Minimum Acceptable Code Coverage

This paper discusses how to decide what percentage of code coverage you need.

Summary

Code coverage of 70-80% is a reasonable goal for system test of most projects with most coverage metrics. Use a higher goal for projects specifically organized for high testability or that have high failure costs. Minimum code coverage for unit testing can be 10-20% higher than for system testing.

Introduction

Empirical studies of real projects found that increasing code coverage above 70-80% is time consuming and therefore leads to a relatively slow bug detection rate. Your goal should depend on the risk assessment and economics of the project. Consider the following factors.

- Cost of failure. Raise your goal for safety-critical systems or where the cost of a failure is high, such as products for the medical or automotive industries, or widely deployed products.
- Resources. Lower your goal if testers are spread thin or inadequately trained. If your testers are unfamiliar with the application, they may not recognize a failure even if they cover the associated code.
- Testable design. Raise your goal if your system has special provisions for testing such as a method for directly accessing internal functionality, bypassing the user interface.
- Development cycle status. Lower your goal if you are maintaining a legacy system where the original design engineers are no longer available.

Many projects set no particular minimum percentage required code coverage. Instead they use code coverage analysis only to save time. Measuring code coverage can quickly find those areas overlooked during test planning.

Defer choosing a code coverage goal until you have some measurements in hand. Before measurements are available, testers often overestimate their code coverage by 20-30%.

Full Coverage Generally Impractical

Although 100% code coverage may appear like a best possible effort, even 100% code coverage is estimated to only expose about half the faults in a system. Low code coverage indicates inadequate testing, but high code coverage guarantees nothing.

In a large system, achieving 100% code coverage is generally not cost effective. Some reasons are listed below.

- Some test cases are expensive to reproduce but are highly improbable. The cost to benefit ratio does not justify repeating these tests simply to record the code coverage.
- Checks may exist for unexpected error conditions. Layers of code might obscure whether errors in low-level code propagate up to higher level code. An engineer might decide that handling all errors creates a more robust solution than tracing the possible errors.
- Unreachable code in the current version might become reachable in a future version. An engineer might address uncertainty about future development by investing a little more effort to add some capability that is not currently needed.
- Code shared among several projects is only partially utilized by the project under test.

Generally, the tester should stop increasing code coverage when the tests become contrived. When you focus more and more on making the coverage numbers better, your motivation shifts away from finding bugs.

Unit, Integration and System Testing

You can attain higher code coverage during unit testing than in integration testing or system testing. During unit testing, the tester has more facilities available, such as a debugger to manipulate data and conditional compilation to simulate error conditions.

Likewise, higher code coverage is possible during integration testing than in system testing. During integration testing, the test harness often provides more precise control and capability than the system user interface.

Therefore it makes sense to set progressively lower goals for unit testing, integration testing, and system testing. For example, 90% during unit testing, 80% during integration testing, and 70% during system testing.

Coverage Metrics

The information in this paper applies to code coverage metrics that consider control structures independently. Specifically, these are:

- statement coverage (line coverage)
- basic block coverage
- decision coverage (branch coverage)
- condition/decision coverage
- modified condition/decision coverage (MCDC)

Although some of these metrics are less sensitive to control flow than others, they all correlate statistically at a large scale.

Formal Standards

DO-178B

The aviation standard DO-178B requires 100% code coverage for safety critical systems. This standard specifies progressively more sensitive code coverage metrics for more critical systems.

Effect of System Failure	Level	Example	Required Code Coverage
Catastrophic	A	Crash	100% modified condition/decision coverage and 100% statement coverage
Hazardous	B	Passenger fatality	100% decision coverage and 100% statement coverage
Major	C	Passenger injury	100% statement coverage
Minor	D	Flight plan change	No code coverage requirement
No effect	E	Entertainment system failure	No code coverage requirement

These requirements consider neither the probability of a failure nor the cost of performing test cases.

IEC 61508

The standard IEC 61508:2010 "Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems" recommends 100% code coverage of several metrics, but the strenuousness of the recommendation relates to the criticality.

Safety Integrity Level	100% entry points	100% statements	100% branches	100% conditions, MC/DC
1 (least critical)	Highly Recommended	Recommended	Recommended	Recommended
2	Highly Recommended	Highly Recommended	Recommended	Recommended
3	Highly Recommended	Highly Recommended	Highly Recommended	Recommended
4 (most critical)	Highly Recommended	Highly Recommended	Highly Recommended	Highly Recommended

The safety integrity level (SIL) relates to the probability of unsafe failure. Determining the SIL involves a lengthy risk analysis.

This standard recommends but does not require 100% coverage. It specifies you should explain any uncovered code.

This standard does not define the coverage metrics and does not distinguish between condition coverage and MC/DC.

ISO 26262

ISO 26262:2011 "Road vehicles -- Functional safety" requires measuring code coverage, and specifies that if the level achieved "is considered insufficient", then a rationale must be provided. The standard recommends different coverage metrics for unit testing than for integration testing. In both cases, the strenuousness of the recommendations relates to the criticality.

For unit testing, three coverage metrics are recommended, shown in the table below. The standard does not provide definitions for these metrics.

Methods	ASIL			
	A (least critical)	B	C	D (most critical)
Statement coverage	highly recommended	highly recommended	recommended	recommended
Branch coverage	recommended	highly recommended	highly recommended	highly recommended
Modified Condition/Decision Coverage	recommended	recommended	recommended	highly recommended

For integration testing, two metrics are recommended, shown in the table below

For integration testing, two metrics are recommended, shown in the table below:

Methods	ASIL			
	A (least critical)	B	C	D (most critical)
Function coverage	recommended	recommended	highly recommended	highly recommended
Call coverage	recommended	recommended	highly recommended	highly recommended

The standard defines function coverage as the percentage of executed software functions, and call coverage as the percentage of executed software function calls.

The automotive safety integrity level (ASIL) is based on the probability of failure, effect on vehicle controllability, and severity of harm. The ASIL does not correlate directly to the SIL of IEC 61508.

The code coverage requirements are contained in part 6 "Product development at the software level."

ANSI/IEEE 1008-1987

The IEEE Standard for Software Unit Testing section 3.1.2 specifies 100% statement coverage as a completeness requirement. Section A9 recommends 100% branch coverage for code that is critical or has inadequate requirements specification. Although this document is quite old, it was reaffirmed in 2002.

FDA requirements for Medical Devices

The United States Federal Drug Administration FDA document General Principles of Software Validation issued Jan 2002 recommends structural testing but does not specify any particular code coverage requirement. The document says only:

> The amount of structural coverage should be commensurate with the level of risk posed by the software.

References

Efficient use of code coverage in large-scale software development, Yong Woo Kim, 2003

Code coverage, what does it mean in terms of quality? Williams et al, 2001

An Empirical Study of the Branch Coverage of Different Fault Classes, Melissa Cline and Linda Werner, 1994

Coverage measurement experience during function test, Paul Piwowarski et al, 1993

How Do You Know When You Are Done Testing?, Richard Bender, 2000

How to Misuse Code Coverage, Brian Marick, 1997

Comparing the Effectiveness of Software Testing Strategies, Basili and Selby, 1987

Understanding the Use, Misuse and Abuse of Safety Integrity Levels, Felix Redmil, 2000