Abstract:

CAI course authors have been faced with the distinct problem of having to learn an instructional coding language before they can get their courses into the computer. A system has been devised so that an author may easily write his course in English on course planning forms and then a pre-processor will generate the coding which will be input for the machine assembler.

Limited Distribution Notice

This report has been submitted for publication elsewhere and has been issued as a member of this Technical Report Series for early dissemination of its contents. As a courtesy to the intended publisher, it should not be widely distributed until after the date of outside publication.

Toward a True Author Entry System for CAI

by

M.W. Dowseyn

Series Editor: N.S.M. Cox

Number 5

November, 1969

© 1969 University of Newcastle upon Tyne.
Printed and published by the University of Newcastle upon Tyne, Computing Laboratory, Claremont Tower, Claremont Road, Newcastle upon Tyne, NE1 7RU.
DOWSEY, Michael William

Toward a true author entry system for CAI. By Michael W. Dowsey. (University of Newcastle upon Tyne, Computing Laboratory, Technical Report Series, No. 5).

Newcastle upon Tyne, University Computing Laboratory, 1969.

12"

Added entries

NEWCASTLE. University. Computing Laboratory.
Technical Report Series, 5

Suggested classmarks (primary classmark underlined)

Library of Congress:
Dewey (17th): 371.3 3078 510.7834
U.D.C.: 681.3 3:371.3

Suggested keywords

AUTHOR LANGUAGES
COMPUTER-ASSISTED INSTRUCTION

Abstract

CAI course authors have been faced with the distinct problem of having to learn an instructional coding language before they can get their courses into the computer. A system has been devised so that an author may easily write his course in English on course planning forms and then a pre-processor will generate the coding which will be input for the machine assembler.

About the Author

M.W. Dowsey is a Research Student in the University Computing Laboratory. The work reported here was undertaken during a Student Training Programme at IBM Corporation, Education Research Department, San José, California, and was supported by IBM(UK) and the Science Research Council.
Introduction

Ever since the advent of Computer Assisted Instruction, the course author has been faced with a new problem. Not only must he have a thorough knowledge of the subject content of the course and be able to reach his intended goals by dint of good presentation and meaningful questions, but he must also know how to effect this objective by writing in an instructional computer language somewhat foreign to his nature. In other words, he must first learn all about the instructional programming language he is to use and what limitations are put on it by the particular computer he is to use, quite a time-consuming activity for an absolute beginner in the art of programming.

As can be imagined, few authors have been persuaded to take up instructional programming for computers when they have had no previous experience. Whereas, for programmed instruction, an author could produce course material with only the help of an editor and, of course, the printing team, with present instructional coding languages he may require a whole team about him to carry out the many stages of getting his course from paper to computer memory. First, he must have an editor to search through his subject matter and English, and then a keypunch operator to produce hard copy of the content to pass on to a coding editor whose sole job is to convert the written English into valid computer program statements. If not coded on-line, these are punched into cards and entered into the computer by an operator and checked for validity in the programming language. If errors exist then, someone must debug the program, make the necessary changes and recheck. The course is then passed back to the author who must check it for inconsistencies in content and logic before it is released to the students.

The above paragraph leads to the conclusion that 'author-entry' languages for CAI are in fact coding languages. The remainder of this paper will report on a new system originally suggested by Dean and Grubb (1969) which will allow an author who has not yet been persuaded to learn an instructional coding language to produce course material without needing so much time and effort from other people. This system allows the generated material to make effective use of the power of a CAI Instructional System which, in this instance, happens to be Coursewriter II for the IBM 1500 System. However, the problem and proposed solutions are generic.
The System also dispenses with the necessity of having a coding editor to produce Coursewriter II statements and then someone to debug the program for programming language errors. This is achieved by simply providing the author, or his secretary, if he has one, with a Course Planning Form on which he may enter his subject presentation, his questions and expected answers, and the corresponding courses of action, in an almost unlimited format.

The Course Planning Form

Up to the present, most authors have prepared course material in one particular format, namely, the presentation of single frames to the student. Then, if he answers successfully, he is allowed to continue to the next frame in sequence, but if he does not, he sees a frame containing remedial information. As this set format for course preparation is in use, the system desired provides a process which will quickly provide the course materials of such a format. Thus, the objective is that any author should be able to prepare CAI course material in a form that, when punched on to cards, is immediately converted to Coursewriter II by this system, which is a pre-processor.

The layout of the course planning form was determined by the technical specifications of the 1510 CRT Display Unit. The 1510 is divided into 32 rows and 40 columns of addressable spaces. However, as two rows are required per character, the 1510 effectively displays 16 rows by 40 columns. This determined the layout of the planning form inasmuch as one space is provided for each character (author-oriented), but half-line shifting for superscripting or subscripting is still available despite the fact that spaces do not exist for this purpose on the form.

However, as has been suggested by Dean (1968), far from imposing a restriction upon the author by insisting he use a 16 by 40 form, its use may produce effective CAI teaching in that most course-authors, accustomed to working with print, are too verbose. They have the tendency to display too much on a single frame, and by generating on ordinary paper material which is later transferred to a planning form, they lose the unity of a frame which must now be spread over several displays.

The form itself is divided into four sections, each easily identifiable.
1. The Identification part, which is obligatory.

2. The Presentation part, which, although not mandatory, is used if any instructional material is to be shown.

3. The Decision part, which decides which frame the student will see next.

4. The Response analysis part, in which the author specifies his contingency prescriptions.

Sections 3. and 4. are alternatives and on no account must they both be filled out.

In an effort to make the form easier to fill in, mandatory parts have been assigned solid boxes or underlining, whereas optional parts are indicated by broken lines.

The numbers contained in parentheses under each entry are indicators of the particular card columns needed for that piece of information. An optical scanner would permit one to bypass this operation completely.

To correspond to the four Sections there are only four types of cards produced: one only for section 1, any number (including zero) for section 2, one only for section 3, and at least two for section 4 - one similar to that for section 3, plus at least one for the response analysis.

Naturally, it is expected that the form, being somewhat of a general nature, will not allow the author to create all types of teaching procedures. However, once the author has become proficient in the instructional coding language, he may then use hand optimization techniques on the code produced from the form via the pre-processor.

Notes for Authors

The following explanatory notes correspond to numbers on the form itself, only there in this instance as an aid. (See Figure 1)

1. The page label MUST appear on every sheet except a continuation sheet (see 9.2.5). It is automatically displayed in columns 34-39 of row 0 to aid the author in debugging. Its format is as follows.

The page label is to be an unsigned integer optionally followed by a string of alphabetic characters, either upper or lower case, the total length being not more than six characters. The purpose of insisting on unsigned integers as labels is to provide notation in ascending order of magnitude, which is the logical sequence of the course written by the author. The first page label in a particular
frame will be the unsigned integer itself but any subsequent pages may be labelled by the unsigned integer followed by a string of letters. This is to allow the author freedom to use learner control techniques as suggested by Grubb (1968). Among other things the student will be able either to branch backward to the last logical frame or skip forward to the next logical frame.

2. Before presentation of any text the author specifies which rows (if any) he wished to be erased.
   2.1 Here he enters the first row of the sequence - it is a number greater than or equal to 0 but less than or equal to 31. If there is no entry a default value of 0 is assumed.
   2.2 Here he enters the last row of the sequence, also a number greater than or equal to 0 but less than or equal to 31.
   Naturally, the number entered in (2.1) must be less than or equal to that entered in (2.2). If no entries at all appear, all 32 rows are erased. Both entries must be zero to produce no erasure at all.

3. If the author desires this point in the course to be a restart point, he checks the box. (At a restart point all current information about the student and the course is stored away in the 1500's student record file. Consequently, if he discontinues, whether through choice or system failure, the system will restart him at this point, with all up-to-date information, when he does decide to return.)

4. The author fills in the form in exactly the way he wishes to present his material. He must not forget that each character requires TWO rows on the screen, but to aid him the form is divided into double rows.
   4.1.1 He writes small letters, capital letters, punctuation, underlining, subscription, superscription, etc., as required.
   4.1.2 To denote where the cursor is to appear on the CRT and the subsequent answer space for a keyboard response, the author fills in the appropriate positions with a 'â€œ'.
   4.1.3 A short-hand form of this is to mark the cursor position with a 'â€œ' and then draw a line throughout the rest of the required answer space.
4.1.4 To denote a light patch area for a light pen response the author fills in the appropriate positions, as shown. Of course there are some necessary restrictions:

4.2.1 The author may not specify both a keyboard and a lightpen response simultaneously.

4.2.2 For a keyboard response, there may be only one continuous answer space; in other words, no embedded characters or blanks. However, responses larger than one line are permitted.

5. The author has the option to specify pause times, in seconds, for each line of the display. The pause provided by the pre-processor after the last line continues until the student presses the space bar to continue.

6. The author should complete either the Decision section of the Response Analysis section, not both. If there was no response required of the student, then the author must fill in the Decision section, but should check only ONE of the branching boxes.

6.1 The 'return point' box is checked in situations such as the following. An author may wish to generate the same comments whenever he receives a correct response. To save repetitive generation of the same coding he may initially jump to a frame where the comments are generated and from that frame 'return' to the next logical frame from which he made the initial jump.

6.2 The 'next logical frame' box is checked when the flow is to be to the first frame in the next logical section. For example, it may be used when there was no question asked, but merely text presented, at label 6. Then the 'next logical frame' is 7. Or, if the author was supplying the correct answer at label 6xy, then the 'next logical frame' is also 7.

6.3 The 'last question' box is checked when the author wishes the student to attempt the last question again. Usually the current frame will be some remedial hint after a non-correct response was diagnosed.
7. The Response Analysis section is filled in when response processing is required after a question. The 'frame named' box is checked and there MUST follow at least one entry in the matching specification list.

8. Here the author enters the page label of the frame that is to follow if the student's response is successfully matched. Naturally it follows the same format as the page label entry in 1.

9. To determine the type of response processing required the author writes into 9(a) one of the following letters: E, K, N, P or U; and then places the actual 'required response' characters into 9(b).

9.1.1 E – an exact keyword match is required. The characters, including any required blanks, are written in 9(b). If 9(b) is left blank, then any string the student enters will provide a match.

9.1.2 K – to allow for misspelling, etc., a 'kernel' match is allowed, but the essential characters being sought (NO blanks) are written in 9(b). For example, if the author seeks 'fortran' then he may enter 'ftrn' in the hope that he may obtain a match even allowing for incorrect characters, missing characters, etc., usually caused by misspelling or typographical errors.

9.1.3 N – this allows a search for a particular numerical value. Two types of checking may be specified;

9.1.3.1 If an exact value is requested, the author specifies the value. A match will occur if the response contains a numerical constant within the interval formed by the value plus or minus half the least significant power of ten given in the value.

9.1.3.2 If a range will suffice the author enters lower bound and upper bound separated by '-' and a match will occur if the response contains a numerical constant greater than or equal to the lower bound and less than or equal to the upper bound.

Note: if the author requires one exact value, i.e. no possible error, then he must use checking of type 2 and make the lower and upper bounds both equal to the exact value sought.
As well as the choice of numerical check the author may logically 'OR' some of each or both by separating each value or range by a ',' . E.g. in 9(b) he may enter: 
-273, 31.2–37.6, -21–-18, 98.4.

9.1.4 P - this signifies a light pen response and in 9(b) the author enters the co-ordinates in the form 'row, column' of any point in the light patch with which he wishes to associate this attempted match.

9.1.5 U - for an unanticipated response. Here 9(b) is left blank.

9.2 A few conditions must be adhered to:

9.2.1 Except for E and U there must be an entry in 9(b) - no default.

9.2.2 Light pen response matching cannot be mixed with any type of keyboard response matching.

9.2.3 For keyboard response matching all 'U's must appear last in sequence, i.e. no E, K or N may follow a U.

9.2.4 For light pen response matching only P may be entered in 9(a) - no U's are allowed. However, the pre-processor produces code which displays on row 30 the suggestion that the student re-answer if he points anywhere except to a specified light patch.

9.2.5 There is no limit to the number of response processing entries there may be. If one form is insufficient, the author may use further forms.

10. To aid in easy recognition of how the student answered, which is supplied in the computer-listed student-response record, a two character response identifier may optionally be entered by the author (e.g. 'cc' for correct or 'ww' for wrong answers - in default 'aa' will be entered).
Notes for Keypunch Operators

There are four types of card, each corresponding roughly to each section of the form, if used. Each input deck is ended by a card containing only '*' in column 1.

1. Identification card
   1.1 Page label: Columns 1-12; use '<' for upshift, '>' for downshift.
   1.2 Start line for erase: Columns 16-17; for 1-digit number use either column.
   1.3 Finish line for erase: Columns 19-20; for 1-digit number use either column.
   1.4 Restart point: any character other than blank in Column 22. All other columns MUST be blank.

2. Presentation card
   The keypunch operator must begin the line at the lowest row number and work down the form if necessary by indexing. This is important, of course, when superscripting has been used.

   for upshift use '<'
   for downshift use '>'

   for keyboard response use 'α' - where author's short hand notation is used all 'α's must be entered
   for light pen response use ':"'
   for index use '│'
   for reverse index use '"'
   for backspace use '%'
   for multiply sign use '⊕'
   for divide sign use '<@'

2.1 Row number: Columns 1-2; for 1-digit number either column may be used. If the keypunch operator does not specify the row number it will default to the 'last one used' +2.

2.2 Text: Columns 6-71, using above notation.

2.3 Continuation: any character in Column 72. This will be necessary if many upshifts, downshifts, backspaces, etc., are needed.

2.4 Pause duration: Columns 75-80. For less than 5-digit numbers any consecutive columns out of columns 75-80 may be used.
3. Decision card

Column 1 only is used on this card; all others should be blank.
3.1 For 'return point' enter 'r'
3.2 For 'next frame' enter 'n'
3.3 For 'last question' enter 'q'
3.4 There is also a fourth alternative which is used when the
    Response analysis section has been filled in. It is the first
    card corresponding to that section: thus, for 'a frame named ...'
    enter 't'.

4. Response analysis card

4.1 Page label: Columns 1-12.
4.2 Type of response matching: in column 14 either 'e', 'k', 'n',
    'p', or 'u' is entered.
4.3 Text: Columns 16-71.
4.4 2-character response identifier: Columns 75-76.

All other columns MUST be blank.
The Action of the Pre-Processor

The pre-processor was written in FORTRAN for three reasons. First, it is a universal language. Second, the ease and speed with which FORTRAN may be debugged was a strong recommendation for its use. Third, as FORTRAN is an easy language in which one programmer can communicate with another, this determined its use since the project was to be reasonably open-ended so that extra facilities and further development could easily be tackled in the future. At the present time it comprises about 1500 basic FORTRAN IV source statements.

The pre-processor's action is as follows:

1. Read in an Identification card. If last card, pass to 12.
2. Check for a valid label name, whether or not erasure is requested and whether or not a restart point is required. Check parameters for COURSEWRITER II 'de'(i) instruction. If no errors exist, write each card image to disk and give listing on printer. Otherwise give appropriate diagnostic message with card number and, in some cases, column number. Also increment the error count. In any case pass to 3.
3. Read in a Presentation card. If last card, pass to 12.
4. Check to see if it is not a Presentation card but a Decision card. If it is a Decision card pass to 6.
5. Check for valid row number (or defaulted row number), response requests, text presentation and pauses. Check parameters for 'dt' (ii) instruction. Either write to disk or give diagnostics. Pass to 3.
6. Check to see if Decision section has been filled in or the Response analysis part. If the former, generate the appropriate branch instruction and pass to 1.
7. Read in a Response analysis card. If last card, pass to 12.
8. Check to see if there is a response type entry or not. If not, it must be an Identification card, so pass to 2.
9. If this is not the first Response analysis card of this series, pass to 11.

(i) display erase.
(ii) display text.
10. From the information given by the Presentation cards, generate the COURSEWRITER II Instructions 'dl'(iii) (only if the keyboard response is a one line insertion) and 'epx'(iv) (where x is blank, for a keyboard response; 'i', for a keyboard insertion response; or 'p', for a light pen response), first checking out the parameters. Write to disk or give diagnostics as necessary. Immediately before the 'epx' the label name of the next logical label in sequence is loaded into return register 0 so that any future branching back to 'return point' will be meaningful. Just after a keyboard 'ep[i]', a macro call of 'ercal' (see section 'Macros Used') is generated.

11. Check for valid label name, see which type of response analysis is requested and generate, if possible, the appropriate instructions.

For 'e' — 'aa'(v) is used
   'k' — a 'ld kernel characters'(vi) followed by a call of function 'keyl'(vii)
   'n' — a call of function 'lt'(viii)
   'p' — 'aap'(ix) is used
   'u' — 'un'(x) is used

However, before the first 'un', a macro call of 'ermany' (see section 'Macros Used') is inserted. After the particular instruction(s) specifying the analysis is generated, the branch to label name is formed. Either write to disk or give diagnostics. Pass to 7.

12. Check the error count to see if there have been any errors. If so, terminate; but otherwise produce a punched card deck from the card images resident on disk, and then terminate.

(iii) display emphasis line.
(iv) enter and process response from keyboard, enter and process response insert from keyboard, enter and process response from light pen.
(v) additional answer.
(vi) load text into buffer.
(vii) keyletter.
(viii) limit.
(ix) additional answer (light pen).
(x) unrecognizable response.
The pre-processor will only find one error per card. When one is detected, the card is 'rejected' and the pre-processor reads in the next card. Naturally, this may produce further errors, but only in the current frame. For each error an appropriate diagnostic message is listed, and the card number, and, in some cases where meaningful, the column number, is also given.

Macros Used

The research which is currently being undertaken by the IBM Education Research Department, San Jose, is geared strongly towards the use of Learner Control techniques.

The main effect for the student is to allow him to move freely throughout the course, whether to skip forward, move backward, proceed to the glossary, return to the course outline map, etc. Consequently, in an effort to co-ordinate the course material produced by the pre-processor to suit the requirements of the learner control coding, the macro 'erhmany' is called before an unanticipated response is assumed. This searches the response to see if the student called upon any of the above mentioned utilities provided for him and acts accordingly. The three parameters necessary for the call (the author does not concern himself with this) are the 'current frame', the 'next logical frame' and the 'last logical frame'. As the author must use sequential numbering for his logical frames those parameters are readily available.

A second facility provided is that in which the student's response, if it contains an arithmetic operator, is fed into an arithmetic syntactic analyzer from which the equivalent reduced value is returned to the calling program to be checked in the response analysis. To effect this, a call is made upon the macro 'ercalc' immediately after the response is entered. No parameters are required for this macro.

Sample Input and Output

There follows part of a simple example demonstrating the use of the pre-processor. In figure 2 is some of the author's input on the course planning form. Figure 3 shows the printout of the keypunch operator's card deck. The corresponding output from the pre-processor is shown in figure 4 with the COURSEWRITER II instructions and also the diagnostic messages.
I'm sure you know that the victor was William the Conqueror. But can you point to his nationality?

English, French, Norman

After this frame, the student should go to:

- The Return Point
- The Next Logical Frame
- The Last Question

(1)

A Frame Named ...

(1)

Enter E, K, N, P or U.

| 3a | if his P response was 20, 16 |
| 3b | if his P response was 21, 9 |
| 3c | if his U response was 23, 32 |
| 3c | if his U response was 16, 71 |

(1-12) (14) (16-71) (75) (76)

Figure 2
PAGE LABEL 3a

From row (16-17) to row (19-20) erased. Restart point? Check if required.

Pause Time in seconds (75-80)

Text (6-71) (72 is continuation)

You certainly know your English history.

After this frame, the student should go to:

The Return Point
The Next Logical Frame
The Last Question

A Frame Named...

Enter E, K, N, P or U.

if his response was

if his response was

if his response was

if his response was

if his response was

if his response was

(1-12) (14) (16-71) (75) (76)

Figure 2 continued
Keypunch operator's card deck

14  (D>‘M SURE YOU MUST KNOW THAT THE VICTOR WAS <W>ILLIAM THE <C>ONQUEROR. <B>UT CAN YOU POINT TO HIS NATIONALITY?  

<ENGLISH <F>RENCH <N>ORMAN

28 <Y>OU CERTAINLY KNOW YOUR <E>NGLISH HISTORY.

28 <V>ERY NEARLY <H>E CAME FROM <F>RENCE BUT WASN’T EXACTLY <F>RENCH. <P>LEASE TRY AGAIN.

28 <G>OOD HEAVENS, <P>ERhaps YOU HAD BETTER READ UP ON THIS.
COURSEWRITER II INSTRUCTIONS AND DIAGNOSTIC MESSAGES

PRR

DT 0, 34/-/-/-/3

DTT 14, 0/-/2, 14/-/38, 0/-<I> MUST YOU KNOW THAT THE VICTOR

DTT 16, 0/-/2, 16/-/34, 0/-<W> WILLIAM THE <C> ONQUEROR. <B>UT CAN

DTT 18, 0/-/2, 18/-/29, 0/-YOU POINT TO HIS NATIONALITY?

PA 100

DTT 20, 1/-/2, 20/-/16, 1/-/-/-/V/-V/-V/-V

DTT 22, 1/-/2, 22/-/16, 1/-/-/-/V/-V/-V/-V

DTT 24, 0/-/2, 24/-/23, 0/-<E> ENGLISH <F> RENCH <N> ORMAN

PAE

LR 4/-/KRO

EPP 9999/-/3

AAP 4, 20, 2, 15/-/N

BR 3A

AAP 4, 20, 2, 8/-/F

BR 3B

RESPONSE REQUEST INCONSISTENT WITH RESPONSE INDICATIONS

CARD NUMBER 53

ONLY 'P' SHOULD BE ENTERED WHEN LIGHT PATCH HAS BEEN INDICATED

CARD NUMBER 54

3A

DT 0, 34/-/-/-/3A

DTT 28, 0/-/2, 28/-/40, 0/-<Y>OU CERTAINLY KNOW YOUR <E> ENGLISH HISTORY

PAE

BR 4

3B

DT 0, 34/-/-/-/3B

DTT 28, 0/-/2, 28/-/36, 0/-<V>ERY NEARLY <H> E CAME FROM <F> RANCE BUT

DTT 30, 0/-/2, 30/-/40, 0/-<W>ASN'T EXACTLY <F> RENCH. <P>LEASE TRY AGAIN

PAE

BR KE

EMBEDDED BLANKS NOT ALLOWED IN LABELS

CARD NUMBER 62 COLUMN NUMBER 2

DT 0, 34/-/-/-/3B

DTT 28, 0/-/2, 28/-/36, 0/-<G> OOD HEAVENS. <P> ERHAPS YOU HAD BETTER

DTT 30, 0/-/2, 30/-/16, 0/-<R> EAD UP ON THIS.

EN

YOU ENTERED 64 CARDS.

116 COURSEWRITER II STATEMENTS HAVE BEEN PRODUCED

AND YOU HAD 5 ERRORS.

CARD OUTPUT HAS BEEN SUPPRESSED.

Figure 4

19
Conclusion

The system reported is a prototype of many possible other systems. Even written in FORTRAN here, any other suitable language may be used and the production of COURSEWRITER II Instructions is incidental to the facilities available in this Department. Using the same principles, pre-processing course planning forms should be possible for any instructional coding language.

References