

# DevOps

Couverture de code

Thomas Ropars

`thomas.ropars@univ-grenoble-alpes.fr`

2025

## Vu précédemment

- Le test est une part importante du développement logiciel
- Tester est une tâche complexe
- Impossible de tester tous les cas

Comment savoir si une suite de tests est complète?

# Couverture de code

## Définition

Décrit le taux de code source testé d'un programme.

## Utilisation

- Fournit une valeur quantitative permettant de mesurer de manière indirecte la qualité des tests
- Met en évidence les parties d'un programme qui ne sont pas testées
- Permet d'ajouter de nouveaux tests

## Couverture de code

Quelle métrique utiliser?

# Critères de couverture

- Statement Coverage

- ▶ Est-ce que toutes les instructions du code ont été exécutées?
- ▶ Similaire: Line coverage, Basic block coverage
- ▶ Est-ce qu'un statement coverage de 100% garantit que les tests sont complets?

# Critères de couverture

- Statement Coverage

- ▶ Est-ce que toutes les instructions du code ont été exécutées?
- ▶ Similaire: Line coverage, Basic block coverage
- ▶ Est-ce qu'un statement coverage de 100% garantit que les tests sont complets?

```
int addAndCount(List list) {  
    if (list != null) {  
        list.add("Sample text");  
    }  
    return list.size();  
}
```

# Critères de couverture

- Statement Coverage

- ▶ Est-ce que toutes les instructions du code ont été exécutées?
- ▶ Similaire: Line coverage, Basic block coverage
- ▶ Est-ce qu'un statement coverage de 100% garantit que les tests sont complets?

```
int addAndCount(List list) {  
    if (list != null) {  
        list.add("Sample text");  
    }  
    return list.size();  
}
```

- ▶ Un Test avec `list != null` donne une couverture de 100%

# Critères de couverture

- Statement Coverage

- ▶ Est-ce que toutes les instructions du code ont été exécutées?
- ▶ Similaire: Line coverage, Basic block coverage
- ▶ Est-ce qu'un statement coverage de 100% garantit que les tests sont complets?

```
int addAndCount(List list) {  
    if (list != null) {  
        list.add("Sample text");  
    }  
    return list.size();  
}
```

- ▶ Un Test avec `list != null` donne une couverture de 100%
- ▶ Et si `list == null` ??
  - Ne pas écrire des cas de tests juste pour satisfaire votre outil de couverture de code

# Critères de couverture

- Decision Coverage

- ▶ Est-ce que les conditions dans les structures de contrôle ont été évaluées à `true` et `false`?
- ▶ Similaire: Branch coverage

- Condition Coverage

- ▶ Similaire à *Decision coverage* mais évalue chaque sous-expression booléenne des conditions de manière séparée
- ▶ Exemple: `if( a==10 || a < b)`

# Critères de couverture

- Path Coverage

- ▶ Est-ce que tous les chemins d'exécution possibles au sein de chaque méthode ont été empruntés?

```
if(A){  
    if(B){}}
```

```
if(C){  
}
```

- ▶ Le nombre de chemin augmente de manière exponentiel avec le nombre de branches. (10 *if* impliquent 1024 chemins possibles)
- ▶ Parfois tous les chemins ne sont pas atteignables.

- Loop Coverage

- ▶ Est-ce que chaque boucle a été exécuté 0, 1 et plusieurs fois?
- ▶ Peut être pris en compte dans le *Path Coverage*

# Critères de couverture

- **Function Coverage**

- ▶ Est-ce que toutes les fonctions du code ont été appelées?
- ▶ Similaire: Method coverage
- ▶ Peut mettre en évidence du code qui ne sert pas

- **Class Coverage**

- ▶ Est-ce que chaque classe a été testée?
- ▶ Une classe est considérée testée par exemple si elle a été chargée et initialisée par la JVM (dépend de l'outil)

# Des outils de couverture de code

Demandez à votre moteur de recherche préféré!

Pour Java:

- Cobertura
- EMMA
- Plugins Eclipse:
  - ▶ eCobertura
  - ▶ Clover
  - ▶ EclEmma

# Exemple de EclEmma

<https://eclEmma.org>

- S'installe depuis *Eclipse Marketplace* (libre)
- Supporte les tests JUnit
- Mesures de couverture<sup>1</sup>:
  - ▶ Instruction (bytecode instruction)
  - ▶ Branches
  - ▶ Lines
  - ▶ Methods
  - ▶ Classes (si au moins une méthode est exécutée)
  - ▶ Cyclomatic Complexity (Mesure de la complexité de la structure d'une fonction, différent du nombre de branches et du nombre de chemins)

---

<sup>1</sup><https://eclEmma.org/jacoco/trunk/doc/counters.html>

# Références

- Notes de P. Labatut
- Notes de B. Bogacki