ParaSoft®
tca®

Maximizing Test-Suite Coverage

The TCA Solution

ParaSoft®

Introduction

The idea behind test coverage analysis is to determine how many of an application's files, functions and statements have been executed. This data can be used during development, and particularly during testing, to give some idea of the overall quality of the testing.

The quality of the checking performed by Insure++ is independent of the amount of effort put in by the development or quality assurance team-- it performs complete and thorough checking on any piece of code that it executes (and also, of course, a significant amount of compile-time checking on code that it merely compiles). As a result, it makes a lot of sense to aim for 100% test coverage if you are using a product such as Insure++, because the testing is much more complete.

For this reason, Insure++ contains the Total Coverage Analysis (TCA) module, which is a component of the Total Quality Software package designed to assure you of 100% execution of your application during the testing and quality assurance phases. TCA provides the following information:

- **Overall Summary**: Shows the percentage covered at the application level - i.e., summed over all program entities.
- **Function Summary**: Displays the coverage of each individual function in the application.
- **Block Summary**: Displays the coverage broken down by individual program statement blocks.

Combining RunTime Error Detection with Coverage Analysis

The Total Coverage Analysis module of Insure++ lets you get “beneath the hood” of your program to see which parts of your program are actually tested and how often each block is executed. In conjunction with a runtime error detection tool and a comprehensive test suite, this can dramatically improve the efficiency of your testing and guarantee faster delivery of more reliable programs.

If you are using Insure++, coverage analysis information will be automatically built into your program. At any time after you have run your code, you can use the `tca` command to find any blocks which have not been executed. Unlike some other coverage analysis tools which work on a line-by-line basis, TCA is able to group your code into logical blocks. A block is a group of statements which must always be executed as a group.

For example, the following code has three statements, but only one block:

```
    i = 3

j = i+3;

return i+j;
```

Advantages of using blocks over lines include:

- Lines of code which have several blocks are treated separately.
- Grouping equivalent statements into a single block reduces the amount of data you need to analyze.
- By treating labels as a separate group, you can actually detect which paths have been executed in addition to which statements.

Figure 1 illustrates the above concepts with a simple test program.

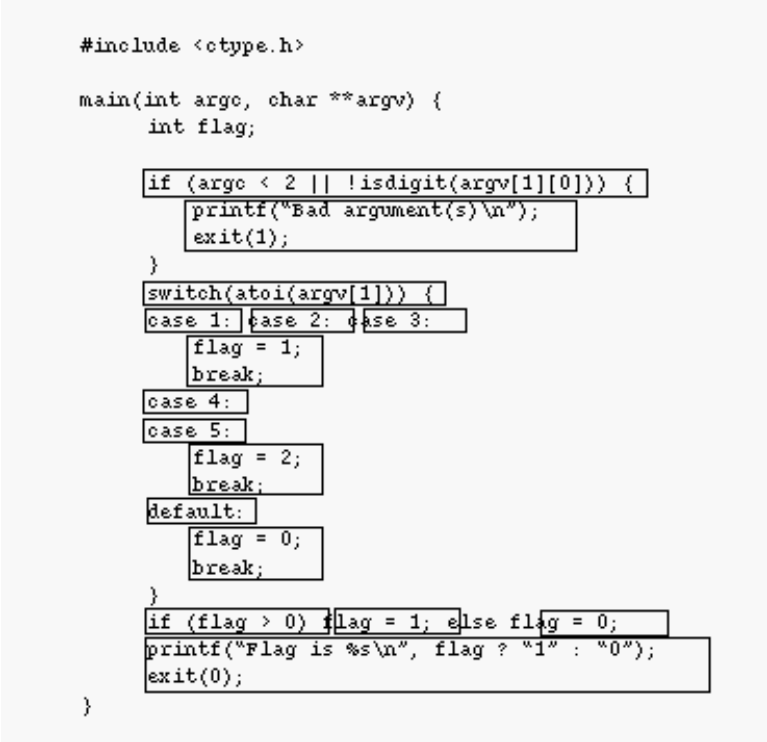

Figure 1. Sample program divided into blocks

Figure 2 shows the TCA output after several test runs with different values. By analyzing this output, you can see which paths have been executed and which have not. Notice that counts are only given at the beginning of each block, and not for each statement within each block.

```
BLOCK COUNTS - by file
============
FILE test.c 87% covered: 2 untested / 14 tested
                #include <ctype.h>
                main(int argc, char **argv) {
                        int flag;
        6 ->            if (argc < 2 || !isdigit(argv[1][0])) {
        1 ->                    printf("Bad argument(s)\n");
                                exit(1);
                        }
        5 ->            switch(atoi(argv[1])) {
        0,2,1 ->                case 1: case 2: case 3:
        3 ->                    flag = 1;
                                break;
        1 ->            case 4:
        0 ->            case 5:
        1 ->                    flag = 2;
                                break;
        1 ->            default:
        1 ->                    flag = 0;
                                break;
                        }
        5,4,1 ->                if (flag > 0) flag = 1; else flag = 0;
        5 ->            printf("Flag is %s\n", flag ? "1" : "0");
                        exit(0);
                }
```

Figure 2. Viewing Blocks Covered During Test Execution

Running Insure++ with TCA is an incredibly powerful combination. Insure++ can only check code if the code was executed. TCA can tell the programmer which statements in the program were executed. By using a runtime error detection tool together with a coverage analysis tool, the programmer can make sure that each block of the code was checked during the testing process and significantly improve the quality of the tested code.

Using the TCA GUI, a developer can analyze coverage by directory, by file, by function,

and by class (See Figure 3). Annotated source code can also be viewed from the TCA GUI.

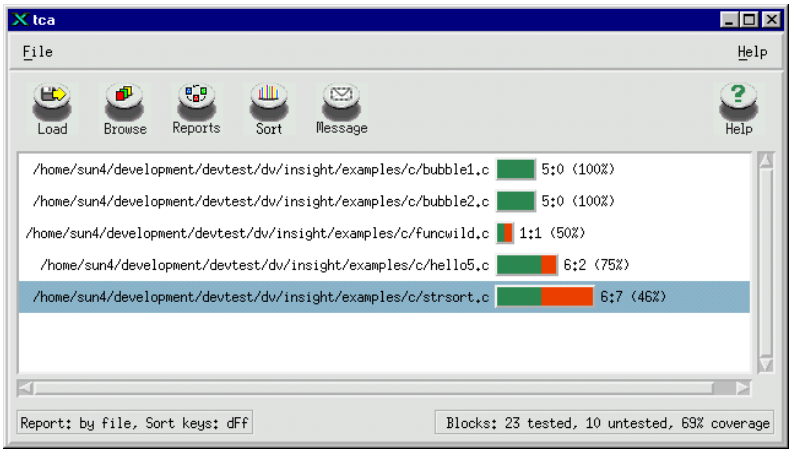

Figure 3. TCA shows you how much code has been tested by Insure++

Conclusion

The quality of the checking performed by Insure++ is independent of the amount of effort put in by the development or quality assurance team-- it performs complete and thorough checking on any piece of code that it executes (and also, of course, a significant amount of compile-time checking on code that it merely compiles). As a result, it makes a lot of sense to aim for 100% test coverage if you are using a product such as Insure++, because the testing is much more complete.

TCA is available now (as an Insure++ add-on) at www.parasoft.com. To learn more about how TCA and other ParaSoft development tools can help your department prevent and detect errors, talk to a Software Quality Specialist today at 1-888-305-0041, or visit www.parasoft.com.