Why Control DCD should be used as a pre-compiler for new Enterprise COBOL 6.2

Marshal Crawford – CEO Marble Computer, Inc.

Enterprise COBOL 6.2 seems to be where the industry is moving. It has the advantage of creating faster execution time code for programs after being compiled and put into production.

It also has the disadvantage of MUCH slower compile times, according to IBM estimates, 5 to 10 times longer than previous compilers. The 5 to 10 times estimates can actually go much higher for COBOL programs with programs with moderate to complex PERFORM usage.

Cleaning up in-house programs and utilizing Control/DCD can be a solution to decreasing COBOL 6.2 compile times.

As mentioned above, with the IBM Enterprise compilers, COMPILE TIME is longer. In addition, compiles also require more address space. If there is a way around doing needless extra compiles, then that should at a minimum, be investigated.

Control/DCD gives unique analysis information beyond what the IBM COBOL compilers produce. It acts as a COBOL pre-compiler in that it does logic and syntax analysis just like the COBOL compiler and it produces ERROR messages for incorrect SYNTAX as does a COBOL compiler.

Rather than compiling the program a few times to remove errors, using Control/DCD as a pre-compiler will allow removing most or all of the errors and at the same time provide extra ANALYSIS information that assists in the proper maintenance of a COBOL program. Below is an example of Error Messages produced by Control/DCD. (The software produces over 1,200 different types of ERRORs.)

```
SQ-NBR  MSG-NBR       DESCRIPTION

138   DCDEJ100-I  PARSING CAN NOT FIND THE BEGIN OF SECOND PART OF NON-88 CONDITIONAL
138   DCDM5J01-W  THE NAME ( WS-END-INVT ) DOES NOT MATCH TO A DATA DIVISION NAME
203   CSEM6IE4-I  VALUE '12 6' INVALID for field at 00102
248   CSEM6KC3-I  MOVE '12T44' may reach field at (0481) via MOVE at 01346
337   CSEM6IB2-I  MOVE HIGH-VALUES may be INVALID for field at 00153
441   CSEM6IA2-I  VALUE SPACES has REDEFINED Numeric Overlap Pos at 00045
```

The time savings in using Control/DCD as just mentioned is ONE reason for using Control/DCD as a pre-compiler.

The ANALYSIS provided by Control/DCD is the OTHER reason. The rest of this white paper is dedicated to the types of UNIQUE analysis that Control/DCD does provide.

A COBOL program has two significant DIVISIONs, the DATA DIVISION and the PROCEDURE DIVISION.

- The DATA DIVISION identifies many data fields along with characteristics or attributes for each field
- The PROCEDURE DIVISION uses these fields in conjunction with each other to complete the task of any one program

It is the logic portion in the PROCEDURE DIVISION where complexity resides and without this complexity, programming would be quite easy and most any programmer would routinely do the task with little effort.

Handling the complexity is where Control/DCD shines:

Programs are routinely structured* using the PERFORM verb in COBOL, where a higher hierarchical performed routine performs a lower one that may easily itself perform one lower hierarchical then itself.

Control/DCD uses indentation in a Forward Tracing chart showing at one glance a complete picture of how each PERFORM in a program relates to each other PERFORM. See Forward Tracing example below. The & character before each Performed-Routine Name is there for unique searching on the PERFORMED ROUTINE name.

```
  #FORWARD TRACING    &FORWARD-TRACING
1    1231    &PROGRAM-ENTRY
2    1267      &C-BUILD-NARR-FILE-TO-MERGE
3    1289      &M-MERGE-THREE-FILES-TO-ONE
4    1560        &FILEIO-ERROR-ROUTINE   --> (4 Performs)
5    1331        &N-SORT-INPUT-PROCEDURE
6    1387          &N220-LOOK-FOR-MATCH   --> (Perform/UNTIL)
7    1408          &N410-TEST-SEQ-IN-RANGE   --> (Perform/VARYING)
(See #4)             FILEIO-ERROR-ROUTINE
8    1453        &P-SORT-OUTPUT-RTN
9    1467          &P-VERIFY-TABLE   --> (Perform/TIMES)
10   1502      &Z-END-ROUTINE
```

Sometimes PERFORMed ROUTINES are used in more than one place. In the above example between lines 7 and 8 is the line beginning '(See #4)' for FILEIO-ERROR-ROUTINE. PERFORMs used from multiple routines need to be followed carefully.

PERFORM ERRORs often exist in a program and they sometimes can go unseen for years. Control/DCD identifies all PERFORM ERRORs, including MAJOR, MODERATE and MINOR ERRORS.

See the example below, where there is a MAJOR perform error where a GO TO leaves the normal path of the Perform Exit to somewhere outside the range of the PERFORM.

```
PERFORM Warnings & Major Errors
 Count   Type & Seq Nbr(s)

  01   GO TOs leaving range of PERFORM      MAJOR PERFORM ERROR
         477
```

The last feature covered in this white paper is how complete ANALYSIS is done for each data field to speed and simplify the understanding of what happens to that data field through the entire program very clearly. See both DESCRIPTION and EXAMPLE below of the analysis created by Marble Computer's DDM (Digital Documentation Manual) on one data field.

DESCRIPTION

The following analysis offers the following information:

- **Data Attributes** (From-To positions, parent 01 record, PIC, etc.) – see 1st part in example.
- *Relevant to this data-name* **Forward-Tracing** (by PERFORMED Routine) – see 2nd part in example.
  - See 'Forward Tracing' under articles on marble web site www.marblecomputer.com
- **Activity by Routine** within Performed Routine - see 3rd part in example.
  - This shows the essence of each Procedure Division Statement referencing this name underneath the Performed-Routine that is belongs to.
  - To understand how the Performed-Routine relates to other Performed-Routines, look at Forward-Tracing, JUST ABOVE Activity.

EXAMPLE

```
(0429) &CR-RED-ACCT                                    / This field: \
     In 13-16 of 01 CR-BILLING-RECORD          ==>    Data Attributes    <==
     In WORKING-STORAGE
     05  CR-RED-ACCT
         PIC S9(7)   VALUE ZEROS
         USAGE COMP-3

 01     PROGRAM-ENTRY                           ==>    Forward Tracing    <==
            02      B-PROCESS-INPUT --> (Perform/UNTIL)
             03         BM-DO-ACCT-VERIFCATION
            02      F-FORMAT-NEW-RECD --> (4 Performs)
             03         FRA-ADJUST-RED-ACCT

  PROGRAM-ENTRY                                    ==> Activity by Routine <==
      MOVE 135 TO CR-RED-ACCT (729)
      MOVE 19 TO CR-RED-ACCT (741)
      MOVE 103 TO CR-RED-ACCT (769)
  B-PROCESS-INPUT --> (Perform/UNTIL)
      IF CR-RED-ACCT = 19 (1135)
      IF CR-RED-ACCT = BR-RED-ACCT @343 (1162)
  BM-DO-ACCT-VERIFICATION
      MOVE 136 TO CR-RED-ACCT (1291)
      IF CR-RED-ACCT NOT = ZERO (1325)
  F-FORMAT-NEW-RECORD
      IF CR-RED-ACCT = BR-RED-ACCT @343 (1721)
  FRA-ADJUST-RED-ACCT
      MOVE 999999 TO CR-RED-ACCT (1983)
```

Below is an analysis in understanding the time savings this Narrative offers several times a day when analyzing a COBOL program.

Steps without Control/DCD Narrative:

1. Look at a statement in the Procedure Division containing one data-field
2. Find all other references to this one data-field in cross reference map of compile listing
3. Make a note of what Performed Routine they are in and then briefly note the Procedure Division activity for later analyzing
4. Build some type of understanding of Perform Routines in relationship to each other
5. If Data Attributes are needed (then make a note of what is needed)

6. Organize everything on one sheet of paper and then review

Average approximate Time: 25 minutes (varying from 10-45 minutes)

Steps with Control/DCD Narrative:

1. Do Find on Narrative for this data-name and review

   Approximate Time: 15 seconds

Percent of time spent doing 2nd activity compared to doing 1st activity is **1%**.

This allows a higher retained focus when using Control/DCD and easier maintenance of the program with less errors when the program is put back into production.

For more detailed information on Control/DCD please visit the Marble Computer web site www.marblecomputer.com/downloads.