ED 071 452 EM 010 718

AUTHOR	Gregory, Carl; Bessent, Authella
TITLE	A Conceptualization of an Improved Authoring Language (IAL). Technical Report Number 11.
INSTITUTION	Texas Univ., Austin. Computer-Assisted Instruction Lab.
SPONS AGENCY	National Science Foundation, Wa. ngton, D.C.
PUB DATE	Sep 71
NOTE	21p.
EDRS PRICE	MF-$0.65 HC-$3.29
DESCRIPTORS	*Computer Assisted Instruction; Program Descriptions; *Programing; *Programing Languages
IDENTIFIERS	IAL; *Improved Authoring Language

ABSTRACT

One of the most difficult tasks in program production of computer-assisted instruction (CAI) has been the transformation of an author's ideas into machine-usable form. Thus, an improved authoring language (IAL) to facilitate the transformation of authors' drafts to machine-usable form was conceptualized. IAL was designed so that an instructional designer can help the author give his course a coherence that will meet programing needs. IAL requires data to be specified completely before the material reaches the programer. The language is a series of commands, many of which could be standard routines in the eventual programing language. Each piece of courseware data is identified, or labeled, so that it can be referenced. Data may be either LOCAL (defined for a given page, template, mudule, or unit), GLOBAL (defined for a given student), or UNIVERSAL (defined for the entire course). Commands are applicable to any subject matter. Many parameters are formulated only loosely, so that authors can, at times, specify material in sentence form. (Author/JK)

THE UNIVERSITY OF TEXAS AT AUSTIN

Computer Assisted Instruction Laboratory

AUSTIN

A CONCEPTUALIZATION OF
AN IMPROVED AUTHORING LANGUAGE (IAL)

TECHNICAL REPORT NO. 11

Carl Gregory and Authella Bessent

September 1971

Supported By:

THE NATIONAL SCIENCE FOUNDATION
Grant GJ 509 X

The University of Texas at Austin
Computer-Assisted Instruction Laboratory
Austin, Texas 78712

Abstract

In previous computer-assisted instruction (CAI) development efforts, one of the most difficult and trying tasks in program production has been the transformation of an author's ideas into machine-usable form. Programmers have been required to make decisions about data structures, branching logic, and control procedures because authors were not able to anticipate precise specifications when the course was being written. Thus, an improved authoring language (IAL) to facilitate the transformation of authors' drafts to machine-usable form was conceptualized.

The design of IAL is such that an instructional designer can aid the author in giving his course the coherence that will meet programming needs. The improved language thus serves as a formal structure that the authoring staff can utilize to organize the author's ideas. Providing a "programming language" structure, IAL requires data to be specified completely before the material reaches the programmer.

The language is written as a series of commands, many of which could be standard routines in the eventual programming language. Each piece of courseware data which is manipulated at any one time must be identified (labeled) so that it can be referenced. Data may be LØCAL (defined for a given page, template, module, or unit), GLØBAL (defined for a given student), or UNIVERSAL (defined for an entire course). Commands are applicable to any type of course, but specific parameters or conditions may be unique to a certain type of subject matter. Other commands may be defined as needed by a given course or a given installation. Many parameters are only loosely formatted, so that authors can, at times, specify material in sentence form.

Introduction

In previous computer-assisted instruction (CAI) development efforts, one of the most difficult and trying tasks in the production of CAI programs has been the transformation of an author's ideas into machine-usable form. Given the available programming languages for a certain computer, the programming staff has usually borne the burden of understanding the author's material and ordering that material logically so that the program may be written. Unfortunately, teachers do not always prepare course material according to programming logic, and are not often aware of the procedures and data that must be soecified completely in order for their course to be coded. Programmers have had to make decisions about data structures, branching logic, and even control procedures because authors have failed to specify them when the course was being written.

On the authoring staff, an instructional designer can be utilized to aid the author in giving the kind of coherence to his course that will fill programming needs. In this regard, the improved language serves as a formal structure that the authoring staff will utilize to order the author's ideas. By giving standard methods in indicating procedures, the language can reduce ambiguity in specification. With a "programming language" structure, the improved language will require data to be specified completely before the material reaches the programmer.

The authoring staff, then, will write a "pre-program program," and so produce material for the programming staff that requires no "debugging" of logic or interpretation. This leaves the programmers free to work with their unique problems of coding, keypunching, loading, and system considerations.

Description

The language is written as a series of commands, many of which could be standard routines in the eventual programming language. Data consists of:

1. Paradigms for control, i.e., student progress, display presentation, data manipulation, answer processing. Paradigms describe general logic processes, e.g., standard Coursewriter branching logic, "Clue" branching logic, how to handle a student response of "help."
2. Routines describing control processes, repeatedly used procedures, etc. Routines, unlike paradigms, are specific sequences of commands, and when called they are "executed" in sequence. Where paradigms are referred to and imply a coding sequence, routines are called like subroutines or macros and define a coding sequence.
3. Variables used for scoring, control, etc.
4. Items in a table, array, or list structure.
5. Lists of tables, arrays, etc.
6. Displays, consisting of the text to be displayed and the position of the display.
7. Any author-defined material specific to his course, e.g., special considerations and procedures requiring his explanation.

Each piece of data which is manipulated at any one time must be identified (labeled) so that it can be referenced. Manipulation consists of:

1. Displaying.
2. Erasing.
3. Inserting.
4. Moving.
5. Altering (adding, subtracting, etc.).
6. Calling (routines).

If data is defined at the beginning of a course unit, it retains its definition in all subdivisions of that unit, i.e., it is LØCAL to that unit as opposed to other units in the course, but it is GLØBAL for all modules within its unit. Data defined for a given module is LØCAL to the module, but GLØBAL to the objectives within the module, etc. Any data defined for an entire course is referred to as UNIVERSAL.

Commands are applicable to any type of course, but specific parameters or conditions may be unique to a certain type of subject matter. Other commands may be defined as needed by a given course or a given installation. Parameters are separated by keyword delimiters, giving a natural language appearance. Many parameters, especially conditions, are only loosely formatted, so that authors at times will specify material in sentence form. This flexibility is necessary to ensure that authors are not restricted in the power of their design. But some keyword format will still be present to limit ambiguity as much as possible.

Manuscript Conventions

Standard symbology in command formats:

1. : : to enclose data identifiers, e.g.,

   :V1: refers to a variable

   :A0003: refers to a message

   :CKSW: refers to a routine

   :V1, V2, A1: refers to three variables

2. [ ] to enclose a literal string when used in a command

3. { } to enclose parameters when calling a subroutine

4. ( ) for subscripting as in

   :TABLE5 (Row, Item):

   :LIST1 (Item):

   e.g.:

   :TABLE5 (3,4): refers to a specific item in an array or table

   :TABLE5 (3): refers to the third row of the table

   :TABLE5 (,4): refers to the fourth item in each row of the table

   :LIST1 (3): refers to the third item in a list

Standard conventions for constructing displays:

1. : : to enclose variables in displays

2. ▨ to indicate "reverse shading," e.g., help

*3. ▭ to indicate keyboard response area

*4. ▨ or $\boxed{\text{display}}$ to indicate light pen response area

5. All labels, message identifiers, response identifiers, etc., are to be indicated elsewhere than the display guide screen grid, i.e., the display grid should not contain anything that is not supposed to be actually displayed.

*These indicate information that is not to be displayed, but must be represented on the display grid. Such information, along with identifiers for the responses, numbering for response areas if there are more than one at a given time, should be in a color other than the display material (which is usually in pencil).

Keyword Definitions

I. Structure definition

1. UNIT) mark the beginning of each unit,
MØDULE) module, objective, and page, and
ØBJECTIVE) identify them by number, e.g.:
PAGE)
UNIT 1
MØDULE 1.1
ØBJECTIVE 1.2.^{1}
PAGE 1.3.2.4

2. RESTART (RST) marks the point where student records are saved.

II. Branching keywords

1. IF :C1: THEN :A1:

where :C1: is the author-described condition
:A1: is the action taken (command sequence) if the condition is true

2. GØ TØ :P1:

where :P1: is a command in the sequence, or in a unique label

e.g., GØ TØ MØDULE 2.3
GØ TØ RESPØNSE A003IB
GØ TØ B3#01

III. Operative keywords (or commands)

1. ADD (AD) ... TØ
2. SUBTRACT (SB) ... FRØM
3. MULTIPLY (MP) ... BY
4. DIVIDE (DV) ... BY
5. SET ... TØ [assigns a value to a piece of data]

```
e.g., SET :A: TØ 1
      SET :B: TØ "The old man"
      SET :TABLE1 (3,2): TØ 5
```

IV. Comparative keywords

[all numeric comparisons]

1. EQUALS (EQ)
2. GREATER THAN (G)
3. LESS THAN (L)
4. GREATER THAN ØR EQUAL TØ (GE)
5. LESS THAN ØR EQUAL TØ (LE)

V. Logical keywords

1. AND — the statement :C1: AND :C2: is true only if :C1: and :C2: are both true; may also be used to sequence actions, e.g.:

```
IF :A: EQUALS :B:
AND IF :B: GE :C: THEN DISPLAY :M1:
AND THEN GØ TØ B3
```

2. ØR — the statement :C1: ØR :C2: is true only if either :C1: is true or :C2: is true, but not both, e.g.:

```
IF :A: EQUALS :B:
ØR IF :B: EQUALS :C: THEN GØ TØ B3
```

3. AND/ØR — the statement :C1: AND/ØR :C2: is true if either :C1: is true or :C2: is true or if both are true (used as in previous example)

4. ELSE — precedes the alternative for an unsatisfied condition, e.g.:

```
IF :A: EQ :B: THEN DISPLAY :M1:
ELSE DISPLAY :M2: AND THEN GØ TØ #01
GØ TØ #02
```

When the IF condition is true, :M1: is displayed, the alternative is ignored, and the sequence branches to #02. When the IF condition is not true, :M2: is displayed and the sequence branches to #01.

There may be a sequence of IF conditions, related by AND, ØR, AND/ØR, that define a set action pattern. ELSE separates this condition-action pattern from the next immediate condition and/or action:

```
IF :A: EQ :B:
AND IF :B: EQ :C:
ØR IF :A: EQ :D: THEN DISPLAY :M1:
```

```
ELSE IF :A: EQ :B:
ØR IF :B: EQ :C:
AND IF :A: EQ :D: THEN GØ TØ #01
ELSE GØ TØ #02
```

:M1: is displayed (1) if A=B and if B=C
or (2) if A=D

sequence branches to #01 (1) if A=B and
if A=D
or (2) if B=C and
if A=D

VI. Evaluative keywords

1. Keywords used to evaluate syntax

a. MATCH — :A: MATCH :B: is true only if A and B are the same character or sequence of characters

b. SIMILAR — the truth of :A: SIMILAR :B: will depend upon similarity criteria specified by the author

c. KEY — indicates a match with a system key, such as HELP, etc., e.g.:

```
IF :RESPONSE: KEY HELP THEN ...
```

d. INDICATE (IND) — used with light pen responses

```
IF INDICATE :P003,1: THEN ...
```

this is a match condition if the student indicates an area numbered 1 associated with response area labeled P003

2. Keywords used to evaluate semantic criteria

a. EQUIVALENT (EQV) :A: EQV :B: is true only if A implies B and B implies A. For example:

beagle implies dog,

but dog does not imply beagle,

so dog and beagle are not equivalent.

Implication may be described by:

if A then B

Equivalence may be described by:

if A then B and if B then A

IMPROVED LANGUAGE COMMANDS

(Keywords are capital letters underlined.)

1. RECØRD [for storing archival data--specified at the beginning of a unit, module, page, or template]

Form: RECØRD :N1: AT :W1:

where

:N1: specifies data to be stored

(test scores, variables, responses, etc.; or the identifier of a list)

:W1: specifies when the records are to be recorded

(restart points, response point, etc.)

2. DEFINE [identifies data specified at the beginning of a unit, module, page, or template; data defined is "local" to thac unit, module, page, or template]

Form: DEFINE :W1:

where

:W1: is VARIABLES

e.g., author (NWA--number of wrong answers
(NT1--number of trials
specifies (etc.

LISTS

(e.g., a list of tables or display identifiers)

TABLES

(e.g., arrays, matrices)

DICTIØNARIES & GRAPHICS

DISPLAYS

(e.g., display guides)

PARADIGMS

(e.g., logical processes)

RØUTINES

(e.g., subroutines)

SPECIAL

(any original author-defined material, e.g., special considerations or conditions, repeatedly used instructions, etc.)

3. SAVE [specifies dynamic data to be saved; specified at the beginning of a unit, module, page, or template]

Form: SAVE :N1: IF :W1:

where

:N1: is the identifier of the data to be saved

:W1: is the condition for saving the data (usually written semantically--see Keyword Definitions)

4. SELECT [specifies parameters for locating an item in a list where all the items in the list are values for a single variable]

Form: SELECT :V1: FRØM :L1: STARTING :N1:

TØ :N2: INCREMENT :N3: UNIQUE

where

:V1: identifies the variable for which the value is being selected

:L1: identifies the list or table used (may be written TABLE:L1: or LIST:L1:, etc.)

:N1: is the starting field (may be omitted; Default: first field)

:N2: is the ending field (may be omitted; Default: final field)

:N3: is the increment:

1--sequentially through the list

2--every other item is selected, beginning with the first

3--every third item is selected, etc.

(may be negative to reverse the order of selection of keywords:

RANDØM--items are selected from the list at random

:R1: --the identifier of an author-defined routine for selecting items

(may be omitted; Default: 1)

UNIQUE--if included, no item from the list will be selected twice

5. SELECT [specified parameters for selecting lines from a table where the items in the line are values for more than one variable]

Form: SELECT :V1, V2, etc.: FRØM :L1: STARTING :N1:

TØ:N2: INCREMENT:N3: UNIQUE

where

:V1, V2, etc.: identify the variables for which the values are being selected

:L1: as in 4

```
:N1: is the starting line )
                          )
:N2: is the ending line   )  as in 4
                          )
:N3: is the increment     )
```

UNIQUE--if included, no line from the table will be selected twice

6. <u>GENERATE</u> [generates random numbers for a variable]

Form: <u>GENERATE</u> :V1, V2, etc.: <u>BETWEEN</u> :N1: <u>AND</u>

:N2: UNIQUE

where

:V1, V2, etc.: identifies the variable(s) for which values are being generated

:N1: is the lower limit

:N2: is the upper limit

UNIQUE--if included, none of the variables will receive the same random value as any other in the command

7. <u>DISPLAY</u> [indicates that text is to be displayed on the screen]

Form: DISPLAY :M1: STARTING LINE :N1: (:W1:)

where

:M1: --

(a) is a predefined message identifier

(b) is a variable identifier

(c) is a written message

:N1: specifies the starting line of the display. If the message is already defined with positioning parameters, :N1: will override

(may be omitted. Default: if no positioning parameters are inherent to the message, the message is displayed starting with the first line available on the screen at the time the command is made. If sufficient space is unavailable, no message is displayed and the author shall be informed at the programmer's discretion, or, if space is available, the message ":M1: too large" will be programmed.)

:W1: is a special consideration in displaying the line, e.g.,

```
CENTER        ) may be defined
              ) semantically
SPELL-ØUT     )
              )
RIGHT TØ LEFT ) by the author (see 2)
```

(may be an identifier for an author-defined routine describing the method of display)

8. <u>ERASE</u> [specifies an area of the screen to be erased]

Form: ERASE :M1:

where

:M1: --

(a) identifies a message previously displayed to be erased from the screen as it was most previously displayed

(b) is :N1:, :N2: when

:N1: is the first line to be erased

:N2: is the last line to be erased and all lines between are to be erased

9. PØSITIØN SLIDE :N1:

SHØW SLIDE :N1: [positions slide and opens shutter]

REMØVE SLIDE :N1: [closes the shutter; must be included if new slide is to be positioned with shutter closed]

where

:N1: identifies a frame on the slide projector

10. PØSITIØN AUDIØ :N1:

PLAY AUDIØ :N1: [positions and plays]

where

:N1: identifies the audio message to be played

11. RESPØNSE [indicates that a student response is to be accepted]

Form: RESPØNSE :L1: :W1:

where

:L1: is an EP identifier; refers to a display guide on which the area for the student response is indicated by a box with the EPID associated

:W1: indicates the method of response:

KEYBØARD

PEN

(may be omitted, Default: KEYBØARD)

12. EVALUATE RESPØNSE [precedes a description of evaluation action and criteria for action; may be a routine defined in 2]

```
Form:           IF :C1:

                ØR IF :C2: THEN :A1:

                JUDGE :W1:

                ELSE IF :C3:

                AND IF :C4:

                ØR IF :C5: THEN :A2:

                JUDGE :W2:

                where

                     :A1:

                     :A2: denote actions to be taken, e.g.,

                     DISPLAY :M1:

                     CALL :R1:

                     :C1:

                     :C2:

                     :C3:

                     :C4: are conditions written semantically
                          by the author
                          (see Keyword Definitions)

                     :W1: is either CØRRECT, INCØRRECT, or
                          UNANTICIPATED (may be written CA,
                          WA, UN, AA (no judgment), or
                          simply as an identifier, e.g.,

                          C1
                          CA1, etc.)

                     (may be omitted)
```

13. CALL [indicates that a routine is to be performed]

CALL :R1: {:P1, P2, P3:}

where

:R1: identifies the routine to be performed

:P1, P2, P3: are parameters to the routine

(may be omitted; if specified the routine must have a list with the same number of items referring to data in the routine)

When a routine is called, the coding sequence branches to the beginning of the routine and proceeds until the end of the routine is reached. The coding sequence then branches to the statement immediately following the CALL command and proceeds. Units, modules, objectives, and pages may all be called as routines.

14. CØNTINUE [indicates that course execution is interrupted until the student presses the CØNTINUE key, space bar, etc., depending on the hardware available]

15. REFER [indicates that a paradigm will describe the logical flow of a coding sequence]

Form: REFER :R1:

where

:R1: identifies the paradigm

The placement of the REFER command determines the coding sequence it will govern. If the command immediately follows a UNIT, MØDULE, ØBJECTIVE, PAGE, or routine identifier, then the paradigm will demonstrate the logical structure of the unit, module, etc. If the REFER command follows an EVALUATIVE command, then the paradigm will govern the answer-processing logic.