CMS AT NORTH CAROLINA STATE UNIVERSITY: TAILORING A TIME SHARING SYSTEM FOR COMPUTER SCIENCE INSTRUCTION

Lionel E. Deimel, Jr.

Department of Computer Science
North Carolina State University
Raleigh, North Carolina 27650

ABSTRACT

The convenience of a time sharing system from the point of view of a computer science instructor is considered. Tools which may be helpful for course administration are described. The experience of the North Carolina State Computer Science Department with its IBM VM/CMS system is considered in detail and its strengths and weaknesses noted.

1. ANCIENT AND MODERN HISTORY

Much has been written about tailoring the interactive environment for more effective programming, both in the academic and nonacademic setting. Less has been said about how that environment can support more effective *administration* of programming classes and thereby improve the quality of instruction. When the Computer Science Department at North Carolina State University (NCSU) began studying the purchase of an interactive computer system to teach its undergraduate programming courses, providing support for the faculty in its teaching was considered an important issue.

Four factors contributed to the departmental concern for faculty support. First, the faculty had many years' experience with a large IBM System/360 batch system and its successors and with IBM TSO time sharing. Certain aspects of these systems--most notably the areas of file and account security--were quite inflexible and had often complicated the work of student and teacher alike. This fact led to the insistence that any new computer system provide relief from such problems. Secondly, a number of administrative tools which had proved quite useful had been developed for the batch system, and it was believed that any new system should provide similar assistance. A third factor was that a new interactive system was not meant to replace the current System/370 providing batch and TSO service, but only to supplement it. Therefore, it was felt that communication between the two systems was essential, to use hardware or software that might be unique on one system or the other, to avoid unnecessary duplication of files, and to allow faculty to use either system for course-related work irrespective of which one was being used by their students. Finally, the increasing Computer Science student population had severely strained the ability of the department to service its course demand. Anything which would ease the teaching or administrative burden on instructors would benefit the teaching mission and, indirectly, the research mission of the department.

At the time the Computer Science Department was studying the purchase of an interactive computer system, the campus Computing Center was doing the same. The Computing Center's goal, conditioned in large measure by financial considerations, was to provide interactive-edit/batch-submission capability for the existing batch system. For various reasons, financial considerations prominent among them, the Computing Center and Computer Science Department pooled available funds and purchased an IBM 4341 processor on which to run CMS under VM. Shortly after the system was delivered, NCSU began running an enhanced system known as VM/SP (System Product). For simplicity, we will usually simply refer to the system as CMS.

2. THE CLEAN SLATE

Like most time sharing systems, CMS has both features which make it an attractive vehicle for teaching beginning programming courses and features which make it unattractive. Also, like most systems, one can add to or modify CMS to give it more desirable characteristics. CMS users at other institutions may be interested in the details of our customizing efforts, whereas users of other systems will want to concentrate on the underlying philosophy of those efforts and interpret them in the context of their own systems. The ideas applied at NCSU do not, in general, rely upon the peculiarities of VM/CMS. Nonetheless, since this operating system is somewhat unusual, a brief description of it is in order.

"VM" refers to the IBM Virtual Machine Facility. This is an operating system which provides many users with "virtual" System/370 computers. Each virtual machine acts like a complete computer system devoted to a single user. A program called CP (Control Program) allots real CPU time on the real CPU to the virtual machines. CP provides commands to manipulate the virtual machines--to log on or log off, change peripheral configurations, manipulate unit-record devices, etc. CMS (Conversational Monitor System) is a single-user interactive operating system which runs on one of CP's virtual machines. It is CMS which provides a file system for each user, processes a command language (called EXEC2), and provides most services one expects of a time sharing system. Disk storage is provided for individual users in the form of "minidisks," segments of real disks permanently assigned to particular users. Access to minidisks by other users is controlled through a system of passwords and entries in a master directory defining individual user privileges. A typical user is given a virtual machine along with a unique userid and password. The virtual machine has read-only access to a number of system minidisks, one or more private minidisks, and some assortment of virtual card readers, consoles, and printers. Other virtual devices may be defined at will.

CMS has many virtues from the viewpoint of a computer science department. The security features of CMS are very good indeed--student files can be compromised only through carelessness (inadvertent disclosure of passwords) or outright collusion. This is an important plus for a system to be used in an academic environment. The ability to tailor the system to one's particular needs through the use of stored lists of commands (called "EXEC's" under CMS) is quite extensive. It is this feature more than any other which allows CMS (or some other time sharing system, for that matter) to be treated as a clean slate on which we may draw a system especially for computer science instruction or for some other purpose.

CMS is not without drawbacks, however. The unique construction of the system, ideal for such activities as the development of operating systems or of assembly language programs generally, imposes unneeded and confusing complexity upon the beginning programming student. The student can be shielded from this complexity with difficulty and then only incompletely. The fact that the user interacts both with CP and CMS can be very confusing, and the use of virtual card readers and printers does not seem especially natural to a user who has not graduated from a card-oriented batch system. Nonetheless, CMS was judged to be an adequate base on which to build a teaching-oriented system, and experience has tended to justify that evaluation.

3. DIFFERENT STROKES FOR DIFFERENT FOLKS

The decision to purchase a computer jointly with the Computing Center provoked both relief and apprehension. The Computer Science Department was quite happy not to be in the computer-running business, a task which naturally devolved upon the Computing Center, yet the faculty was concerned about being at the mercy of an organization whose stated objectives for the new computer were quite distinct from its own. Nevertheless, these differences in goals were never hidden, so that it was understood by all concerned that different classes of users would have to be treated differently.

Two major techniques were used for varying the level of service provided users. One of these was the varying of virtual machine characteristics. In particular, the amount of virtual storage available to an individual user can be expanded or restricted to allow or disallow the running of certain processors. (On other systems, comparable mechanisms are usually available.) The other technique used was to place processors (PL/I, Pascal, PL/C, WATFIV, etc.) on different minidisks. Access to these minidisks must be granted explicitly in the master user directory. This latter technique is the more useful, as it allows even students in different classes access to different selections of software. In practice, beginning courses have had available a restricted set of processors, whereas more advanced classes (for example, the data structures course) were given much more freedom in their use of processors.

Because the needs of Computer Science users and other users of the system were so obviously divergent, a special Computer Science "system" minidisk was established on which to place software being provided to Computer Science faculty and students. An EXEC to access this disk was placed on one of the "normal" system minidisks. This gave the Computer Science Department complete control over its own software--neither the Computer Center's co-operation nor acquiescence was required to add, delete, or modify software used by the department. The disadvantage of this scheme, of course, is the ongoing maintenance problem thereby created for the department, a burden not to be dismissed lightly.

As we mentioned earlier, students in different classes can be given access to different software. This tailoring can be done for individual accounts also, of course, but the fundamental group having common needs distinct from other groups is the class. Computer Science faculty planning for the advent of CMS on campus focused much attention on the class unit. Previously, classes had been granted single accounts for batch or TSO access, and monitoring individual resource usage and providing individual file security was difficult. (Many schools, to their credit, have always granted individual accounts to students, a generally more satisfactory arrangement.) CMS allows students in a given class to own their private minidisks, access the Computer Science minidisk (as can any user, in fact), and also have read-only access to a minidisk reserved for the use of their particular class. The instructor, whose account is established at the same time as those of his students, has read/write access to the class disk. He may place any files or software he wishes to make available to the class on the class minidisk.

4. IN NUMBERS THERE IS GRIEF

Having to generate hundreds of student accounts each semester is no trivial matter. The task is complicated by students adding courses late, changing sections, or dropping courses after they have begun. (In the latter case, the student's computer account is to be deleted.) These tasks clearly need to be automated as much as possible and must be performed in a timely manner. The approach taken at NCSU toward this problem reflects the realities of the situation--at the start of the semester, reasonably reliable roll information is available from the registrar's office, but thereafter adds and drops need to be handled on a case-by-case basis. The Computing Center arranged to obtain at the beginning of each semester a magnetic tape containing class rolls for courses using CMS. Software was then developed to generate accounts and passwords for each student and to generate individual ID cards to be distributed in class with this information. From this point on, each instructor is responsible for his own students' accounts. He has the ability to create and destroy accounts as required and to generate ID cards for new accounts. This arrangement, only partially implemented as of spring semester 1982, is very attractive for the instructor, as most of the account maintenance work is done for him by the Computing Center without any special effort on his part. At the same time, however, when the need for an account maintenance transaction is acute, as when a student adds his course and needs computer access immediately, the instructor needs to fill out no paperwork nor rely on someone in another department across campus. In fact, a grader is usually available to perform the simple tasks needed to create the new account.

Even in small classes, but especially in classes of (apparently) ever-increasing size, record keeping can be burdensome. For a number of years, the Computer Science Department has used a program, GRADE75, for maintaining grade records and for computing and displaying grades. A related program generates random ID numbers and ID cards for students, and GRADE75 produces various grade listings by these ID numbers, which can be publicly posted. It was natural to want to have these programs available under CMS, and their installation was easily accomplished. GRADE75, of course, requires a class roll. Therefore, a CMS EXEC was created to take the roll file of names and userids, as created originally by the Computing Center from registration information and later modified by the instructor, and create the appropriate grading program roll file. This constitutes something less than an integrated data base approach to record keeping problems, but has proved quite effective. At least for now, a more grandiose system does not appear to be justified. The scheme has the advantage of having the *identical* grading programs available on both computer systems, thereby minimizing maintenance headaches. This consideration mitigates against further integration of function, for example, the production of ID cards containing both account information and the GRADE75 ID number. Note that it was decided not to use a single ID number for both purposes, since although there is often need for one student to tell another his userid, most students are unlikely to want to divulge a code number which will allow a fellow student to examine all their grades.

5. KEEPING THE LINES OF COMMUNICATION OPEN

Computer systems tend to be organic, ever-developing entities. This is especially true of newly-installed ones. It is important to keep users abreast of changes taking place--of software added, changed, or deleted, or of solutions to recurring problems encountered by an inexperienced user community. Different groups have different needs for information, however. At the highest level, CMS provides for the system administrator to send broadcast messages to all users, but this is a crude device for dissemination and cannot be used effectively except for short, vital notices. Since the Computer Science Department, as well as the Computing Center, is now in the software business, it too has need for broadcast messages for its users. Even Computer Science instructors have a similar need. What teacher has not realized an hour after the last class before a programming assignment is due, the *crucial* point he failed to mention in his lecture? Obviously, the system broadcast message service is not the answer to everyone's need to publish information. The user, insofar as he is a member of each group--of users generally, of Computer Science users, of members of a specific class--has need of two kinds of information: time-critical headlines and detailed explanations to be examined at greater leisure. On NCSU CMS, we have tried to meet both kinds of needs. An EXEC named BULLETIN prints out Computer Science headlines from a special file on the Computer Science minidisk. The MESSAGES EXEC does the same for a class, by displaying the contents of a similar file on the minidisk shared by class members. More extensive notifications are available using an EXEC called NEWS, which handles items of interest in the categories of "general," "CSC," and "class." (Other categories can be established if needed.) NEWS displays a menu of available notices from files on a system, the Computer Science, or the class minidisk. It also keeps track of what items a user has been informed of by storing the data of the last invocation of NEWS in a user file named PROFILE DATA, which also contains information about the user required by certain other EXEC's.

When a CMS user logs on, commands in an EXEC called PROFILE are executed if that user has a file of that name on his minidisk. (Similar features on other systems are common.) This PROFILE helps serve the functions both of setting up the virtual machine configuration required by a member of a particular group, and of communicating necessary information to those who need it. All users are encouraged to place certain commands in PROFILE EXEC to match terminal characteristics with CMS output. In addition, Computer Science students are told to execute NCSCSC and CSCPROF from this file. These commands invoke EXEC's to make the Computer Science minidisk part of the user's virtual machine (NCSCSC) and to perform tasks on a list which has been changing as greater experience with the system is gained (CSCPROF). CSCPROF executes BULLETIN, MESSAGES, and NEWS for information dissemination. (A user who is not a class member has no class-related information displayed thereby, of course.) Also, if the instructor has provided an appropriately named EXEC on the class disk, it too is executed, allowing him to provide a special environment for his class. This feature has proved quite useful.

Communication is not a one-way affair. Users need to be able to report bugs and make suggestions, and students sometimes need to communicate with one another (admittedly, not always for legitimate reasons) or with their teacher. Reports of system problems, user complaints, and suggestions may be submitted to the Computing Center and Computer Science Department through an EXEC called SUGGEST. CMS provides an extensive file-transfer capability, but Computer Science has made available EXEC's called GIVE and TAKE to provide an easier-to-use mail service among users. The ability of an instructor to receive messages from his class can prove very helpful, as in the case where his students discover his elision from his lecture before he does, and GIVE and TAKE provide this service. Of course, the instructor or his agent needs to log on regularly to assure that messages are received.

The faculty concern with communication facilities to the existing System/370 used by NCSU has already been mentioned. As implemented, this has taken the form of a conduit to the batch input stream from CMS. Output from batch runs submitted through this channel return either to the CMS user who originated the job or it can be routed anywhere normally-entered batch jobs may be sent. This use of CMS is the primary one of interest to the Computing Center, though it has been used quite successfully in at least one Computer Science course, a COBOL course which required a compiler not available on CMS. The greatest use of the facility by Computer Science users has been by faculty members who have done most work on CMS, but have used the other system for access to special equipment such as printers with upper- and lowercase characters, for access to files created on the System/370, etc. To facilitate tasks such as these, EXEC's have been created to send files to the System/370 (EXPORT), obtain files from that system (IMPORT), archive files to tape (TOTAPE), and so forth. These EXEC's create the necessary control card images to accomplish their respective tasks. Information needed to assemble the jobs comes from EXEC parameters, from terminal prompts by the EXEC's, and from data (such as batch machine account and password) from the user's PROFILE DATA file.

6. NO ONE SHOULD TEST HIS OWN PROGRAM

For a number of years, the Computer Science Department has, in many of its classes, collected student programs and graded them based on their performance processing data not seen by the student in advance. The author and others have made a case for this procedure elsewhere [1,3,4]. Among the advantages of this practice is the ability to run programs many times against different data, to rerun programs to test for initially unanticipated errors, and even to rerun a specific program modified by the instructor to verify that a program error has been identified correctly. The student being tested in such a way learns to write programs to specifications and is forced to test his programs more thoroughly than he might otherwise.

Collecting programs also discourages and helps detect certain forms of cheating. On any system where a student can edit his output (CMS among them), the instructor cannot be sure that output he receives from a student is really output from the program which purportedly produced it. The more evaluation relies on examination of program output, the more serious this uncertainty becomes. Generally, "old fashioned" batch systems made cheating through manufacture of bogus output or editing of actual output uneconomical for the potential offender. An interactive system can make this technique one of the simplest ways of "correcting" minor bugs, however. Finally, if the instructor has all student programs available to him in machine-readable form, automated checks for plagiarism can be run (see, for example, [4] and [6]), and the programs can be archived for future reference and comparison should collusion be discovered among a group of students on a later assignment.

Programs TODISK and WATLOAD formed the core of the prototype program-collection system used by Computer Science at NCSU [1]. This system was succeeded by a more general facility called the TODISKG/RUNCLASS system, batch-oriented software upon which a corresponding CMS capability has been built. The heart of the NCSU CMS program-collection facility is a special virtual machine, CTODISKG, which runs "disconnected" at all times. (The CTODISKG machine is an interrupt-driven program which does nothing unless service is explicitly required of it.) When the student is ready to turn in his program for grading, he types a line such as

```
TODISKG MYPROG PASCAL A (COURSE 102 SECTION 2 PA 3
```

This particular entry, for example, requests that file MYPROG PASCAL A be submitted in fulfillment of programming assignment 3 for section 2 of CSC 102. After CTODISKG processes his request, usually in a matter of seconds, the student receives a message to the effect that his program has been received. If this is not the first submission by the student for assignment 3, the message indicates that the file just sent replaces an existing file sent earlier. As files are received by CTODISKG, they are time-stamped and stored on CTODISKG's own private minidisk. The time stamp provided by CTODISKG allows the instructor complete flexibility in setting deadlines for turning in assignments. In practice, due times are at least as likely to be established at night or in the wee hours of the morning as during the day. Students can submit programs only on their own behalf (CMS prevents misrepresentation of identity by the sender) and only for courses for which they have been authorized. (See below.) Note that CTODISKG provides effective general-purpose one-way communication from student to instructor, and it has been used for messages as well as programs. At least one instructor has considered using it as a general work-submission facility for homework, program specifications, etc. The availability of a text-processing program (Script) on the NCSU CMS system makes this especially attractive to students and not unattractive to instructors.

Authorization to use CTODISKG by student and instructor alike must be explicit. The instructor requests the service of the departmental computing coordinator (as the system is maintained by Computer Science). The coordinator then places the instructor's userid, course, and section in an authorization table used by CTODISKG. As for other functions, the instructor is then responsible for maintaining the list of authorized students in his class. This is simply done using an EXEC and the userid list from the Computing Center, as for the grading program. The instructor types a command such as

```
MAKEROLL COURSE 102 SECTION 2
```

to send CTODISKG the student roll. Other EXEC's allow the instructor to inquire as to which students have already submitted programs or to request that all student submissions be sent to his virtual machine. Normally, CTODISKG only sends programs not sent previously, so that late programs may be processed without interference by previously graded assignments, although this feature may be overridden when necessary. The instructor may also request that programs for a certain assignment stored on CTODISKG's disk be deleted.

Once the instructor has his students' programs in hand, he must run them for grading. For this, several EXEC's have been written for creating batch jobs either for CMS using the CMSBATCH facility or for the System/370. In the latter case, output may be printed or examined through the facilities of either computing system at the discretion of the instructor.

7. MISSING LINKS

The NCSU CMS system provides quite a lot of assistance to the Computer Science instructor. He may easily add or delete student accounts, has good two-way communication with his students, including the ability to collect and run programs from his students with little effort on his part, can maintain his grade records easily, and has convenient access to another computer system.

Some features that might be useful have not been provided. The collection of programs by the instructor allows for the examination of those programs by some form of plagiarism checker. Such a feature was in fact planned in the prototype system TODISK [1], but it was never implemented. It is unclear just how useful such tools can be, particularly in courses after the first programming course. The fact that NCSU CMS is not now being used for the first programming course somewhat discourages implementation of such a program at this time.

A more significant omission may be the unavailability of statistical information about student use of the machine. At present, no meaningful summaries of account usage by his students is provided instructors either on- or off-line. In some cases, this information can be very helpful in identifying students who are not working hard enough, those working too hard (fulfilling assignments with great difficulty or performing other tasks on the computer which may or may not contribute to their computer science education), or simply working in a counterproductive way. Such information has been helpful in the past at NCSU in confirming cases of plagiarism by showing that a student did insufficient work on the computer to explain his program. Although statistical summaries of number of logons, CPU time used, and the like are more often than not used in a crude, *ad hoc* way, some experiments have attempted to make more meaningful such measurements. For example, Hsia and Petry report on a pedagogically-oriented program-development environment which constrains the student to work in a prescribed, systematic way [5]. The system not only encourages good work habits, but also collects data on the student's progress, which could then be made available to the instructor. Systems such as this one certainly bear further investigation. They do have the disadvantage of requiring extensive development time themselves and necessarily incorporate some particular design philosophies, though perhaps they can be script-driven to the extent that changes in pedagogy can be incorporated easily into them.

The Computer Science Department has made no attempt to provide any automatic checking of student program output. The department has placed great emphasis in recent years on program style, documentation, and structure [2], so that checking output for "correctness" requires relatively little of the total time spent grading programs. Instructors do not seem to find the omission a hardship.

One item on the original NCSU CMS project list which has not yet been implemented is a public assignment calendar. Academic computing facilities are subject to severe demand peaks caused by the occurrence, and particularly the simultaneous occurrence, of class project deadlines. Although students quickly learn that response time and terminal availability may be degraded as an assignment deadline approaches, they cannot easily anticipate the slowdowns caused by assignments in other courses. If an updated schedule of due dates for assignments in all classes were available on line, students could better manage their time. As an added benefit, the demand peaks imposed on the system could be reduced thereby. Appropriate software to implement this idea will probably be developed as time permits.

8. SECOND THOUGHTS

NCSU CMS is new and experience with it is limited. (The first regular Computer Science classes began using it in January 1981.) It is clear, however, that the planning effort expended by the department has paid off. Instructors do indeed have a powerful tool at their command, and some of them are beginning to understand its real potential. Actually, the greatest problem which has been encountered has been the difficulty of providing adequate faculty training. Handouts have been prepared explaining functions of interest to instructors, and one workshop has been held. Unfortunately, CMS must be learned before the use of any software on it can be, and the learning process takes time. On-line help is available for all EXEC's and there are a few people around who can answer critical questions when emergencies arise. Instructors have not always known just what has been available to them, however, and thus have not always known the right questions to ask. These problems should diminish as experience is gained with the system. Meanwhile, some amount of "advertising" may be in order to promote intelligent use of the facilities which have been provided.

The feature most in need of enhancement is the ability to generate batch jobs for each student, in order to evaluate programming assignments. The initial software devised for the purpose relied on the IBM-supplied pseudobatch facility CMSBATCH, which has proved to be quite unsatisfactory. Other CMS installations have replaced CMSBATCH with locally-devised software, and NCSU may eventually do the same. Meanwhile, the link to the System/370 has been utilized as the most reliable channel to a batch processor, and an EXEC named RUNCLASS has provided an adequate means to utilize this option. Unfortunately, RUNCLASS requires the instructor to generate a certain amount of control card information in the form of IBM JCL statements, a task universally disliked. This method may be replaced in the future by a more interactive mode of operation which queries the instructor about his needs and constructs JCL statements for him.

GRADE75 must be replaced eventually. (One student project is now attempting to provide such a replacement.) The current program is batch-oriented and does not allow the instructor the capability of specifying different grading schemes and immediately seeing their effects. Nor does the present implementation allow students to see their grades unless the grades are posted on a bulletin board. Why not simply have the grades on the class disk and avoid the printing process altogether?

The student-teacher communication channels which have been made available constitute one of the more exciting features of NCSU CMS, and instructors are devising novel uses for them. Some homework has been collected this way, for example. More interesting has been the alternative of placing example programs on the class minidisk rather than typing them and handing them out. Not only is this usually easier, but students can run and even modify the resulting "handout."

An interesting variation on the class account structure involving a shared minidisk has been the use of such a class by the teacher and lab instructors teaching sections of CSC 101, the introductory programming course at NCSU. All the lab instructors have been given read/write access to the class minidisk and update separate grade files for their respective sections. A simple EXEC is then used to collect these files and run GRADE75 for the entire class. MESSAGES and NEWS are used by these people as an electronic mail service, a particularly helpful service for the teacher's contacting the lab instructors, who are part-time workers and full-time students.

The class structure used by Computer Science has been quite effective, though perhaps it is not the ideal arrangement. Limited resources have prevented even consideration of one other scheme--giving each Computer Science major his own account as a freshman and allowing him to keep it until he graduates. Class affinity would then be defined as it is for CTODISKG--with an explicit authorization file. This idea is extremely attractive because students would have more incentive to gain facility with a system seen as a general-purpose tool available to them for four years.

In this same vein, it has been gratifying to see the enthusiasm with which students have taken to using CMS. This seems to be due to a number of factors, not the least of which is the new system syndrome. Beyond this, however, CMS's EXEC2 command language, which is quite powerful, has encouraged student experimentation. Students have also enjoyed feeling closer to the "action"--the people in charge of their machine are not across town or even across campus, but in fact just down the hall in the Computer Science Department. This has been of benefit to student and department alike. Students have accumulated valuable experience writing EXEC's for CMS, and much of the Computer Science software now in use has been contributed voluntarily by enthusiastic students.

Undoubtedly, much remains to be done and redone to make NCSU CMS a convenient tool for Computer Science instructors. The effort is clearly worthwhile, however, and it is recommended that faculties elsewhere consider whether they could benefit from similar support on their interactive systems. Besides, we may better prepare students to provide society with the powerful, labor-saving computer tools it deserves if we show them that we know how to provide such tools for ourselves.

ACKNOWLEDGEMENTS

It is impossible to list all those people who have had a hand in making NCSU CMS what it is today. The efforts of Computer Science faculty and especially Computer Science students have been invaluable. The Computing Center staff has been very cooperative also, and our debt to members of the Triangle Universities Computation Center staff who worked on a short-lived CMS implementation at their facility is also substantial. To former student Kurt Deitrick, however, must go a special note of appreciation for his efforts directed

toward developing the basic structure of much of the NCSU system, as well as for providing much of the implementation effort.

REFERENCES

1. Deimel, L. E. and Clarkson, B. A. The Todisk-Watload System: a convenient tool for evaluating student programs. Proc. 16th Annual SE Regional ACM Conference, April 1978, pp. 168-171.

2. Deimel, L. E. and Pozefsky, M. Implementation of programming standards in a computer science department. Proc. 17th Annual SE Regional ACM Conference, April 1979, pp. 142-143.

3. Deimel, L. E. and Pozefsky, M. Requirements for student programs in the undergraduate computer science curriculum: How much is enough? *SIGCSE Bulletin 11*, 1 (February 1979), pp. 14-17.

4. Grier, S. A tool that detects plagiarism in Pascal programs. *SIGCSE Bulletin 13*, 1 (February 1981), pp. 15-20.

5. Hsia, P. and Petry, F. E. A systematic approach to interactive programming. *Computer 13*, 6 (June 1980), pp. 27-34.

6. Robinson, S. S. and Soffa, M. L. An instructional aid for student programs. *SIGCSE Bulletin 12*, 1 (February 1980), pp. 118-129.