Softwarequalitätssicherung

Codeüberdeckung

Gruppenmitglieder: Dennis Jongebloed
Julian Hinxlage
Johannes Theiner

Semester: Sommersemester 2020
letzte Änderung: 23. April 2020

1 Codeüberdeckung

1.1 Welche Metriken werden von EclEmma angeboten?

1.1.1 C0 Coverage - Anweisungsüberdeckung

Die kleinste Einheit in JaCoCo sind einzelne Java-Bytecode-Anweisungen. Die Anweisungsabdeckung enthält Informationen über die Anzahl an Code, die ausgeführt oder die nicht ausgeführt wurden. Diese Metrik ist unabhängig von der Quellformatierung und ist immer verfügbar, auch dann, wenn keine Debug-Informationen in den Klassendaten vorhanden sind.

$$Anweisungsüberdeckung = \frac{Anzahl\ \ der\ \ durchlaufenen\ \ Anweisungen}{Gesamtzahl\ \ Anweisungen} \times 100\%$$

1.1.2 C1 Coverage - Zweigüberdeckungs

JaCoCo berechnet Zweigüberdeckung für alle if und switch Anweisungen. Diese Metrik zählt alle solcher Zweige in einer Methode und bestimmt die Anzahl der ausgeführten oder fehlenden Zweige. Die Zweigüberdeckung ist immer verfügbar, auch wenn keine Debug-Informationen in den Klassendaten vorhanden sind.

$$Zweigüberdeckung = \frac{Anzahl\ \ der\ \ durchlaufenen\ \ Zweige}{Gesamtzahl\ \ Zweige} \times 100\%$$

1.1.3 Zyklomatische Komplexität

Zyklomatische Komplexität gibt an, wie viele Testfälle höchstens nötig sind, um eine Zweigüberdeckung zu erreichen. Laut McCabe sollte die zyklomatische Komplexität einer Methode nicht höher als 10 sein, da es schwieriger wird den Code nachzuvollziehen und zu testen.

$v(G) = e - n + 2$

G = Kontrollflussgraph
e = Anzahl der Kanten von G
n = Anzahl der Knoten von G

JaCoCo berechnet die Zyklomatische Komplexität mit die Anzahl der Branches und die Anzahl der Entscheidungspunkte.

$v(G) = b - d + 1$ G = Kontrollflussdiagramm
b = Anzahl der Zweige von G
d = Anzahl der Entscheidungspunkte von G

1.1.4 Zeilen

Für alle Klassen die mit debug Informationen kompiliert wurden, können Abdeckungsinformationen für einzelne Zeilen berechnet werden. Eine Zeile gilt als ausgeführt, wenn mindestens eine dieser Zeilen zugewiesene Anweisung ausgeführt wurde. Abhängig von der Formatierung des Codes, kann sich eine Zeile eines Quellcodes auf mehrere Methoden oder Klassen beziehen. So kann die Zeilenanzahl der Methoden nicht einfach addiert werden, um die Gesamtzahl für die enthaltende Klasse zu erhalten. Dies gilt auch für die Zeilen mehrerer Klassen in einer Datei. JaCoCo berechnet die Zeilenabdeckung für Klassen und Quelldateien basierend auf den tatsächlich abgedeckten Zeilen.

1.1.5 Methoden

Jede nicht abstrakte Methode enthält mindestens eine Anweisung. Eine Methode gilt als ausgeführt, wenn mindestens eine Anweisung ausgeführt wurde. Jacoco arbeitet auf bytecode Ebene, so werden Konstruktoren und statische initializers als Methoden gezählt.

1.1.6 Klassen

Eine Klasse gilt als ausgeführt, wenn mindestens eine der Methoden ausgeführt wurden. Hierbei muss man drauf achten, dass JaCoCo Kontruktoren und statische initializers als Methoden betrachtet. Da Java Interfaces statische initializers enthalten können, werden solche Schnittstellen auch als ausführbare Klassen betrachtet.

1.2 Wie verhalten sich diese Metriken zu den Whitebox-Testverfahren?

- Der Anweisungstest führt zu einer hohen Anweisungsüberdeckung.
- Der Zweigtest führt zu einer hohen Zweigüberdeckung und hohen Anweisungsüberdeckung.
- Der Bedingungstest führt zu keinem hohen Wert einer Metrik.
- Mehrfachbedingungstest optimiert sowohl die Anweisungsüberdeckung als auch die Zweigüberdeckung.
- Definierter Bedinungstest optimiert auch sowohl die Anweisungsüberdeckung als auch die Zweigüberdeckung.
- Auf die Zyklomatische Komplexität, Zeilen, Methoden und Klassen haben die Tests keine Auswirkungen.

2 Codeüberdeckungsmessung

2.1 Messwerte

Metrik	Coverage	Covered	Missed
Instruction	98,1%	103	2
Branch	68,2%	15	7
Line	93,3%	14	1
Method	100%	3	0
Type	100%	2	0
Complexity	50%	7	7

Tabelle 1: Überdeckung Unit Tests

Metrik	Coverage	Covered	Missed
Instruction	98,%	103	2
Branch	68,2%	15	7
Line	93,3%	14	1
Method	100%	3	0
Type	100%	2	0
Complexity	50%	7	7

Tabelle 2: Überdeckung Anweisungsüberdeckung

Metrik	Coverage	Covered	Missed
Instruction	100%	105	0
Branch	72,7%	16	6
Line	100%	15	0
Method	100%	3	0
Type	100%	2	0
Complexity	57,1%	8	6

Tabelle 3: Überdeckung Zweigüberdeckung

Metrik	Coverage	Covered	Missed
Instruction	100%	105	0
Branch	72,7%	16	6
Line	100%	15	0
Method	100%	3	0
Type	100%	2	0
Complexity	57,1%	8	6

Tabelle 4: Überdeckung definierter Bedingungstest

2.2 Bewertung

- Es gibt kein Verfahren welches 100% Abdeckung produziert, nur mit einer Kombination ist dies möglich.
- Einzelne Bedingungen für eine Abdeckung von 100% sind gewöhnungsbedürftig (Konstruktor muss verwendet werden).

3 Codeüberdeckungsmessung im Projekt

PMD nutzt das Tool Coveralls, um die Codeüberdeckung zu ermitteln.

3.1 Welche Überdeckung wird erreicht ?

Die Gesamtabdeckung des Projektes ist 45%.
Dabei schwankt die Abdeckung der einzelnen Module für die verschiedenen Programmiersprachen.
Java hat z. B. 81.71% coverage, JavaScript hat 27.72% und Swift nur 0.27%.

3.2 Welche Überdeckungsmaße werden verwendet ?

Coveralls ermittelt wie viele Zeilen an Code bei der Ausführung erreicht werden. Die anderen Metriken wie Anweisungen oder Zweige werden nicht ermittelt.

3.3 Ist die Codeüberdeckung angemessen ?

Die Abdeckung ist mit 45% unserer Meinung nicht ausreichend.
Dabei ist aber z.b. die Abdeckung des Java Moduls, auf welches PMD an meisten Wert legt, mit 81.71% in Ordnung.