Communications of the IIMA

2003

COBOL on Broadway: an Innovative Approach to Teaching Programming Concepts

Tanya Goette
Georgia College & State University

Follow this and additional works at: http://scholarworks.lib.csusb.edu/ciima

Recommended Citation

COBOL on Broadway: an Innovative Approach to Teaching Programming Concepts

Tanya Goette

Georgia College & State University, ISC Dept., CBX 12, Milledgeville, GA 31061
478-445-5721 tanya.goette@gcsu.edu

ABSTRACT

Students often have a difficult time grasping programming concepts the first time they are exposed to a programming language. This paper shows how role-playing can be used to demonstrate COBOL programming logic to students. The students role-play parts of a COBOL program to participate in the execution of the programming statements. "Seeing" how the program executes and data is stored seems to enhance the students' understanding of the program.

INTRODUCTION

Tell me and I'll forget; show me and I may remember; involve me and I'll understand.

Chinese Proverb

Most information systems majors enter their first programming class without any prior knowledge of programming languages. They may have some experience with HTML, but they have never encountered decisions, loops, or file processing. Many students find these concepts to be quite foreign and very difficult to comprehend.

Our university teaches COBOL as the students' first language course. The typical textbook covers each of the four COBOL divisions (Identification, Environment, Data, and Procedure) in order and then uses a small sample program to show how these divisions work together using procedural programming to turn input from a file into output on a printed report. Some students (the "natural born" programmers) understand this concept easily, but the majority of students just cannot seem to put the big picture together.

This seems to be the critical juncture in the course. If the student does not understand how the COBOL program executes one line at a time in the Procedure Division using the data described in the Data Division, then there is no way the student will be able to add to the program the more complex topics of decisions, loops, and tables. After teaching COBOL several times, it became clear that it was essential to use an innovative approach to convey this information to the student. This paper explains how a classroom skit may be used to convey a COBOL program's execution.

RELEVANT LITERATURE

There are many creative teaching methods documented in various books and journals. Research shows that having the students involved in activities often help them retain and understand material better. Passive students do not learn as much as active students (Eckstein, 2000; Eckstein, Bergin, & Sharp, 2002).

One of the more creative approaches is using drama to make events real to students (O'Day, 2001; Braund, 1999; Taylor, 1998). This is frequently done in literature, history, and law, and has been used in computer science as well (Dean & Hinchey, 1995; Jones, 1987; Levine, 2000; Andrianoff & Levine, 2002). When learners are involved in a physical analogy it helps them understand rather abstract concepts (Eckstein, Bergin, & Sharp, 2002).

Andrianoff and Levine (2002) use role-playing to teach the concepts of object-oriented programming in an introductory class. Students are given scripts to use as they take the parts of objects in an object-oriented computer program. Several scripts are used during the course in order to increase the students' understanding of objects, encapsulation, inheritance, and polymorphism.

This case study explains how drama or role-playing may be used to teach procedural programming concepts in COBOL. The skit gives each student a way to be actively involved in learning.

PROGRAM INFORMATION

A brief explanation of a COBOL program is needed in order to understand the skit. The Identification and Environment Divisions describe the basic information about the program and the files being used in the program. The Data Division defines the data through the use of variable names and types. COBOL has two main data types. An X symbolizes text data while a 9 is used for numeric data representation. A Picture clause is used to indicate each variable's length and type. This data is located in various storage locations and is then used by the Procedure Division.

The Procedure Division consists of the actual program instructions. These instructions manipulate the data in order to produce output. Figure 1 shows a complete COBOL program. A program like this one is written on the board and explained to the students before the skit begins.

```
IDENTIFICATION DIVISION.
PROGRAM-ID. Sample.
AUTHOR. Goette.

ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
SELECT Student-File ASSIGN TO DISK "A:stu.dat"
```

```
   ORGANIZATION IS LINE SEQUENTIAL.
SELECT Report-File ASSIGN TO DISK "A:Sample.rpt".

DATA DIVISION.
FILE SECTION.
FD  Student-File.
01  Student-Record        PIC X(53).
FD  Report-File.
01  Output-Record         PIC X(80).

WORKING-STORAGE SECTION.
01  Student-Record-In.
  03 SSN-in               PIC X(9).
  03 First-Name-In        PIC X(15).
  03 Last-Name-In         PIC X(20).
01  Eof-And-Hold-Fields.
  03 EOF-Flag             PIC X VALUE "N".
01  Page-Heading.
  03             PIC X(5) VALUE SPACES.
  03             PIC X(19) VALUE
     "Student Information".

01  Column-heading-1.
  03             PIC X(5) VALUE SPACES.
  03             PIC X(10) VALUE "First Name".
  03             PIC X(5) VALUE SPACES.
  03             PIC X(9) VALUE "Last Name".
01  Detail-Line.
  03               PIC X(5) VALUE SPACES.
  03 First-Name-Out      PIC X(15).
  03               PIC X(3) VALUE SPACES.
  03 Last-Name-Out       PIC X(20).

PROCEDURE DIVISION.
Main-Para.
   PERFORM Initialize-Para
   PERFORM Process-Para UNTIL EOF-flag = "Y"
   PERFORM Close-Files-Para
   STOP RUN.

Initialize-Para.
   OPEN INPUT Student-File
      OUTPUT Report-File
   WRITE Output-Record from Page-Heading AFTER PAGE
   WRITE Output-Record from Column-Heading-1 AFTER 3.
Process-Para.
   READ Student-File INTO Student-Record-In
    AT END
       MOVE "Y" TO EOF-Flag
    NOT AT END
       PERFORM Output-Para
   END-READ.
Output-Para.
   MOVE First-Name-In TO First-Name-Out
   MOVE Last-Name-In TO Last-Name-Out
   WRITE Output-Record FROM Detail-line AFTER 2.
Close-Files-Para.
   CLOSE Student-File
        Report-File.
```
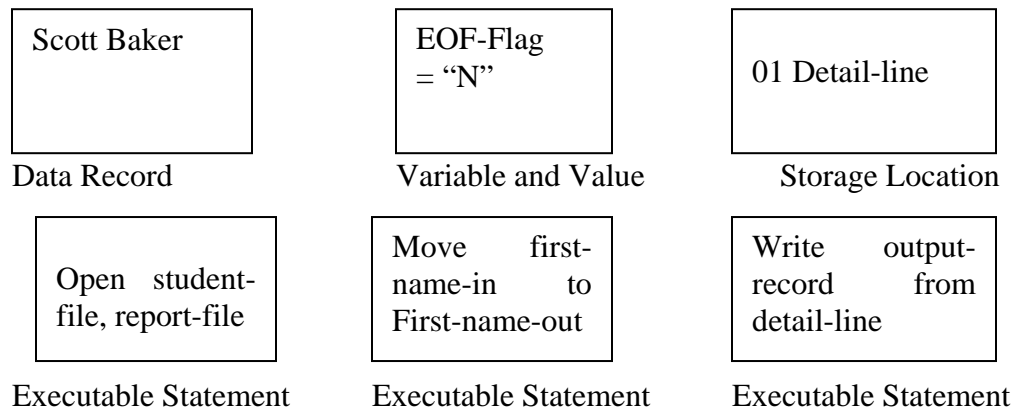
Figure 1: A Sample COBOL Program

This simple program opens the files and reads a student's record until there are no more records. When a record is read, the information is moved to the output area and then

written to the report-file. After the last record has been processed, the files are closed and the program ends.

THE SKIT

Students are asked to volunteer (or are drafted) to play roles in the COBOL skit. The roles are data records, Working-Storage variables, Data Division storage locations, and Procedure Division executable statements. Approximately 15 students are given signs with words on them indicating their part in the skit. Figure 2 shows examples of some of the signs.

Scott Baker	EOF-Flag = "N"	01 Detail-line
Data Record	Variable and Value	Storage Location
Open student-file, report-file	Move first-name-in to First-name-out	Write output-record from detail-line
Executable Statement	Executable Statement	Executable Statement

Figure 2: Sample Signs

The students that are data records take their place outside the closed classroom door while the other students are assembled in processing order at the front of the classroom. The skit begins with the Open statement in the Procedure Division, and the door to the classroom is opened. Execution of the statements continues until the Read statement is reached. At this line, the first student with a data record sign enters the classroom and stands behind the input record storage sign. The Not At End statement is performed and execution proceeds to the output paragraph.

Program execution continues. When the Move statement is reached, the student moves from the input record storage location to the output record storage location (from standing behind a student holding the input sign to standing behind the student holding the output sign). When the Write line is executed, the student moves to the report location. This movement allows students to visualize how data is moved within the program. Then the EOF-flag is checked. Because its value is "N", the next record is read (another student comes in and repeats the process).

Execution proceeds through the loop until the EOF-flag is "Y" and no more records (students) are in the data file. There are usually three to four students playing the part of the data records. Students can see that the file is empty and that the value of the EOF-

flag variable has changed to "Y". Students can see how the program execution completes.

After the program has completed, the volunteers sit down and new volunteers are used to repeat the skit again. If the class is small, some students are used again, but in a different role. Some students may also hold multiple signs if needed.

The students are asked at the conclusion of the two performances if the COBOL skit helped them to understand more about the way the program reads records and executes lines of code. While no formal analysis of the students' comprehension of the program has been performed, almost all the students agree that they now have a better understanding of program execution.

CONCLUSIONS

This skit allows students to "see" how data moves through a COBOL program and where it is stored. It shows students the importance of data names and gives them an understanding of paragraphs, loops, and flags (indicators). This skit has been used for over five years in COBOL classes of various sizes. It seems to work best in a class size of 25-30 students because students do not have to be assigned multiple signs and they can watch once and participate once.

The involvement of students in the learning process enables them to understand and retain more of the information presented. By using dynamic learning, abstract concepts can be made more concrete. Future research may include evaluating the students' understanding of the COBOL program's execution before and after the skit is performed.

REFERENCES

Andrianoff, S. & Levine, D. (2002). Role playing in an object-oriented world. *SIGCSE Bulletin*, The 33rd SIGCSE Technical Symposium on Computer Science Education, 121-125.

Braund, M. (1999, Sept.). Electric drama to improve understanding in science. *School Science Review*, 81(294), 35-41.

Dean, N. & Hinchey, M. (1995). Introducing formal methods through role playing. *SIGCE Bulletin*, 27(1), 302-306.

Eckstein, J. (Retrieved August 26, 2003). Pedagogical pattern #7: Incremental role play. Online. http://www-lifia.info.unlp.edu.ar/ppp/pp7.htm.

Eckstein J., Bergin, J., & Sharp, H. Eds. (Retrieved August 26, 2003). Patterns for active learning. *Submission to the PPP pattern language project*, Online. http://www.pedagogicalpatterns.org.

Jones, J. (1987). Participatory teaching methods in computer science. *SIGCE Bulletin*, 19(1), 155-160.

Levine, D. (2000, May). Helping students through multiplicities. *The Journal of Computing in Small Colleges*, 15(5), 285-291.

O'Day, S. (2001, April). Creative drama through scaffolded skits in the language arts classroom. *Primary Voices K-6*, 9(4), 20-25.

Taylor, P. (1998), Redcoats and Patriots: Reflective Practice in Drama and Social Studies. Dimensions of Drama Series.