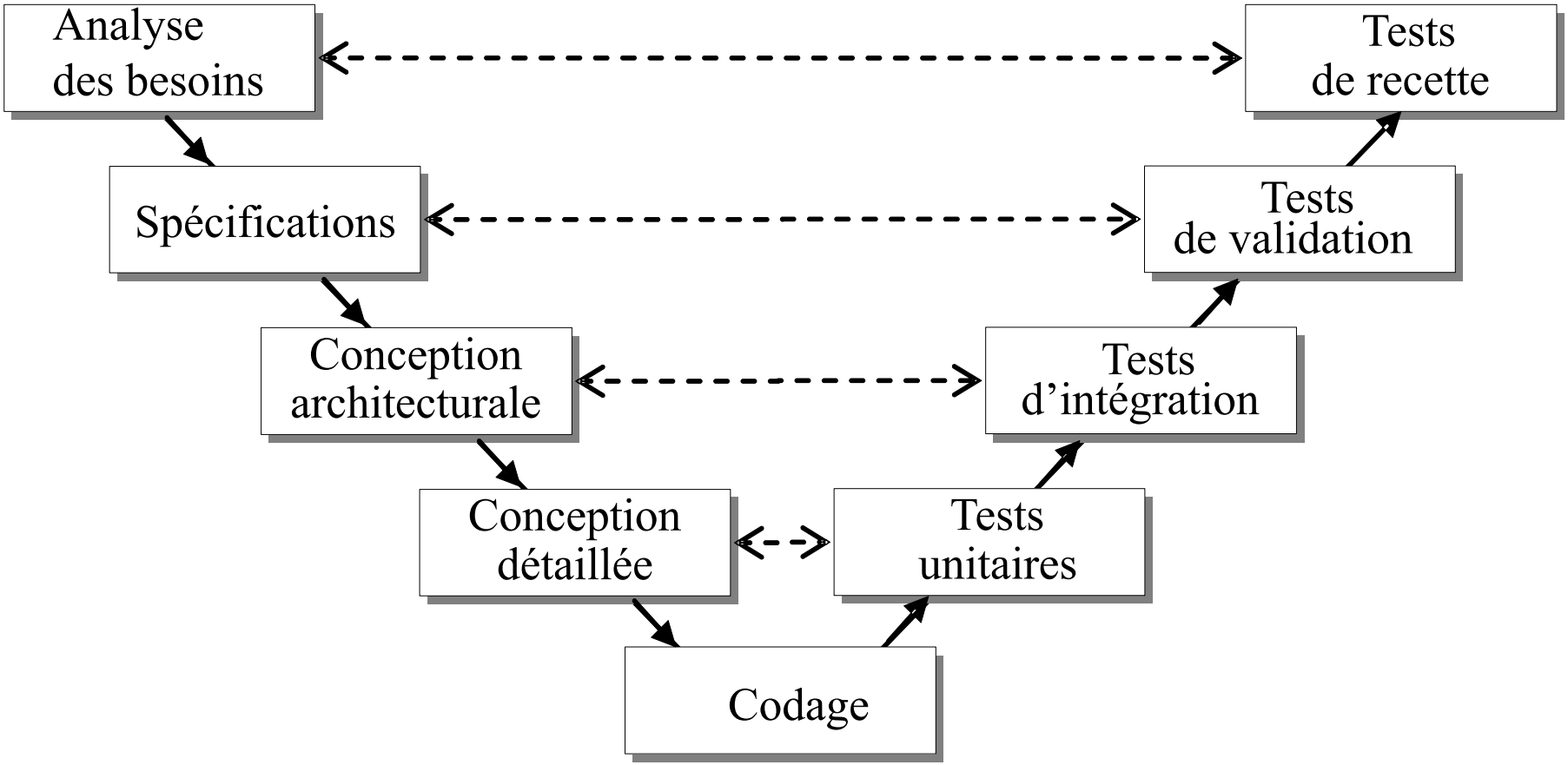
  - Exceptionnel :



Test de logiciel : terminologie

- Etapes de test
  - Unitaire, intégration, système, recette
- Phases transversales
  - non régression, performance
- Différente dynamicité, les tests peuvent être
  - Dynamique / statique
- Test du comportement
  - Nominal / Exceptionnel
- Création de tests avec une approche
  - Fonctionnelle / structurelle
  - Chacune ayant plusieurs techniques de création/sélection de cas de test

Le test dans un cycle de développement en V



Etapes de test : Test unitaire

- Vérification d'une unité indépendamment des autres
- Vérifier intensivement les unités

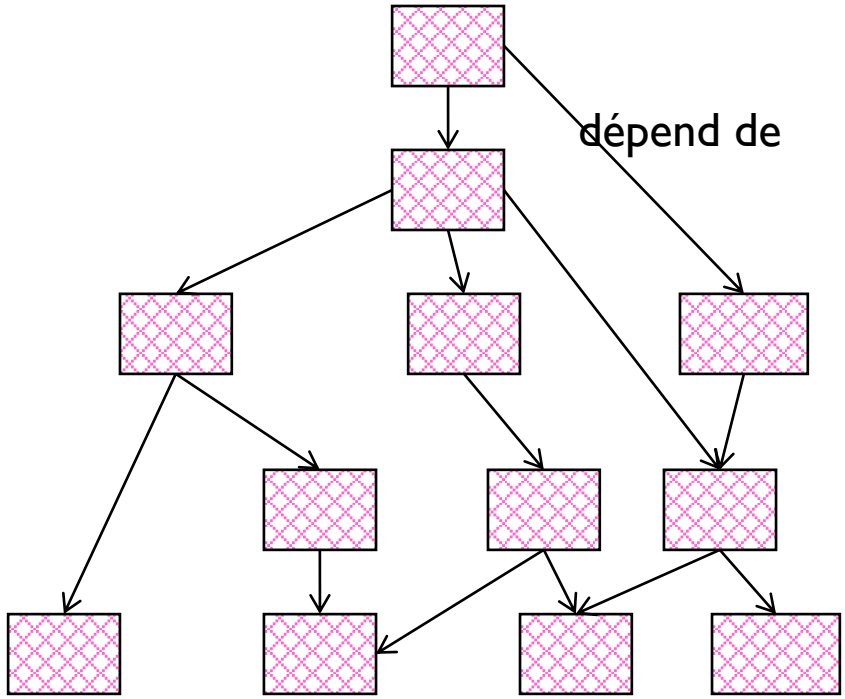
- Pour un langage procédural
  - unité de test = procédure
- Dans un contexte orienté objet
  - unité de test = classe

Etapes de test : Test d'intégration

- Choisir un ordre pour intégrer et tester les différents modules du système

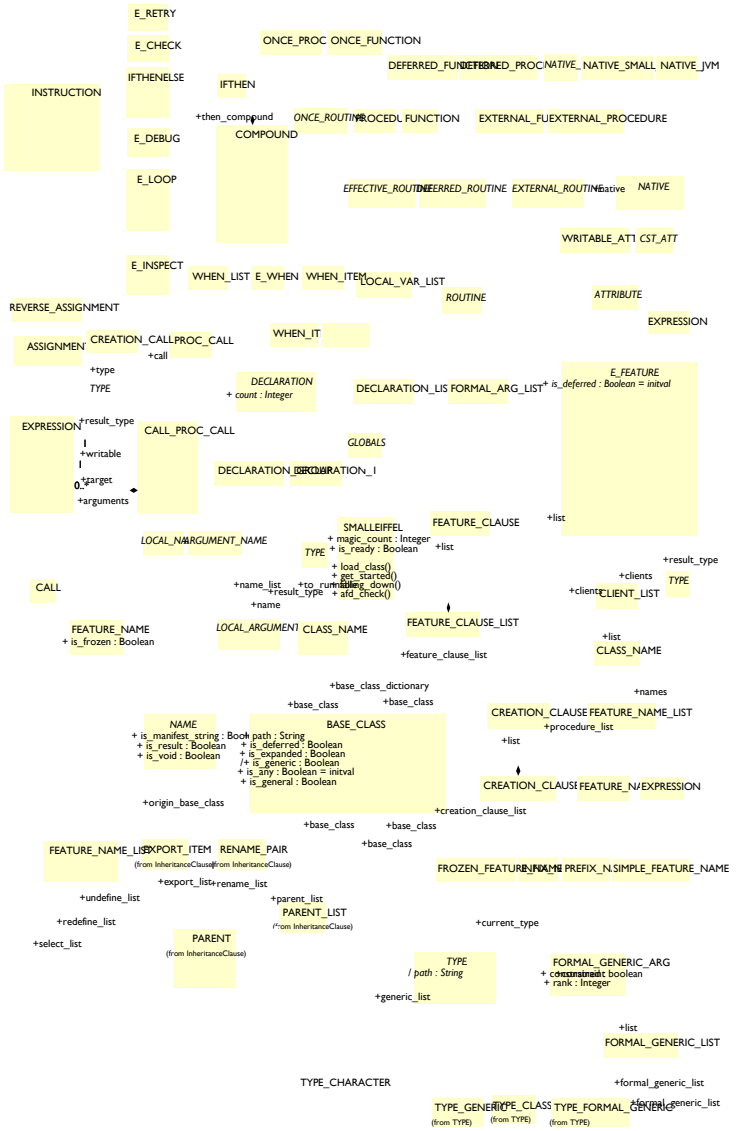
- Cas simple: il n'y a pas de cycle dans les dépendances entre modules

- Les dépendances forment un arbre et on peut intégrer simplement de bas en haut



Etapes de test : Test d'intégration

- Cas plus complexe: il y a des cycles dans les dépendances entre modules
- Cas très fréquent dans les systèmes à objets
- Il faut des heuristiques pour trouver un ordre d'intégration



Etapes de test :

- Test de validation ou système
  - Valider la globalité du système
    - Les fonctions offertes
    - A partir de l'interface
  - Remet en cause les exigences, la spécification

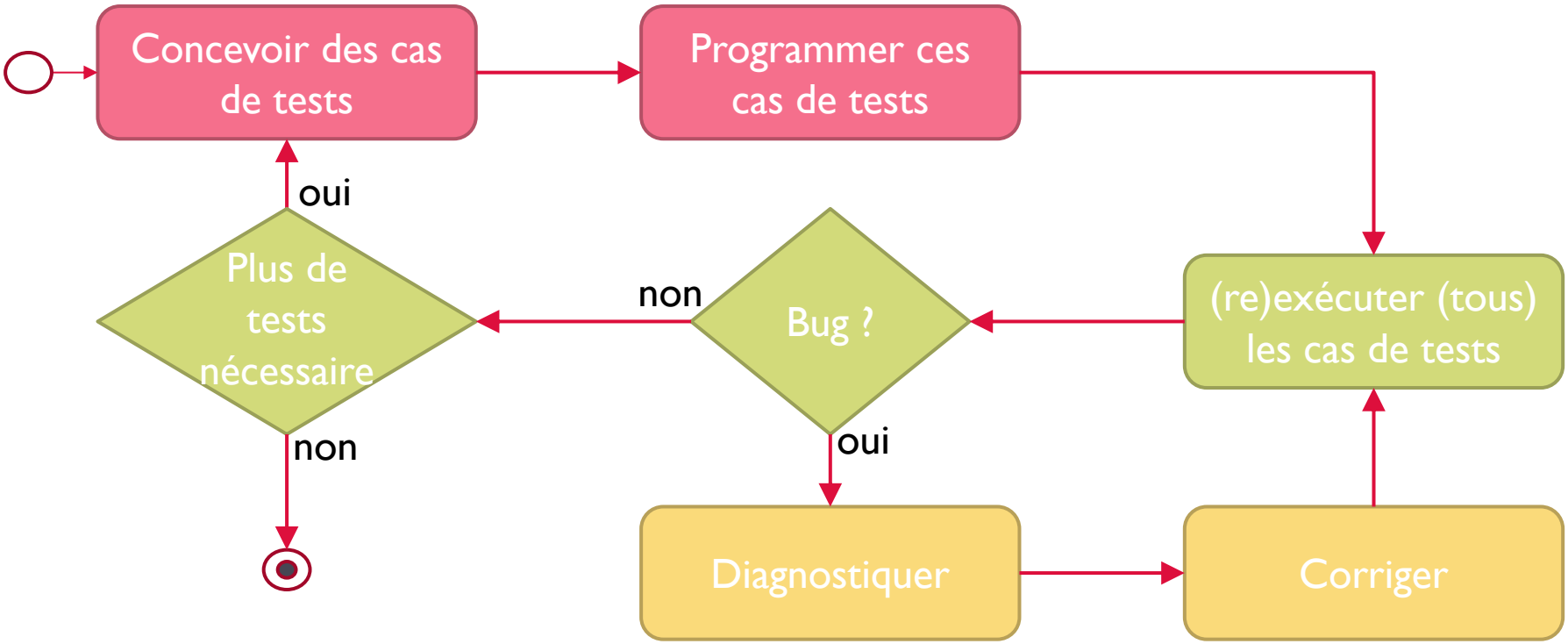
- Test recette
  - Effectué avec la MOA ou par la MOA
  - Test avec utilisateur final

Phases transversales de test :

- Test de non régression
  - Consiste à vérifier que des modifications apportées au logiciel n'ont pas introduit de nouvelle erreur
    - vérifier que ce qui marchait marche encore
  - Dans la phase de maintenance du logiciel
    - Après refactoring, ajout/suppression de fonctionnalités
  - Après la correction d'une faute
- Test de performance
- Etc.

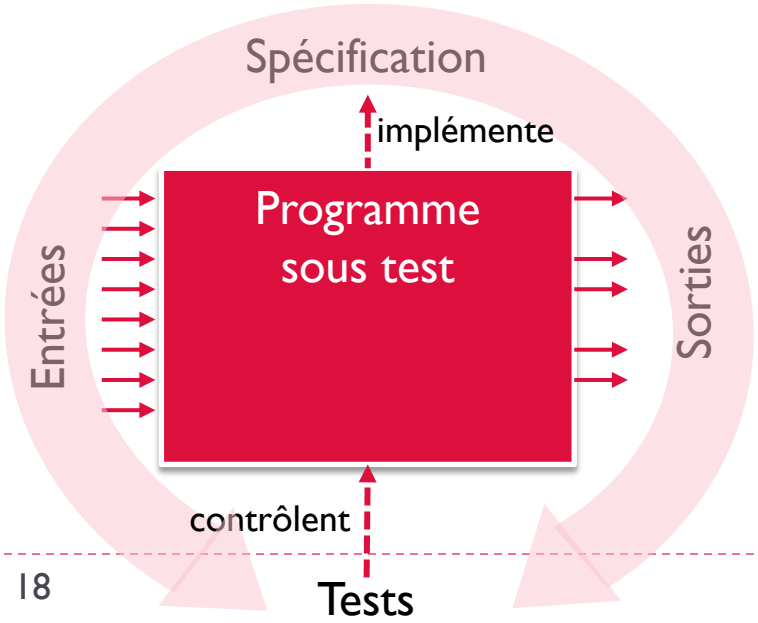
Quelle exécution des tests ?

- Test statique (cf QD4 en BUT3)
  - relecture / revue de code
  - analyse automatique
- Test dynamique en automatisant les tests



Création de tests avec une approche :

- Test par approche fonctionnelle (test boîte noire)
  - Exploite la description des fonctionnalités du programme



- Test par approche structurelle (test boîte blanche)
  - Exploite la structure interne du programme