Case Consult

The Evolution En@bling Experts

Our Products

ANALYZER

Test Coverage Monitor for C/C++, COBOL, Java and PL/I

ANALYZER

Test Coverage Monitor for C/C++, COBOL, Java and PL/I

Description

ANALYZER measures the test coverage of C/C++, COBOL, Java and PL/I programs and provides dynamic quality assurance in software development and program maintenance.

Dynamic and systematic testing increases the quality of applications. **ANALYZER for C/C++, COBOL, Java and PL/I** is instrumental in helping to avoid many of the problems which normally occur during production launches of new or altered applications.

Dialog language
English

Areas of application
Software development
Software maintenance
Quality assurance
Software security
e-Commerce
Tuning
Test environment

Languages supported
C
COBOL
Java
PL/I

Platforms

C/C++, COBOL and PL/I versions available for
MS-DOS
OS/2
Windows 3.x, 9x, 2000, NT, XP
UNIX
AIX
IMB OS/390

Java version available for
most platforms supported by Java

You think there's no such thing as 100 % certainty? Think again. With **CC's ANALYZER** you have everything you need to thoroughly test your applications before making them operational. There's no "let's see what happens" when **ANALYZER** is part of the application testing process. You can finally make this transition with complete security and minimal manual input.

ANALYZER is a test coverage monitoring tool - secure, efficient, exact and comprehensive. **ANALYZER** works automatically and is the perfect dynamic quality analysis tool.

ANALYZER for C/C++, COBOL, Java and PL/I

- ❏ provides exact and logically structured methods for testing applications
- ❏ organizes and optimizes all information needed thoughout test cycles
- ❏ provides dynamic quality assurance in software development and program maintenance

This systematic testing automatically defines and sets high quality standards which directly lead to a drastic reduction in the problems that often come up during operational introduction of new or altered applications.

The need for security and efficiency in test monitoring continues to grow exponentially to new business possibilities. New markets, e-business, e-commerce and the resulting proximity to customers demand dynamic changes in the world of IT development and production. This makes costs, time management and resources key factors in economic success.

And this redefines what it is that companies need to focus on:

- ❏ improved quality and shortening of test cycles
- ❏ objective assessment of test processes
- ❏ transparent production of test results
- ❏ documentation of reliable information on test runs

ANALYZER for C/C++, COBOL, Java and PL/I offers all this and more.

Characteristics

ANALYZER is the quality assurance tool to measure test coverage of applications of software development and program maintenance.

Selected characteristics are:

Measure the percentage of code executed

- ❏ Measure the comprehensiveness of code executed in development, test and production environments
- ❏ Show comprehensiveness of program runs by the number of code intervals in program

Identify code not executed

- ❏ Identify specific code not executed in a test run
- ❏ Count number of times each command was executed

Increase the quality and optimization of test data elements

- ❏ Provides all information required for compiling quality test data and for optimizing existing data
- ❏ Substantially increases quality of test data during development and modification/enhancement processes

Specify test range

- ❏ Analyzes entire systems or selected individual programs and intervals
- ❏ Targeted testing of test coverage range during maintenance task

Tailor and automate code insertion

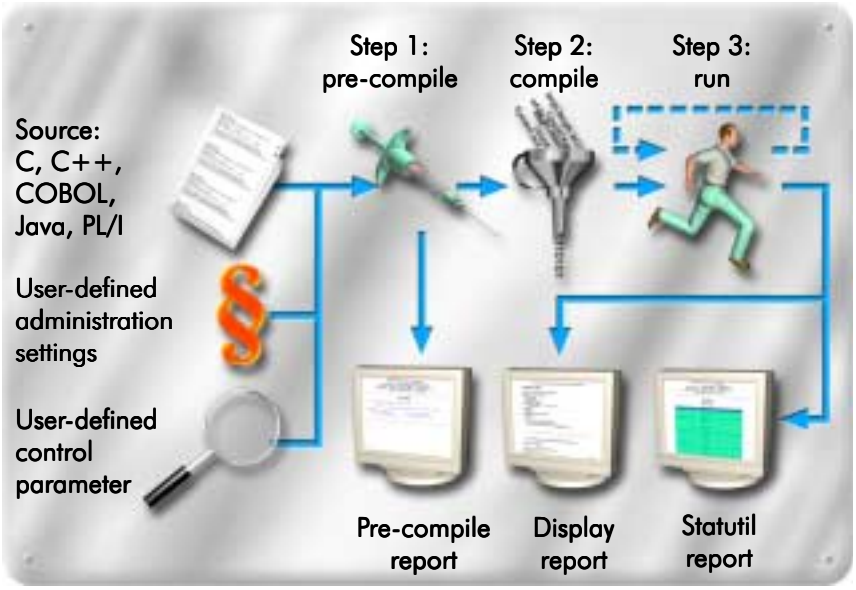
- ❏ Perform specific automated code modifications, e. g. special forms of debugging or tracing
- ❏ Automated insertion of comments or customized program measurement

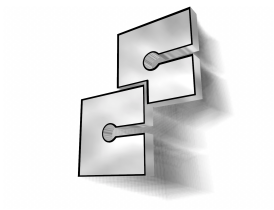
Produce transparent results in HTML and text format

- ❏ Report showing the total number of intervals executed in the test and the percentage of the total of all intervals that this executed set represents

What else does ANALYZER do?

- ❏ ANALYZER for Java analyzes all Java classes (Applets, Servlets, Beans, etc.)
- ❏ Provides reliable information on the quality of test data and test processes
- ❏ Helps to optimize test data
- ❏ Trace function in targeted test runs for easy localization of errors
- ❏ Eases fast and smooth introduction to foreign programs through quality test run information with trace functionality
- ❏ Uncovers overlaps in test runs
- ❏ Shows test coverage of one or more test runs
- ❏ Eases recognition of program processes with customized test data
- ❏ Assists in reduction of unproductive test runs by producing quality analysis results
- ❏ Offers reliable test documentation with possibility to synchronize to pre-defined test end criteria





ANALYZER for C/C++ – HTML-Report

Licenced by: CASE CONSULT DEVELOPMENT

(C) CC GmbH 2002 Version: 1.0
Date: 2002-06-30 16:16:15

Display Summary

Source Input File: **count.c**
Statistics File: **statfile.dat**

Runs	Run Date	Intervals	Executed	Percent
Test 1	2002-06-30-16.07.47.00 / 2002-06-30-16.07.47.00	7	2	28.57 %
Test 2	2002-06-30-16.10.11.00 / 2002-06-30-16.10.11.00	7	6	85.71 %
TOTAL	2002-06-30-16.07.47.00 / 2002-06-30-16.10.11.00	7	7	100.00 %

ANALYZER for C/C++ - Report for Test Number: 1

Top

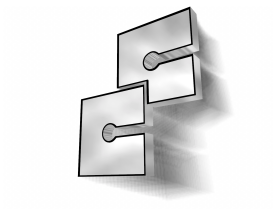
Source Input File: **count.c**
Statistics File: **statfile.dat**

```
        void CountChars (FILE *ptr) {
********    char ch;
*       *   int count = 0;
               while (!feof(ptr)) {
********       ch = fgetc(ptr);
               if (!feof(ptr))
********          count++;
            }
********    printf("No. of chars counted = %d\n", count);
        }
        void main(int argc, char *argv[]) {
(000001)    FILE *fptr;
            char filename[100];
            if (argc > 1) {
********      strcpy(filename, argv[1]);
*       *     fptr = fopen(filename, "r");
*       *     CountChars(fptr);      fclose(fptr);
            }
            else
(000001)       printf("Enter the file name as arg to count chars\n");
        }
```

```
Out of    7 Intervals,     2 ( 28.57%) were executed
     1 Test Run(s) represented from 2002-06-30 16.07.47.00
                                 to 2002-06-30 16.07.47.00
```

ANALYZER for C/C++
Test Coverage Monitor

HTML Sample Report



Top

ANALYZER for C/C++ - Report for Test Number: 2

Source Input File: count.c
Statistics File: statfile.dat

```
         void CountChars (FILE *ptr) {(000001)    char ch;
            int count = 0;
               while (!feof(ptr)) {
(000722)       ch = fgetc(ptr);
               if (!feof(ptr))
(000721)          count++;
            }
(000001)    printf("No. of chars counted = %d\n", count);
         }
         void main(int argc, char *argv[]) {
(000001)    FILE *fptr;
            char filename[100];
            if (argc > 1) {
(000001)      strcpy(filename, argv[1]);
              fptr = fopen(filename, "r");
              CountChars(fptr);      fclose(fptr);
            }
            else
********       printf("Enter the file name as arg to count chars\n");
         }
```

Out of 7 Intervals, 6 (85.71%) were executed
1 Test Run(s) represented from 2002-06-30 16.10.11.00
to 2002-06-30 16.10.11.00

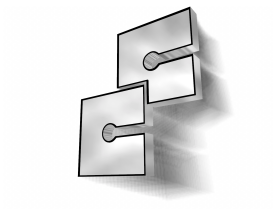
Top

Accumulated Results of Test Runs

Source Input File: count.c
Statistics File: statfile.dat

```
         void CountChars (FILE *ptr) {
(000001)    char ch;
            int count = 0;
              while (!feof(ptr)) {
(000722)       ch = fgetc(ptr);
               if (!feof(ptr))
(000721)          count++;
            }
(000001)    printf("No. of chars counted = %d\n", count);
         }
         void main(int argc, char *argv[]) {
(000002)    FILE *fptr;
            char filename[100];
            if (argc > 1) {
(000001)       strcpy(filename, argv[1]);
               fptr = fopen(filename, "r");
               CountChars(fptr);      fclose(fptr);
            }
            else
(000001)       printf("Enter the file name as arg to count chars\n");
         }
```

Out of 7 Intervals, 7 (100.00%) were executed
2 Test Run(s) represented from 2002-06-30 16.07.47.00
to 2002-06-30 16.10.11.00



ANALYZER for COBOL – STATISTICS REPORT

LICENSED BY: CASE CONSULT DEVELOPMENT

(C) CC GmbH 2002 VERSION: 5.2
DATE: 2002-03-28 14:15:32

PROGRAM: DEMO2

RUN DATE	INTERVALS	EXECUTED	PERCENT	INSTRUMENTATION DATE
2002-03-28(13:48:17)	5	4	80%	2002-03-28(11:52:50)
2002-03-28(13:53:36)	5	2	40%	2002-03-28(11:52:50)

---------- REPORT TOTALS ----------

PROGRAM	RUNS	RUN DATE	INTERVALS	EXECUTED	PERCENT
DEMO2	2	2002-03-28	5	5	100%
TOTAL:	2		5	5	100%

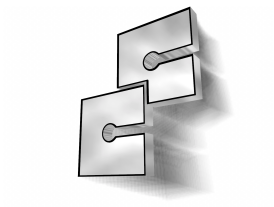
101 - REPORT COMPLETED

ANALYZER for COBOL - DISPLAY REPORT FOR TEST NUMBER: 1

SOURCE INPUT FILE: DEMO2
STATISTICS FILE: STATFILE

```
                 PROCEDURE DIVISION USING PARM1, PARM2.
                PARA-1.
(000001)            MOVE 'DEMO2' TO PARM1
                    MOVE 'Y' TO MAIN-ENTRY-SW
                    MOVE +1234 TO PARM2.
                    IF PARM2 = 9999
(000001)                DISPLAY 'PARM2 HAS AN IMPOSSIBLE VALUE'
                        GO TO PARA-EXIT
                    END-IF
********            EXIT PROGRAM.
                PARA-EXIT.
(000001)            DISPLAY 'ABORTING PROGRAM'.
                PARA-EXIT-X.
(000001)            EXIT PROGRAM.
```

OUT OF 5 INTERVALS, 4 (80%) WERE EXECUTED.
89 RECORDS READ FROM SOURCE FILE
1 TEST RUN(S) REPRESENTED FROM 2002-03-28(13:48:17.35)
TO 2002-03-28(13:48:18.89)

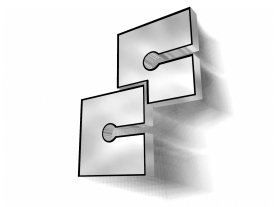


ANALYZER for COBOL - DISPLAY REPORT FOR TEST NUMBER: 2

```
SOURCE INPUT FILE:   DEMO2
STATISTICS FILE:     STATFILE


                PROCEDURE DIVISION USING PARM1, PARM2.
                PARA-1.
(000010)            MOVE 'DEMO2' TO PARM1
                    MOVE 'Y' TO MAIN-ENTRY-SW
                    MOVE +1234 TO PARM2.
                    IF PARM2 = 9999
********                DISPLAY 'PARM2 HAS AN IMPOSSIBLE VALUE'
*      *                GO TO PARA-EXIT
                    END-IF
(000010)            EXIT PROGRAM.
                PARA-EXIT.
********            DISPLAY 'ABORTING PROGRAM'.
*      *        PARA-EXIT-X.
********            EXIT PROGRAM.

OUT OF    5  INTERVALS,    2 ( 40%) WERE EXECUTED.
         89  RECORDS READ FROM SOURCE FILE
          1 TEST RUN(S) REPRESENTED FROM 2002-03-28(13:53:36.05)
                                     TO 2002-03-28(13:53:37.49)
```

ANALYZER for COBOL - ACCUMULATED RESULTS OF TEST RUNS

```
SOURCE INPUT FILE:   DEMO2
STATISTICS FILE:     STATFILE


                PROCEDURE DIVISION USING PARM1, PARM2.
                PARA-1.
(000011)            MOVE 'DEMO2' TO PARM1
                    MOVE 'Y' TO MAIN-ENTRY-SW
                    MOVE +1234 TO PARM2.
                    IF PARM2 = 9999
(000001)                DISPLAY 'PARM2 HAS AN IMPOSSIBLE VALUE'
                        GO TO PARA-EXIT
                    END-IF
(000010)            EXIT PROGRAM.
                PARA-EXIT.
(000001)            DISPLAY 'ABORTING PROGRAM'.
                PARA-EXIT-X.
(000001)            EXIT PROGRAM.

OUT OF    5  INTERVALS,    5 (100%) WERE EXECUTED.
         89  RECORDS READ FROM SOURCE FILE
          2 TEST RUN(S) REPRESENTED FROM 2002-03-28(13:48:17.35)
                                     TO 2002-03-28(13:53:37.49)
```



ANALYZER for Java – HTML-Report

LICENSED BY: CASE CONSULT DEVELOPMENT

(C)CC GmbH 2002 VERSION: 1.0
DATE: 2002-01-07 15:12:47

1. DISPLAY

DISPLAY SUMMARY

RUNS	RUN DATE	INTERVALS	EXECUTED	PERCENT
Test 1	2002-01-07-15.06.36.890000 / 2002-01-07-15.06.37.380000	6	2	33
Test 2	2002-01-07-15.07.12.750000 / 2002-01-07-15.07.13.140000	6	5	83
TOTAL	2002-01-07-15.06.36.890000 / 2002-01-07-15.07.13.140000	6	6	100

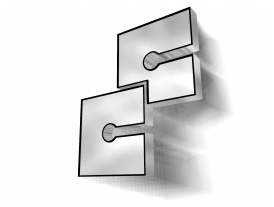
Top

ANALYZER for Java - REPORT FOR TEST NUMBER: 1

SOURCE INPUT FILE: Count
STATISTICS FILE: Statfile-Application

```
         import java.io.*;
         public class Count {
            public static void CountChars(Reader in) throws IOException {
********       int count = 0;
*      *       while (in.read() != -1)
********          count++;
*      *
********       System.out.println("No. of chars counted = " + count);
*      *

            }
            public static void main(String[] args) throws Exception {
(000001)       if (args.length >= 1)
********          CountChars(new FileReader(args[0]));
*      *       else
*      *
(000001)          System.err.println("Enter the file name as arg to count chars");
            }
         }
```

```
OUT OF    6 INTERVALS,     2 ( 33%) WERE EXECUTED
         66 RECORDS READ FROM SOURCE FILE
     1 TEST RUN(S) REPRESENTED FROM 2002-01-07-15.06.36.890000
                                 TO 2002-01-07-15.06.37.380000
```



Top

ANALYZER for Java - REPORT FOR TEST NUMBER: 2

SOURCE INPUT FILE: Count
STATISTICS FILE: Statfile-Application

```
         import java.io.*;
         public class Count {
            public static void CountChars(Reader in) throws IOException {
(000001)       int count = 0;
               while (in.read() != -1)
(000447)          count++;
(000001)       System.out.println("No. of chars counted = " + count);
            }
            public static void main(String[] args) throws Exception {
(000001)       if (args.length >= 1)
(000001)          CountChars(new FileReader(args[0]));
               else
********          System.err.println("Enter the file name as arg to count chars");
*       *
            }
         }
```

```
OUT OF    6 INTERVALS,     5 ( 83%) WERE EXECUTED
         66 RECORDS READ FROM SOURCE FILE
     1 TEST RUN(S) REPRESENTED FROM 2002-01-07-15.07.12.750000
                                TO 2002-01-07-15.07.13.140000
```

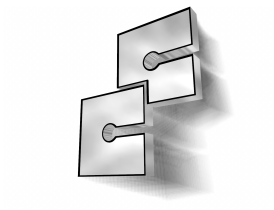
Top

ACCUMULATED RESULTS OF TEST RUNS

SOURCE INPUT FILE: Count
STATISTICS FILE: Statfile-Application

```
         import java.io.*;
         public class Count {
            public static void CountChars(Reader in) throws IOException {
(000001)       int count = 0;
               while (in.read() != -1)
(000447)          count++;
(000001)       System.out.println("No. of chars counted = " + count);
            }
            public static void main(String[] args) throws Exception {
(000002)       if (args.length >= 1)
(000001)          CountChars(new FileReader(args[0]));
               else
(000001)          System.err.println("Enter the file name as arg to count chars");
            }
         }
```

```
OUT OF    6 INTERVALS,     6 (100%) WERE EXECUTED
         66 RECORDS READ FROM SOURCE FILE
     2 TEST RUN(S) REPRESENTED FROM 2002-01-07-15.06.36.890000
                                TO 2002-01-07-15.07.13.140000
```



ANALYZER for PL/I – STATISTICS REPORT

LICENSED BY: CASE CONSULT DEVELOPMENT

(C) CC GmbH 2002 VERSION: 5.2
DATE: 2002-03-28 16:12:47

PROGRAM: DEMOP2

RUN DATE	INTERVALS	EXECUTED	PERCENT	INSTRUMENTATION DATE
2002-03-28(15:03:35)	4	3	75%	2002-03-28(14:26:09)
2002-03-28(15:07:23)	4	2	50%	2002-03-28(14:26:09)

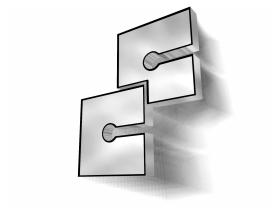
---------- REPORT TOTALS ----------

PROGRAM	RUNS	RUN DATE	INTERVALS	EXECUTED	PERCENT
DEMOP2	2	2002-03-28	4	4	100%
TOTAL:	2		4	4	100%

101 - REPORT COMPLETED

ANALYZER for PL/I - DISPLAY REPORT FOR TEST NUMBER: 1

```
SOURCE INPUT FILE:   DEMOP2
STATISTICS FILE:     STATFILE

          DEMOP2: PROCEDURE(PARM1,PARM2) REORDER;
(000001)
  NON-PROCEDURAL CODE SUPPRESSED
          PARM1 = 'DEMOP1';
          MAIN_ENTRY_SW = '1'B;
          DISPLAY('PARM2');
          IF PARM2 = 9999 THEN DO;
(000001)      DISPLAY('PARM2 HAS AN IMPOSSIBLE VALUE');
              GOTO PARA_EXIT;
              END;
********  RETURN;
          PARA_EXIT:
(000001)  DISPLAY('ABORTING PROGRAM');
          RETURN;
          END DEMOP2;

OUT OF    4 INTERVALS,     3 ( 75%) WERE EXECUTED.
         59 RECORDS READ FROM SOURCE FILE
          1 TEST RUN(S) REPRESENTED FROM 2002-03-28(15:03:35.17)
                                     TO 2002-03-28(15:03:35.25)
```



```
    ANALYZER for PL/I - DISPLAY REPORT FOR TEST NUMBER:   2

SOURCE INPUT FILE:   DEMOP2
STATISTICS FILE:     STATFILE

          DEMOP2: PROCEDURE(PARM1,PARM2) REORDER;
(000010)
  NON-PROCEDURAL CODE SUPPRESSED
          PARM1 = 'DEMOP1';
          MAIN_ENTRY_SW = '1'B;
          DISPLAY('PARM2');
          IF PARM2 = 9999 THEN DO;
********      DISPLAY('PARM2 HAS AN IMPOSSIBLE VALUE');
*       *     GOTO PARA_EXIT;
*       *     END;
(000010)  RETURN;
          PARA_EXIT:
********  DISPLAY('ABORTING PROGRAM');
*       *  RETURN;
*       *  END DEMOP2;

OUT OF    4 INTERVALS,     2 ( 50%) WERE EXECUTED.
         59 RECORDS READ FROM SOURCE FILE
          1 TEST RUN(S) REPRESENTED FROM 2002-03-28(15:07:23.62)
                                     TO 2002-03-28(15:07:23.78)
```

```
       ANALYZER for PL/I - ACCUMULATED RESULTS OF TEST RUNS

SOURCE INPUT FILE:   DEMOP2
STATISTICS FILE:     STATFILE

          DEMOP2: PROCEDURE(PARM1,PARM2) REORDER;
(000011)
  NON-PROCEDURAL CODE SUPPRESSED
          PARM1 = 'DEMOP1';
          MAIN_ENTRY_SW = '1'B;
          DISPLAY('PARM2');
          IF PARM2 = 9999 THEN DO;
(000001)      DISPLAY('PARM2 HAS AN IMPOSSIBLE VALUE');
              GOTO PARA_EXIT;
              END;
(000011)  RETURN;
          PARA_EXIT:
(000001)  DISPLAY('ABORTING PROGRAM');
          RETURN;
          END DEMOP2;

OUT OF    4 INTERVALS,     4 (100%) WERE EXECUTED.
         59 RECORDS READ FROM SOURCE FILE
          2 TEST RUN(S) REPRESENTED FROM 2002-03-28(15:03:35.17)
                                     TO 2002-03-28(15:07:23.78)
```

Case Consult

Consulting • Software • Training

CC in ...

USA
Case Consult Corporation
East Main Street 176, Suite 5
Westborough, MA O1581
Telefon +1-8OO-836-7772
info-usa@caseconsult.com

Germany
CC GmbH
Flachstraße 13
65197 Wiesbaden
Telefon +49-611-942O4O
info-europe@caseconsult.com

CC GmbH
Alter Fischmarkt 5
2O457Hamburg
Telefon +49-4O-32325OO
info-europe@caseconsult.com

India
Case Consult (India) Pvt. Ltd.
Technopark Campus
Trivandrum 695 581
Telefon +91-471-7OO176
info-india@caseconsult.com

South Africa
Case Consult (Pty) Ltd.
Cabernet House - East
Brandwacht Office Park
Stellenbosch
P O Box 12676
7613 DIE BOORD
Telefon +27-21-8O9-216O
info-africa@caseconsult.com

www.caseconsult.com