TEACHING ON-LINE APPLICATION PROGRAMMING USING CICS

by: Sandra Poindexter
Management, Marketing, CIS
Northern Michigan University
Marquette, Michigan 49855

ABSTRACT: The intent of this paper is to encourage the teaching of on-line programming skills and to provide suggestions and a model as to how this can be accomplished in a University course. The effectiveness of the model is supported by the results of student evaluations and questionnaires. The vehicles used in this model are an IBM 4381 mainframe and the CICS Command Level language.

KEYWORDS: CICS, Online Systems, Mainframe Programming

TRANSITION TO ON-LINE PROGRAMMING

Universities have traditionally concentrated on batch applications when teaching programming methods and languages on mainframe computers. This was due to several factors: entry-level programmers generally worked in the batch environment, the relative ease of establishing and controlling the mainframe computer resources for batch as compared to an on-line environment, and the coding complexity of on-line programming. All of these were valid at one time, but trends are changing. Exposure to on-line mainframe programming will give graduates an edge in career placement and promotion.

Students are comfortable with software packages utilizing menus, prompts, and terminal input/output. Most universities have on-line systems for administrative functions and therefore already possess the necessary hardware and software for programming on-line applications. If those resources can be accessed by the academic side of the university the added cost for offering such a course on the mainframe is not significant. While there are various systems software for developing on-line systems, this paper focuses on CICS.

Still, on-line systems do possess hurdles. The systems software and maintenance of the operating system is much more complex than the standard academic batch environment. The computer operators require additional training, and resources are more sensitive to program failures. The program concepts and language interfaces are more sophisticated. However, these hurdles can be jumped given sufficient planning, cooperation, and patience.

ACADEMIC ON-LINE ENVIRONMENT

There are some very real problems that must be addressed in establishing a CICS on-line environment for student use. A program logic error under CICS can easily cause the entire CICS region to crash preventing all users of that region from executing until it is restored by an operator. Under batch each program normally runs independently from others and an abend affects only the individual with the logic error. Student design often lapses into a series of the "lets try this and see what happens" technique. This cannot be done with CICS without losing the availability of the computer and the patience of staff, instructor, and students.

Secondly, if the academic side is allocated a fixed amount of computer resources the CICS software will eat substantially into the space available for all academic users. Given the small percentage of students taking the CICS course this allots them a disproportionate amount of resources and can affect the throughput of other students.

Lastly, there is a continual reliance upon the computer operators and systems programmers. They will be responsible for the initial creation (not a small task), problem resolution, and maintenance of the abundance of tables under CICS. Few people are ever overjoyed with the idea of taking on additional work even if they recognize the merits of the overall goal.

To address these concerns the systems programmer at this university isolated the CICS students into their own test region. When the region crashes they affect only each other, not other users of CICS. The procedures for starting-up and restoring CICS, and terminating specific tasks and users were given to the academic lab operators who had to be trained in their use. Students contact the lab operators, not the Computer Center, in case of problems. EDF (Execution

Diagnostic Facility), an on-line debugging tool, was placed in the test region. Students are instructed in its use to prevent abends and infinite loops. (One caution here: EDF is very costly in terms of resources and just two or three simultaneous users could bring the test CICS region to a halt.)

CICS is table-driven software. All files, programs, terminals, task execution transaction id's, etc. are defined and stored in tables in advance of any programming. Therefore, a designated four character trans-id, entered by the terminal operator, triggers execution of a compiled program. In the case of student applications all CICS table entries were preassigned by the systems programmer. For example, group trans-ids were G1T1, G2T1, etc. for the PCT (Program Control Table) entries. This minimizes CICS table revision during the semester. Fifteen sign-on access lines were established for thirty students. Too many users on this scaled-down system at once reduces response time and space allocations may be exceeded causing CICS to crash.

Pseudo-conversational programs are the only accepted solutions in order to conserve resources with suspended tasks. Students are repeatedly told to walk through their initial design and logic changes prior to implementing them, and to use the CICS manuals as research aids. Peer pressure helps prevent carelessness because they are all aware of the impact of such carelessness on the class at large. Throughout the semester it important to encourage humor and a sunny attitude as ways to handle the stresses of an unpredictable system and uncooperative teammates.

COURSE DELIVERY

Goals and Policies.

The scope of this course is to teach the CICS Command Level language and its associated on-line logic concepts. Data communications and terminal hardware are briefly discussed when technical background is needed. The focus is CIS applications. Figure 1 is a sample syllabus showing course description, topics list, and policies. COBOL fluency is expected because CICS does not stand alone, but is integrated into a host language. For the class at large, COBOL was chosen over PL/1 and Assembler since our CIS students take one year of COBOL. The Systems Analysis & Design course includes the on-line topic and is a building block for this course.

Universities have traditionally concentrated on batch applications when teaching programming methods and languages on mainframe computers. ... but trends are changing. Exposure to on-line mainframe programming will give graduates an edge in career placement and promotion.

Teams are used as suggested by the systems programmer. It has several advantages over individual work: 1) fewer communication lines, CICS table entries, VSAM files and space resources are required as there are fewer actual programs to be tested; 2) team members benefit from each others perspective; and 3) debugging time and CICS downtime may reduce with two or more people having to agree on logic.

The first time the course was taught students were allowed to choose a teammate. Many simply selected the person sitting next to them and later found that incompatibilities existed. A midsemester team switch was done. In subsequent offerings the teams were assigned by the instructor based upon three criteria: GPA, available hours (mornings, afternoons, or evenings), and whether the student was a procrastinator or "go-getter". This latter method worked much better. A midsemester switch was still done since it provided an acceptable way for teams to separate as friends. The problems associated with the use of teams has to be handled on an individual basis. Frequent quizzes are given to separate the students into truer grade categories. Class participation and attendance are naturally high due to the complexity and interesting nature of the material.

Course Resources.

Six CICS books were reviewed in the text selection process. They were CICS Primer by Ryan (1), CICS Made Easy by LaBert (2), CICS/VS Command Level With ANSI COBOL Examples, 2nd ed. by Lim (3), CICS Command Level Programming by Jatich (4), On-line Systems Design and Implementation Using COBOL and CICS (5) by Kacmar, and a two-volume series CICS For The COBOL Programmer by Doug Lowe (6). The Ryan text was chosen once due to its building block approach and clearly written narratives, but students desired more complete examples and in-depth material. The two-volume series CICS For The COBOL Programmer by Doug Lowe has also been used, but has the potential disadvantage of requiring students to purchase multiple texts. Some of the others have a reference guide orientation, and are more suited to a professional programmer than CIS students. A new textbook, Essentials of CICS/VS by Robert Lowe (7), seems thorough and was well received by students. The books not adopted are on reserve in the library as outside reading material for various topics. All the above books are based upon COBOL as the host language.

The text is supplemented with the following handout material: Executing a transaction in a CICS region; Using EDF (Execution Diagnostic Facility); Common problems, Design standards; CICS commands summary; BMS (Basic Mapping Support) macros summary; and SDF (Screen Definition Facility) summary. The Common problem list was created as problems arose and solutions found. Items are added to the list by students and shared among their classmates.

Many aspects of CICS and EDF are difficult to grasp without visual aids. Once or twice a week a microcomputer with a modem and overhead projector is used in class to dial into the IBM mainframe

and demonstrate compilations, new copies, executions, EDF and assignment expectations.

Assignments.

Early in the course, the student needs to see a distinction between prior programming courses and CICS. The first assignment, given early in the semester, is to key a textbook example, submit it for translation and compilation, and execute it. After this, simple problem sets at the end of chapters are used as starting points for assignments. The class is asked to offer logical enhancements which then become part of the specifications. Other more complex problems follow.

An alternative to textbook problems is to create assignments based upon real applications from the Computer Center. Since on-line programs are usually very brief, it is possible to give eight or nine assignments (though most require at least two programs). The Computer Center typically has a backlog of user requests, some of which have a low priority and will not be addressed by the staff programmers in the foreseeable future. These make good problems for student assignments because there is no hurry to finish the work and the staff does not feel threatened by student work. Starter assignments are to develop menu and inquiry programs. Later tasks involve file maintenance and browsing problems. Such applications especially help to provide more realism and increase student satisfaction.

Every assignment has to be completed to a working state or a grade of Incomplete is given for the course. The student has until the end of the semester to finish incomplete assignments. No additional points are added to their grade, but this requirement helps to ensure that class members completing the course have mastered all topics covered.

Examinations.

Quizzes were given approximately every two weeks over two to four topics. Each quiz contained four or five short answer questions or short problems. Where the topic was appropriate, some quizzes also included multiple choice questions. In cases where one member of a team was carrying their group, the quiz scores usually differentiated the members.

EFFECTIVENESS OF THIS MODEL

Two instruments were used to determine the effectiveness of the course. The first instrument was a four page questionnaire where students were asked to state detailed opinions on the following five topics: teams, grading methods, assignments, course content, and general comments. Each topic was divided into one or more specific questions. Ten points were given for the completed document in order to provide an incentive for thoughtful answers rather than an unsupported yes or no. Figure 2 summarizes the results of the questionnaire given for only one semester offering (total number of students was 20).

The second survey took the form of an objective, eighteen question evaluation which was given to each student. The responses were a rating on a scale of 1 to 4, with 4 being the highest. Figure 3 shows the outcome of the evaluations given during two different semester offerings (total number of students was 48). Only those questions pertaining to the course content and related presentation are included in Figure 3.

While admittedly these are small samples, they do indicate the overall success of the course as well as problem areas which need to be addressed. The most pronounced areas which need improvement are in the available textbooks; none of the text used were rated highly. Several students did comment that the newly published Robert Lowe (7) text seemed to be both easier to understand and used as a reference.

CONCLUSIONS

Both the students and the instructor have been pleased with the CICS course. Student interest remains high throughout the semester. It is not uncommon for students to research topics and bring questions to class, or to go beyond the program specifications when doing assignments.

I strongly endorse the teaching of on-line programming skills. CICS on an IBM mainframe is not the only methods available to accomplish this task; other methods such as high powered databases and 4th generation languages can, and should, be offered. If a university does not have the mainframe support for these courses microcomputer versions are an alternative. However, if graduates are tending to be placed in mainframe-oriented computer centers every effort should be made to give them opportunity to learn these packages while still in school. Increasingly, students are going to be short-changed unless they receive some knowledge in the on-line arena.

REFERENCES/FURTHER READINGS

1. Ryan, Liz. CICS Primer. Chicago: SRA, 1986

2. Lebert, Joseph. CICS Made Easy. NY: McGraw-Hill, 1986

3. Lim, Pacifico. CICS/VS Command Level with ANS COBOL, 2nd ed. NY: Van Nostrand Reinhold, 1986.

4. Jatich, Alida. CICS Command Level Programming. NY: Wiley, 1985.

5. Kacmar, Charles. On-line Systems Design and Implementation (Using COBOL and Command Level CICS). Reston, VA: Reston Publishing, 1984.

6. Lowe, Doug. CICS For The COBOL Programmer Part 1 and Part 2. Fresno. CA: Mike Murach & Assoc., 1985.

7. Lowe, Robert. Essentials of CICS/VM. Dubuque, Iowa: Wm. Brown Publishing, 1988.

Figure 1

SYLLABUS: CICS Application Programming

Description:

The course is intended to build experience and competence in working with interactive, on-line business applications. Topics of on-line concepts, system design and programming will be covered. Emphasis will be programming on the IBM mainframe using CICS command level and COBOL languages.

Text: CICS For The COBOL Programmer Parts I and II, Doug Lowe
On-Line Systems Design & Implementation, Kacmar (On reserve)
Essentials of CICS/VM, Bob Lowe (On reserve)

Prerequisites: Advanced COBOL, Systems Analysis & Design, or consent of instructor

Outline:	Week	Text Reference
I. General Concepts	1-2	
A. Data Communications		Chp. 1
B. CICS		Chp. 2,3
		Handouts: Batch/Online, CICS Components, Transaction & COBOL Executions
QUIZ I		
II. Terminal I/O	3-4	
A. Procedures for Development		Chp. 4
B. Basic Mapping Support		Chp. 5
		Handouts: Common Problems
QUIZ II		
III. Basic Processing	5-6	
A. Basic Commands		Chp. 6
B. Design Considerations		Chp. 7
		Kacmar Chp. 2,4
		Handouts: Trans. Design
QUIZ III		
IV. Program Controls	7-8	
A. Debugging & Testing		Chp. 10
B. Control Command & CICS Storage		Chp. 8 Topic 1
		Handouts: Abend Codes
QUIZ IV		
V. File Processing	8-10	
A. SDF		Chp. 8 Topic 2
B. File I/O Commands		Chp. 9
C. File Inquiry/Maintenance		On Reserve: IBM Screen Def.Facility Manual
		Handouts: Update Logic, SDF Execution
QUIZ V		
VI. Advanced Topics	11-14	On Reserve: Book 2 Doug Lowe
A. Transient Data		Chp. 5
B. Temporary Storage		Chp. 4
C. Browsing		Chp. 1
D. Advanced Mapping		Chp. 6,7
QUIZ VI		

Grading: Assignments 60% / Quizzes (5) 40%

Figure 2

DETAILED SURVEY RESULTS

TOPIC: TEAMS

1. List the benefits you personally received as the result of the team concept.
   Answers: easier problem solving; someone to bounce ideas off; learning to work with others; learning to compromise; seeing new styles; pooling ideas; tutoring help; more effective time usage; oral communication improvement; motivation.

2. List the handicaps you felt occurred as the result of the team concept.
   Answers: meeting time difficulty; forced to forego preferences; disparity in commitment; intimidating partners; disparity in abilities; dependence upon others.

3. Given the above, do you agree or disagree with the team concept?
   Answers: Yes 20 No 0 (Some of the yes's were reluctant).

4. What suggestions do you have for improving the team concept?
   Answers: Pick your own partners (5); Instructor assign teams (6); Switch teams midsemester, optional (13); Instructor assign teams for first assignment only (2).

TOPIC: GRADING

5. Should team member evaluations be considered as part of a grade?
   Answers: Yes 8 No 11

6. Should there be fewer tests covering more information and worth more points, ie. two tests and a final?
   Answers: Yes 0 No 18

TOPIC: ASSIGNMENTS

7. State your opinion as to the worthwhile nature of the assignments and what aspects of them you enjoyed the most.
   Answers: Very useful (18); Build upon one another (5); Mapping the screens; Being given general specifications and allowed to customize; Update program (3); Browse program (3); Good variety.

TOPIC: COURSE CONTENT

8. Should SDF be incorporated prior to the last assignment? (note: the intent is NOT to teach SDF, but to convey the fact that tools exist. The BMS foundation is more crucial)
   Answers: Incorporate SDF earlier (10) Incorporate SDF at end (9)

9. Should more design philosophy, or telecommunications hardware/software topics be substituted for programming technique topics?
   Answers: On-line design topics should remain in the Systems Design course; CICS environment topics could be included in handouts.

TOPIC: GENERAL COMMENTS

10. If you have other comments, criticisms, improvements that are not addressed above, please add them here.
    Answers: Enjoyed the class (7); Should be a required course; Wish the system would crash less often; Disliked the idea of receiving an Incomplete if assignments were not done (2); Higher percentage of grade for assignments; More time for assignments (2)

Figure 3

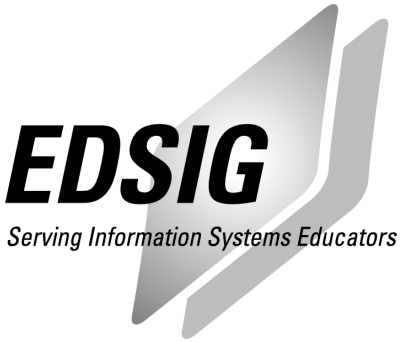
Student Course Evaluation

Questions		Mean Response
1. The objectives and purpose were kept clear during the course		3.53
2. Exams and other evaluations were consistent with the course		3.47
3. Overall, I would rate the textbook(s) as excellent	(Ryan)	2.36
	(Doug Lowe)	2.09
4. I would rate the outside readings as very useful		2.76
5. The course improved my ability to solve problems		3.45
6. An overall plan for the course was followed		3.58
7. Hand-in assignments stimulated my learning		3.67
8. The instructor was responsive to student reactions and comments		3.58
9. The instructor taught in an interesting way		3.34
10. The topics presented in this course stimulated my interest to learn		3.52
11. This course challenged me to think		3.70
12. This course emphasized thinking rather than memorizing		3.55
13. I enjoyed taking this course		3.40
	Overall Mean	3.37

ABOUT THE AUTHOR

Sandra Poindexter is an Assistant Professor of Computer Information Systems at Northern Michigan University in Marquette, Michigan. She has worked as a programmer and systems analyst, taught a variety of CIS undergraduate courses, and coordinated major revisions to CIS undergraduate curricula. Her professional interests involve reviewing CIS textbooks and writing case studies/problem sets for textbooks. She also served as one of the judges for the 1988 EDSIG Case Study Competition.

Information Systems & Computing
Academic Professionals



STATEMENT OF PEER REVIEW INTEGRITY

All papers published in the Journal of Information Systems Education have undergone rigorous peer review. This includes an initial editor screening and double-blind refereeing by three or more expert referees.

ISSN 1055-3096