**JaCoCo.**

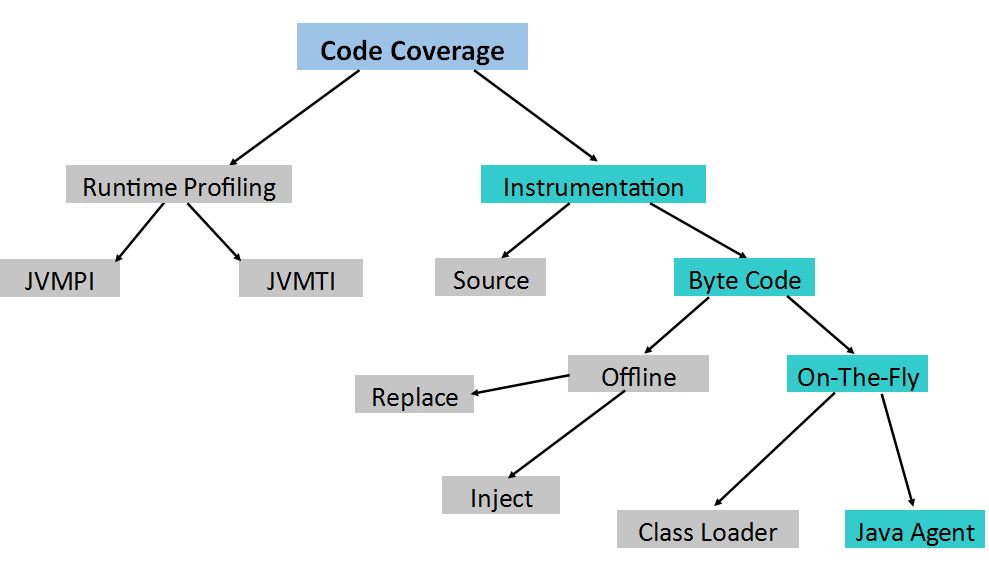
**Overview.**

JaCoCo, an open-source toolkit which operates at byte code level to measure, analyse, and report code coverage for Java projects. The aim of code coverage tools is to determine which segments of the program have been tested by recording the lines of code which execute successfully during a running test.

***Coverage Counters.***

Code coverage is calculated in coverage metrics (measurements which are usually expressed as a percentage). JaCoCo calculates coverage metrics using a collection of different counters to deliver effective analysis of software by extracting information from Java class files as the code is being executed (On-The-Fly). The information collected is then mapped back to the source code and a detailed visualisation of the code analysis is generated.

There are various methods of collecting code coverage information and different application techniques are used. The diagram below illustrates the techniques available and highlights the techniques used by JaCoCo.



***Instructions (C0 Coverage).***

Instruction coverage extracts information about the quantity of code which has been missed or executed by counting single Java byte code instructions, the smallest unit count JaCoCo is capable of processing.

***Branches (C1 Coverage)***

JaCoCo analyses branch coverage for *if* and *switch* statements. The number of branches in a method are counted to record the amount which are missed or executed. The results are highlighted as:

* No coverage – No branches in the line are executed (red).
* Partial coverage – A portion on branches in the line are executed (yellow).
* Full coverage – All branches in the line are executed (green).

***Cyclomatic Complexity.***

Cyclomatic complexity is defined as the minimum number of paths that can generate all possible paths through a method. The process is used to summarise the complexity of a class, package or group. The complexity value then indicates the number of unit test cases necessary to fully analyse the software.

***Lines.***

Coverage for individual lines can be calculated for complied class files which include debug information. When compiled, an individual line can contain numerous byte code instructions – each source line is logged as executed when one or more instructions attached to the line have been executed. The results are highlighted as:

* No coverage – No line has been executed (red).
* Partial coverage – A portion of the instructions in a line have been executed (yellow).
* Full coverage – All instructions in the line are executed (green).

***Methods.***

A method is treated as executed when one or more instructions associated with the method have been executed (non-abstract methods consist of one or more instructions). Due to the fact that JaCoCo operates at byte code level, it counts static initialisers and constructers as methods.

***Classes.***

When one or more of its methods have been executed, a class is considered as executed. As stated earlier, JaCoCo counts static initialisers and constructers as methods – static initialisers may be included in Java interfaces types and can also be counted as executable classes.