**SE401: Software Quality Assurance and Testing**

**Activity 9.1: Code Coverage Analysis using EclEmma**

# Objective

* Understand the importance of Code coverage in assessing testing of software deliverables
* Use a code coverage tool to determine untested segments of source code.
* Measure code coverage in Java using the EclEmma tool.
* Construct new test cases to increase code coverage.
* Use external models to generate test cases for a project.

# Introduction

One of the hallmarks used for software testing is the concept of code coverage. While the exact relationship is often elusive, research has generally shown that the greater the code coverage during testing, the greater the reliability of the deployed software will be once the system is completed. There has been significant study of the relationship between code coverage and the resulting reliability of the source code. Garg and Del Frate indicate that there is a strong correlation between code coverage obtained during testing and software reliability, especially in larger programs. The exact extent of this relationship, however, is unknown, but the general trend is consistent, though highly non-linear, as is shown in Figure 1.

Piwowarski, Ohba, and Caruso indicate that 70% statement coverage is necessary to ensure sufficient test case coverage, 50% statement coverage is insufficient to exercise the module, and beyond 70%-80% is not cost effective. Hutchins indicates that even 100% coverage is not necessarily a good indication of testing adequacy, for though more faults are discovered at 100% coverage than 90% or 95% coverage, faults can still be uncovered even if testing has reached 100% coverage. These recommendations, however, are quite old and reflect trying to do code coverage from an acceptance testing approach, not a unit testing approach.

Extreme Programming advocates and agile programming methods tend to endorse 100% method coverage in order to ensure that all methods are invoked at least once, though there are also exceptions given for small functions that are smaller than the test cases would be. Method coverage, however, is a very weak coverage measure. Stronger coverage methods include statement coverage and branch coverage.

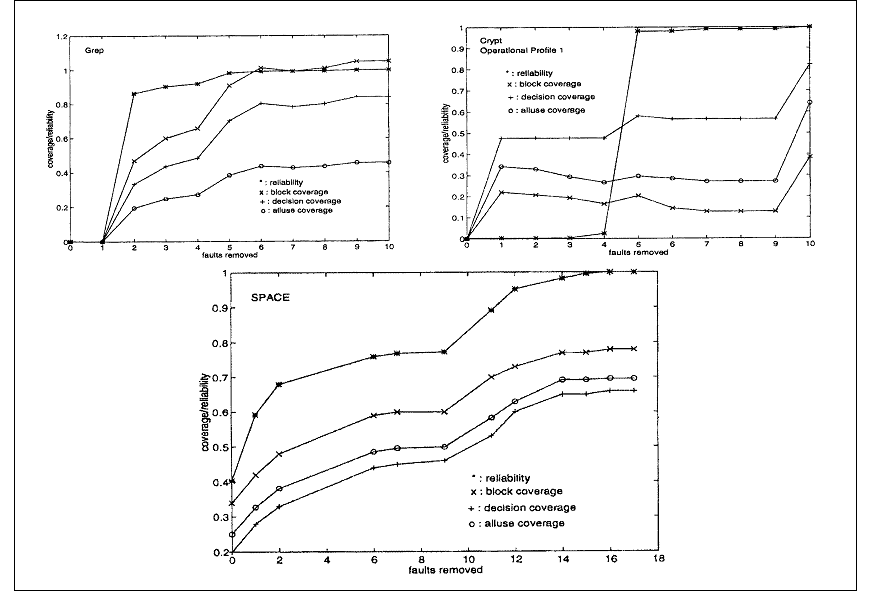
[](https://camo.githubusercontent.com/e50e9377f502228d88588bebe485fdcb12b6eaad706fbf777fce12000a6af360/687474703a2f2f692e696d6775722e636f6d2f6231426a5736472e706e67)

Figure 1. The relationship between coverage and reliability.

We will be using the EclEmma plug-in for code coverage that collects metrics about the code coverage obtained when you execute your program.

# Deliverables

* Submit the source and test cases along with code coverage reports.
* Add a file called write-up to include your name, email, include the following:
  + A screenshot of your final code coverage
  + An analysis of why, with specifics, you could not achieve 100% statement coverage.
* Your grade will primarily be based upon the level of coverage achieved, and the quality of your analysis.