Behavior Research Methods, Instruments, & Computers
1988, 20 (2), 206-217

Micro Experimental Laboratory: An integrated system for IBM PC compatibles

WALTER SCHNEIDER
University of Pittsburgh, Pittsburgh, Pennsylvania

Micro Experimental Laboratory (MEL) is a third-generation integrated software system for experimental research. The researcher fills in forms, and MEL writes the experimental program, runs the experiments, and analyzes the data. MEL includes a form-based user interface, automatic programming, computer tutorials, a compiler, a real-time data acquisition system, database management, statistical analysis, and subject scheduling. It can perform most reaction time, questionnaire, and text comprehension experiments with little or no programming. It includes a Pascal-like programming language and can call routines written in standard languages. MEL operates on IBM PC compatible computers and supports most display controllers. MEL maintains millisecond timing with high-speed text and graphics presentation. MEL provides a systematic approach to dealing with nine concerns in running an experimental laboratory.

Developers of any software system for experimentation must trade off considerations of flexibility, learnability, ease of use, precision, experiment generation time, and experimental quality control to maximize system performance. Incorporation of advances in hardware and software technology allow for quantum jumps in terms of system performance given the above considerations. The Micro Experimental Laboratory (MEL) optimizes these trade-offs within the constraints of personal computers with 640K of memory, human factors considerations about interface technology, fourth-generation software technology, and the use of automatic programming techniques.

MEL is a third-generation psychology software system that builds on the lessons learned from earlier programming systems. The first generation of programming languages in psychology were typically specialized process control languages working in small memory partitions (e.g., Schneider & Scholz, 1973; Scholz, 1972) operating with less than 16K of memory on machines such as the DEC PDP-8 or the IBM 1800. The second generation had two major variants. The first was the *subroutine approach*, which involved a series of subroutines that are called from FORTRAN or Pascal programs (e.g., Osgood, 1985, 1988) running on DEC PDP-11s or Apple IIs with 64K of usable memory per process.[1] The second variant was the *stimulus presentation approach* (Eamon, 1982; Poltrock & Foltz, 1982, 1988), in which fixed displays were presented in prespecified orders on Apple II computers. The orders could be either specified within one of three prescribed paradigms, as in CEDATS (Eamon, 1982), or explicitly enumerated, as in APT (Poltrock & Foltz, 1982, 1988). The subroutine package approach was flexible but required that graduate students be trained as programmers and involved long development times to generate experiments. The stimulus presentation approach allowed fast experiment specification at the cost of reduced flexibility and a great deal of typing to enumerate all the stimuli and the stimulus orderings.

MEL is a third-generation psychology software system that operates on IBM PCs with 640K of memory and a hard disk.[2] The order of magnitude increase in memory from the previous generation enables much greater user support in developing, debugging, documenting, and analyzing experiments. In the 1980s, new software development techniques were generated to speed software creation. These are referred to as fourth-generation languages, code generators, or automatic programming systems. These are systems in which the user specifies the problem in a format that closely matches the application environment, using terms already known by the user. A code generator program converts the specification into more traditional computer source code (e.g., Pascal). Program generators can significantly reduce program development time compared with conventional programming. MEL incorporates a code generator to execute routine experimental functions (e.g., stimulus display, response collection, condition randomization, and data logging). Many experiments can be done without the experimenter's writing any code. If specialized displays or stimulus randomization procedures are needed, the user can incorporate code as needed.

MEL incorporates a form-based user interface (see Schneiderman, 1987) that allows the user to fill in blanks in forms to specify experiments and request help on every blank, if needed. This type of user interface provides for structured problem descriptions, self-documentation

Reprint requests can be sent to Walter Schneider, Learning Research and Development Center Building, 3939 O'Hara St., University of Pittsburgh, Pittsburgh, PA 15260. For detailed descriptions of the Micro Experimental Laboratory software and demonstration floppies, contact Psychology Software Tools, Inc., 511 Bevington Rd., Pittsburgh, PA 15221 (phone 412-244-1908). The cost of licensing the system varies, depending on features requested and quantity ordered. Single licenses are $295 for the student edition, $495 for the professional edition.

of specifications, and default specifications of parameters, and it greatly reduces the learning time required to use the system (see Ogden & Boyle, 1982; Smith, 1982).

MEL is an integrated software system for professional and student experimental research using IBM PC compatible machines. MEL can run nearly all discrete-trial experimental paradigms (e.g., reaction time, questionnaire, and text comprehension experiments). Discrete-trial paradigms are ones in which there is a precisely timed trial followed by an intertrial interval in which some variability (e.g., 0.1 sec) is acceptable.[3] The user writes experiments by filling in forms, specifying information in a manner consistent with the way a psychological researcher thinks about the problem (e.g., by entering independent and dependent variables, blocks, trials, stimulus lists, questionnaires, etc.). The form specifications are converted into source code using automatic programming techniques. The source code is compiled into an object code. The object code is run, presenting stimuli and collecting responses with millisecond accuracy. The data are logged for later analysis. The analysis programs produce plots, tables, lists, and statistical tests (e.g., ANOVAs, correlations, planned comparisons, group tests). The data can also be exported to other statistical packages. As an integrated package, the experimenter fills in a form, and the information on the form is transmitted to programs that generate the experiment, compile the code, manage the data, and analyze the results. All these programs use the same information, eliminating the need for the experimenter to retype the information.

MEL was designed to solve nine concerns that are common in professional cognitive laboratories and were present in my own laboratory. First, MEL allows rapid experimental development and modification. Basic research involves exploring the unknown. In my laboratory, we generally produce and test multiple pilot paradigms before a procedure clearly illustrates the phenomena of interest. In second-generation systems of the subroutine type, experiment generation is probably the most expensive item in completing an experiment, in terms of time and money. Even with a good subroutine library, a typical experiment requires 600 lines of code. Software industry studies show that a professional applications programmer, on average, generates about 10–65 lines of debugged and documented code per day (see Yourdin, 1975, p. 143). At 30 lines per day, 160 h of development time by a professional programmer are required for implementing a new paradigm. In psychology laboratories, graduate students do most of the experimental programming. The students must first learn to be programmers and then they can program about one novel paradigm each semester. The long development time and difficulty of learning the system have detrimental side effects. Students often work with someone else's paradigm, using code they do not understand.

MEL incorporates automatic code-generation techniques. The knowledge of how to program routine experimental procedures from psychological specifications is incorporated into a program. MEL writes the program for the experimenter at the rate of about 100 lines of code per second. Automatic programming can greatly reduce the experiment generation time relative to the time required for the subroutine call approach. In MEL, a typical paradigm can be developed in less than a day, suggesting a productivity increase of a factor of 20 (8 h vs. 160 h). I have implemented new paradigms in less time than it took to read the article describing the paradigm. Undergraduates have generated 600 lines of documented and debugged code in an afternoon. A graduate student generated a new paradigm on one computer in about the same time as it took to recompile and link an old paradigm using a second-generation subroutine library approach.

The second concern is providing precision and speed in experiment presentation. The subroutine approach provides flexibility, but generally requires special hardware and strict adherence to programming rules. If the experimenter calls subroutines in an inappropriate order or performs long-duration calculations at the wrong time, erroneous timing will result. MEL reprograms the standard IBM hardware clock to maintain millisecond accuracy. The code generator will produce only debugged code that maintains the precise order of commands to eliminate coding timing errors. MEL also cues all precise timing events in an event table so events are executed precisely, independent of the time needed to generate the events or log the data. The event table's cuing of commands also greatly speeds the output of critical events. For example, MEL writes characters 82 times faster than do write statements in Microsoft Pascal.

The third concern is providing flexibility in experimental procedures. MEL provides greater flexibility than stimulus presentation packages because the experimenter is not limited to a prescribed set of paradigms. The form specifications in MEL are powerful enough to implement most computer experiments without code. In a survey of 104 experiments published in *Perception & Psychophysics* that used computers, Butler (1988) estimated that 70% could be authored with MEL without writing any code. In addition, with some code added, MEL can perform nearly all the experiments that can be run on a PC using discrete-trial procedures. If special functions are needed, the experimenter can write code. MEL includes a Pascal-like language with special functions for real-time experimentation. With a few lines of code, an experimenter can program graphics, interface voice keys, read the position of the mouse, change character sizes, draw polygons, or calculate trigonometric functions to position characters. If a student knows no programming, the faculty member or an advanced student can spend a few minutes writing the needed code segment to enable a specialized function. To maintain complete flexibility, the experimenter can call procedures written in standard languages such as Pascal, C, FORTRAN, and BASIC.

The fourth concern is quality control assurance. Testing of complex or real-time programs generally takes longer than generating the programs originally (see

Yourdin, 1975). A researcher must carefully test a program to verify that the experimental procedure exactly matches the procedure reported in the method section. MEL simplifies human checking of the program, verification of the conditions of the experiment, and verification of the timing of the experiment. To check a student program using the subroutine method, a faculty supervisor would have to desk check the code and run a series of test runs on the experiment. In the stimulus presentation method, a faculty supervisor would have to check the typing and the ordering of all the stimuli. In APT (Poltrock & Foltz, 1982, 1988), for example, this involves checking lists of arbitrary numbers and letters to be sure that there are no errors. In MEL the critical segments of a program (e.g., the forms for stimulus presentation and response collection) are typically specified in two or three forms. These can be checked in a few minutes, rather than spending an hour checking the resulting code. The ordering of conditions is typically specified on a single blank to determine how the stimuli are sampled. Stimulus lists include only the part of the stimulus that changes (e.g., the list of words, rather than the full screens including the words and instructions, as is the case in stimulus presentation systems). The stimulus list also includes the independent variable specification to facilitate checking (e.g., the stimulus "1\CAT" may code a condition 1 word stimulus with the word "CAT" in a lexical decision experiment). The invariant parts of displays (e.g., the instructions on the target frame) need only be typed once. Most important, the code generator and the programs in MEL have been extensively debugged in the process of running over 100 studies collecting over a million observations of data.

MEL provides debugging aids for internal checking of variables and precise timing to allow an experimenter to verify the accuracy of the experiment. MEL verifies that a valid value is logged on each trial or block. Any missing value or out-of-range value is flagged, causing termination of the experiment. The experimenter can set an option to automatically display or print the trial number and the trial's independent and dependent variables at the end of each trial. This allows verification of whether the executed trial matches the way the data were recorded. A single command changes all timing from milliseconds to seconds. With this option, one can verify that a 20-msec period is 20 msec by timing seconds with a wristwatch. Commands can be entered to simulate responding so that internal events can be timed (e.g., measuring how many milliseconds it takes the computer to put up 12 lines of text). If a program is run in automatic execution mode, the computer performs either the correct responses or the random responses, so one can let an experiment run all night to verify that there are no random subject sequences that can crash the program.

The fifth concern is that nonprogrammers should be able to generate experiments. A large proportion of the graduate students and undergraduates who might want to work in a psychology laboratory have only minimal programming skills. The form-based approach used in MEL can be quickly learned by nonprogrammers (see below).

The sixth concern is to minimize time and resources needed to teach researchers to program experiments. Probably the second largest expense of running a computer-controlled cognitive laboratory is the cost of training students. In the second-generation subroutine approach, new graduate students required 6 to 12 months before they could be trusted to program a novel paradigm without someone else helping them or checking the work. In fact, this cost was so high in the subroutine approach that when I moved between institutions, the cost of training a new staff was greater than buying the equipment to set up the new laboratory. One year's support for a graduate student (including indirect costs and tuition) can purchase 5 to 10 IBM PC class machines. MEL reduces training costs in six ways. First, the system is form based so there is less to learn. Second, over 20 computer tutorial lessons provide a self-study environment in which to learn the system. Third, there are extensive manuals, which detail the system for the novice (100 pages) and expert user (300 pages). These manuals were developed using protocol-aided revision (see Schriver, 1984), in which the draft manuals and tutorials are given to novice users by an independent testing organization. Students learned the system with only the manuals to aid them. Whenever the student had difficulty, the manuals were revised. Fourth, an extensive sample program library provides examples of how to specify classic experimental procedures. Fifth, exercises and quizzes are included to extend and assess student knowledge of the system. Sixth, task diagrams have been developed to provide a quick reference to using the system. These diagrams provide a step-by-step listing of goals and actions to specify an experiment. This type of diagram was developed in industry (see Poller, Friend, Hegarty, Rubin, & Dever, 1982) and has been found to greatly reduce training times for learning technical procedures.

The seventh concern is enhancing communications with colleagues and exporting experimental procedures to other laboratories. Easy communication of exact experimental procedures can improve scientific communication, enhance replication, and simplify extension of experiments. Developments in computer science have been greatly aided by the advent of standardization of programming languages. Think how much progress is hindered for an algorithm to be developed on one campus and not be exported to another campus. This lack of exportability is typical in psychology. Currently a program developed on one campus can rarely be easily run and modified on another campus. MEL runs on most IBM PC compatibles and supports over 30 display controllers. Most psychologists in the United States have access to an IBM PC compatible computer and hence can run experiments that operate on a standard IBM PC. To enhance communication, a demonstration copy of MEL can be distributed

without charge for the purposes of nonprofit scientific communication.[4] The form-based specification language allows colleagues to unambiguously determine the exact experimental specification by reading a small number of forms. I hope that in the future it will become commonplace to offer demonstration floppies with reprint requests. The reader should note that copying a floppy costs about as much as photocopying 10 pages of a manuscript.

The eighth concern is rapid exploratory data analysis of experiments. Since experiments must often be extensively pilot tested, it is important to go rapidly from running the experiment to graphs, tables, and ANOVAs of the data. MEL is an integrated system that includes graphics, descriptive statistics, and inferential statistics. Minutes after running the subject, the data can be plotted on the screen for the user to determine if the results are reasonable. Individual subject's plots can be examined to determine if all the subjects understand the task and are performing adequately. Reaction time distributions can be plotted to determine if long responses are biasing the results and should be filtered out. ANOVAs and planned comparisons can be used to determine whether effects are significant. Analyses can be set up for automatic execution as soon as the subject completes the experiment. Data can be exported to other statistical packages if additional analyses are needed.

The ninth concern MEL alleviates is that it allows graduate students to quickly and cheaply set up experimental laboratories on their own, even if they have limited research funds.[5] An unfortunate problem with the current system for graduate training is that most researchers are trained in the best laboratories, where students have far better hardware and technical support than they can expect to have when they set up their own laboratories. The result is that new junior faculty researchers often spend most of their first year trying to set up their own laboratories. Danny Kahneman, University of California, Berkeley, has described the usefulness of MEL to graduating students, stating that "MEL turns a PC equipped with fairly standard features into a first-rate professional laboratory tool. Every young Ph.D. trained in MEL will therefore be able to set up a functioning laboratory with no delay and very little cost" (personal communication).

AUTHORING EXPERIMENTS

To author an experiment in MEL, a researcher fills in the blanks on a series of forms. There are seven steps to getting an experiment ready to run. First, the names of the independent and dependent variables must be listed. Second, the experiment must be outlined with sample displays drawn, the stimuli listed, and responses specified. Third, an outline of the forms needed for the experiment must be generated. Fourth, the forms are to be filled in. Fifth, the skeleton experiment is tested. Sixth, the stimulus lists for the experiments are added. Seventh, all the data storage is checked to verify the conditions and data logging of the experiment.

MEL allows incremental refinement of experiments. The experimenter first generates the simplest possible experiment (e.g., one having only one stimulus with a single response). After the simple version of the experiment is running correctly, the additional complexity is added incrementally (e.g., both responses, all the stimuli, and the instructions are added). Incremental refinement speeds program generation (see Yourdin, 1975) by limiting the number of new specifications at each stage. It also provides the author with a rapid, top-level view of how the experiment will look to allow assessment of the appropriateness of the paradigm. MEL makes incremental refinement easy due to the speed of going from code to a running experiment. Conversion of a set of forms into a running program occurs in less than 10 sec. A switch from a running experiment back to editing the forms occurs in 1 sec.

Filling in Forms

Most experiments can be implemented by filling in a small number of forms. The author types in the blanks of the forms. This form of human-computer dialog has been found to be easier to learn and faster to specify than command language or menu-based dialogs (e.g., Ogden & Boyle, 1982). The blanks in MEL provide psychologically meaningful key words to prompt the experimenter to fully specify an experiment. Many of the blanks are filled in automatically with default values. By pressing a help key (F1), a description of the effect of that field is displayed. If the blank has a fixed number of options, the author can press the "+" key to sequence through the list of possible options for the form (e.g., pressing the "+" key on the "FOREGROUND COLOR" field will display "RED," "GREEN," etc.). The author needs to type very little to specify a complete experiment. A complete choice reaction time experiment can be implemented with as few as 96 keypresses. (Note that a single line of text is 80 characters.)

To specify a typical reaction time experiment requires filling in four types of forms. The EXPERIMENT form specifies the independent and dependent variable names, the abstract, and the major events. The BLOCKS form specifies the number of blocks, the block conditions, and the block events. The TRIALS form specifies the number of trials, the trial conditions, and the trial events. The INSERT form specifies the list of conditions and the stimuli for the experiment. Figures 1-4 show the basic four form types as they would be filled in for a four-choice reaction time experiment.

The forms in Figures 1-4 specify the choice reaction time experiment. The numbers with square boxes in Figures 1-4 provide labels to simplify explanation. In the following text, the numbers with square brackets around the number refer to the labeled blank in the figures. The

reader is encouraged to read each paragraph completely and then examine the indicated blanks of the forms. The underlined text on the form illustrates where the experiment author can enter information. The blanks without squares are defaulted and were not typed by the author.

The EXPERIMENT form (Figure 1) includes the author's name [1], experiment abstract [2], independent variable name [3], and dependent variable name [4]. The experiment events include an orientation frame [5], blocks [6], and a goodbye frame [7]. The BLOCKS form (Figure 2, top) includes a comment describing the BLOCK [8], the number of blocks [9], and the block events [10]. The TRIALS form (Figure 2, bottom) contains a comment [11], number of trials [14], and trial events [17 and 18]. In addition, the trial form indicates whether error trials are to be rerun at the end of a trial [16].

MEL provides a convenient method for specifying, selecting, and utilizing stimulus lists. The method uses *inserts* to insert text and parameters in forms before they are executed. Most experiments require that stimuli be selected from a list to be presented as changing stimuli embedded in a series of displays that are fixed. For example, in a choice reaction time experiment, there may be four displays, each of which includes the words "push button" and one of the letters "a," "b," "c," and "d." In MEL the author could complete four separate frames for the four stimuli. However, this would be time-consuming and may lead to errors. With inserts, the author can type only the stimuli or parts of the display that vary from trial to trial. The author can also identify the inserts by indicating to what condition of the independent variable each stimulus belongs. This is done by the use of *slots*. Each stimulus set has a number of slots. The INSERT form [19] (Figure 3) lists the condition letter pairs for the choice reaction time experiment. A number [20] encodes the condition number that is logged for later anal-

```
PAGE 1                    EXPERIMENT SPECIFICATIONS #1
1 AUTHOR Bill James          CREATION DATE 10-31-87 LAST UPDATE 11-02-87
  FILES:  EXP choicert DATA choicert  INSERT choicert  INCLUDE
  BACKUP DISK VOLUME                  DEBUG normal      SPARE
2 ABSTRACT This is a basic choice reaction time experiment.  The subject
  performs 2 blocks of 4 tials in a 4 choice reaction time task in which
  the subject presses the letter of the stimulus.  The subject receives
  accuracy and reaction time feedback.
  NAMES OF: BLOCK INDEPENDENT VARS  1:      2:      3:      4:
   (to be logged for later analysis) 5:      6:      7:      8:
         BLOCK DEPENDENT VARIABLES   1:      2:      3:      4:
   (logs as ACcuracy,SElection, RT)  5:      6:      7:      8:
        TRIAL INDEPENDENT VARIABLES  1:stimnum 3      3:      4:
   (to be logged for later analysis) 5:      6:      7:      8:
        TRIAL DEPENDENT VARIABLES    1:resp 4  2:      3:      4:
   (logs as ACcuracy,SElection, RT)  5:      6:      7:      8:
EVENT   TYPE  FORM ID      COMMENT                MISC. INSERT FIELD
5  1  frame  1        welcome to subject
6  2  block  1        perform 2 blocks
7  3  frame  5        thank subject, goodbye
```

Figure 1. Sample EXPERIMENT form for labeling the experiment, labeling the independent and dependent variables, and listing the top-level events of the experiment. The experimenter programs an experiment by filling in blanks. The words in capital letters identify the function of each blank. The underlined text represents potential experimenter input. The blanks with square boxes were typed by the experimenter. The other blanks were filled in as defaults by the system. See the text for a description of each of the blanks that the experimenter typed in.

```
PAGE 1                    BLOCKS SPECIFICATIONS #1
8  COMMENT do 2 blocks of trials
   BLOCK INSERTS                SEQUENCE none     NUMBER OF BLOCKS 2 9
   VALUES OF BLOCK INDEPENDENT VARS  1:      2:      3:      4:
    (to be logged for later analysis)5:      6:      7:      8:
   EVENT  TYPE   FORM ID      COMMENT                MISC. INSERT FIELD
10 { 1  frame  2        block orientation
   { 2  trial  1        do 4 choice RT

PAGE 1                    TRIALS SPECIFICATIONS #1
11 COMMENT present 4 trials for the block 13
   TRIAL INSERTS 1 12       SEQUENCE random   NUMBER OF TRIALS 4 14
   VALUES OF TRIAL INDEPENDENT VARS  1:{T1} 15 2:      3:      4:
   TO BE LOGGED FOR LATER ANALYSIS   5:      6:      7:      8:
   RERUN ERROR TRIALS no 16
   EVENT  TYPE   FORM ID      COMMENT                MISC. INSERT FIELD
17   1  frame  3        warning stim, get ready
18   2  frame  4        present probe,get resp.
```

Figure 2. Sample BLOCK (above) and TRIAL (below) forms. These specify the selection of INSERT stimuli for the block/trial, the number of blocks/trials, the values of any block/trial variables, and the events for the block/trial.

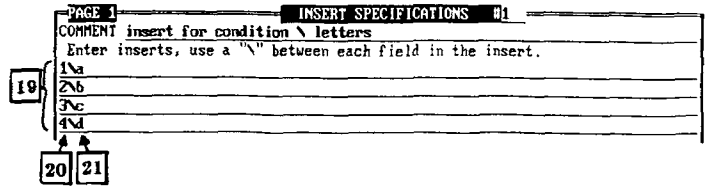

Figure 3. Sample INSERT form lists the independent variables and conditions for each stimulus that are used during each trial. In this case, the inserts have two slots separated by a " \ " character. The first slot gives the number of the condition, the second the stimulus for that condition.

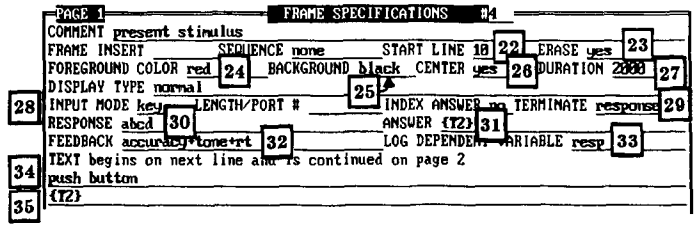

Figure 4. Sample FRAME form specifying how to display, how to collect the response, and what to display. This specifies the probe display for a choice reaction time experiment in which the subject sees the words "push button" and a letter (determined by the trial insert slot 2 "{T2}") on each trial (see text for details).

ysis. A letter [21] designates the stimulus that will be displayed. The insert "2\b" indicates a condition 2 trial in which the "b" stimulus is displayed. The selection of trial inserts is specified on the trial form. The inserts can be used on any form to either display text or set parameters on the form (e.g., the duration of a display).

The TRIALS form (Figure 2) specifies what stimuli to select for the trial, how to select the stimuli, and to what values the independent variables should be set for the trial. The TRIAL INSERT blank [12] specifies from which INSERT form to select the stimuli (in this case, from INSERT 1). The stimuli are to be selected at random without replacement [13]. They can also be selected in a fixed order. The first trial's independent variable is set to the first slot of the selected stimulus [15]. The insert slot is designated by the input "{T1}." The letter, "T," indicates that this insert was selected on a TRIALS form. The number "1" indicates the slot of the selected stimulus. Thus, if the stimulus selected for this trial is "2\b," the {T1} is a "2" and {T2} is a "b." MEL also allows inserts to be selected on BLOCK, FRAME, QUESTIONNAIRE, TEXT, and SUBJECT (for between-subject variables) forms.

The FRAME form (Figure 4) specifies the experimental display, response collection, and feedback. There are many blanks on this form that provide options to control the display presentation and response collection. For a choice reaction time experiment, the FRAME form would specify the probe stimulus. The FRAME form specifications for displaying the text include the START LINE of the text [22], ERASE [23] of the previous display, the FOREGROUND [24] and BACKGROUND [25] color, and the DURATION [27] in milliseconds.

The FRAME form specifies input responding, including the following: the input will be a single keypress [28]; the display will TERMINATE on the response [29] or duration, whichever comes first; the acceptable RESPONSE keys are "abcd" [30]; and the correct answer is determined by the trial insert slot 2 "{T2}" [31]. If the selected insert on a trial were "2\b," then the trial insert slot 2 would be "b," and thus the correct answer for the trial would be "b." The FEEDBACK specification "accuracy+tone+rt" [32] indicates that the subject will receive accuracy, tone, and reaction time feedback displays. The dependent variable to be logged for later analysis [33] is the "resp" variable. This causes each trial to log the variables of "respRT" for the reaction time, "respAC" for the accuracy, and "respSE" for the response key.

The FRAME form also includes what to display. The text "push button" [34] will appear in red (FOREGROUND COLOR [24]) on the 10th line (see START LINE [22]) of the screen centered (see CENTER [26]). The trial insert slot 2 "{T2}" [35] will appear centered on the 11th line. If the selected insert on a trial were "2\b," the subject would see a display with "push button" centered on the 10th line and "b" centered on the 11th line.

The FRAME form allows inputs from typed strings, computer mice, and external hardware (e.g., a voice key). Graphics can also be specified (see below). The display can be output so that it is blinking, is in multiple colors on color monitors; or is in reverse video, highlighted, or underlined on monochrome monitors. The display output can be synchronized to the refresh cycle of the display monitor, and the entire screen can be blanked while

the characters are being written so the subject does not see the characters being sequentially written.

Figures 1–4 show the specifications of the core procedure for a choice reaction time experiment that required the filling in of five forms. A complete experiment would add orientation displays, requiring four FRAME forms. The set of forms provides a concise, exact, humanly readable description of an experiment.

Code Generation

After the author fills in the forms in Figures 1–4, pressing the generate key (F3) causes MEL to check the specifications for consistency across forms and to write the code for the experiment. MEL incorporates an expert system-like set of rules to verify the consistency of the experiment. For example, if the author specified three independent variables for each trial and only two had been set on a trial, MEL explains the error and positions the cursor on the TRIAL form where the additional variable must be added. If the author specified a display duration that is less than a single refresh of the display monitor (17 msec for color displays and 20 msec for monochrome displays), MEL points out that such displays are not possible for standard displays. After the checks have been performed, MEL writes the source code for the experiment.

MEL converts the psychologist's form description into code, appropriately translating meaningful names into source code commands and parameters and reordering the information to produce the appropriate sequencing of commands to maintain precision timing of events. Computer source code is necessarily linear and often involves specifying commands with parameters that seem arbitrary to the novice user. In contrast, the forms are organized into conceptual categories with meaningful names.

MEL creates code that is syntactically correct. The code passed to the compiler generally compiles without errors.[6] One of the thrills of using MEL is when a user authors his/her first experiment and writes, via automatic code generation, a 200-line program without compiler errors on the first pass. This is something I doubt that I ever did using the subroutine method of experiment generation. With MEL, this is now a routine practice for new undergraduates in the laboratory.

Graphics

In many experiments it is important to provide graphic stimuli to the subject. MEL supports a full range of graphics commands that will display on over 30 graphics adapters, including the IBM standards of CGA, EGA, and VGA graphics boards. Experimenters can either specify graphics with Turtle-like graphics primitives (as in the Logo language) or import drawings that were drawn or digitized with paint programs for the PC. These can be intermixed with the text displayed with FRAME forms. On machines with advanced display controller cards (e.g., IBM EGA or VGA), the color palette can be remapped and images can be placed on virtual pages of the display board to enable rapid (1/60 of a second) full-screen display changes. This allows tachistoscopic presentation of stimuli and complete control of the size and color of the displayed images. These advanced cards also support a limited animation capacity that can be used for studies of perceptual movement and shape transformation.

Figure 5 illustrates the use of graphics for a mental rotation experiment. Cooper's (1975) subjects compared polygons rotated at increments of 60°. In MEL single-character commands can move a Turtle around the display, making shapes. For example, to draw a circle, the commands would be "HM150,120 SC100" to Hide the Turtle, Move to a point on the screen (150 horizontal, 120 vertical), Show the Turtle, and draw a Circle of radius 100 points (see Figure 5). To draw a polygon, the command is Fill Polygon with the move commands for the points of the polygon within parentheses ["FP(M109,104 ... M112,163)"; see Figure 5].

Code for Complex Operations

Experiments can require complex display and calculation requirements that necessitate writing of computer source code. An important feature of MEL is that experimenters can intermix their code with code generated

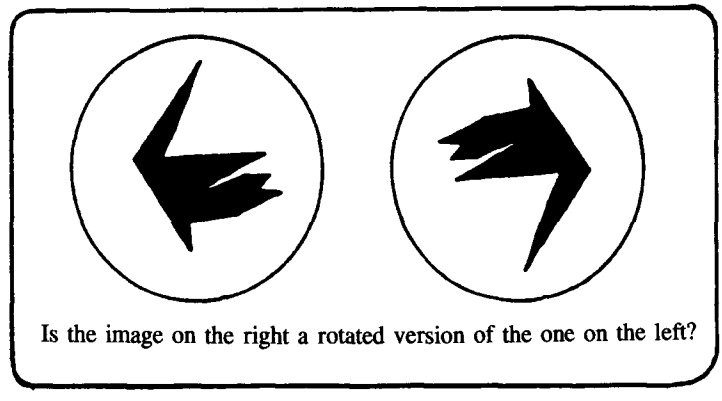

Figure 5. Example graphic display for Cooper's (1975) mental rotation experiment. The display contains text and graphics. This represents a screen dump from an IBM EGA graphics display card. See text for details.

CODE FORM FOR CIRCULAR DISPLAY

```
! Set radius of ring based on visual angle, points and length of monitor
x_ring = TAN(FLOAT(visual_angle))*FLOAT(view_distance) * x_points/x_length
y_ring = TAN(FLOAT(visual_angle))*FLOAT(view_distance) * y_points/y_length

! Set X and Y positions of the 16 points of the circular display
FOR i = 1 TO 16 DO
 BEGIN
   clock_angle = FLOAT(i) * 22.5
   x[i] = x_center + ROUND(sin(clock_angle) * x_ring)
   y[i] = y_center - ROUND(cos(clock_angle) * y_ring)
 END

! Place the letters in the letter array and randomly permute the letters
DIVIDE_INSERTS(letter,'A\B\C\D\E\F\G\H\I\J\K\L\M\N\O\P')
PERMUTE_STRING(letter,1,16)

! Set character rotation, width, height, and slant
GRAPHICS_TEXT(0,char_width,char_height,0)

! Set colorA to black, write characters sequentially in black (invisible)
GRAPHICS_COLOR(colorA)
SET_PALETTE(colorA,black)
FOR i = 1 TO 16 DO GRAPHICS_DISPLAY(x[i],y[i],letter[i])

! Wait till display refresh at top of screen, turn all char white
WAIT_TOP
SET_PALETTE(colorA,white)
```

SAMPLE DISPLAY FROM EGA MONITOR

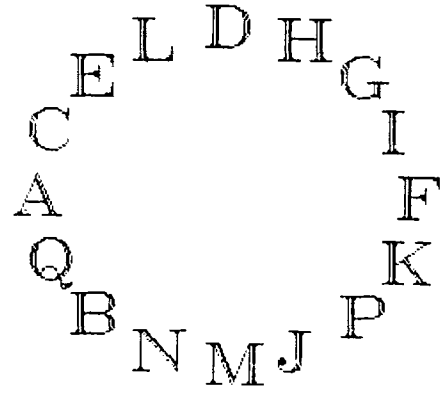

Figure 6. Sample code (top) and resulting display (bottom) for positioning characters in a circular display with a given character size and retinal distance from the fixation cross. The code positions each character on a 640 × 350 pixel grid and writes the characters individually in black so they are invisible and then turns all the characters to white to expose them simultaneously. The lines beginning with "!" are comments. MEL language keywords and function name are in capital letters; the user-defined variable names are in lowercase.

by MEL. This allows the use of forms to generate the routine code for most randomization, data logging, stimulus selection, presentation of orientation displays, response collection, and feedback. The experimenter adds code to perform nonroutine display operations, branching, and calculations. A typical complex experiment may combine 50 lines of code written by the experimenter with 500 lines written by MEL.

MEL supports a Pascal- or BASIC-like language, providing most of the flexibility available in standard computer languages.[7] Experimenters who have learned a standard computer language such as Pascal, BASIC, or FORTRAN can learn to write in the MEL language in a day. MEL supports over 200 commands, including multiple variable types (e.g., REAL, INTEGER, ARRAY__OF__STRING), control statements (IF, FOR, REPEAT), structured coding (BEGIN, END), input/output commands (READ, WRITE), functions (SQRT, LOG, COS, ARC__TAN), stimulus output (DISPLAY, CLEAR, TONE), graphics (LINE, FILL__ARC,

SET_PALETTE, GRAPHICS_FONT), timing (WAIT, LATENCY, TIME_RESOLUTION), arithmetic (+, −, *, /, REMAINDER), statistics (SUM, MEAN, STANDARD_DEVIATION), randomization (PERMUTE, RRANGE, RANDOM_STIMULUS), data logging (TRIAL_VAR, LOG_TRIAL), special input/output (PORT_IN, ARRAY_PORT_OUT), and system-maintained variables (ELAPSED_TIME, SUBJECT_NAME, TRIAL_NUMBER).

Figure 6 illustrates the use of code to place 16 characters on a clock face so all the characters are at a fixed retinal locus from the center and are presented on a single screen refresh. This code is for a display controller that supports changing the colors of pixels (i.e., dots) on the screen via altering the color palette (i.e., a set of numbers in EGA or VGA display controllers that set dots with a given color designation to a specific hue). To accomplish simultaneous presentation of all the characters, the characters are written in black first; then when the display monitor is at the top of the screen, all the characters are changed from black to white.

Text Comprehension

The text comprehension facilities in MEL allow easy measurement of reading times. The experimenter can limit and record the reading of a page, line, phrase, or word. The subject presses a key to present the next unit of text on the display. Words or phrases can be presented either on a moving window (i.e., individual words or phrases are made visible as they would appear in normal reading) or in a rapid sequential visual presentation (RSVP; i.e., individual words or phrases are presented centered on the screen). The resulting responses show a close relationship to eye movement reading times (see Just, Carpenter, & Woolley, 1982). To author a text comprehension experiment, the author fills in a TEXT form indicating the type of display format (i.e., page, line, moving window, RSVP), the maximum display time per unit of text, and the text. MEL then presents the material for the maximum display time or until the subject responds. The reading time for each unit is recorded for later analysis.

Text comprehension experimental procedures can be combined with other types of procedures. For example, one could load up short-term memory with three to five digits before presenting the text and recording reading times. To study priming, one could present the text with a TEXT form and then examine the effects on priming in a lexical decision experiment specified with a TRIAL and FRAME. Similarly, one might present some text and then use QUESTIONNAIRE forms to measure text comprehension.

Questionnaire Experiments

The questionnaire facility in MEL enables rapid generation and administration of computer-controlled questionnaires. MEL supports bipolar, multiple-choice, open-ended, and matching questionnaires. Different question types with different numbers of alternatives can be intermixed in any order. Computer administration of questionnaires eliminates the hand coding of questionnaire data that occurs in paper-and-pencil format questionnaires or the inflexibility and potential for subject coding errors that occur when using computer-readable answer sheets. Computer administration of questionnaires provides many options not available in paper-format questionnaires. Some of the questionnaire features provided in MEL include recording the reading time, returns, and changes to a question; random ordering of the questions and alternatives between subjects; limiting the time any question can be examined; incorporating color graphic images; optional forcing the subjects to answer each question sequentially and controlling whether they can return to a question; branching to questions based on a response to a previous question; subject-controlled priority marking so the subject can label questions for review; optional answer feedback; and immediate scoring of responses. The questionnaires can be combined with real-time experiments or text comprehension experiments. For example, subjects could be given hyperactivity questionnaires and then given a test of how well they can focus attention.

DATA ANALYSIS

MEL incorporates a variety of analysis options to allow rapid generation of graphs, tables, and analyses of experiments. Output options include line and bar graphs, multidimensional tables, lists, group tests, correlations, regressions, and ANOVAs. Data can also be exported to standard statistical programs, such as BMDP, SPSS, SASS, and Power Stat, and to spreadsheet programs, such as Lotus. The descriptive statistics provide means, medians, totals, variance, standard deviations, skewness, minimum, maximum, number of observations, sum of squares, proportion, cumulative proportion, and cumulative average. The ANOVAs provide for up to 10-way designs, equal and unequal numbers of observations, nested designs, repeated measures, fixed and random effects, general linear model, and planned comparisons.

The analysis system is form based to simplify specification, reduce learning time, provide self-documentation of analyses, and allow single-line running of analyses. An analysis is specified by filling in a series of forms analogous to an experiment's being specified in a series of forms. An analysis is composed of PLOT, TABLE, LIST, GROUP, CORRELATION, ANOVA, and EXPORT forms. The user interface is similar to the experiment forms. For example, defaults for routine information are pretyped, pressing the help key provides help on each blank, forms are checked for internal consistency before they are executed, and pressing the generate key causes the analysis to be performed.

An experimenter can call up previously generated forms to perform standard analyses. For example, the faculty supervisor can generate the standard analysis forms used for a paradigm. Then undergraduate subject runners can

call up the previous forms to perform daily analyses. The faculty supervisor can examine the resulting graphs, tables, and analyses to verify that the experiment is proceeding as expected. If problems appear (e.g., a subject responds with too high an error rate), the problem can be dealt with while the experiment is still progressing. In fact, with the use of command files, the analyses will be executed automatically after the subject completes a session.

INSTRUCTION WITH MEL

MEL is a powerful research tool that can be used in undergraduate and graduate instruction. It has been used in freshman experimental methods courses, graduate laboratory courses, and research seminars. MEL gives undergraduates the ability to modify and develop their own experiments even in their first laboratory course. In many psychology methods courses, students get bored after running half a dozen canned experiments. With MEL, undergraduates can develop and run experiments they design themselves. Having to design one's own experiment makes many abstract concepts (e.g., counterbalancing, subject assignment, control, number of observations, statistics) become much more concrete.

The amount of time it takes to learn MEL depends on the complexity of the experiments being executed, one's computer expertise, and the availability of other knowledgeable users. MEL is much easier to learn than traditional programming languages. A student taking an introductory computer course at the University of Pittsburgh typically spends 160 h in class, reading, and programming machine problems. Learning to author experiments in MEL is estimated to require a total of 2 h for the student to run and analyze canned experiments, 8 h to modify experiments, 16 h to generate simple experiments and do inferential statistics, and 25–50 h to produce novel experiments requiring code.[8] Introductory laboratory courses can use the system, allocating 2 to 8 h to learn the system, whereas an advanced lab may require 20 h. These estimates are provided to give the reader a ballpark estimate of how much time is needed. About two-thirds of the estimated time should be allocated for hands-on experience with the computer. If machines are limited, students can often work in pairs with little loss of learning, assuming they take turns using the computer and do not proceed until both students can perform the task. If the system is used in a classroom, it is critical that the teaching assistants know MEL fairly well (e.g., have 40 h of experience) and are available as consultants in the lab room while students do the exercises.

MEL provides seven aids to ease learning of the system and to facilitate using the system for undergraduate instruction. First, the system is form based and provides help along the way. After students have learned a few basics about editing, the experiments generally compile and run without errors. Students spend most of their time learning the structure of experiments and verifying that the experimental procedure executes as intended.

The second learning aid is that computer tutorials take students step-by-step through learning the system. The tutorials present instructions and explanations to the students while they are executing MEL programs. Learning from the tutorials is similar to having a very knowledgeable friend introduce the package and explain each option. The tutorials also verify every keystroke to keep the students from getting into trouble and indicate when an inappropriate key has been pressed. The tutorials include editing forms; generating simple reaction time, questionnaire, text comprehension, and graphics experiments; drawing graphic images; precise timing of tachistoscopic displays; randomization and counterbalancing; graphic and descriptive statistical output of data; inferential statistics for correlations, *t* tests, ANOVAs; and incorporating code in experiments. There is a manual that goes along with the tutorials to describe what will be learned and provide homework exercises.

The third aid is provided by the manuals/text books that describe how to use the system as a psychologist. There are two manuals. The *User's Guide* is written for undergraduate and graduate students with minimal computer experience. It describes how to author experiments in MEL, research methods of using computers in psychology, test experiments to verify the procedure, deal with human subjects in computerized experiments, analyze data, and schedule subjects. Experimental procedures that are frequently used in psychology experiments are described. Runnable source code is provided for procedures, including randomizing stimuli, counterbalancing, tachistoscopic displays, dual task procedures, stimulus onset asynchrony, voice key input, and up-down psychophysical procedures. The *Language Reference Manual* (over 200 pages) details all the commands in the language and provides sample program segments. This is designed for advanced users who have previously learned some programming language, such as BASIC, Pascal, FORTRAN, or C.

The fourth aid is an experiment library with a laboratory manual. The library contains 20 experiments, including many classics. These are included in source form so that students may adapt the experiments as exercises. The laboratory manual includes exercises to help the student understand the phenomena being tested. The experiments include sensory icon decay, short-term memory, sentence processing, problem solving, verbal learning, mnemonics, personality tests, free recall, recognition memory, human factors design, reading comprehension, blind spot mapping, visual search, automaticity, mental rotation, signal detection, organization of memory, and spacing effects.

The fifth aid is the provision of computer tests that examine students' knowledge of the MEL. To be a skilled researcher, one must know the precision and flexibility of one's research tools. The tests are provided as MEL questionnaires. They assess how well a student knows the system and provide a laboratory director an objective as-

sessment of whether students should be trusted to author experiments on their own.

The sixth aid is a public domain experimental library of experiments that run with MEL. Researchers are encouraged to submit experiments to the library. These are distributed at cost and may be copied freely. A department can collect a large experimental library that students can use to demonstrate experiments on themselves and to develop extension experiments.

The seventh aid is an instructional tips pamphlet that provides comments on how to set up an undergraduate or research laboratory. The pamphlet includes suggestions on hardware, software, special interfaces (e.g., voice keys), subject booths, readings, and a collection of comments from other instructors.

ADVANCED FEATURES

MEL is an open architecture, making it possible to combine MEL code with other code to enhance its operation. MEL can call subroutines written in standard programming languages such as Pascal, C, assembly, and FORTRAN. This simplifies supporting special-purpose devices. For example, a researcher might want to incorporate a special display controller, eye movement monitor, and brain wave recording. This can be done by writing device drivers in a standard programming language and calling these routines from within MEL. Writing device-driven code requires having a skilled programmer available. Sample programs are provided to illustrate how to extend the MEL architecture.

COMPATIBILITY AND EQUIPMENT REQUIRED

MEL will run on most IBM PC, XT, AT, or PS/2 or compatible computers. The minimal subject development station is an IBM PC with 384K of memory, a floppy disk drive, and a monochrome monitor. The recommended subject station is an IBM PC with 640K, a floppy disk drive, EGA graphics adapter, and monochrome or color monitor. The recommended development/analysis station is an IBM XT with 640K, floppy disk, hard disk, EGA graphics controller, monochrome or color monitor, floating point chip, and dot matrix printer. MEL will support a pen plotter for plots if it is available.

MEL operates on most IBM PC clones. It has been tested on all of the IBM product line and individual models from Leading Edge, AT&T, Zenith, Radio Shack (other than the 2000), Compaq, and PC Limited. However, there are many off-brand clones, and some are not compatible. A researcher should run the demonstration programs for MEL to verify compatibility.

MEL is compatible with a wide range of display controllers. Over 30 display controller boards are supported. The compatibility with the IBM line of controller boards includes the CGA, EGA, and VGA cards. The CGA boards provide only very limited graphics support. The EGA and VGA boards provide support for remapping the color palette and writing to virtual pages allowing rapid display changes.

SUMMARY

MEL is a third-generation integrated software system for experimental research. MEL is optimized to take full advantage of the 640K memory, hard disk, and graphics adaptors commonly available on IBM PC computers. It incorporates fourth-generation programming techniques and a forms-based interface that are not available in any previously reported psychological research packages. MEL is designed for executing professional-level psychological research. MEL is very flexible and can program most experiments without code, and nearly all with some code. MEL maintains millisecond precision of timing intervals. MEL is limited to paradigms that have an intertrial interval in which some variability (e.g., 0.1 sec) can be tolerated. MEL enables rapid experiment development, provides experimental precision and speed, allows great flexibility and range of experimental techniques, implements quality control assurance, is usable by nonprogrammers, reduces training time needed to learn to author experiments, facilitates communication of experimental procedures with colleagues, speeds data analysis, and allows researchers to quickly and cheaply set up professional laboratories. Experiments are specified in MEL by filling in forms. Reaction time, text comprehension, and questionnaire experiments are supported. MEL incorporates an analysis package providing graphs, tables, lists, inferential statistics, and data export to other statistical packages. Learning the system is facilitated by a forms interface, computer tutorials, system manuals, a laboratory manual with sample experiments, student knowledge tests, a public domain experimental library, and an instructional tips pamphlet. MEL is an expandable system allowing incorporation of code written in other languages. The system is compatible with most IBM PC clones and graphics controllers.

MEL is a tool that allows a psychology research laboratory to utilize staff time to do psychological research rather than expending a great deal of effort training programming skills and developing software. The design of MEL incorporates the lessons learned during 15 years of experience in computerized psychological research over three generations of computer equipment. The speed and ease of use of the system can empower students and researchers with minimal computer skills to develop, execute, and analyze complex computerized experiments in short periods of time.

REFERENCES

BUTLER, D. L. (1988). A critical evaluation of software for experiment development in research and teaching. *Behavior Research Methods, Instruments, & Computers*, **20**, 218-220.

COOPER, L. A. (1975). Mental rotation of random two-dimensional shapes. *Cognitive Psychology*, **7**, 20-43.

EAMON, D. B. (1982). CEDATS: A cognitive experimental design and testing system. *Behavior Research Methods & Instrumentation*, **14**, 142-145.

JUST, M. A., CARPENTER, P. A., & WOOLLEY, J. D. (1982). Paradigms and processes in reading comprehension. *Journal of Experimental Psychology: General*, **111**, 228-238.

OGDEN, W. C., & BOYLE, J. M. (1982). Evaluating human-computer dialog styles: Command vs. form/fill-in for report modification. In *Proceedings of the Human Factors–26th Annual Meeting* (pp. 542-544). Santa Monica, CA: Human Factors Society.

OSGOOD, G. (1985). *Apple-Psych software program* (Report No. 6). Eugene, OR: University of Oregon, Cognitive Science Laboratory.

OSGOOD, G. (1988). Generalizing the Apple-Psych system. *Behavior Research Methods, Instruments, & Computers*, **20**, 155-157.

POLLER, M. F., FRIEND, E., HEGARTY, J. A., RUBIN, J. J., & DEVER, J. J. (1982). *Handbook for writing procedures* (Tech. Rep. selection code 700-242). Indianapolis, IN: Western Electric Distribution Center (IDC).

POLTROCK, S. E., & FOLTZ, G. S. (1982). An experimental psychology laboratory system for the Apple II microcomputer. *Behavior Research Methods & Instrumentation*, **14**, 103-108.

POLTROCK, S. E., & FOLTZ, G. S. (1988). APT PC and APT II: Experiment development systems for the IBM PC and Apple II. *Behavior Research Methods, Instruments, & Computers*, **20**, 201-205.

SCHNEIDER, W., & SCHOLZ, K. W. (1973). Requirements for minicomputer operating systems for human experimentation and implementation on a 4K PDP-8 computer. *Behavior Research Methods & Instrumentation*, **5**, 173-177.

SCHNEIDERMAN, B. (1987). *Designing the user interface: Strategies for effective human*-computer interaction. Reading, MA: Addison-Wesley.

SCHOLZ, K. (1972). Computerized process control in behavioral science research. *Behavior Research Methods & Instrumentation*, **4**, 203-208.

SCHRIVER, K. A. (1984). *Revising computer documentation for comprehension: Ten exercises in protocol-aided revision (CDC Tech. Rep. No. 14). Pittsburgh, PA: Carnegie-Mellon University, Communications Design Center.*

SMITH, S. L., (1982). *User-system interface design for computer-based information systems* (Tech. Rep. ESD-TR-82-132). Bedford, MA: USAF Electronic System Division.

YOURDIN, E. (1975). *Techniques of program structure and design*. Englewood Cliffs, NJ: Prentice-Hall, Inc.

NOTES

1. The PDP-11s can address much more memory, but the compilers provide very limited support of the extended memory.

2. MEL will operate with 384K with two floppies; however, it is optimized for 640K with a hard disk.

3. MEL is not appropriate for tasks with continuous input and feedback, such as a psychomotor tracking task.

4. The demonstration copies are limited to demonstration and do not allow data collection.

5. This concern was pointed out to me by Danny Kahneman at the University of California.

6. If the users add their own code (see below), they may produce syntactical errors that the compiler will detect.

7. The language does not support recursion or record-type definition as does Pascal.

8. These estimates are based on informal observations at four test sites that have used MEL for instructing undergraduates. Note the variability is large, with the standard deviation being probably half of the mean. The upper limit is typical of students without programming experience, the lower limit of students who have had at least one programming course. If the students need only modify previously written code rather than write the code from scratch, nonprogrammers can use code after 30 total hours of instruction and practice.