Automated Test Coverage: Take Your Test Suites To the Next Level

Presented By

Jay Berkenbilt

Apex CoVantage

Presentation Objectives

- Test coverage concepts
- Advantages of automated test coverage
  – Code tests the test suite
  – Powerful code review tool
  – Protection against test suite decay
- Details of one coverage system
- Tips and real-world examples

Testing Philosophy

- Designing for testability
- Self-testing software
- Automated test suites
  - Coded incrementally by developer
  - Passage strongly indicates working software
- Complemented by
  - External, Alpha, Beta testing
  - Banging on the keyboard

Test Suite Functionality

- Test case passes when:
  - Program produces expected output and exit status
- Test suite passes when:
  - All test cases pass
  - Correct number of test cases seen
  - Test driver terminates normally

Coverage Terminology

- *Coverage Case*
  - Event in the code to be exercised during testing
  - Belongs to a coverage scope
- *Coverage Scope*
  - Way of assigning coverage cases to a specific test suite
  - Allows multiple test suites to exercise different coverage conditions in one body of code

Test Coverage System Overview

- Ensures code is exercised by test suite as developer intended
  - *Code tests the test suite*
- Three lightweight components
  - Coverage calls: inside code
  - Coverage case registry: one per scope
  - Coverage analyzer: part of test system
- Integrates with test framework

Differences From Other Systems

- Programming technique
  - Coverage calls explicitly coded by developer
- Complementary to black-box code coverage checkers
- As detailed as programmer wants
- Coverage code permanently lives with test suite

A Quick Example

- A normal code fragment:

```
if (condition){
    do_something(val1);
} else {
    do_something_else();
}
```

- Depends upon `condition` and `val1`

Example With Coverage

- Now with coverage:

```
if (condition){
    TCOV::TC("example", "something",
             (val1 > 5 ? 0 : 1));
    do_something(val1);
} else {
    TCOV::TC("example", "something else");
    do_something_else();
}
```

Coverage Calls

- Function call
- Three arguments
  - Scope name (string literal)
  - Coverage case name (string literal)
  - Numeric argument (may be expression)
- Tells system that the coverage case happened

Coverage Call Numeric Argument

- Distinguishes multiple conditions of a call
- Example:

```
TCOV::TC("example", "something",
         (val1 > 5 ? 0 : 1));
```

- Scope is "**`example`**"
- Coverage case name is "**`something`**"
- Numeric argument depends upon value of ***`val1`***

Coverage Case Registry

- Flat file: ***`scope.testcov`***
- Lists each coverage case in *scope*
  - **`coverage case name`** ***`n`***
  - ***`n`*** is maximum numeric argument value
- Lists other allowable scopes
  - These scopes are ignored when seen in code
  - Portects aganist tpyogarphicial erorrs
- Checked against code for consistency

Example Registry

<u>**example.testcov**</u>

```
ignored-scope: my_other_prog
something 1
something else 0
exception caught 0
```

Coverage Analyzer

- Short program called by test framework
- Called at beginning of test suite to compare registry with code
- Called at the end of test suite to compare registry with observed calls

Coverage Analysis At Beginning

- Ensure each of the following:
  - Each registry entry is unique
  - Each coverage call is in valid scope
    - Current scope
    - Explicitly ignored scope
  - Every coverage call is in registry
  - Every registered case appears exactly once
- If errors, abort test suite
- Otherwise, coverage system is active

Coverage Analysis At End

- Ensure each of the following:
  - Every registered case was called at least once with each numeric value
  - No unexpected coverage cases appeared
- If errors:
  - Report sorted list of missing cases
  - Report sorted list of extra cases
  - Fail overall test suite
- Otherwise, coverage analysis passes

Coverage Call Implementation

- Check scope argument against environment; return if mismatch
- Otherwise, append coverage case name and number to output file
- Current scope and output file name come from environment variables
  - Variables set by test framework when coverage is active
- Implemented in just a few lines of code

Additional Test Suite Functionality

- Coverage analysis succeeds when:
  - Registry matches code
  - Each coverage case seen at least once
  - No extra coverage cases seen
- With coverage, test suite passes when:
  - All test cases pass
  - Correct number of test cases seen
  - Test driver terminates normally
  - *Coverage analysis succeeds*

Example Code: `search`

- Searches through a sorted array
- Uses either linear scan or binary search
- Boundary conditions:
  - Which search method?
  - Found or not found?
  - First item, last item, middle, too low, or too high?
- Paper goes into more detail; code available for download

Example: Unprintable Characters

- Conversion software converts unprintable characters in input to “?”.
- Test file with unprintable characters edited for another purpose, unprintable characters (accidentally) removed
- Coverage system alerted developer that unprintable character code was no longer exercised in test suite

Example: Issues With No Articles

- Program generates XML for articles in a journal issue
- Excludes some articles based on specific rules
- Special case for issues where all articles are excluded
- One test issue exercises this condition

Issues With No Articles (cont'd)

- Change to exclusion rules resulted in one article in special test issue no longer being excluded
- Coverage case "all articles excluded" failed
- Offending article removed from test issue
- Coverage system ensured all coverage cases still exercised after removal of article

Example: Worker Thread

- Main thread draws image, worker thread does computation on image
- User does operation that needs results: main thread waits until results are ready
- Exercised in test suite by creating complex image for which computations take a long time

Worker Thread (cont’d)

- New fast machine: worker thread finished too fast
- All test cases passed, but coverage system reported “wait for worker thread” coverage case missing
- Artificial delay, triggered by environment variable, added to exercise waiting condition in one instance

Code Reviews

- Should include review of test suite
- Reviewer sees tricky path in code, wonders whether test suite exercises it
- Find out: add a coverage case
- Benefit:
  - Guarantees developer must fix test suite
  - Provides permanent assurance that test suite will always cover this case

Test Coverage Tips and Tricks (1)

- Exploit lexical sorting of analyzer output
  - Put source file name at beginning of coverage case name
  - Use conventions like putting “ERR” after file name for error conditions
  - Lexical sorting causes missing case report to be grouped together by functionality

Test Coverage Tips and Tricks (2)

- Write coverage code while writing main code
  - Developer is most aware of boundary conditions
  - Prevents forgetting to code a test case—better than keeping a check list
- Use numerical argument for “can’t happen” cases
  - Numerical argument greater than *n* generates extra coverage case report

Performance and Security

- Performance impact of inactive coverage calls is usually negligible, but there’s still a function call
- User can force application to append coverage cases to a file
- If coverage calls must be excluded from released version, use conditional compilation
- Make sure coverage calls have no side effects
- Run test suite with coverage analysis off just to be sure

Obtaining Example Code

- Downloadable example:
  - Implementation of simple test framework with coverage support
  - Implementation of `search` program with test suite
  - Code works natively in UNIX or Windows
  - Test code works in UNIX or Windows with Cygwin 32
- Download paper, presentation, and code at

http://www.ql.org/pstt/testcov