Test Coverage Metrics

Craig A. Damon

March 10, 1997

Papers:


Basic Question

How Reliable Is A “Tested” Program?

Will A Test Suite Find All or Even Most Bugs?

Want a Metric (Criteria) To Say How Thoroughly A Test Suite Tests A Program
Background

A Complete Test Suite Containing Every Possible Input Would Be Guaranteed To Find Every Bug

This Is Clearly Almost Always Too Expensive, And Frequently Infinite

An Ideal Test Suite For A Given Program Would Contain One Test For Each Actual Bug, But ...

Bugs Aren’t Known

So We Want A Test For Each Possible Bug
Basic Definitions

Goal is a way to check whether every possible bug of the program is covered by some test in the finite test suite.

Called “Reliable” test suite if guaranteed.

Called “Almost Reliable” if some possible bugs are only very likely to be covered.

Criteria is a method for checking whether a test suite is (approximately) reliable.

A criteria which can be satisfied for every program is called “Applicable.”
Statement / Block Coverage Criteria

Is Every Statement Executed By Some Test?

Block Coverage Is Effectively The Same, But More Efficient

Very Easy To Implement And Understand

Not Reliable:

   Obvious Missing Paths (Loops, Else-less Ifs)

   No Consideration Of Context

Not Applicable:

   “Dead” Code

   Error Handling Code
Branch Coverage Criteria

Is Every Possible Decision Made At Every Branch?

Relatively Easy To Implement And Understand

Adds Obvious Missing Paths From Statement Coverage

Not Reliable:

   No Consideration Of Context

Not Applicable:

   “Dead” or Error Code
Path Coverage Criteria

Is Every Possible Path Executed By Some Test?

Infinite Number Of Paths If Any Loops

Various Reduction Techniques

Most Common Is Path Fragment Coverage, Where Every Possible Path Is A Combination of Some Sequence of Path Fragments

Not Reliable:
  Any Reduction May Remove Key Combination

Not Applicable:
  Infeasible Paths
Mutation Coverage

Is Every Possible Variation Of The Program Tested?

Possibly Huge Numbers Of Mutations

Can Show Other Interesting (Mis-)Features Of Code

Not Reliable:

- Non-Local Combinations
- Missing Mutation Combinations
- Way Off Errors
- Missing Cases

Not Applicable:

- “Dead” Code
Data Flow Coverage Criteria

Is Every Possible Usage Of Every Possible Definition Tested?

Feasible Only For Local Data Flow

Can Use Approximations At Expense Of Some Reliability

Really A Kind Of Path Coverage Reduction Technique

Not Reliable:
  
  Non-Local Def/Use Pairs

  Combinations of Usages

Not Applicable:

  Infeasible Paths
Currently Available Tools

Some Commercial Offer Only Statement/Block Coverage

Better Tools Give Branch Coverage

A Few Give Some Form Of Path Coverage

One Commercial Tools (LDRA) Claims To Provide Some Unspecified DF Coverage

GCT Does Some Mutation Testing As Well As Statement And Branch

atac Does Basic Data Flow Coverage As Well As Statement And Branch
Big Issues

Which Criteria To Choose

Which Covers More Is Current Hot Topic

How Much Of Test Suite Can Be Generated?

Not Everything

Heuristics vs Well Founded

Higher Density Of Bugs In Heuristics

Thoroughness vs. Cost

Misleading Numbers in Coverage

Other Types Of Programs

Event Oriented

Object Oriented

Data Structure Oriented

Black Box vs White Box Testing

Missing Functionality

Other Types Of Problems
Performance

Usability