LA COUVERTURE DE CODE

Qualité de developpement : Analyse, outils et démonstration pratique

Mohamed el Salah AISSAOUI
Jenna-Louisa DJAOUEL
Audrey JULIEN

26/03/2025

SOMMAIRE

QU’EST-CE QUE LA COUVERTURE DE CODE ?

La couverture de code permet d’évaluer quelles partie d’un programme, projet ou d’un bout de code sélectionné sont réellement exécutées lors des tests.

- Détection des parties du code inutilisées et non éxécutées.
- Identification des chemins critiques non testés.
- Amélioration de la qualité logicielle et réduction des bugs.
- Optimisation des performances et de la maintenabilité.
- Impact sur l'éco-responsabilité du code en supprimant le code inutile (réduction du code mort).

DIFFERENTS TYPES DE COUVERTURE DE CODE

1. **Line Coverage** : Teste si chaque ligne du programme est exécuté au moins une fois
2. **Branch Coverage** : Vérifie si toutes les branches (conditionnelles) (if/else) ont été exécutées
3. **Function Coverage** : Vérifie si toutes les fonctions sont appelées
4. **Condition Coverage** : Vérifie les valeurs booléenne dans les conditions
5. **Path Coverage** : Analyse les différents chemins d'exécution

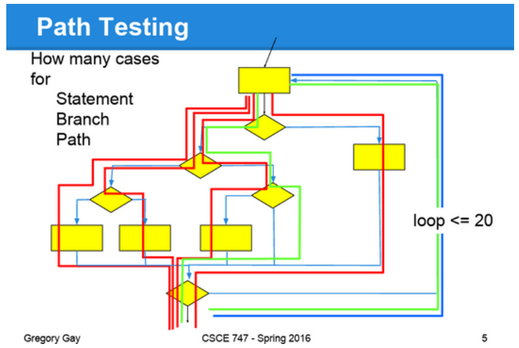
DONC C'EST IMPORTANT POUR :

EVITER DU CODE MORT

Des fonctions ou des variables inutiles ou plus utilisés qui alourdissent le projet

EVITER LES BUGS DE PRODUCTIONS

Des chemins critiques non testés peuvent causer des erreurs



Exemple d'occurrence de test dans un path

EVITER LE MANQUE D'OPTIMISATION

Manque d'optimisation : Code non exécuté = potentiel ralentissement ou failles de sécurité

MINIMISER LES IMPACTS SUR CI/CD

Mauvaise qualité des livrables, versionning altéré => déploiement plus risqué

EXEMPLES PLUS PRÉCIS DE PROBLEMES LIÉS A LA NON COUVERTURE DE CODE DANS UN PROJET

CODE MORT

- Oublies
- Fonctions non appelées
- Conditions toujours vraies ou fausses

```cpp
int main() {
    int x = 5;
    if (x > 10) {  //ici la condition ne sera jamais vraie
        std::cout << "ce message ne sera JAMAIS affiché." << std::endl;
    }
}
```

Exemple d'une condition toujours fausse

```cpp
int main() {
    int nombre = 42;  //variable inutilisée
    return 0;
}
```

Exemple d'une variable jamais utilisée ou appelée

EXEMPLES PLUS PRÉCIS DE PROBLEMES LIÉS A LA NON COUVERTURE DE CODE DANS UN PROJET

BUGS DE PRODUCTIONS

- **Authentification d'un utilisateur**
  - **Traitement de paiement**
    - **Condition de sécurité**

```
if (password != null && password.length() > 8)
```

Mais si on ne prend pas tout en compte, le programme pourrait planter.

```
password = null, password = ""
```

EXEMPLES PLUS PRÉCIS DE PROBLEMES LIÉS A LA NON COUVERTURE DE CODE DANS UN PROJET

MANQUE D'OPTIMISATION

Code non exécuté ou inutile = potentiel ralentissement ou failles de sécurité

```cpp
int main() {

    char motDePasse[8];
    std::cin >> motDePasse;  //l'utilisateur entre un mdp
    //sans limitation
}
```

Pas de controle sur l'entrée d'un mot de passe : faille de sécurité

```cpp
int main() {

    for (int i = 0; i < 1000000; i++) {
        std::cout << "Courd de qualité de dev" << std::endl;  //à l'aide
    }
}
```

Boucle INUTILE et qui ralentit le programme

"L'injection SQL= menaces visant la sécurité des applications web connectées à une BDD. Manque de contrôle et de validation des entrées utilisateur permet l'injection de code SQL malveillant exécuté dans la BDD, contournant ainsi les mesures de sécurité en place."

Source : https://kincy.fr/faille-securite/

EXEMPLES PLUS PRÉCIS DE PROBLEMES LIÉS A LA NON COUVERTURE DE CODE DANS UN PROJET

IMPACTS SUR CI/CD*

*CI/CD Continuous Integration/Continuous development

Mauvaise qualité des livrables, versionning altéré => déploiement plus risqué

Scénario : quelqu'un crée une fonction , ne la test pas et la push dans le main

```
int division(int a, int b) {
    return a / b;
}
```

Mais si b = 0

Une fonction maintenant inutilisée ou une fonctionnalité mal testée avant le déploiement peut mener à saboter l'entièreté du projet

COMMENT ASSURER UNE BONNE COUVERTURE DE CODE :

- **Différents outils pour différents langages**

JaCoCo, Clover Coverage : Java

arquillian-blade-example — Sessions

arquillian-blade-example

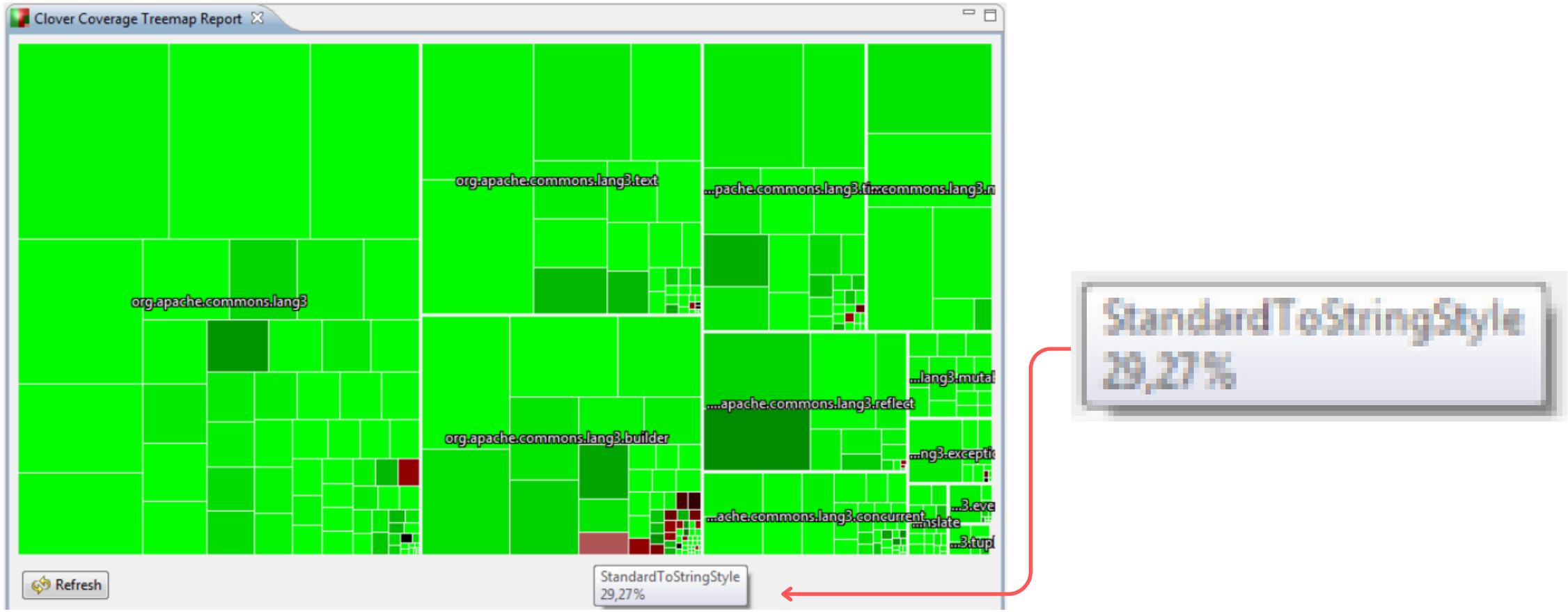
Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
com.liferay.arquillian.sample.portlet		100%		n/a	0	2	0	15	0	2	0	1
com.liferay.arquillian.sample.service		100%		n/a	0	2	0	2	0	2	0	1
Total	0 of 60	100%	0 of 0	n/a	0	4	0	17	0	4	0	2

Created with JaCoCo 0.7.9.201702052155

Exemple d'un rapport de JACOCO pour JAVA

COMMENT ASSURER UNE BONNE COUVERTURE DE CODE :

- **Différents outils pour différents langages**



Couverture Treemap Report de Clover Coverage

COMMENT ASSURER UNE BONNE COUVERTURE DE CODE :

- **Différents outils pour différents langages**

istanbul : JavaScript JS, TypeScript TS

```
-------------------------|---------|----------|---------|---------|-------------------
File                     | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
-------------------------|---------|----------|---------|---------|-------------------
All files                |   39.39 |    30.56 |   35.71 |   39.39 |
 src                     |       0 |      100 |       0 |       0 |
  index.js               |       0 |      100 |       0 |       0 |           4,5,6,9
 src/example-1           |   42.86 |    29.17 |   42.86 |   42.86 |
  App.jsx                |   64.29 |       40 |      50 |   64.29 |    20,21,23,24,27
  app.jsx                |       0 |        0 |       0 |       0 |... 21,23,24,27,30
  example-1.js           |   85.71 |       75 |   66.67 |   85.71 |                15
 src/example-1/components|   44.44 |     37.5 |      25 |   44.44 |
  Location.jsx           |   88.89 |       75 |      50 |   88.89 |                16
  Name.jsx               |       0 |        0 |       0 |       0 |... 13,15,16,25,29
 src/example-2           |     100 |       50 |     100 |     100 |
  example-2.js           |     100 |       50 |     100 |     100 |                 5
 src/example-3           |       0 |        0 |       0 |       0 |
  example-3.js           |       0 |        0 |       0 |       0 |     4,5,7,8,11,14
-------------------------|---------|----------|---------|---------|-------------------

=============================== Coverage summary ===============================
Statements   : 39.39% ( 26/66 )
Branches     : 30.56% ( 11/36 )
Functions    : 35.71% ( 5/14 )
Lines        : 39.39% ( 26/66 )
================================================================================
```

Exemple d'un rapport de istanbul pour JavaScript

COMMENT ASSURER UNE BONNE COUVERTURE DE CODE :

- **Différents outils pour différents langages**

PHPUnit Xdebug : php

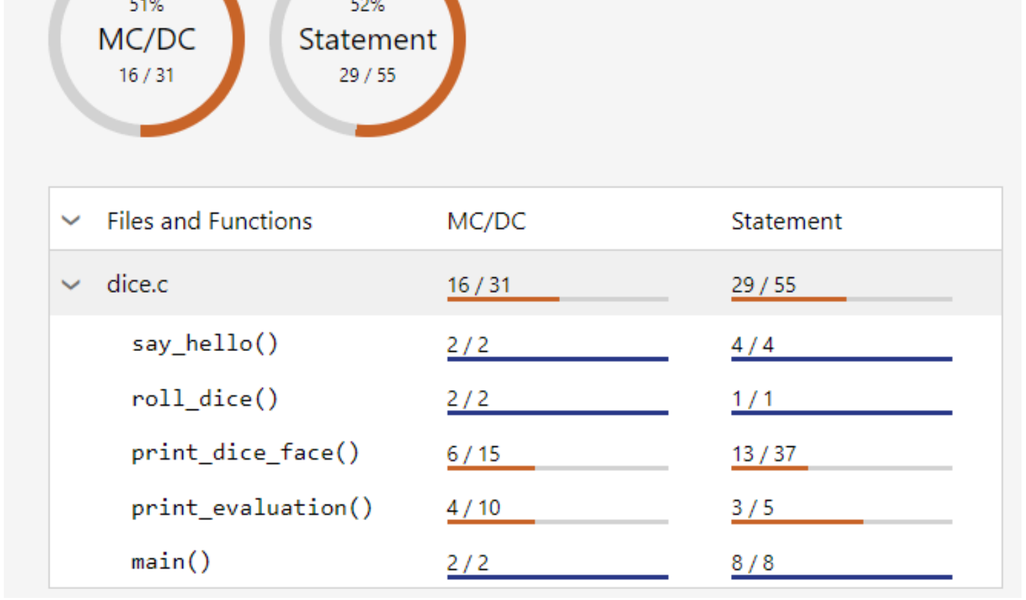
```
→  test-coverage php artisan test --coverage

  PASS  Tests\Unit\ExampleTest
  ✓ that true is true

  PASS  Tests\Feature\ExampleTest
  ✓ the application returns a successful response

  Tests:  2 passed
  Time:   0.32s

  Console/Kernel 19 ..................................................... 66.7 %
  Exceptions/Handler .................................................... 100.0 %
  Http/Controllers/Controller ........................................... 100.0 %
  Http/Kernel ........................................................... 100.0 %
  Http/Middleware/Authenticate ..........................................   0.0 %
  Http/Middleware/EncryptCookies ........................................ 100.0 %
  Http/Middleware/PreventRequestsDuringMaintenance ...................... 100.0 %
  Http/Middleware/RedirectIfAuthenticated ...............................   0.0 %
  Http/Middleware/TrimStrings ........................................... 100.0 %
  Http/Middleware/TrustHosts ............................................   0.0 %
  Http/Middleware/TrustProxies .......................................... 100.0 %
  Http/Middleware/ValidateSignature ..................................... 100.0 %
  Http/Middleware/VerifyCsrfToken ....................................... 100.0 %
  Models/User ........................................................... 100.0 %
  Providers/AppServiceProvider .......................................... 100.0 %
  Providers/AuthServiceProvider ......................................... 100.0 %
  Providers/BroadcastServiceProvider ....................................   0.0 %
  Providers/EventServiceProvider ........................................ 100.0 %
  Providers/RouteServiceProvider 49 ..................................... 90.9 %

  Total Coverage ........................................................ 57.6 %
```

Exemple d'un rapport de couverture de Xdebug pour PHP (Laravel)

COMMENT ASSURER UNE BONNE COUVERTURE DE CODE :

- **Différents outils pour différents langages**

Testwell CTC++ : et



Exemple d'un rapport de Testwell CTC++ pour C

GCOV , L’OUTIL EN C++

Pourquoi utiliser GCov ?

- Détecter le code mort (fonctions inutilisées, branches jamais exécutées)
- Améliorer la qualité du code en s'assurant qu'un maximum de cas sont testés
- Optimiser le programme en supprimant le code inutile
- Bonne documentation fournie par Gcov et Lcov

GCOV , L’OUTIL EN C++

Installation

gcov et lcov sont des outils de couverture de code utilisés pour analyser l'exécution du code source en C/C++.

- gcov : Outil de couverture de code inclus avec GCC.
- lcov : Interface graphique pour gcov qui génère des rapports en HTML.

Sur Linux

sudo apt update
sudo apt install gcc

Sur Mac-OS

xcode-select --install

Sur Windows

Installez MySys2.
Installez lcov manuellement en téléchargeant les sources depuis le github officiel ou en utilisant:
pacman -S mingw-w64-x86_64-lcov
Ajoutez le script lcov au PATH.

UTILISATION CONCRÈTE DANS UN PROJET

FTCPP

```
Bievenue sur FTCPP
220 Welcome to the DLP Test FTP Server

Enter username: dlpuser
331 Please specify the password.

Enter password: default password taken for testing
230 Login successful.

227 Entering Passive Mode (44,241,66,173,4,119).

Data connection: 44.241.66.173:1143
150 Here comes the directory listing.
```

UTILISATION CONCRÈTE DANS UN PROJET

Comment fonctionne gcov et lcov ?

Compilation avec support de couverture :

Pour générer des informations de couverture, le programme doit être compilé avec gcc en utilisant les options -fprofile-arcs -ftest-coverage :

```
g++ -fprofile-arcs -ftest-coverage -o FTCPP.exe main.cpp function.cpp -lws2_32
```

On execute le programme:

```
./FTCPP.exe
```

On utilise Gcov:

```
gcov FTCPP-main FTCPP-function
```

UTILISATION CONCRÈTE DANS UN PROJET

Génération d’un rapport html avec lcov

Capture les données de couverture:

lcov --capture --directory . --output-file coverage.info

Génère un rapport:

genhtml coverage.info --output-directory coverage_report

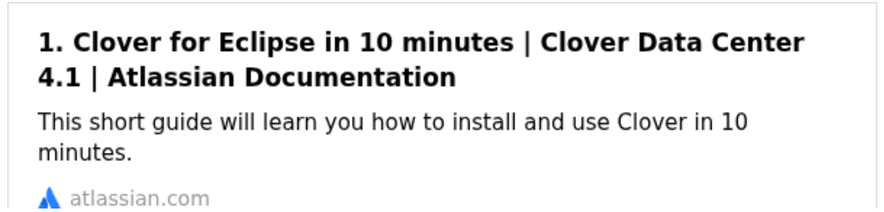
ouverture du rapport dans un navigateur:

“navigateur” coverage.html

CE QU'IL FAUT RETENIR

Est-ce qu'un programme est utilisé entièrement, et si oui est ce qu'il est BIEN utilisé ?

SOURCES



Source : https://kincy.fr/faille-securite/
Failles de sécurité

https://fr.wikipedia.org/wiki/Chemin_critique
- Chemin critique informatique

chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://www.verifysoft.com/de_Testwell_CTC++_at_a_Glance.pdf

https://verifysoft.com/en_ctcpp.html

chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://www.inf.ed.ac.uk/teaching/courses/st/2017-18/Path-coverage.pdf

chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://www.inf.ed.ac.uk/teaching/courses/st/2017-18/Path-coverage.pdf