Code coverage

Code coverage is a measure used in software testing. It describes the degree to which the source code of a program has been tested.

Code coverage was among the first methods invented for systematic software testing. The first published reference was by Miller and Maloney in *Communications of the ACM* in 1963.[1]

Code coverage is one consideration in the safety certification of avionics equipment. The standard by which avionics gear is certified by the Federal Aviation Administration (FAA) is documented in DO-178B.[2]

Coverage criteria

To measure how well the program is exercised by a test suite, one or more *coverage criteria* are used.

Basic coverage criteria

There are a number of coverage criteria, the main ones being:[3]

- **Function coverage** - Has each function (or subroutine) in the program been called?
- **Statement coverage** - Has each node in the program been executed?
- **Decision coverage** (not the same as **branch coverage**.[4]) - Has every edge in the program been executed? For instance, have the requirements of each branch of each control structure (such as in IF and CASE statements) been met as well as not met?
- **Condition coverage** (or predicate coverage) - Has each boolean sub-expression evaluated both to true and false? This does not necessarily imply decision coverage.
- **Condition/decision coverage** - Both decision and condition coverage should be satisfied.
- **Parameter Value Coverage** [5] - In a method taking parameters, has all the common values for such parameter been considered?

For example, consider the following C++ function:

```cpp
int foo (int x, int y)
{
    int z = 0;
    if ((x>0) && (y>0)) {
        z = x;
    }
    return z;
}
```

Assume this function is a part of some bigger program and this program was run with some test suite.

- If during this execution function 'foo' was called at least once, then *function coverage* for this function is satisfied.
- *Statement coverage* for this function will be satisfied if it was called e.g. as `foo(1,1)`, as in this case, every line in the function is executed including `z = x;`.
- Tests calling `foo(1,1)` and `foo(0,1)` will satisfy *decision coverage*, as in the first case the 2 `if` conditions are met and `z = x;` is executed, while in the second case, the first condition (x > 0) is not satisfied, which makes the code `z = x;` not executed.
- *Condition coverage* can be satisfied with tests that call `foo(1,1)`, `foo(1,0)` and `foo(0,0)`. These are necessary as in the first two cases `(x>0)` evaluates to `true` while in the third it evaluates `false`. At the same time, the first case makes `(y>0)` `true` while the second and third make it `false`.

- *Parameter Value coverage* (PVC) is the idea that all common possible values for a parameter are tested. For example, a string could be any of these commonly: 1) null, 2) empty, 3) whitespace (space, tabs, new line), 4) valid string, 5) invalid string, 6) single-byte string, 7) double-byte string. Failure to test each possible parameter value may leave a bug. Testing only one of these could result in 100% code coverage as each line is covered, but only one of seven options are tested, there is only 14.2% PVC.

In languages, like Pascal, where standard boolean operations are not short circuited, condition coverage does not necessarily imply decision coverage. For example, consider the following fragment of code:

```
if a and b then
```

Condition coverage can be satisfied by two tests:

- `a=true, b=false`
- `a=false, b=true`

However, this set of tests does not satisfy decision coverage as in neither case will the `if` condition be met.

Fault injection may be necessary to ensure that all conditions and branches of exception handling code have adequate coverage during testing.

Modified condition/decision coverage

For safety-critical applications (e.g., for avionics software) it is often required that **modified condition/decision coverage (MC/DC)** be satisfied. This criterion extends condition/decision criteria with requirements that each condition should affect the decision outcome independently. For example, consider the following code:

```
if (a or b) and c then
```

The condition/decision criteria will be satisfied by the following set of tests:

- a=true, b=true, c=true
- a=false, b=false, c=false

However, the above tests set will not satisfy modified condition/decision coverage, since in the first test, the value of 'b' and in the second test the value of 'c' would not influence the output. So, the following test set is needed to satisfy MC/DC:

- a=**false**, b=**false**, c=true
- a=**true**, b=false, c=**true**
- a=false, b=**true**, c=**true**
- a=true, b=true, c=**false**

The bold values influence the output, each variable must be present as an influencing value at least once with false and once with true.

Multiple condition coverage

This criterion requires that all combinations of conditions inside each decision are tested. For example, the code fragment from the previous section will require eight tests:

- a=false, b=false, c=false
- a=false, b=false, c=true
- a=false, b=true, c=false
- a=false, b=true, c=true
- a=true, b=false, c=false
- a=true, b=false, c=true
- a=true, b=true, c=false

- a=true, b=true, c=true

Other coverage criteria

There are further coverage criteria, which are used less often:

- **Linear Code Sequence and Jump (LCSAJ) coverage** - has every LCSAJ been executed?
- **JJ-Path coverage** - have all jump to jump paths[6] (aka LCSAJs) been executed?
- **Path coverage** - Has every possible route through a given part of the code been executed?
- **Entry/exit coverage** - Has every possible call and return of the function been executed?
- **Loop coverage** - Has every possible loop been executed zero times, once, and more than once?
- **Parameter Value Coverage** [5] - For each parameter in a method, have the most common possible parameter values been tested?

Safety-critical applications are often required to demonstrate that testing achieves 100% of some form of code coverage.

Some of the coverage criteria above are connected. For instance, path coverage implies decision, statement and entry/exit coverage. Decision coverage implies statement coverage, because every statement is part of a branch.

Full path coverage, of the type described above, is usually impractical or impossible. Any module with a succession of n decisions in it can have up to 2^n paths within it; loop constructs can result in an infinite number of paths. Many paths may also be infeasible, in that there is no input to the program under test that can cause that particular path to be executed. However, a general-purpose algorithm for identifying infeasible paths has been proven to be impossible (such an algorithm could be used to solve the halting problem).[7] Methods for practical path coverage testing instead attempt to identify classes of code paths that differ only in the number of loop executions, and to achieve "basis path" coverage the tester must cover all the path classes.

In practice

The target software is built with special options or libraries and/or run under a special environment such that every function that is exercised (executed) in the program(s) is mapped back to the function points in the source code. This process allows developers and quality assurance personnel to look for parts of a system that are rarely or never accessed under normal conditions (error handling and the like) and helps reassure test engineers that the most important conditions (function points) have been tested. The resulting output is then analyzed to see what areas of code have not been exercised and the tests are updated to include these areas as necessary. Combined with other code coverage methods, the aim is to develop a rigorous, yet manageable, set of regression tests.

In implementing code coverage policies within a software development environment one must consider the following:

- What are coverage requirements for the end product certification and if so what level of code coverage is required? The typical level of rigor progression is as follows: Statement, Branch/Decision, Modified Condition/Decision Coverage(MC/DC), LCSAJ (Linear Code Sequence and Jump)
- Will code coverage be measured against tests that verify requirements levied on the system under test (DO-178B)?
- Is the object code generated directly traceable to source code statements? Certain certifications, (i.e. DO-178B Level A) require coverage at the assembly level if this is not the case: "Then, additional verification should be performed on the object code to establish the correctness of such generated code sequences" (DO-178B) para-6.4.4.2.[2]

Test engineers can look at code coverage test results to help them devise test cases and input or configuration sets that will increase the code coverage over vital functions. Two common forms of code coverage used by testers are statement (or line) coverage and branch (or edge) coverage. Line coverage reports on the execution footprint of testing in terms of which lines of code were executed to complete the test. Edge coverage reports which branches or code decision points were executed to complete the test. They both report a coverage metric, measured as a percentage. The meaning of this depends on what form(s) of code coverage have been used, as 67% branch coverage is more comprehensive than 67% statement coverage.

Generally, code coverage tools and libraries exact a performance and/or memory or other resource cost which is unacceptable to normal operations of the software. Thus, they are only used in the lab. As one might expect, there are classes of software that cannot be feasibly subjected to these coverage tests, though a degree of coverage mapping can be approximated through analysis rather than direct testing.

There are also some sorts of defects which are affected by such tools. In particular, some race conditions or similar real time sensitive operations can be masked when run under code coverage environments; and conversely, some of these defects may become easier to find as a result of the additional overhead of the testing code.

Software tools

Tools for C / C++

- McCabe IQ [8]
- BullseyeCoverage [9]
- Gcov
- Insure++
- Cantata++
- NuMega TrueCoverage
- LDRA Testbed
- Tcov
- Tessy
- Testwell CTC++
- Trucov
- Parasoft C++test [10]
- froglogic's Squish Coco [11]

Tools for C# .NET

- McCabe IQ [8]
- DevPartner
- JetBrains dotCover [12]
- Kalistick
- NCover
- TestDriven.NET [13]
- Visual Studio 2010 [14]
- Parasoft dotTEST [15]

Tools for Java

- McCabe IQ [8]
- Cobertura [16]
- Clover
- DevPartner
- EMMA
- Jtest
- Kalistick
- LDRA Testbed
- JaCoCo [17]
- CodeCover [18]
- JMockit Coverage [19]

Tools for JavaScript

- McCabe IQ [8]
- coveraje [20]
- JSCoverage [21]
- script-cover [22]
- Code Coverage in JSTestDriver [23]

Tools for Haskell

- Haskell Program Coverage (Hpc) toolkit [24]

Tools for Perl

- McCabe IQ [8]
- Devel::Cover [25] is a complete suite for generating code coverage reports in HTML and other formats.

Tools for PHP

- McCabe IQ [8]
- PHPUnit, also need Xdebug to make coverage reports

Tools for Python

- McCabe IQ [8]
- Coverage.py [26]
- Figleaf [27]
- Pester [28]

Tools for Ruby

- McCabe IQ [8]
- CoverMe [29]
- RCov [30]
- SimpleCov [31]

Tools for Ada Ada (programming language)

- McCabe IQ [8]
- GNATcoverage [32]

Hardware tools

- Aldec
- Atrenta
- Cadence Design Systems
- iSYSTEM
- JEDA Technologies
- Mentor Graphics
- Nusym Technology
- Simucad Design Automation
- Synopsys

References

[1] Joan C. Miller, Clifford J. Maloney (February 1963). "Systematic mistake analysis of digital computer programs". *Communications of the ACM* (New York, NY, USA: ACM) **6** (2): 58–63. doi:10.1145/366246.366248. ISSN 0001-0782.
[2] RTCA/DO-178B, *Software Considerations in Airborne Systems and Equipment Certification, Radio Technical Commission for Aeronautics,* December 1, 1992.
[3] Glenford J. Myers (2004). *The Art of Software Testing, 2nd edition*. Wiley. ISBN 0-471-46912-2.
[4] Position Paper CAST-10 (June 2002). *What is a "Decision" in Application of Modified Condition/Decision Coverage (MC/DC) and Decision Coverage (DC)?* (http://www.faa.gov/aircraft/air_cert/design_approvals/air_software/cast/cast_papers/media/cast-10.pdf)
[5] http://www.rhyous.com/2012/05/08/unit-testing-with-parameter-value-coverage-pvc/
[6] M. R. Woodward, M. A. Hennell, "On the relationship between two control-flow coverage criteria: all JJ-paths and MCDC", Information and Software Technology 48 (2006) pp. 433-440
[7] Dorf, Richard C.: *Computers, Software Engineering, and Digital Devices*, Chapter 12, pg. 15. CRC Press, 2006. ISBN 0-8493-7340-9, ISBN 978-0-8493-7340-4; via Google Book Search (http://books.google.com/books?id=jykvlTCoksMC&pg=PT386&lpg=PT386&dq="infeasible+path"+"halting+problem"&source=web&ots=WUWz3qMPRv&sig=dSAjrLHBSZJcKWZfGa_IxYlfSNA&hl=en&sa=X&oi=book_result&resnum=1&ct=result)
[8] http://www.mccabe.com/
[9] http://www.bullseye.com
[10] http://www.parasoft.com/jsp/products/cpptest.jsp
[11] http://www.froglogic.com/squish/coco
[12] http://www.jetbrains.com/dotcover/
[13] http://testdriven.net/default.aspx
[14] http://msdn.microsoft.com/en-us/library/ms182496.aspx
[15] http://www.parasoft.com/jsp/products/dottest.jsp
[16] http://cobertura.sourceforge.net/
[17] http://www.eclemma.org/jacoco/index.html
[18] http://codecover.org/
[19] http://jmockit.googlecode.com/svn/trunk/www/tutorial/CodeCoverage.html
[20] https://github.com/coveraje/coveraje
[21] http://siliconforks.com/jscoverage/
[22] http://code.google.com/p/script-cover/
[23] http://code.google.com/p/js-test-driver/wiki/CodeCoverage
[24] http://www.haskell.org/haskellwiki/Haskell_program_coverage
[25] http://search.cpan.org/perldoc?Devel::Cover
[26] http://nedbatchelder.com/code/coverage/
[27] http://darcs.idyll.org/~t/projects/figleaf/doc/
[28] http://jester.sourceforge.net/
[29] https://github.com/markbates/cover_me
[30] https://github.com/relevance/rcov
[31] https://github.com/colszowka/simplecov

[32] http://www.adacore.com/gnatcoverage/

External links

- Branch Coverage for Arbitrary Languages Made Easy (http://www.semdesigns.com/Company/Publications/TestCoverage.pdf)
- Code Coverage Analysis (http://www.bullseye.com/coverage.html) by Steve Cornett
- Code Coverage Introduction (http://www.javaranch.com/newsletter/200401/IntroToCodeCoverage.html)
- Development Tools (Java)/ Code coverage (http://www.dmoz.org//Computers/Programming/Languages/Java/Development_Tools/Performance_and_Testing/Code_Coverage) at the Open Directory Project
- Development Tools (General)/ Code coverage (http://www.dmoz.org//Computers/Programming/Software_Testing/Products_and_Tools) at the Open Directory Project
- FAA CAST Position Papers (http://www.faa.gov/aircraft/air_cert/design_approvals/air_software/cast/cast_papers/)
- Gcov (http://gcc.gnu.org/onlinedocs/gcc/Gcov.html) with graphical summaries LCOV (http://ltp.sourceforge.net/coverage/lcov.php) and text/XML summaries gcovr (https://software.sandia.gov/trac/fast/wiki/gcovr)

Article Sources and Contributors

Code coverage *Source*: http://en.wikipedia.org/w/index.php?oldid=492402018 *Contributors*: 194.237.150.xxx, Abdull, Abednigo, Ad88110, Agasta, Aislingdonnelly, Aitias, Aivosto, AliveFreeHappy, Alksub, Allen Moore, Altenmann, Andreas Kaufmann, Andresmlinar, Anorthup, Attilios, Auteurs, Beetstra, BenFrantzDale, Billinghurst, Bingbangbong, BlackMamba, Blacklily, Blaxthos, Centic, Chester Markel, Cindamuse, Conversion script, Coveragemeter, DagErlingSmørgrav, Damian Yerrick, Derek farn, Didgeedoo, Digantorama, Dr ecksk, Ebelular, Erkan Yilmaz, Faulknerck2, FredCassidy, Gaudol, Ghettoblaster, Gibber blot, Greensburger, HaeB, Henri662, Hertzsprung, Hob Gadling, Hooperbloob, Hqb, Hunghuuhoang, Infofred, JASpencer, JJMax, Jamelan, JavaTenor, Jdpipe, Jerryobject, Jkeen, Johannes Simon, JorisvS, Jtheires, Julias.shaw, JustAnotherJoe, Kdakin, Kku, Kurykh, LDRA, LouScheffer, M4gnum0n, MER-C, Materialscientist, Mati22081979, Matt Crypto, MehrdadAfshari, Millerlyte87, Miracleworker5263, Mittgaurav, Mj1000, MrOllie, MywikiaccountSA, Nat hillary, NawlinWiki, Nigelj, Nin1975, Nintendude64, Nixeagle, Ntalamai, Parasoft-pl, Penumbra2000, Phatom87, Picapica, Pinecar, Ptrb, Quamrana, Quinntaylor, Quux, RedWolf, Roadbiker53, Robert Merkel, Rpapo, RuggeroB, Rwwww, Scubamunki, Sdesalas, Sebastian.Dietrich, Sferik, SimonKagstrom, Smharr4, Snow78124, Snoyes, Stoilkov, Suruena, Taibah U, Technoparkcorp, Test-tools, Testcocoon, Tiagofassoni, TutterMouse, U2perkunas, Veralift, Walter Görlitz, Walterkelly-dms, WimdeValk, Witten rules, Wlievens, Wmwmurray, X746e, 249 anonymous edits

License