COMPUTER LITERACY SCOPE AND SEQUENCE MODELS
A CRITICAL REVIEW OF TWO APPROACHES

Herman Fischer
Litton Data Systems*
Van Nuys, CA 91409

Introduction

Two computer literacy scope and sequence models are reviewed, one by Gary Bitter [ACM SIGCSE Bulletin, Fall 83] and another by the Los Angeles Unified School District [Draft, 3/83]. Both models are described and then compared to each other. In both cases, the models separate general literacy issues from expanded computer competency topics, though not clearly for the same reasons. The competency, or programming, curricula seem to have technical errors which are correctable; primarily these are a tendency to use concepts and skills from the last decade, in a time when we must prepare students for employment in the next decade. Comments are provided on the lack of educational curricula material, on the general apprehension of educators towards literacy programs, and on other specific problems perceived.

Two Scope and Sequence Models

The Bitter approach focuses on a bipartite model. The two parts are Computer Awareness and Computer Programming. Topics are listed as specific details, e.g., fourth graders are introduced to Flowcharting, and BASIC commands print, rem, let, input, and goto. The Los Angeles (LA) Model, called a continuum, is split into four hierarchies, Awareness, Knowledge, Competency, and Expertise. The LA program describes its steps in terms of goals, rather than as specific topical items. A fourth grader "uses appropriate software to enhance skills in various subject areas, e.g., mathematics, science, etc.," or for flowcharting, a sixth grader "describes standard flowchart symbols," and "reads a flowchart." One could say that the Bitter model is, in a sense, more specific, while the LA model builds in a greater degree of interpretational freedom for its users (not to comment on the merits of either style).

The Bitter model is more computing oriented while the LA model is more applications oriented. For example, Bitter notes that fourth graders "can become thoroughly familiar with the term hardware, describing it as...," while LA stresses use: Fifth graders "demonstrate how to insert the disk, turn on the computer, and boot a program... stop, escape from, and continue a program as needed."

Both models will be shown to use obsolete technical ideas, focusing on flowcharting, not making proper use of LOGO, and in the Bitter curriculum, ending with PASCAL rather than any of its current derivatives.

Misquoting Papert

Papert, in Bitter's article, is implied to advocate that "the primary advantage of early computer education is that very young children have not yet developed computer phobia that may inhibit computer training in later years." It is learning phobia that Papert is concerned with. I feel that Papert made a strong point that computers teach children how to think, and how to understand their thought processes; they become epistemologists because they learn how to approach the solving of problems. This, not worries about keyboard phobia, is the primary advantage of using computers in a child's education. And this drastically requires reorienting the primarily BASIC-aligned sequence models to ones which build upon and expand epistemological benefits of computers in a child's education. (I believe no youngster will have keyboard phobia anyway; the world of television commercialdom has seen to it that all will have at least some subconscious worship of computer game playing and enough keyboard hacking desires to get mom and dad to buy the hyped computer products along with sugar coated cereal.)

---

* the author presents views which are strictly personal and have neither been reviewed by his employer or the ACM.

What's Wrong with BASIC

There is nothing wrong with BASIC, as a computer language, if there is nothing better around. Designed for incredibly tight memory restrictions of the computers of the early 1970's, the language yields unstructured and often disorganized programs, with obscure data organizations. LOGO, proposed to be taught first to the very youngest, allows problem structuring, abstraction and information hiding, and data structuring. Yet LOGO is dropped from curricula in both models after the children draw spirals or manipulate sprites. LOGO would be an ideal language to teach problem decomposition, object (real-world) oriented program structuring, and list-oriented data structuring. A fourth grader educated on this track today will become employable in a post-Ada, or a Modula world, maybe even a world where Artificial Intelligence becomes commonplace in programming applications. One educated in BASIC will need to retrain, to "unlearn" the negative habits whch arise from unstructured and non-object oriented problem approaches.

In 1982, one could argue that LOGO was not available to the masses. Today all the popular home and office computers support it, even the $199 processors. For the IBM models, there is even a free subset which is redistributable without copy restrictions, called LadyBug. While many vendors distribute BASIC free, with LOGO available on that same basis, even the argument that is was too expensive, becomes moot.

Teaching Programming versus Problem Solving with Computers

Today, a youngster grown up and joining the ranks of computer professionals, can become a "programmer," whereupon he might earn $10,000 to $28,000, or he can become a "software engineer," whereupon his salary limits can go up into the $50,000's. Programming is a skill where, given a specification, the practitioner produces (hopefully debugged) program code. Software engineering is often thought of as an art, where, given a set of requirements, the practitioner devises a solution, which usually results in a specification, design, and implementation in code. Software engineers get paid so much more because they can attack a problem, design a solution, and follow it through implementation. Papert's approach, teaching children to discover thought processes related to their world (objects, to the computer scientist), and to break complex problems into simpler ones (house becomes triangle and square; face becomes circles and arcs) inadvertently practices software engineering instead of programming (coding). Once children are started on the right track, why stop and switch to the wrong subject?

Why Are There No Educational Materials

A trip to any popular bookseller will yield shelves primed with hundreds of books on BASIC programming. One is lucky to find half a dozen on LOGO. And looking for books geared to primary and secondary educators, there are only one or two on LOGO (if they are stocked at all).

I visited the MIT AI Laboratory last summer, expecting to come away with an armful of curriculum materials suitable for the intermediate instruction in LOGO, past the spiral and sprite stages, and not at the sophistication of physics and artificial intelligence researchers. Instead, there was an insular and possibly lofty attitude, one where not even the MIT Education department uses its AI Laboratory's products. Lists of districts using LOGO seemed populated mostly with selected vendor-sponsored test schools. Educational focus between the very young, or learning disabled, and the advanced users, seemed absent. That is needless and a pity.

The Bitter Model

My understanding of the Bitter model is shown in Table 1. The figure shows, for each topic of the two tracks of the model, the starting grade indicated. Most topics are only slated for introduction in the starting grade, with expansion and further activities in the next three to eight years. Thus, when reading the topics, one must bear in mind that some of the apparently quite early starts are indeed only "start-ups," and not intended to imply mastery or comprehensive familiarity with a topic.

Table 1. Bitter model, tracks and topics.

Starting Grade	Awareness Track	Programming Track
K	What a computer is Following directions Vocabulary	Programming programmable devices Turtle graphics
1	What a computer can do Learning to use a computer Using the keyboard	Turtle graphics (moving shapes on screen)

2	Computer advantages " disadvantages Computers in our lives	Turtle graphics (rotating shapes) LOGO (sprites)
3	History of computers And, Or logic; parts of computers	Logo animation (turtle geom & sprites) Logo for problem solving
4	Definitions of hardware and software Binary representations Flowcharting symbols Storyboarding a program	BASIC: Print, Remark, Let, Input, & Goto; Writing formulas in BASIC String data in BASIC Relations "<" and ">"
5	Computer generations (tube, transistor, IC) How a "counter" works	BASIC: If-Then, On-Goto Read-Data Word Processing
6	Mainframes vs mini's vs micro's. Processing data Languages and applications Iteration (looping)	BASIC: For-Next, Random Problem solving Graphics
7	Modeling, to develop plan to solve general problem Robotics exposure Social issuses Data base use fundamentals	BASIC: arrays, functions
8	Computer crime Algorithms and their usage	BASIC: 2 dimensional arrays: sound & color
9	Discuss computer capabilities Career and vocation planning	Simulation programming BASIC: matrix, files PILOT: introduction
10	Data collection and interpretation, reporting Artificial intelligence, concept and applications	Pascal: introduction
11	Systems, types vs. brands Statistical sampling	PILOT: programming
12	Societal impact of computers Privacy protection	Pascal: programming Data bases: updating

LA Model

My understanding of the LA model is shown in a hierarchy chart, figure 1, because the LA model is not tied to specific grades; instead it is goal oriented and appears to have a variation of three to five grades from earliest to latest grade when subject may be introduced. Shown in braces are the grade level where the topics in the area are suggested to be started, followed by a pair of numbers representing the lowest (presumably optional) starting grade and the highest grade ending the program.

Comparison of Models: Topics and Starting Ages

The common elements of the two models are next compared to show where topical agreement exists, and how diverse the opinions of start-up age varies.

The LA model's hierarchy is used here, because the reader has the Bitter model above, and because the LA model has a more hierarchical (topic development continuum versus age progression) organization.

Topics are described, followed by the coding LA:x,y B:z. The codes mean that the LA curriculum has the x-th grade for general start-up, the y-th grade for optional earliest start-up, and the Bitter model has the z-th grade for its start-up. Where an item has an asterisk in an entry it means that the topic is not required or not present in the curriculum. Where a topic has a question mark entry, it means that the topic probably is covered but cannot be clearly identified as to which grade. Not every topic of both models is separately identified; several have been grouped to make the table more readable.

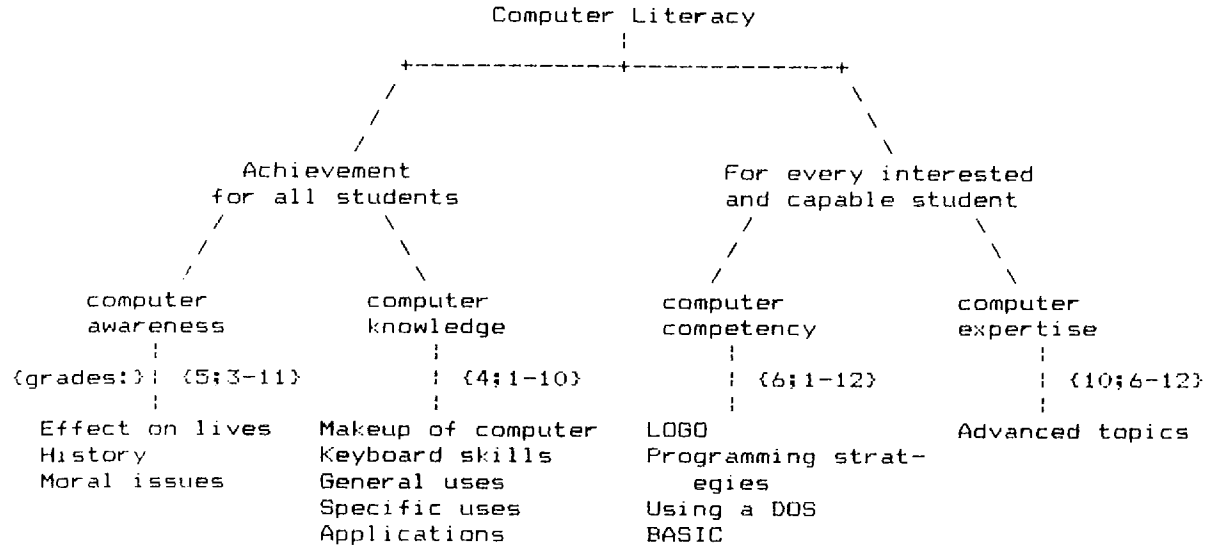

Figure 1. Hierarchy chart showing topic organization of LA model, and grade ranges in which they may be taught.

Computer Awareness

Computers effect on our lives

Recognize applications of computers	LA:5,3	B:2
Recognize ways used in daily life	LA:7,4	B:2
Identify computer-related occupations	LA:8,6	B:9
Limitations and capabilities of computers	LA:8,6	B:2
Reading related books, magazines	LA:8,3	B:*

History of computers

Characteristics of each generation	LA:6,4	B:5
Differentiate mainframe, mini, micro	LA:8,5	B:6

Moral issues and computer use

Copyright laws	LA:8,6	B:8
Advantages and dangers of data bases	LA:10,8	B:12
Legal issues from widespread usage	LA:10,8	B:8

Computer Knowledge/Operation

Makeup of a Computer

Identify components	LA:5,1	B:3
Use prepared program in a microcomputer	LA:4,3	B:1
Definitions, hardware, and software	LA:4,3	B:4
How it works (input/output, etc.)	LA:5,4	B:3
How a "counter" circuit works	LA:*,*	B:5
Capabilities of types and sizes, by brand	LA:7,5	B:11

Keyboard skills

Letters and numbers	LA:3,1	B:1
Special-purpose keys	LA:5,2	B:?
Correct keyboarding skills	LA:5,2	B:*

General uses of computer

Care, how to handle floppy and boot	LA:5,3	B:1
Run from catalog (directory), or menu, etc.	LA:6,4	B:*
Recognize simple error messages	LA:7,4	B:?

Specific uses

Usage of drills, tutorials, and simulations	LA:4,2	B:?
Problem solving program	LA:6,4	B:?

Word Processing (These topics are in LA's)	LA:7,5	B:5
Use Data Base (track for all students)	LA:9,6	B:7
Use spreadsheet (but in Bitter's pro-)	LA:9,7	B:*
Use utilities (gramming track.)	LA:9,6	B:*

Applications

Know major applications	LA:8,5	B:6
Exposure to robotics	LA:*,*	B:7
Artificial intelligence	LA:*,*	B:10

Computer competency (program for those who desire it

LOGO

Simple programs, graphical design	LA:*,K	B:K
Text programs, dialogues	LA:*,3	B:11
Sound in program	LA:*,3	B:*
Sprites: movement and direction	LA:*,*	B:2

Programming strategies (BASIC)

Flowcharts	LA:6,4	B:4
Predict output from a listing	LA:7,4	B:5(?)
Fundamental programming	LA:6,4	B:4
Sound and color (graphics) programming	LA:6,4	B:8
Structured programming	LA:7,4	B:*

Computer expertise

Advanced computer utilization

Multiple languages skills	LA:10,8	B:10
Evaluate hardware, software for own use	LA:11,9	B:11
Programs using algorithms, graphics	LA:11,9	B:8
Formatting techniques	LA:11,9	B:*

Both models have a great number of parallels. They stress teaching what might be called the "social study" of computers, and they stress providing programming skills to the interested and capable students. They neither effectively stress using computers to help children discover learning (more to follow), nor provide plans for using computers to help learning disabled to progress better. And what is most neglected is building a a track for teaching children how to approach real world problems, be they simple or complicated, and using subtle but incredibly important "methodologies" to break the real world into objects, actions, functions, and attributes which can be expressed in a series of increasingly more detailed but hidden subproblems, until the complete solution is at hand. A major point of "Mindstorms" is missed.

Is it fair to compare these two models by citing the starting ages for the various topics? That is hard to say. Some classrooms are filled with bright self motivated children while others have children who will never really read, even after becoming adults. Papert seems to show that both can benefit from the process of discovering non-Euclidian geometry and basic programming in Turtle Graphics. Yet neither of the above curricula stress this benefit of computers in the classroom. Maybe neither can, because Papert feels [ACM '83 Plenary Lecture] that one computer per five students is needed to achieve results he describes. (LA's average is one per 206 students [LA Times, 11/27/93].) Few districts will survive taxpayer fury if they try to create special assessments to bring the number of computers up this high.

We are faced with two curricula models which have spelled out academic details, and programming introduction, with the goal of "literacy" instead of the goal of direct benefits to the students in their learning and developing process. And a dilemma that there are neither teaching materials nor sufficient computing hardware to prepare the educators for either the planned course (above) or the "better" course (Papert's goal of creating epistemologists out of children).

Apprehensive Educators

I have consulted with several primary and secondary educators, in the LA area (just because that is where I live), and find a consistent reaction when they read either of the literacy curricula: they appear frightened of the thought that they have neither any concept of what the mandatory and optional topics are about, nor any hope of being able to master these new topics while carrying out their day to day chores. Some have basic

programming skills, say sufficient to add up numbers in BASIC or draw up houses and faces in LOGO. Strangely no one had ever used either a spreadsheet or database program. They do not have any feel for the material which needs to be known to present the topics in the list, REGARDLESS OF GRADE OF START-UP. Furthermore, there does not appear to be any competent teaching materials for preparation of the subjects (with the possible exception of vendor-dependent and vendor-limited material supplied by several of the manufacturers).

Another element of fright seems to arise when these educators eventually relize that the children they will educate WILL have the skills outlined above, and many self starters will have them earlier than the ages indicated; the prospects of BASIC and LOGO hackers in the classroom, either because of the questions they might ask, or the feared antisocial behavior machine-bound concentration might yield, are of significant concern. And then, probably every now and then, a hacker of Defense Department ARPAnet fame might come about and maliciously destroy the other students' work, alter grades, or cause havoc extremely difficult to detect and prevent.

Must We Push Technological Obsolescence

Both curricula abandon LOGO after early grades and use BASIC to teach programming. That anachronism in technology needs to be corrected. It may be political suicide to tell a parent just learning BASIC that he has started on the wrong language; that his child will program LOGO instead. But the realities require that, for the programming experience to be useful, it should be in a structured environment, appropriate for real-world (object) orientation, with modularity, hiding (of subprocedures), and abstraction. Big words which children will not know, but nonetheless do practice in drawing LOGO happy-faces. They cannot perform these necessary methodological practices in BASIC. (I feel that I am hyperventilating on this point by now.)

Bitter's model introduces PASCAL to tenth graders. PASCAL became very popular among college educators in the 1970's, and has formed the basis which underlies a number of new languages. PASAL never had great commercial or business acceptance (a different language called "C" has that acclaim, today). However, two of the newer languages, which owe roots to PASCAL are important, not necessarily to teach this year or next, but for near term planning to replace PASCAL as the "advanced language" of the model. Wirth, the "father" of PASCAL, more recently "fathered" Modula 2, a strongly typed language with numerous important advantages over PASCAL. Though Modula 2 compliers are just coming of age on microcomputers (as is the case with LOGO), there appears to be a need for tests and teaching materials for this language. Separately, the Defense Department, working with a world wide set of experts, recently completed specification of the Ada language, also a strongly typed language derived from PASCAL. Numerous commercial and defense suppliers are preparing for the need to have educated Ada programmers on their staffs in the upcoming years; training materials for that language are presently slanted for the well-experienced, not the beginners. Teaching PASCAL may eventually be somewhat like teaching Latin; modern languages have their roots in it, but speaking it will not order one a meal in a restaurant.

(If one were to replace PASCAL with a current language for the immediate short term, then there probably is only one choice--"C." A vocational student trained in "C" probably has as assured a future as a COBOL programmer had twenty years ago.)

Flowcharting is a part of both curricula. Flowcharts are relics of the 1960's which went out before "top down" programming, which also went out in the previous decade [Ken Orr, Dallas AdaTEC, 10/20/83]. There are graphical design methods, and they are important to the methodologies which use them. There is considerable dispute over which (and whose) methodologies are better. Educators should be able to get some advice and help from the design methodology experts in setting up curricula (and staying current) in the design aids field. And they should bury the old flowchart for good.

Both curricula teach computer generations. Does a fifth grader care about tube computers, transistor computers, or integrated circuit computers? Will this make him a better person for society? Is knowledge of the binary system even relevant? No microcomputer user needs to know about tubes, or about binary numbers, to use spreadsheets, wordprocessors, or data bases.

Did Anybody Notice CAI Missing

Neither program has an element where computers are used to replace teachers, "computers programming children" as Papert would call it. That is incredibly positive! Large numbers of advocates of computers in primary and secondary grades assume that the computer will be used for drill, for tutoring, and to perhaps alleviate the "teacher shortage." Typical experience with Computer Assisted Instruction (CAI) shows that young

students have short attention spans, get bored, and quit early. College level students may have better track records with non-human tutors. But for primary and secondary levels, unless there is some breakthrough, technologically, in CAI, that usage of computers is best omitted as a major element of curricula. There will always be large numbers of CAI-type software programs available; they will supplement teaching, tutoring, and assist students with special learning requirements. But the main point is, they will not form the basis for a major part of a curriculum.

Will There be a Camelot

Planning curricula will be difficult; honing them will be a lengthy trial and error process. Some school boards will have success stories (particularly where they are well-heeled), and for every success story, large numbers of districts will create educator discontent, upset the funding sources, and build a few squeaky wheels who will complain endlessly.

Camelot, if one will ever arise, will do so because of perseverance, or perhaps because the district was burned badly enough to become ambitious in correcting its curricula. I doubt that any one-vendor district will have success; vendor hardware gets obsolete so fast that there are no benefits from picking the temporal "best." Hopefully successful districts will be compatible (software-wise) between the multiplicity of equipment which eventually will be available. And adaptable to changing languages, methods, and technology. Only time will tell.

****

BATCH LOGO - continued from page 16

3. Accomplishment Survey

SUMMER COMPUTER INSTUTUTE 1983

WORKSHOP ON STRUCTURED PROGRAMMING FOR JUNIOR HIGH STUDENTS

Accomplishment Survey

Name _____ Approximate Age _____

The final task of this workshop is a hands-on questionnaire. This is not a test. Although you may find some of the questions challenging, please try to answer all of the questions to the best of your ability. Your answers will help us to assess the success of this workshop.

Concepts in LOGO

1. Write the PL/1 code to draw an equilateral triangle. (remember to approach the angle of each turn from the turtle's point of view.)

2. If the procedure RIGHT were removed from LOGO:
   a. Could you still draw any figure that you were previously able to draw?
   b. If your answer to above was yes, write the PL/1 code for a procedure called RITE to perform the work of the missing procedure RIGHT.

3. Using LOGO, draw a circle of any size you wish. Hand in your printout (your WIDJET listing file).

PL/1 Language

Explain the purpose of the following PL/1 constructs:

1. DECLARE
2. DO WHILE
3. DO I = 1 TO 5; contrast this statement with DO WHILE.
4. CALL
5. GET LIST
6. /* */

Concepts in Programming

1. What is the purpose of using procedures?
2. What are arguments and parameters? Explain their importance.
3. Explain the importance of documentation.