A Methodology for Incorporating Programming Management Concepts Into a Cobol Course

Earl Chrysler
CIS Department, Quinnipiac University
Hamden, CT, 06518, USA

Abstract

A student in the first course in COBOL is typically taught the syntax of the language and basic processing logic, e.g., creating and updating files with file maintenance data and transactions and printing accounting and/or management reports. This paper presents a methodology for introducing the student to programming management procedures such as establishing program naming conventions, utilizing source statement library procedures for file definitions, and designing, performing and documenting thorough program tests. These techniques will not only assist students in developing valuable habits and recognizing the value of such programming management techniques, but make them aware that these techniques should be in place in the IS area in which they program or manage programmers. That is, the first COBOL course can be one of those courses that contains concepts of value not only in their entry-level position but in IS positions they hold later in their careers. The specific programming management procedures as they relate to COBOL programs are presented and examples are discussed.

Keywords: Programming management, career skills, COBOL standards, program testing

1. BACKGROUND

Graduating IS students typically enter into positions such as Computer Programmer, Programmer/Analyst Trainee, Systems Analyst Trainee or Business Analyst Trainee. Depending on their interests, they may progress to positions such as Programming Manager, Project Manager, Systems Manager, and eventually IS Director and Chief Information Officer.

Research performed by the author has found that the value of the content of specific IS courses may have primary value in one's entry-level position, in a higher-level position later in one's career, or in both types of positions. Although the first course in COBOL is generally thought of as primarily a skills course, it presents the opportunity to become a course which provides students also with knowledge that will become useful at a later stage in one's IT career.

2. PROGRAMMING MANAGEMENT TECHNIQUES

Program naming conventions
In audits of systems performed by the author it has been typically found that COBOL programmers have designed their own file definitions, including their own field names, for all records, for files that are used in several applications. Further, the author has rarely found naming conventions for files, records and fields published in an organization and, if published, only occasionally adhered to or enforced. This is an area where those teaching a COBOL course can include the requirement of adhering to naming conventions as part of a programming assignment.

Student use of naming conventions
When a student designs an FD for a file, the name of the file, the names of all records within the file and the names of all fields within all records are required to conform to the naming conventions provided to the class. A report file would have required two entries. One entry would have been for the FD in the File Section. Another entry would have been for all of the print line layouts, which would be copied into the Working Storage Section. A copy of the naming conventions the author provides and requires adherence to is shown in Appendix A. The students discuss the value of naming conventions in class from a management perspective for future reference.

Use of source statement library file definition (FD) entries
It is relatively commonplace to have changes occur to record layouts over time. For example, new fields may be added, old fields may be deleted and coding structures may change. If each time a program is written to access a file the programmer develops a unique FD, this implies that whenever a change occurs in a record layout each program that accesses this file must be

located, its FD must be changed and the program re-compiled. In many organizations the information regarding the files accessed by each program is not routinely documented.

If, instead, a master FD had been created for each file and this information was noted, a significant amount of time would have been saved. When a programmer was to write a program that was to access an existing file, the programmer would merely have had to COPY the FD from the source statement library into the program. Further, whenever the record layout needed to be changed, only the master FD would have needed to be revised, then all programs that accessed the file would have been re-compiled, COPYing in the revised FD. In order to have the latter capability, however, there would need to be someone charged with monitoring the library of master FD's.

When a new file is created, the FD would be placed in the FD source statement library and the file would be added to the FD master list and a Program-File Matrix, which indicates all FD's used in each program.

Whenever a record layout needed to be changed (1) it only needed to be revised in one location and (2) no program revisions would be required using this methodology. As a consequence, all programs that accessed a specific version of a file whose record layout has been modified, which was now documented, only needed to be retrieved from the source statement library and re-compiled. The revised record layout would have been automatically incorporated into the program. Procedure Division changes only would have been required if the revised record layout required any modifications to the program logic.

Student use of the source statement library
To incorporate this programming management technique into the COBOL course, the student is required to develop an FD for each file used in the first file-processing program and place the FD's in his/her personal Source Statement Library. Since there are several later programs that require processing many of the same files, the students quickly realize the laborsaving value of the Source Statement Library. The value to an IS facility as changes occur to record layouts over time is discussed to assure that students understand the long-term value of this procedure.

Program testing
Occasionally one will go to a firm's web site and, upon selecting a specific link and clicking on it, get a message that a Javascript error has occurred. This implies that the programming management technique of program testing and test documentation has obviously not been followed.

Student program testing and test documentation
As an introduction to the concept of program testing, students are provided with the data for all records to be created in a keyed-sequential master file. They are required to incorporate a routine in the program that prints each record after it has been built, but before it has been written to disk. They are also required to put an "INVALID KEY" clause for the write statement that displays the key field of a record that cannot be added to the file due to an error, along with a message describing the nature of the error. The key field of one of the records to be created is purposely placed in an out of sequence manner on the handout used by the students to enter the file-creating data, which will cause the invalid key clause to be executed. The students are not informed of this.

The students are required to print a listing of the file that has been created and compare it to the printout created while the program was executing and asked to (a) compare the two printouts and (b) explain why they differ. Since they assumed that their professor has provided them with correct data to be entered, they have difficulty resolving the difference between the two printouts. Interestingly, most students ignore their invalid key clause message on their monitor or they come to my office to tell me "something went wrong" in their program because "this message appeared on my screen".

In subsequent programs students are required to update their keyed-sequential master file and create reports from records in the file. For an update program, records are provided that must be processed against the master file. For each record, the students must look at the record and a printout of the original master file and determine (a) what the purpose of the record is (by its record code) and (b) what processing should occur. If the update record is valid, they are aware of what processing should transpire and they are required to prepare a description of what should occur and then describe how the reader verifies that it did occur by referring the reader to the printout of the master file after the program has executed. If the update record is not valid, they are to describe what message appeared on the screen to inform the user of the nature of the error and the data that identified which record was in error.

For report printing programs, the students must again refer to the master file printout and prepare a description of what should appear on the report how the reader verifies that the required data did in fact appear on the report. An example of the type of documentation the students are to provide appears in Appendix B.

By exposing the students to the documentation of the processing of valid and invalid records and creating reports, they learn that one must never assume that one's program is error-free. They also learn that submitting a program which compiles without a syntax error but contains logic errors may only be embarrassing and cost a few points in a class, but can have much more severe consequences when one is developing production programs. During class discussions the various types of edits that must be performed on incoming data are considered. For each type of edit, students are required to discuss how a record could be designed to test to assure the program detected the erroneous data and reported the error condition in an appropriate manner to the user.

3. SUMMARY AND CONCLUSION

This paper presents a methodology whereby the content and format of the first course in COBOL can be modified to incorporate additional student activities. These additional activities change the objective of the course. Instead of being primarily concerned with assuring a student becomes skilled in the use of the language to perform business-processing tasks, the course now includes programming management concepts. The value of these programming management concepts in terms of increasing programmer productivity is demonstrated by their personal experiences. The value of the concepts to assuring a well-managed IT area is explored in class discussions.

APPENDIX A

Class naming conventions

Use the following method for naming files, records, and fields in the file section:

Files: XXX-YY-FILE

Where XXX is the system indicator, for example PAY for a payroll system, INV for an inventory system, and where YY is the file indicator, OM for old master (for legacy sequential file applications), NM for new master (for legacy sequential file applications), M for input-output master, TR for transactions, FM for file maintenance, UD for update, and RP for report. For example, INV-OM-FILE would indicate that the file is a sequential inventory master file and is the version that will be input to a file maintenance, transaction or updating process. The version of the additional new file that exists after the file maintenance, transaction or update processing has been completed would be designated as INV-NM-FILE.

Records: XXX-YY-ZZZ...Z- RECORD

Where XXX and YY are the same as for the file and where ZZZ...Z is a descriptive name of the purpose of the record if multiple record types exist for the file. For example, INV-TR-WITHDRAWAL-RECORD indicates that the record is an inventory transaction file record that contains data regarding a withdrawal from inventory. When only one type of record exists for a file, the ZZZ...Z portion of the record name would not be used, e.g., INV-TR-RECORD.

Fields: XXX-YY-ZZZ...Z-FFF...F

Where XXX, YY, and ZZZ...Z are the same as for the record and

where FFF...F is a descriptive name for the data contained in the field. For example, INV-TR-WITHDRAWAL-PART-NO indicates the field contains the part number of a record which is an inventory transaction file record regarding a withdrawal from inventory. When only one type of record exists for a file, the ZZZ...Z portion of the field name is not used, e.g., INV-TR-PART-NO.

Use the following method when naming lines and fields of reports that are defined in the Working-Storage section:

The layouts of all print lines for a report are to be at 03 or higher level numbers, and the 01 level will be a group item that identifies the name of the report or print-out, for example:

```
1    ANNUAL-W2-FORMS
```

The layout of all the print lines, following the standards below, will be COPYed at this point in the source code program.

When naming a 03 level entry in working storage that is the image of a print line of a report, use an abbreviation of the report name followed by -RP followed by a description of the type of print line image, e.g.

STATUS-RP-TITLE-LINE,
STATUS-RP-COL-HDG-LINE-ONE,
STATUS-RP-COL-HDG-LINE-TWO,
STATUS-RP-DETAIL-LINE,
STATUS-RP-DEVICE-TOTAL-LINE,
STATUS-RP-GRAND-TOTAL-LINE.

The fields within a print line image should be named following the same pattern as for fields within records, for example the field name

STATUS-RP-DETAIL-PART-NO

would be used for the field that would contain the part number in the detail line of the status report.

APPENDIX B

*	NOTE: Below are examples of how you are to prepare a narrative.

The purpose of file maintenance record #1 is to add a new record to the customer master file with a customer code of NWD001200. Note that on the old customer master there is no record with a customer code of NWD001200 so this is a valid attempt to add a new record. Note that on the new customer master file a record with a customer code of NWD001200 with the data from file maintenance record #1 is present.

The purpose of file maintenance record #2 is to revise the address of the customer master file record with a customer code of SWS004500. Note that on the old customer master file record with customer code SWS004500 the address is 12090 S. CEDAR STREET and on the new customer master file record with customer code SWS004500 the address is that from the file maintenance record, 1400 PROMENADE WAY.

The purpose of file maintenance record #3 is to delete the customer master file record with a customer code of SED012000. Note that there is no record on the old customer master file with a customer code of SED012000. Therefore, examination of the source code will show that an error message was displayed on the screen that informed the user that no matching record was found for this file maintenance record and it was not processed.