# 3 The Value of Code Coverage

## 3.1 Essential Step / Reveals Untested Code

It is an essential step in the testing process to determine the code coverage that was reached during testing. Code coverage measurement is important with respect to software quality, because it would reveal if e.g. a part of the code was never executed during testing. This would be a hint for "dead code" or insufficient test cases.

The value of code coverage may be increased if not only the reached percentage is considered, but also single test cases are investigated. Did the test case execute the expected path?

## 3.2 Can Indicate Project Progress

If the total coverage of a project is monitored over time, this can be an indication for the progress of the project.

For instance:

(1) If the percentage of the total coverage grows, this could mean testing is catching up.

(2) If the percentage of the total coverage is getting smaller, this could mean new code is added without testing it appropriately.

However, this discussion is beyond the topic of the paper at hand.

## 3.3 Two Weaknesses of Code Coverage

However, even if you have reached the desired percentage, you must keep two weaknesses of code coverage in mind:

(1) Code coverage measurement cannot detect omissions, e.g. missing or incomplete code.

(2) Code coverage measurement is insensitive to calculations.

### 3.3.1 Cannot Detect Omissions

For instance, the specification for a function could state:

"The maximum input value the algorithm can handle is 500. If the input value is greater than 500, it shall be set to 500."

If the implementation of the function misses to check the input value, and therefore also misses to take an appropriate action if the value is greater 500, code coverage measurement will not detect the missing check, even if a test case with the input value 501 is executed.

### 3.3.2 Insensitive to Calculations

Consider the following function.

```
long double sin(long double x_deg)
{
      int i;
      long double temp, x_rad;
      int sign = -1;

      x_rad = x_deg / 180 * pi ;
      temp = x_rad;

      for(i=3; i<=(MAX_FAC-2); i+=2)
      {
            temp += sign * pot(x_rad,i) / fac(i);
            sign *= -1;
      }
      return(temp);
}
```

Fig. 34 A function that calculates the sinus value of its input

A single test case with the input x_deg == 0 (i.e. an angle of zero degrees) returns 0, what is the correct and expected result. Code coverage measurement of this single test case yields 100% branch coverage, 100% MC/DC, and 100% MCC. Although you have reached 100% coverage for all measures, you have no evidence that the sinus calculation is correct. (A completely different function could be implemented, e.g. signum(). It would also return 0 for an input of 0).

This is an example where 100% code coverage is not enough.