# Metric Functionality Testing

## Manual Data Analysis

To test the functionality of these metrics, a Java class that contained contribution points for all of the applicable metrics was manually analysed, and the found results were compared against the data generated by MAIJ, to verify the correctness of the metrics. The code analysed was pulled from a public Github project, which will be linked below this section, as well as the full class that was analysed.

## Class Metrics

LOCMetric: This metric is a measure of the lines of code inside a class. This is measured by every line of code that contributes to the body of a class, e.g. anything inside the class declaration. This excludes import statements, as they do not form part of the class declaration in Java.

**Manual Data: 108**

**Program Generated Data: 108**

**MATCH**

NOSMetric: This metric is a measure of the number of statements inside of a class or method. This is measured by every assignment statement or return statement. This excludes variable declarations without a data assignment.

**Manual Data: 12**

**Program Generated Data: 12**

**MATCH**

NOAMMetric: This metric is a measure of number of accessor methods contained within a class, e.g. any method that provides access to, or returns, internal class data, whether that data is public, protected or private, and accessible through direct indexing.

**Manual Data: 10**

**Program Generated Data: 10**

**MATCH**

FanOutMetric: This metric is a measure of how many non-primitive data types are introduced or declared inside of a class or a method. In Java, this is any internal data type that is starts with a capital letter as opposed to lowercase, e.g. “int” vs “Integer”, or any imported data types or structures.

**Manual Data: 1**

**Program Generated Data: 1**

**MATCH**

NOPAMetric: This metric is a measure of the amount of public accessor methods contained within a class. This differs from the NOAMMetric, as it contains only public accessor methods, so any protected or private methods used for internal data manipulation, such as singleton instance generators.

**Manual Data: 10**

**Program Generated Data: 10**

**MATCH**

ATFDMetric: This metric is a measure of access to foreign data, e.g. all data that can be accessed from outside of the class. This is contributed to by any access through direct indexing to a field, or through an accessor method, for any class or method declared out of the scope of the class.

**Manual Data: 0**

**Program Generated Data: 0**

**MATCH**

### Project Link

<https://github.com/structurizr/java/releases/tag/1.0.0-RC5>

Code Excerpt 1

## Method Metrics

The method level metrics were tested on a different method to the class level metrics, a large method inside of a library from google used for pattern matching. The large methods inside of this class made it ideal for evaluating the class and method level metrics.

CycloMetric: This metric is a measure of cyclomatic complexity of a method, the number of different traversal paths that exist within a code block. This increments every time there is a choice to be made, such as an “if” statement or a “switch” statement.

**Used Method:** diff\_CleanupMerge(LinkedList)

**Manual Data: 34**

**Program Generated Data: 34**

**MATCH**

MaxNestingMetric: This metric is a measure of maximum nesting of code paths contained within a method. It increments every time there is a statement nested within another statement, either through a choice or through a loop.

**Used Method:** diff\_CleanupMerge(LinkedList)

**Manual Data: 6**

**Program Generated Data: 6**

**MATCH**

### Project Link

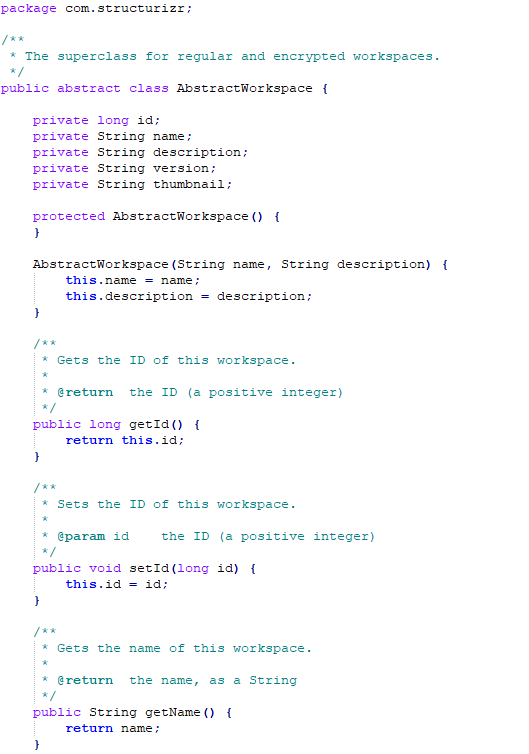
<https://github.com/google/diff-match-patch>

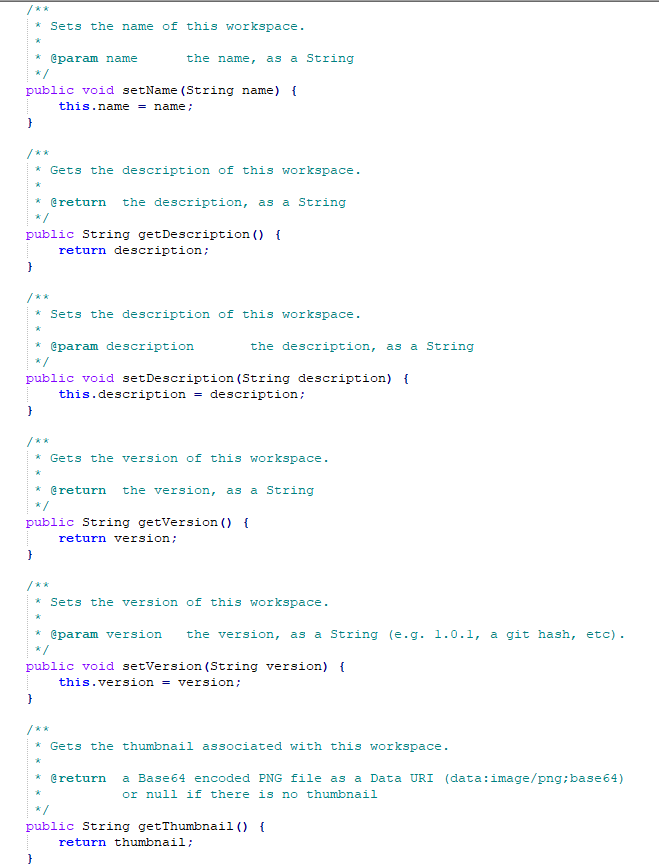
Code Excerpt 2

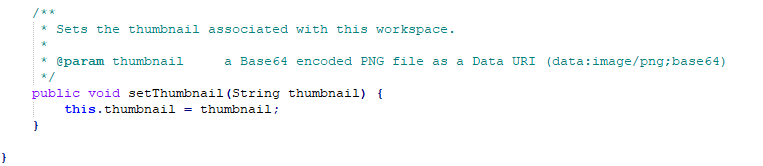
## Code Excerpt 1

### Relative Path:

\structurizr-core\src\com\structurizr\AbstractWorkspace.java







## Code Excerpt 2

Relative Path

/java/src/name/name/fraser/neil/plaintext/diff\_metch\_patch.java

