VIRGINIA POLYTECHNIC INSTITUTE

AND

STATE UNIVERSITY

DEVELOPING CURRICULUM FOR USE IN AN INTERACTIVE COMPUTATIONAL ENVIRONMENT

by

John Heafner, Computing Center

John A. N. Lee, Department of Computer Science

Sharon A. Shrock, Learning Resources Center

Report No. CS 79004-E

September 17, 1979

Abstract

The advent of the interactive computer classroom and laboratory on the campus leads naturally to the increased usage of the computational facilities in undergraduate education. This report reviews a number of alternative plans for the integration of these facilities into various curricula, and provides outlines for the development of course syllabi which are appropriate.

Introduction

Referring back to 1955 when IBM introduced the 704 computer, John Backus, the originator of the FORTRAN language said: "... the effect [of] ... speeding up operations by a factor of ten ... left inefficiencies nowhere to hide." [1] So it is in attempting to use interactive classroom facilities to supplement the course offerings in any class; there is no room for the inefficiencies to hide. As most computer users are aware, computer systems tend to be very unforgiving, and the natural tendency of a teacher to "wave off" a difficult point to be considered at some later point in time is thwarted by the uncompromising nature of the machine. This

---

[1] Backus, John, "The History of FORTRAN I, II and III", Preprints of the Conference on the History of Programming Languages, ACM SIGPLAN Notices, August, 1978.

really is not much different from other situations in which teachers utilize mechanical devices to supplement their presentations. In fact, if one regards the computer as merely another experimental device, then teaching in this environment is not so much different from that of the science teacher conducting "real time" laboratory experiments.

The purpose of this booklet is to examine some of the opportunities which the interactive laboratories and classrooms provide, and to lay a groundwork for faculty to prepare for their effective use of these facilities.

Table of Contents

Interactive Computing

The "Encyclopedia of Computer Science" [2] defines Interactive Computing as a mode of processing which supports programmers who wish to develop programs in real time , correcting errors as soon as the latter are detected by the computer. Interactive Computing is often combined with the Time Sharing mode of operation so that several users can (apparently) simultaneously use the computer system.

Besides providing a tool for programmers, there is a tremendous potential to be attained in the use of interactive computation facilities in an educational environment. Most obviously is the direct use of the computer for some form of computer aided, directed, managed or augmented system whereby the actual learning experiences are not directly gained from an instructor. On a much lower level, the access to an interactive system can be considered as simply another tool to be used by students in their out-of-class activities, much in the same manner as some classes utilize a batch computational facility. Somewhere in between these two extremes are many styles of usage of the interactive computer classrooms, including not only laboratory-style learning activities, but also enhanced lectures.

The fundamental difference between a batch system and an

---

[2] Ralston, A. and Meek, C.L., (Eds), Encyclopedia of Computer Science, Petrocelli/Charter, New York, 1976.

interactive system lies in the communication between the system and the user. A common misconception is that interactive processing is accomplished as soon as terminals are added to a system. However, only when the proper software is also installed so that the user can interact with the processing during the processing cycle, is the system truly interactive. Of fundamental concern to the interactive user is the response time of the system. It has been found that delays in response of greater than 5 seconds are intolerable to many users who are used to what is called "conversational mode" computing. For others who are familiar with batch processing, delays of 10 seconds are perfectly acceptable. However, where a student is being presented with material to learn, certain delays can distract him from the task. Better interactive systems have delays which are no more than "conversational pauses", which are generally regarded by the television industry as 2 to 3 seconds.

The pauses inherent in an interactive/time sharing system are the result of both hardware and software factors, and the "friendliness" of the system is almost totally under the control of the software designer. In the case of educational systems, the software designer includes the faculty member who is designing a learning experience for his students. In this report we include a number of checklists which we hope will assist the user of interactive systems to have the system respond to the needs of the user

without intolerable delays or offensive responses.

As in so many situations, the commercial implementation of interaction is a step behind where the original interactive systems stood. For example, the early Dartmouth Time Sharing System (DTSS) provided a BASIC environment where each BASIC language statement was syntax checked on entry. Thus a student knew as soon as he had finished typing the line that he had not made any syntax errors. In 1964, this same system provided direct interaction with a running program so that if an error was found during execution, the system did not simply abort the run but instead suspended execution, saving the state of the machine before the offending statement was executed, and offered the user the option of correcting the statement there and then. These corrections were not substituted into the program immediately but were saved for the user to substitute on command. Unfortunately such highly interactive features have been lost in the last 15 years.

The interactive environment provides the user with the ability to interrogate the status of execution of processors in a language which is at his level, not the machine's. Thus octal dumps are replaced by listings of the contents associated with the various identifiers used in the program, expressed in the symbolic terms used by the programmer and in a number representation scheme which is specified in declaration statements in that program.

There is a danger in using interactive programming as a

tool in program development in that there is a tendency to rush into trying things without thinking ahead. Using our notion of the similarity between the interactive computer laboratory environment and that of the science laboratory, it is fair to say that the scientist rarely tries something just to see what happens; instead he predicts the result and then verifies his prediction or explains the differences if any. Dartmouth College interactive laboratory has a large sign at the front which says:

SITTING AT A TERMINAL IS NO SUBSTITUTE FOR THINKING

and we might add that

USING AN INTERACTIVE LABORATORY IS NO SUBSTITUTE FOR TEACHING

In a batch environment the user provides the complete set of processors and data, prays that they are compatible and launches into a period of uncertainty which can only have one of two responses; it works or it doesn't. The process of program or system development in a batch system tends to be to develop large chunks of program and to test these simultaneously so as to get out all the bugs. In the interactive environment it is preferable to develop the same systems piecemeal, to validate each segment separately and then to combine the segments into a whole only when the pieces are ready. This is possible because of the apparent fast turn-around which accompanies interactive processing. The user does not have to wait for considerable periods

between development steps.

Different users will have different strategies of development which may include dividing the problem into distinct separate modules which will each be developed separately, or to design an incremental system of development wherein each stage builds on the previously validated system.

Batch vs Interactive Computing

The transfer of systems from the batch environment to the interactive system requires the addition of many new features which were not either necessary or feasible before. The fact that the system is interactive means that the majority of input will be from the terminal keyboard. Whereas in the batch system the incorrect input of data is grounds for the fatal termination of the program, in an interactive system the recognition of such an error is merely a cue for the program to request the "proper" data. The language processing systems within the interactive system will provide normal error activities, but in most cases will regard "out of range" data as a cause for the termination of execution. Thus the developer of an interactive program must provide his own input validation routines, testing for data in the appropriate range (which may be quite different from that which is valid for the machine) and for requesting a re-input of the data in the correct form. This latter action may require the user to re-input a whole line or to merely correct a single item. There are significant advantages to be gained for "one-at-a-time" input where the user supplies only one item in response to each request (suitably output to prompt him) so that the checking can be accomplished effectively. The input procedure should also permit the user to enter non-valid but special data to signal some special action which

he desires. For example, after entering a set of data the user may decide to abort the processing of the rest of the program since he has already recognized some special feature which invalidates further computation. However, if the normal procedures which request re-input of invalid data have been incorporated the user may find himself in a endless loop from which he cannot escape except by hanging up the phone or switching off the terminal. Thus an escape of recognizing the key-word "quit" is a necessity. In some systems, the use of the "break" key can be similarly programmed.

Output to an interactive terminal needs special attention also. On one hand the slow speed of a 30 characters per second terminal can usefully be invoked to place material in front of a student at about normal reading speed. However, the same program operating on a 200 characters per second machine is totally useless; the material passes from sight so fast that the reading time is lost. Similarly, using a small video-screen, the amount of useful information that can be displayed is about 20 lines, anything more than that being scrolled before the recipient has a chance to do anything with it. In an interactive system, output should be kept to a minimum. Where possible an interactive system should only output key data and then store other generated information in a location where the user can access it on demand. For example, it would be far better to plot a graph of a function and store the discrete data points for

interactive investigation by the user, than to overwhelm him with too much data.

Interactive Program Development Checklist

INPUT CONSIDERATIONS

- O Input amount is sufficiently small to be input from a keyboard
- O Separate input items into one-at-a-time requests
- O Add prompting statements before each input and give required format where absolutely necessary (free form input is better)
- O Validate input after each request and repeat until OK or until it is obvious that the user doesn't understand the question
- O Add an escape to each input which will permit user to abort the run
- O Provide an alternative response which is equivalent to the user asking for more explanation of the request -- such as "help" or "?"
- O Consider whether there is a default response which could be used if the user doesn't care -- identified perhaps by a response of simply "return"

- Is any input dependent on intermediate results? If so ensure output of intermediate results before requesting input.

OUTPUT CONSIDERATIONS

- Insert initial output statements to announce the processor and confirm correct version
- Reduce output to a minimum
- Save additional output in file for user inspection
- Provide routines for inspection of additional output or give sufficient documentation to enable user to use an editor to examine the remainder of the output
- Make sure output will fit onto one "frame" of the terminal
- Check line length of typical terminal to prevent "folding" of output
- Prettyprint

HELP FILES

- Provide a separate help file for the user to view before using system
- Break help file into sections so that user does not have to view whole file
- Give section listing in first section

- Provide a file for users to report problems and for others to check on "known" errors

INSTALLING THE PROCESSOR

- Set access rights to appropriate subset (probably read and execute only)
- Give access to appropriate subset of users
- Protect source and listing from unauthorized access
- Validate that version identification in source and output are same
- Use optimizing compiler option for production version

POTENTIAL LANGUAGE CONVERSION PROBLEMS

- Programming Language implementation is compatible with previous system (preferably a standard language)
- Word length change does not invalidate results
- System provided subroutines function as before
- Rounding and truncation do not affect expected results
- Logical unit numbers and names match
- Range of loops unaffected
- Length of identifier names satisfactory
- (in PL/I) name of segment and name of procedure are same

- (in PL/I) eliminate "options main"
- Give segment name the suffix of the processor to be used
- Check correct use of quotes (") and apostrophes(')

Modes of Usage

Out of Class Assignments

Prior to the development and implementation of the Interactive Classroom system, the majority of undergraduate computer usage was centered around the Remote Job Entry (RJE) stations on campus. Class assignments were then given to students to execute on the system in the batch environment. This was accomplished in two steps; firstly the teacher, in class, and totally divorced from the real time operation of the system, introduced sufficient skills in machine operation and (perhaps) programming, to provide a foundation on which to build the assignments. It was then up to the student to deliver to the RJE station the appropriate data in order to solve his out-of-class assignment. Being a batch environment, any errors in either the data or the program were reported back to the student "en masse" and it was his problem to make the corrections off-line to obtain a valid run. In all of this the instructor was divorced from the actual operation of the system, except to answer consulting questions.

This same process can be carried on with the Interactive Laboratories wherein the instructor can give the necesary instructions in class, off-line, and leave all the actual processing to the "spare-time" of the student. However, one important question needs to be asked before this transfer

from the batch oriented system to the interactive system is contemplated:

> "What can be accomplished additionally using the interactive system that could not have been accomplished in the batch environment?"

For example, if the processing to be accomplished is the mere tedious repetitive operations over previously prepared data with no options on either the method of execution or the selection of some algorithmic process, then there is no advantage to interactive processing. On the other hand, if the assignment is to use the prepared data as a data base for the student to experiment over in order that he might discover the most appropriate tools for this particular experiment, then the interactive system may be most appropriate.

From an administrative piont of view, it is imperitive that some consideration be given of the cost differential between batch and interactive usage. In general, batch processing is based on CPU usage whereas terminal usage is based (in the main) on connect time. Further, the availability of a University provided RJE system may not be offset by department provided terminals.

Electronic Submission of Assignments

Through the use of the communications devices in the interactive system, students can be given the opportunity to deliver their assignments to the instructor electronically. Provided that the instructor sets up conventions for naming programs and other documents which will permit him to differentiate between submissions, then this system will be highly effective. Since the interactive system records the date and time of each message which it relays through the system, the submission of assignments on time can easily be verified. This also means that the faculty member does not have to physically collect assignments.

Through the use of "compose" or "script" (depending on the system used) students in any class can submit themes for grading. Similarly, where the system permits the user to automatically record (or "log") his terminal session, the student in a laboratory exercise can submit exactly what happened during his experiment. In the same manner, if the student has problems and wishes to ask a question through the mail system, the courses consultant can be provided with an exact copy of all the actions the student took prior to getting the problem for which help is needed.

BEWARE however that assignments do not pile up to such an extent that the storage system is overtaxed. Remember that a submission is a copy of a file or segment and uses additional storage. Delete all files after grading.

Out of Class Assignments Checklist

- O Objectives of assignment can be met with the use of the computer
- O Students have sufficient experience to complete assignment
- O Assignment statement clearly describes any necessary system segments to be used
- O Copy of assignment exists as an accessible file on the system
- O Where necessary, limited subset of commands has either been implemented or expanded
- O Help segments on the assignment have been prepared and will be updated as necessary
- O Establish a mail box for student enquiries
- O Establish a consultant's communication system
- O If electronic submission is to be used, rules regarding program/segment naming have been established
- O Grading system for assignments has been established and has been transmitted to the students

In Class Teaching and Experimentation

The use of the interactive classroom is probably the most difficult to envisage. The proper inter-relation between the teacher and the system is difficult to attain, the coordination of teacher and machine being difficult to control. From prior experience we know that the best use of the system (from a system point of view) is when the students are all busy using the terminals. However, the teacher needs some time to explain what to do, even if he has conducted a preparatory period. The natural reaction of the students, sitting at a terminal, is to play. At this point the teacher loses the primary battle of "getting their attention." Thus the actual presentation of new material by a teacher in this environment is difficult. At its best the teacher comes off second best as merely a "suggester" of things to do or a consultant when that which was expected does not work. One can only conclude then that in this environment the best teacher is the one that has the attention - the terminal.

In the early stages of using the interactive system, the teacher has no option but to attempt to compete with the system so as to enable the students to gain experience in using the system. Ideally, this initial experience will be gained quickly and universally by all students on campus so that this initial "start-up" will not be necessary in every class.

Without initiating a complete Computer Aided Instruction (CAI) system, which is the subject of the next section, classroom environments can be developed in which the learning experiences can be controlled by the teacher but which are provided by the system. One such scenario might be to provide a two class sequence in which the first class is spent in the interactive laboratory and in which each student, or group of students, experiment over a fixed data base (read-only) with various packages. In the next class period then they can compare their results outside of the terminal environment. Conversely, the first class period may be effectively used to design a set of experiments for each group to accomplish to achieve similar goals.

The more passive use of the system in a teaching environment is possible where the faculty member has prepared a set of notes which are displayed on the screen as he progresses through the lecture. These serve both as a guide to his presentation but also provide visual aids to the students at the appropriate time. In such an system the keyboards of the terminals are inoperative until released by the instructor for appropriate input.

The use of the terminal system for the simulation of other interactive environments can be highly beneficial. Typically such systems are used in the simulation of business activities through the use of a myriad of business games where each group makes decisions on a timely basis. The results of these decisions then influence a global

environment and result in the modification of the status of each groups "holdings." The development of such systems can be extended to many other decision making environments besides business, though the algorithms for such games are best known in that field.

Considerable success has been achieved by establishing the class period as a laboratory experience where the problem has been carefully laid out in advance. For example, in a programming class it is a good learning experience for students to read programs that are well constructed. Using this tenet one classroom exercise is to give the student a good example of a program which is accessible in a file (or segment) and to make some extensions to that program. This requires both an understanding of the language and an ability to understand the underlying algorithm which the program expresses. Once a grounding in reading good programs has been established it is then possible to give the student a poor program and for him to re-write it, or an incorrect program to debug. With experience the instructor can construct programs which contain "progressive" bugs; that is, bugs which only reveal themselves as each prior one is resolved.

Sample Classroom Learning Assignment

DEPARTMENT OF COMPUTER SCIENCE CS4Ø52 WINTER 1979

Objective: to write a PASCAL program to investigate the declaration attributes of arrays.

Using the framework of a PASCAL program which is stored under the project directory as "pascal_frame", write a program which will declare an array to be two-dimensional with the first subscript varying from 5 to 15 and the second from 1 to 1Ø. Fill each element of the array with the value computed as the sum of its subscripts. Print out the contents of the array. Use all three versions of the loop control statements - do while, repeat until and for.

Problems:

If a type had been declared as:

```
type it = array[1..1Ø] of integer;
```

what would be the effect of the statement

```
var that: array[5..15] of it; ?
```

What is the difference between the above pair of declarations and the following:

```
var that: array[5..15,1..1Ø] of integer; ?
```

Show by two programs, the differences (or similarities) between these statements.

Using the same programs as above, investigate the differences between write and writeln.

What is the maximum size of any output line on your terminal?

What is the error message which is produced when an invalid subscript is used (i.e. a subscript out of range)?

What is the significance of the dumped data after an execution time error?

I want to use an array (two dimensional) which has integers in one dimension and reals in the other. How would I do this? Show by a program that you can do this!

Checklist for Classroom Activity Planning

- O Objectives of classroom period are established
- O Segments or datasets for students to use have been prepared and the appropriate access set
- O Progression of activities is logical and match the subobjectives of the class
- O Dummy segments for each stage of the classroom progression exist for those students who do not have time or ability to complete prior activity
- O Submission arrangements have been established
- O Classroom instructor has terminal for his exclusive use and can access it easily from front of room
- O Classroom is scheduled
- O Plan B is ready in case system is inoperative at this time period

Computer Aided Instruction (CAI)

[3] Multics supports a system for the assistance in writing interactive programs which was developed by Roy Kaplow at M.I.T. named TICS (Teacher-Interactive Computer System) which is aimed at facilitating the authoring of [general] programs. The system includes particular features for creating instructional software, and in that application it is intended for direct use by teachers or other persons whose expertise lies in the subject matter being addressed, but not necessarily in computer programming. TICS is implemented in two components: an author system and a delivery system. The former provides the tools for writing, investigating, editing, and trying out programs. The latter provides a special environment for student use of the programs. The author can work on his description of a program through an on-line terminal. He can allow "try out" access to others who then can use, but not alter, the program, while it is still being developed and even if it contains structural errors! An author may work on as many programs as he chooses and, conversely, it can be arranged that any number of authors can work on one program.

When a program is finished, the author initiates a transformation process (analogous to compilation), which is

---

[3] This review is taken from: Kaplow,R. et al, Computer Assistance for Writing Interactive Programs: TICS, Proc. ACM Mtg., August 1973, pp 309-315.

called compression. That process comprises the various steps of ordering the data base, searching for structural errors, extracting only those elements which are needed for the execution-time description of the program, and formatting them into a highly coded, compressed form. This is used in the delivery system.

The commands for creating the program itself are probably the simplest part of the language; the following sequence is intended simply to give its flavor and is shown without the shorthand conventions that comprise the true [actual] language.

```
set the working node "name_1"

ask "What is the date of Washington's Birthday?"

if response = "february 22"
        then print "That's right!"
             and go to node "name_2".

if response = "february 19"
   and response (in node 17) = "old enough"
       then do flag = 1 + flag
            and print "Yes, in America they will even change
                       a President's birthday to make it
                       a long weekend."
            and go to node "name_3".

if response = "february 12"
   and variable > 2.3
       then call subroutine_name(flag, variable)
            and hint "That's some other President's
                      name. Please try again.".
```

As a computer aided instruction system, TICS can provide a learning environment which the author can vary according to the needs of the students, can provide automatic grading and response, and will permit the student to progress at any

rate or a rate determined by the author.

Much of the power of TICS is left for the teacher to invoke and to use ingeniously. A limited set of pre-programmed packages are available and it is anticipated that cooperation between faculty with similar problems but differing environments can lead to some universally useful systems.

Common or Garden Uses

The everyday usage of the interactive classrooms by students who may not necessarily be registered for a course which encompasses computer usage is now possible. This may involve such simple activities as the use of the system as a large calculator for those who don't have a pocket calculator, or the preparation of themes for a writing assignment providing thereby simple editing facilities as are provided in many business offices today.

One of these typical uses is the editing facilities of the computer and the ability to format the output in various ways. Included in such formatting systems can be the specialized requirements for (say) theses. Text editing systems are becoming more and more common in not only newspaper offices but also as aids for secretaries and writers. The two systems, script (for IBM systems) and compose (for Multics), use similar formatting specifications (such as for centering titles, indenting sections, paragraph

indents, etc.) and thus are a good starting point for students to learn the two stage process of programming: specification and execution.

Other Uses Checklist

- O As a calculator
- O As a Text Editor
- O As a Text Formatter

The Systematic Design of Instruction

Effective instruction requires careful planning. While this is true of instruction in any setting, even more thorough planning is required when interactive computing is a part of the instructional strategy. Computers, when used appropriately, are capable of delivering subtle, yet complex instruction -- teaching in a manner that no human instructor could, even in a one-on-one situation with a student. The computer has infinite patience, and is not likely to be irritated by repetitive responses. However, unlike the human teacher who can modify behavior to better respond to the student, the computer can only respond to anticipated student actions and the student's ability to correctly operate the interactive terminal.

There are (at least) three ways in which instruction which is to be delivered via interactive computing can fail:

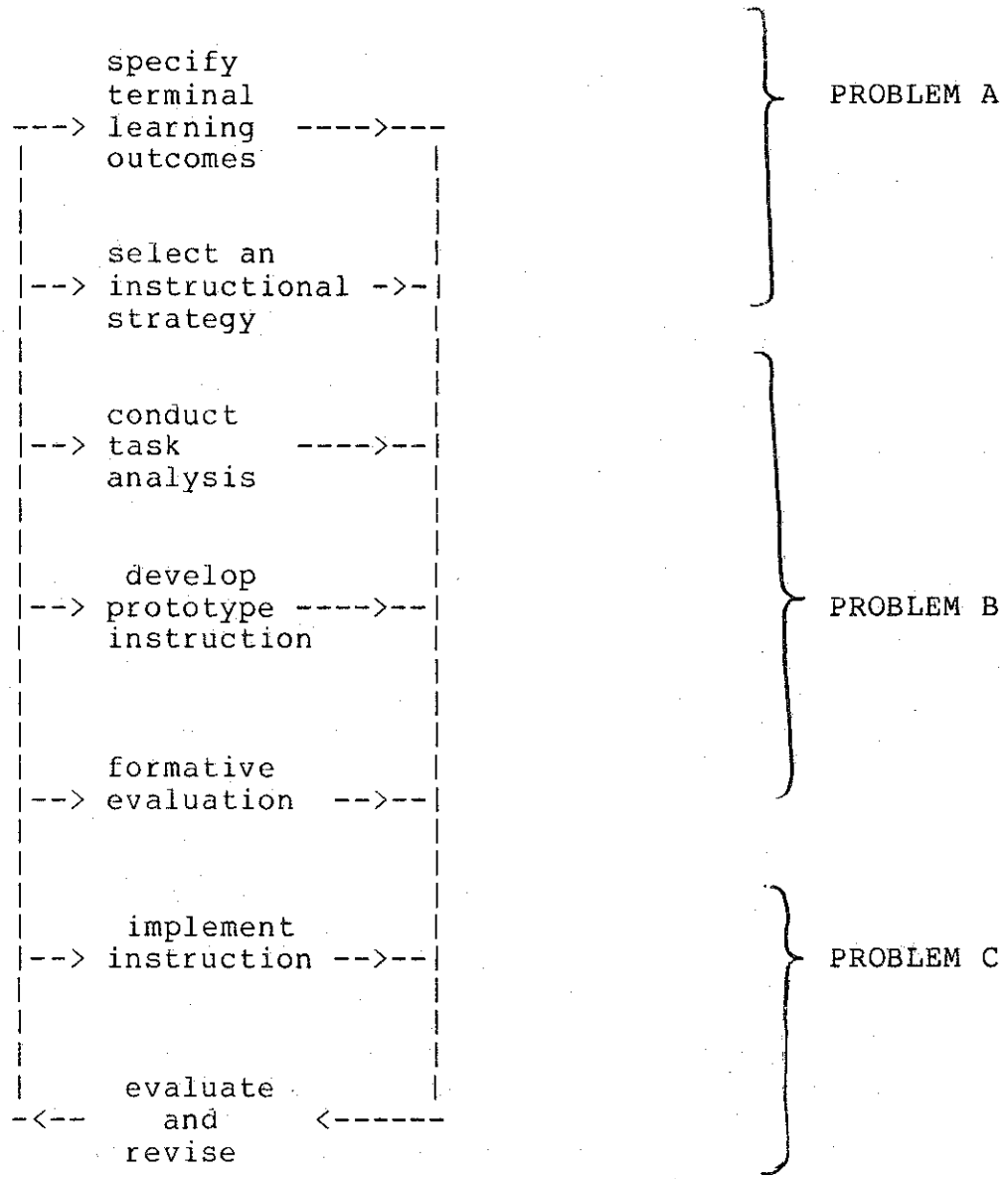
A. The instruction can be addressed to inappropriate objectives, that is, its potential can be squandered toward instructional ends that could be more effectively met by some other method.

B. The instruction can be poorly organized or be otherwise confusing -- puzzling and then frustrating the student with its inability to provide further clarification.

C. Students can be so poorly informed about the terminal's essential operating procedures or the logistics of integrating computerized instruction into the course that they cannot access the instruction properly in the first place.

The best safeguard against all three of these problems is

the systematic design of instruction. Below is a simple diagram depicting the process and indicating how the various stages of the design procedure relate to the three problems described above.



Specify terminal learning outcomes

It is important to note the first step in the design procedure, that is specifying the terminal learning outcomes. Knowing what it is that you want students to be able to do as a result of instruction precedes deciding how you are going to teach them to do it -- which is step 2. Interactive computing is one alternative amongst many in the set of instructional strategies. In general, quality instruction is in jeopardy when the instructional strategy -- in this case interactive computing -- is selected before one has a very clear statement of the desired results of instruction. Failing to have this information before selecting a strategy is very much like adopting a solution before identifying the problem. Poorly delivered computerized instruction often results from an instructor's using the medium in spite of intended learning outcomes that suggest another strategy.

Therefore, the suggestion here is to be disciplined about writing instructional objectives for your course. There have been many books written about how to write instructional objectives; some of these are quite helpful and are referenced at the end of this manual. Regardless of whose rules you might use, there are two recommendations upon which virtually everyone agrees.

1. Write down your intended learning objectives (outcomes). You are probably deluding yourself if you think that you "have them well enough in mind". Having course outcomes in writing will make later creation of the instruction and evaluation of the lesson much

easier.

2. Write your intended learning outcomes in terms of the students' behavior. If you write outcomes in terms of what you or the computer is going to do, you remove your attention from the only point at which effectiveness can be determined -- that is, the students' behavior.

The objectives related to any segment of an instructional program can be developed by completing the following outline:

The purpose of this segment is to:

_____

_____

_____

As a result of this segment, the student will be able to:

_____

_____

_____

Select an instructional strategy

After you have determined the major intended learning outcomes of your course, you are in a position to decide whether interactive computing is an appropriate instructional device. What kinds of objectives lend themselves to instruction by interactive computation?

For a moment, think of computerized instruction as a stimulus-response-feedback process. The system provides a stimulus to which the student responds. The system then provides the student with feedback regarding the appropriateness of the response or the effect of that response upon other data or variables. Those objectives

best suited to computer aided instruction are those whose attainment requires one or both of the following elements:

1. Very frequent stimulus-response-feedback cycles

The objectives which require these cycles tend to be the development of lower level cognitive skills such as remembering or applying rules upon which the system can provide drill and practice. A student practicing at a terminal is provided with many more opportunities to respond than is a student in the typical classroom situation where responses are requested actively only once or twice per class period. Examples of these lower level objectives are:

Given an English word, student will be able to spell correctly its Spanish equivalent.

Given a bank of data, the student will be able to compute the mean, mode, median and standard deviation of that data.

2. Very complex, individualized feedback.

The objectives which require this feedback tend to be the development of higher order cognitive processes such as complex application of a number of principles or analysis of a problem involving several variables. Instructional simulation is a good example of the application of computers toward these higher level objectives. This strategy uses the computer's unique capacity for complex calculation to simulate real world relationships among variables. Working with an instructional simulation allows the student to

manipulate a microsm, viewing directly the effects of one or a number of factors upon other variables. The simulated relationships may be so complicated that a human teacher would not be able to compute them without the assistance of a computer let alone compute them repeatedly for each student as he manipulates the variables in an attempt to discover the principles that link them together. Examples of higher order objectives that suggest the use of computer aided instruction are:

> Given a simulation which depicts the effects of costs, profits, investment, marketing techniques, public relations, and general economic conditions upon the maintenance of a successful business, the student will develop a corporate budget that maximizes the profits wihtout allowing the corporate image to drop below some quantified minimal level.
>
> Given a simulation of a physical process, the student is to investigate the effects of altering certain variables, in much the same way as he would probe a system in a physical laboratory.

Conduct task analysis

Once you have specified the major learning outcomes in terms of the expected student behaviors, you can proceed to break these objectives down into their component sub-objectives -- known by the systems designers as the process of successive refinement. This analysis will assist you in creating the instruction. A good way to do this is to ask yourself "What would a student have to know in order to meet this major objective?" The answer to this question should

result in one or more additional objectives that are subordinate to the first. Each of these sub-objectives can be submitted to the same question and further be broken down until you reach the knowledge level that your students possess upon entering the course.

For example, if it is the purpose of the course to demonstrate the techniques of automatic type setting and pagination, with the development of skills related to the layout of the "printed word", it will first be necessary for the students to have the necessary skills in English composition regarding the use of such things as footnotes, paragraph indentation, etc. before it is meaningful for them to understand the techniques of typography. If a computer is to be used in this demonstration, then the pre-skills must also include knowledge of how to access the system and some basic knowledge of the storage system. The insertion of type-setting instructions into a manuscript also then requires the user to understand the difference between the data to be type-set and the coded instructions which are to be obeyed by the type-setting system.

This type of task analysis is particularly important when planning computer assisted instruction. It is your best guard against the "Type B" error described on page 30 Because the student will be interacting with a machine instead of a person, gaps in knowledge must be anticipated and appropriate support built into the instruction. Otherwise, the student will reach an impasse in progressing

through the lesson. When the task analysis is complete, you should have a comprehensive list of everything -- every student behavior -- that the instruction must teach. Furthermore, a hierarchical task analysis such as the one described above will generally suggest the order in which the knowledge components -- concepts, principles, specific applications of principles, etc. -- should be presented to the students.

Develop prototype instruction

At this point you are ready to write a tentative instructional program.

Formative Evaluation

When you have completed a prototype of the instruction, formative evaluation is appropriate. Evaluation at this stage of instructional design is called "formative" because the results are used to refine the instruction before it is implemented in class. Formative Evaluation requires that you observe closely a small number (say 2 or 3) appropriate students as they work through the instruction. The working environment should simulate the expected "real world" instructional setting as closely as possible. The students must be actively encouraged to verbalize their thinking as they work so that you can note carefully the source of any misunderstandings that develop.

To yield accurate information for improving instruction, formative evaluation must have the following characteristics:

1. The evaluation must be tied closely to the complete set of objectives generated by the task analysis, the subordinate objectives as well as the terminal ones. Attainment of each of these objectives must be assessed if the instruction is to be revised with any precision.

2. The evaluation must involve students who are representative of the population of students who will be using the finished lesson. Never use a graduate student or a colleague as a trial student for formative evaluation; no one should be in that role who might be able to correct deficiencies in the instruction by drawing unknowingly upon advanced knowledge of the subject matter.

After the formal evaluation data has been collected, the instruction should be revised to remove any errors and to add any clarification suggested by the evaluation results. Be prepared for extensive revision. Several formative evaluation-revision cycles may be necessary before the instruction is ready for a final validation and implementation.

You should recognize that formative evaluation is yet another means of addressing problem B above; it is an excellent empirical means of determining the confusing points in an instructional sequence. Furthermore, when conducted with truly representative students, formative evaluation can give you valuable insight into how well the instruction works mechanically, that is, problem C. Formative evaluation is essential to the development of sound computer assisted instruction because it is impossible for the unaided author to anticipate <u>all</u> the ways in which the instruction can be subverted, distorted, misunderstood,

or just plain "not found" by students.

Implement instruction

After the instruction has been revised to correct every problem detected during formative evaluation, the instruction is ready for implementation in its intended setting.

Evaluate and Revise

Evaluation of its effectiveness under the constraints of academic reality is, of course, very important. Even this evaluation is somewhat formative in nature since one should be prepared to revise the instruction based on its results. Therefore, the evaluation must assess all the learning objectives. However, in contrast to the evaluation discussed previously, this evaluation data will be based on the performance of many more students and under relatively independent circumstances.

The recognition of the need to be ready to revise an instructional segment should lead the instructor to recognize the need for documentation of the steps and decisions made in developing the original version of the segment. It is very difficult to revise anything if you don't know why you did it that way in the first place! It is better to have a separately written description of the system than to attempt to reconstruct it each time you recognize the need for revisions.

Because you will not be observing each student during the use of the instruction, unlike during formative evaluation,

the means of data collection must be carefully planned. Fortunately the computer has several features which are particularly useful for this purpose. For example, it is possible to observe the "droppings" of a student in a much more critical manner than in other classroom situations. The instructor can be given the opportunity to "rummage" through the files the student creates and to observe the rough work which undertaken in order to achieve the objectives of the learning activity. By this means the instructor can examine partial results, incomplete assignments and intermediate experiments as well as simply

receiving the final copy of the assignment material.

Checklist for Systematic Design of Instruction

- O Terminal learning outcomes stated in terms of student behavior.
- O Task analysis (specification of sub-objectives) completed.
- O Prototype of instruction "tried out" with a small number of students (formative evaluation).
- O Instruction revised based on results of formative evaluation.
- O Instruction implemented.
- O Effectiveness of implemented instruction evaluated.
- O Instruction revised based upon the students' ability to perform the previously specified learning outcomes.

Evaluation Techniques and Tools

The TICS system has built into it various facilities to enable the author to record several types of data regarding a student's use of the system. These include the verbatim recording of student responses as well as scores on tests, the number of incorrect responses before a correct response is selected, and other pertinent data. The majority of these data items are directly under the control of the author and are not "automatic".

The use of the mail system in Multics permits the student to send both messages and assignments to the teacher and gives the student the ability to correct input (but with the modification of the date sent to the date of the latest update) and protection against misuse by unauthorized users. However, this system is limited to the mailing of segments which do not exceed four thousand words (one length).

Provided each student establishes a mailbox during the initial access to be workspace, facilities are available to broadcast course-wide messages for each student to receive. By this means, updates on assignments, notices of class meetings etc. can be issued to all registered members of a class.

Each month the computing center issues a usage report which provides the faculty member with information on system usage which can be used as a management-by-exception tool. A similar report can be obtained on a daily basis under

Multics by Project Adminstrators.

In general, the recording of data on student activities under a project is the responsibility of the author or faculty member and must be "programmed" just as other aspects of computer assisted instruction.

System features available to assist in evaluation

- (Multics) Project Usage Reports
- (TICS) Records of responses -- verbatim
- (TICS) Cumulative scores on multiple choice tests
- (TICS) Selected data chosen by the author
- (Multics) Verbatim records of terminal sessions

Resource Scheduling and Allocation

Faculty Qualifications

To utilize the system and to administer a Multics account, a faculty member must be familiar with the system and have takem the Project Adminstrator's course.

Room Scheduling

The room scheduling is handled by the registrars office. A faculty member turns in their request to the registrar.

Resource Allocation (Pie Slicing)

The Multics system is set up in a pie slicing algorithm. During the day the classrooms get 75% ofthe available resources.

Computing Center Courses Available

Introduction to Multics:

This course is an introductory level course for Multics. The basic information for getting started with Multics is provided.

Project Administration:

This course provides faculty and graduate students information concerning the administration of a project. The Project Administrator can control what project members can do on the system and how much disk space they are allowed.

Other courses are available on CMS and other aspects of the IBM systems.

Instructional Development Facilities

The Learning Resources Center provides a comprehensive media resource service to support the University's instructional, research, and extension programs. The goals of the center are to develop the necessary resources, both human and technological, to meet the demands for improved communication and expanded learning opportunities, to encourage systematic analysis and design of instructional formats to enhance student learning, and to respond to perceived faculty needs for audio-visual media support.

As a primary mission, the staff of the center analyzes instructional needs and provides planning, production, and warehousing of resources to facilitate communication and improve the quality of instruction. These tasks are accomplished through the three divisions of the center: instructional development, media services, and educational systems. While each division provides services of a specific nature, cooperative and coordinated efforts by the staff fulfill faculty and student needs.

During the past several years there has been a noticeable increase in the awareness of media by the faculty. It is apparent that a wide knowledge of the center exists; over 5Ø percent of the faculty use services from the center at some time.

A personal goal of each member of the center's staff is to provide the best quality service possible in fulfilling the University's media requirements. Suggestions from faculty for the kinds of services needed and the quality of those services have been invaluable over the years.

In a broad sense, the Instructional Development Division (IDD) of the Learning Resources Center deals primarily in HUMAN resources. IDD is the division responsible for working directly with faculty members who are creating or revising course materials. IDD helps faculty members implement new approaches to instruction and aids faculty in assessing the effectiveness of either instructional materials or the instructional process.

Media Services provides a wide variety of instructional support services. These include the purchase, rental and local production of instructional materials, and the provision of projection and audio equipment needed to use these materials.

The Educational Systems Division supports the university with television production and distribution facilities. Programs are produced in studio and on location. Staff producers/directors assist faculty in adapting ideas and wishes into usable television programs. A variety of approaches to instructional objectives are possible, and experiementation in

program format is encouraged. Program distribution can be through the closed-circuit cable TV system which serves 180 classrooms, or through departmental self-study facilities.

The Virginia Tech Computing Center has an IBM 3032, IBM System/370 model 158 central processing unit (CPU) and a Honeywell 68/60.

The IBM 3032 runs the OS/VS2 MVS (Multiple Virtual Storage) operating system with JES2 (Job Entry Subsystem 2) as job scheduler. Information Management System (IMS) runs on this CPU.

The IBM 370 runs under the control of Virtual Machine Facility/370 (VM/370). Under VM, Conversational Monitor System (CMS) with APL, Speakeasy, and several major language processors provides interactive support.

The QUICKE processor runs under the control of MVS and JES2 providing service for several high-level languages, a dialect of 370 assembler language, and almost any one-step processor. The QUICKE facility is designed to handle a large volume of small jobs with short run times (fifteen seconds or less) and is available to local users at terminals in 130-A Burruss and to users at all remote workstations. For more information on the use of QUICKE, contact User Services, extension 6154.

MVS batch services are available to remote users via

card-reader/printer terminals and via computers that function as JES2 workstations.

Terminals are located at colleges throughout the state. Hardware support exists for dial-up BSC IBM 2780/1130/S360 MOD 20 compatible terminals and for dial-up ASR 33/35 (ASC II) compatible terminals.

The Honeywell 68/60 provides interactive computing under Multics (Mult iplexed I nteractive C omputing S ervice). Multics' primary purpose is to support undergraduate instruction.

Time sharing services are available in three forms:

* VM/CMS is a time sharing system that allows interactive programming. The software available under CMS includes VS APL, Speakeasy, major language processors, and a text-formatting program called SCRIPT.

* Honeywell Multics is an interactive system which provides interactive programming and debugging with major languages, such as FORTRAN and PL/I.

* IMS (Information Management System) is a control system, used in administrative applications only,

designed to implement medium to large data bases in a multi-application environment. This environment is created to accommodate both online message processing and conventional batch processing, either separately or concurrently.

Hardware Configuration at the Computing Center

The current configuration of the Computing Center is given in the following chart, effective July 1, 1979.

CURRENT VIRGINIA TECH COMPUTING HARDWARE

UNIT	DESCRIPTION	NUMBER
IBM		
3032	IBM System 3032 (6.0 megabytes)	1
3158	IBM System 370 Model 158 (4.0 megabytes)	1
3830	Disk Controller	1
3350	Dual Spindle Disk Drive (800 megabytes)	14*
3330	Dual Spindle Disk Drive (400 megabytes)	2*
3803	Tape Controller Drive	1
3420	Tape Drive (800/1600)	5
2914	I/O Switching Unit	1
3540	Flexible Disk Reader	1

2821	Controller	1
3505	Card Reader	1
3525	Card Punch	1
2501	Card Reader	1
3272	Display Controller	4
3284	Printers	4
3705	Communications Processor	1
1200	Versatec Electo-Static Plotter	1
1051	Calcomp Plotter (11" and 30")	1
1403-N1	Line Printer (1000 LPM)	1
3211	Printer (2000 LPM)	2
3811	Printer Control Unit	2

Honeywell

8868	Honeywell Multics Processor (1.0 megawords)	1
451	Disk Drive (200 megabytes)	2
190	Disk Drive (100 megabytes)	8
1200	Printer (with upper/lower case)	1
500	Tape Drive (800/1600)	2
6624	Datanet Communications Processor	1

Other

1380	Memorex Communications Processor	1
9000	Develcon Dataswitch	1

* Each dual spindle disk drive accomodates two disk packs.

Remote Processing Hardware

A number of remote terminals belong to certain academic and administrative departments which use the Computing Center's VM/CMS and IMS systems. Access to IMS data bases is available through IBM 3277 or equivalent video display terminals or with 3284 remote printers. The remote equipment using VM/CMS includes video, graphics, and hardcopy terminals using EBCDIC or ASCII codes.

The College of Education in University City Office Building operates a Data 100 RJE terminal. The College of Agriculture in Hutcheson Hall, the College of Arts and Sciences in McBryde Hall and in Robeson Hall, the College of Engineering in Randolph Hall, and the Learning Resources Center in Derring Hall, operate HP 2100 stations consisting of a minicomputer, a card reader, and line printer(s). The Computing Center is responsible for the operation of equipment at the locations mentioned. In Whittemore Hall, the College of Engineering maintains and operates a station with similar capability.

A remote job entry station provides a user with a nearby terminal for submitting jobs into the main system and

obtaining printouts at the RJE station, thereby avoiding a trip to the Computing Center in Burruss Hall. The remote stations are open to all Virginia Tech computer users. The schedule for remote stations is posted at the remote stations and in the Consulting Room in Burruss.

Minicomputers

There are over three dozen minicomputers on the Virginia Tech campus. The vendors include Digital Equipment Corporation (DEC), Hewlett Packard, General Electric, Raytheon, and Packard Bell. The uses for minicomputers are quite varied and include programming or instruction in programming, communications to remote lab equipment, graphics for design and study, lab control, and ground wind measurement. In the College of Engineering Computer Laboratory there is also one EAI-580 analog computer used in a hybrid configuration with a digital minicomputer. The Computer Science Department has a Hewlett-Packard 2100 minicomputer to be used for training in time sharing, systems architecture, and operating systems.

Interactive Labs

Two classrooms have been set aside as interactive labs available for undergraduate instruction.

120 Robeson is a graphics lab with:

* 5 Hewlett-Packard 2648s
* 5 Tektronic 4013s
* 2 Hughes C9s
* 1 Versatec 1640 Printer/Plotter
* 1 Numonics Digitizer

116 McBryde is a non-graphics lab with:

* 20 Hewlett Packard 2641 terminals.

Available Publications

Honeywell Information Systems Publications

Order No.	Manual Title
AG90	MPM Introduction
AG91	MPM Reference Guide
AG92	MPM Commands and Active Functions
AG93	MPM Subroutines
AG94	Multics PL/I Language Specifications
AK95	APL Users' Guide
AL40	Multics Introductory Users' Guide
AM82	BASIC
AM83	Multics PL/I Reference Manual
AS43	Multics COBOL Users' Guide
AS44	Multics COBOL Reference Manual
AT58	Multics FORTRAN
AW32	Multics SORT/MERGE
AZ98	WORDPRO Reference Guide
CC70	Multics FORTRAN Users' Guide

Virginia Tech Computing Center Publications

Five kinds of publications are available to Computing Center users: user's guides; miniguides; Fastline, a weekly two-page newsletter available to all users of the computer; Feedback, which contains information from previous Fastlines that has not yet been incorporated into the appropriate user's guide or miniguide; and the Log, a free monthly newsletter available to users and other interested persons.

Under CMS, the Help System provides online documentation for CP and CMS commands, for EDIT subcommands and EDIT macros, and for commands and EDIT macros implemented at Virginia Tech. Under CMS, type:

```
help help
```

to find out how to use the Help System.

User's Guides

The Computing Center provides user's guides to aid in the use of programs and facilities available at the center.

Title of User's Guide	Index Number
Assembler G	3
CMS	5
Versatec and CALCOMP Graphics	6
Handbook for SCRIPT Users	12
Introduction to CMS and the CMS Editor	8
Introduction to the Computing Center	1
Introduction to SCRIPT	9
OS/VS2 JCL and JES2 Statements	2
PDP-11/360 Simulator	13
PL/C	20
Postal Address Labels System (PALS)	19
University of Waterloo SCRIPT Reference Guide	10
Sub-account System	15
SYSPUB User's Guide	11
WATBOL	17
WATFIV	4

Miniguides

In most cases, each miniguide provides information about one program product, utility package, or Computing Center service. In a few cases, descriptions of more than one program are combined into one miniguide.

Title of Miniguide	Index Number
IEBCOPY (IBM Utility Program)	3
IEBGENER (IBM Utility Program)	4
IEBPTPCH (IBM Utility Program)	5
IEFBR14 Program and WIPEOUT	51
PRPU, PRPUSEQ, and CONVERT (Card Deck Maintenance)	29
SyncSort	6
THESIS (Prints Input Data, Spaced for Thesis Binding)	28
TPPRINT (Tape and Sequential Disk File Printing Program)	26
VPILIST, LISTPDS, VPIPDS, and VPIPROGM	25

Fastline

Fastline is the weekly newsletter of the Virginia Tech Computing Center. It contains current user information. Copies are available in the Computing Center Library (132 Burruss), at the main computer room window, in 124 McBryde, in 102-B Hutcheson, in 2103 Derring, in 143 Whittemore, and in 100-J Randolph. Fastline can be printed at remote

locations, with QUICKE, or on a CMS terminal.

Feedback

Feedback consists of reprints of articles published in Fastline which contain information that will eventually be incorporated into a miniguide or user's guide. Once the information has been added to a miniguide or user's guide, the corresponding article in Feedback will be deleted. Information from Fastline published on Thursday will usually be added to Feedback by 12 noon on the following Monday.

Feedback will be useful primarily to the user who misses seeing several issues of Fastline. Feedback may also be useful to someone who prints a group of user's guides or miniguides. By printing a copy of Feedback at the same time, the user has copies of updates to be made to the user's guides all in one place, instead of scattered through back issues of Fastline.

The Log

The Computing Center publishes a monthly newsletter, the Log, to inform users and other interested persons of

computing developments at Virginia Tech. To subscribe to this free newsletter, send your name and address to:

Virginia Tech Computing Center
Editor, Computing Center LOG
Virginia Polytechnic Institute and State University
Blacksburg, Virginia 24061

Other References

[On Interactive Computing]

Bork, A., and Marasco, J., "Modes of Computer Usage in Science", THE Journal, Vol. 4, No. 2, Feb. 1977.

Bork, A., "Interactive Learning: Millikan Lecture, American Association of Physics Teachers", Am. J. Phys. 47(1), Jan 1979, pp. 5-10.

Haynes,J., A Tale of Two Computers, COMPUTER, IEEE Computer Society, New York, May 1977.

Kennedy,T.C.S., The Design of Interactive Procedures for Man-Machine Communication, International Journal of Man-Machine Studies, V5, 1974, pp. 304-334.

Weinberg, G., The Psychology of Computer Programming, Van Nostrand Reinhold Co., New York, 1971.

Lee,J.A.N., "So! You want to use Multics", Dept. of Computer Science, Virginia Tech, March 1979.

[On Instructional Objectives]

Bloom, B.S., Taxonomy of Educational Objectives - Handbook I: Cognitive Domain, David McKay Co., Inc., New York, 1956.

Davies, I.K., Objectives in Curriculum Design, McGraw-Hill, New York, 1976.

Gronlund, N.E., Stating Behavioral Objectives for Classroom Instruction, MacMillan, London, 1970.

Mager, R.F., Preparing Instructional Objectives, Fearon, Belmont CA, 1962.

[On Systematic Design]

Davies, I.K., Competency Based Learning: Technology, Management and Design, McGraw-Hill, New York, 1973.

Davis, R.H., Alexander, L.T., and Yelon, S.L., Learning System Design: An Approach to the Improvement of Instruction, McGraw-Hill, New York, 1974.