The Domain-Analysis Based Instruction System

Geoffrey I. Webb

Department of Computer Science, LaTrobe University, Bundoora, Victoria

Abstract

At the past two CALITE conferences I have described a methodology for creating knowledge-based CAL. This paper describes how that methodology has evolved. The Domain-Analysis Based Instruction System, a CAL system that utilises the methodology is then described in detail.

INTRODUCTION

At the first CALITE Conference, Tom Richards, Bob Hooke and I presented a paper on ECCLES, a CAL system that uses explicit decision trees to define expert analyses of complex domains (Richards, Hooke and Webb, 1983). Inspired by the conference, I came to wonder what would result if an ECCLES like system utilised its underlying structure to represent the epistemological structure of the domain rather than the sequence of actions to perform in an analysis of a domain. A new methodology for CAL practise called Feature-Network Based Courseware Design was born! At the past two CALITE conferences I presented papers describing this methodology (Webb, 1984, 1985). This paper describes the evolution of that methodology and the CAL system that has been developed to demonstrate it. This system is called the Domain-Analysis Based Instruction System (DABIS).

The 1984 paper was written as I was initially developing the methodology and system described herein. The methodology that was described in that first paper generated lessons on a subject domain directly from a knowledge-base that fully described all aspects of that domain that were required for the instructional process. The knowledge-base was an enhanced system-network. (The system-network is a formalism developed by M.A.K. Halliday (1973) to describe linguistic systems.) The system-network described the epistemological structure of the domain. The enhancements provided textual descriptions of each aspect of that epistemological structure that could be used to develop or evaluate the student's comprehension of the domain. From this knowledge-base it was possible to generate many different forms of lesson and to perform extremely sophisticated analyses of the student's understanding of the domain. A system was developed that demonstrated that the methodology worked as claimed.

Following the publication of this paper my ideas about this methodology underwent several major transformations. First, on further formalising the basic knowledge structure on which the methodology was based, I realised that it had several minor but significant differences to system networks as described by Halliday. I called my variant of system networks, feature networks. This formalism is described in detail in Webb (1986b). This latter formalism is used in my last CALITE paper.

A more significant change resulted from my realisation that the power of the methodology was not a direct result of its generation of lessons directly from a knowledge-base. Rather, I came to realise that the power of the methodology could be attributed to three factors:

1. the utilisation of a simple knowledge-representative formalism to manage flow of control within a body of instructional material;
2. the utilisation of a simple knowledge-representation formalism to evaluate student's comprehension of the subject matter of a lesson; and
3. the utilisation of one general CAL treatment of a subject domain to provide many specific treatments of discrete examples from that domain. This approach to CAL, courseware abstraction, is discussed in detail in Webb (1986a).

These realisations lead me to revise the requirement on my methodology that lessons be completely generated from a knowledge-base. Instead, the knowledge-base is now used only to manage flow of control within the instructional material and to evaluate students' comprehension of the subject matter of the lesson. Thus, the feature-network on which a lesson is based is no longer in any manner enhanced. A lesson's knowledge-base is purely a description of the epistemological structure of a domain of knowledge. Separate CAL treatments of each aspect of a domain are provided. These are called tutorial interfaces. A tutorial interface is associated with each node in the feature network for a lesson. A tutorial interface can have either or both of the following roles.

1. to increase the student's understanding of how the epistemological aspect of the domain represented by the node relates to specific examples from the domain; and
2. to evaluate the student's understanding thereof.

The feature network is used to determine the flow of control between these tutorial inter faces and, in conjunction with the specific evaluations provided by the tutorial interfaces, to provide detailed analyses of the student's understanding of the domain.

A second system has been developed that embodies this revised conception of the feature network based courseware design methodology. I will refer to this system as DABIS (as opposed to the prototype DABIS system which is the first version of the system.)

DABIS currently consists of three programs:

1. an authoring program. This is used to create and modify all aspects of a lesson.
2. a presentation program. This is used to present a lesson to a student.
3. a lesson printing system. This produces a hardcopy listing of a lesson for reference purposes.

A fourth program will be developed as soon as the time becomes available to do so. This will be the student analysis program which will perform the detailed student analysis mentioned above. (This program exists for the prototype DABIS system, so its development for the current system shall be quite straight forward.)

DABIS is designed with three distinct classes of user in mind:

1. the lesson author;
2. the course administrator; and
3. the student.

The first two of these classes of user are frequently not clearly distinguished in the CAL literature. The lesson author creates a resource which is to be used to teach students. The course administrator is the person who makes that resource available to the student, typically, a teacher. Sometimes, the lesson author and course administrator will be the one and the same person, but this is probably the exception rather than the rule. The course administrator may wish to use the lesson as a stand alone program or to call it from within a tailored CAL environment or other software. Thus, it is desirable to have a great deal of flexibility in the manner in which a lesson can be accessed and used.

DABIS allows many tutorial interfaces to be associated with the one node of the network. This allows the same information to be examined in many different ways to suit differences in students' learning styles and educational objectives. However tutorial interfaces at particular nodes often assume that particular treatments of particular points have been given by tutorial interfaces at different nodes. Therefore, the selection of tutorial interface on a node by node basis is not acceptable. Hence, DABIS organises tutorial interfaces into groups called views. A view contains a set of tutorial interfaces, one for each node of the network. A view should provide a consistent treatment of a domain.

The information pertaining to each example from a domain that a lesson can examine is stored as an item. An item has a name and a description of the features from the domain that it exhibits. In addition, it may specify values for text macros that have been defined for the lesson. When examining an item, any text macro that is displayed will have substituted on the screen in its place the value that has been defined for it for that item.

DABIS stores each lesson as a collection of files. One file describes the feature network; each view is described by a separate file; one file is used to record the items; and another to record the lesson sets. In addition, several index files are maintained to speed up access to various files. A directory is created for each lesson which is called the lesson directory. All of the files that describe a lesson are stored in its lesson directory. Another directory is maintained for each lesson in which all records of student's interactions with that lesson are stored. This is called the record directory.

In the remainder of this paper I will discuss the life cycle of a typical lesson developed using DABIS and indicate how each component of the system operates.

The first step is obviously to create the lesson. This is done by running the authoring program.

The first action taken in the authoring program is to name the new lesson. This name is used to create a new lesson directory. Next, the feature network for the lesson is created. First the author must specify the name and type of each node in the network. Next, for each node, each of its children must be listed, When this is complete, a feature network will have been described. The author is now free to modify the feature network, create any number of views for the lesson, or create items for the lesson. Typically, the next step will be to create a view. This is because it is usually necessary to associate with items information that is required by the views. Creating the items after the views means that the exact requirements will be known when the items are created.

The first step in the creation of a view is to specify a name. Next, the authoring system steps the author through each node of the network and allows the author to specify a tutorial specification for it. A tutorial specification is created by selecting a sequence of commands under menu control. The commands that are currently supported are listed in Table 1. These commands will be significantly extended in the near future.

Many commands take parameters, such as the text to be displayed by a display command. Having selected such a command by menu selection, it is necessary for the author to type in the value for the parameter. This is one of the few exceptions to the almost exclusive use of menu driven control in the authoring system.

After creating each view the author is once again free to proceed with development of any aspect of the lesson. S/he may wish to return to the feature network editor and modify some aspect of the epistemological description of the domain. S/he may wish to specify further views. Alternatively, s/he may proceed to the item editor and start creating the items for the lesson.

Once in the item editor, creating an item only requires specifying its name, the features that it exhibits and the values to set for each text macro when the item is examined.

The item editor may be exited at any stage and any other component of the system entered. Typically, however, the author will remain in the item editor until sufficient items have been specified and then enter the lesson set editor.

The lesson set editor is used to divide the items into groups that are used to control the order in which the items are presented to the students. Up to twenty-six lesson sets, referred to by the letters of the alphabet, may be used. Each item may, but need not, be assigned to any one of these sets. Items that are not assigned to a lesson set are made available for remedial purposes.

Having specified the lesson sets, the lesson is now complete. The author may still, of course, alter any aspect of the lesson at any time. The author may also at this stage wish to use the lesson printing system to obtain a hard copy print out of the lesson with which to validate it.

The lesson may now be presented to students. When the lesson presentation system is run, the terminal type, the name of the terminal description file, and a unique identifier for the student, must be passed to it as parameters or specified as variables in the program's operating environment. Optional parameters are the names of the lesson and view that are to be presented. If the lesson is not specified then the student may select from those available by menu selection. If the view is not specified, the first view to have been created for the lesson is used.

Flow of control within a lesson is determined at two levels. At a macro level it is determined by the selection of which items to present and what order to present them in. At a micro level, it is determined by selecting which tutorial specifications to invoke for an item and the order in which to invoke them.

The selection of items proceeds as follows. The lesson sets can be considered as a queue of queues. In the outer queue, the first lesson set is at the head of the queue. Within each lesson set, the first item within that lesson set is at the head of the queue. The first item is selected for presentation by popping the first item from the first lesson set. The next item is selected by popping the next item from the first lesson set. This continues until no items remain in the first lesson set, when it is popped from the outer queue and the process continues with the next lesson set. The same process is repeated for successive lesson sets until all lesson sets have been used. The lesson is then completed. When a student fails to correctly analyse an item that has been presented, before the next item is selected, the following actions take place:

1. An item is selected from those available for revision which has the same set of features as that for which the error occurred. This item is added to the end of the first lesson set.
2. An item is selected at random from the second lesson set and added to the end of the first lesson set.
3. Finally, the item for which the error was made is added to the end of the second lesson set.

This ensures that several general features are enforced. First, for every item that the student cannot correctly analyse, revision is received in the form of another item being selected with the same combination of features and the same item being re-presented following a delay period. Second, the author is able to grade the items into a general order in which they are to be presented. Revision is intermixed with items from further up the ordering of items.

Having selected an item the following procedure is followed to determine flow of control between the tutorial interfaces. A set of pending nodes is maintained. For each item, this is initialised to the top most node in the network. The next step is repeated as long as nodes remain in the pending set.

> A node is removed from the pending set. Any node may be selected so long as, if it is a conjunctive entry condition, all of the nodes that it descends from have already been traversed. Once a node has been selected, its tutorial interface is involved. The tutorial interface will return one of the following values - continue, block, halt, or exit. If the tutorial interface returns continue then, unless the node is a feature choice, all of the children of the current node are added to the pending set. If the node is a feature choice, the correct feature for the current item is added to the pending set. If the tutorial interface returns block, then none of the node's children are added to the tutorial interface. This indicates that further examination of this aspect of the network is fruitless and prevents it from occurring. If the tutorial interface returns halt then the examination of the current item is terminated. If exit is returned then the current terminal session is finished.

The algorithm that is used to select nodes from the set of pending nodes ensures that depth first traversal of the network occurs. This provides a clearer examination of the domain than does any alternative traversal strategy.

Command	**Description**
Return.	Ends the current tutorial specification
Display - Temporary	Displays a string on the terminal. The string is erased by either clear command.
Display - Long Term	Displays a string on the terminal. The end of the string becomes the end of the long term display.
Clear - Temporary	Clear the screen from the end of the long term display. (This command is automatically executed at the end of each tutorial specification.)
Clear - Long Term	Clear the entire screen.
Multiple Choice - Closed	May only be used in a tutorial description for a feature choice. A single character is associated with each feature of the feature choice. These characters are called tags. The student is requested to enter a character. This is repeated until one of the tags is entered. The student's selection is then noted.
Multiple Choice - Open	As with closed multiple choice except that a tag is also associated with the option of not choosing a feature.
Multiple Match - Closed	May only be used in a tutorial specification for a feature choice. A template is associated with each feature. The student is requested to enter a string. If the string matches the template, the student is considered to have selected that feature. If the string does not match any template, an error message is displayed and the student must enter another string and the process is repeated. The error message is defined separately for each closed multiple match command.
Multiple Match - Open	Identical to the closed multiple match except that if the student's string does not match a template, the student is considered to have not selected a feature and the command is completed.
Respond	May only be used in a tutorial specification for a feature choice. Has n x (n + 1) matrix of texts where n equals the number of features for the feature choice. The text associated with the correct feature and the student's choice (where the student's choice may either be a feature or no choice) is displayed.
Pause.	Wait for the student to press the space bar before continuing

Table 1: The Tutorial Commands

The procedure followed for flow of control within the instructional materials guarantees the following features about the treatment that a student receives of an item:

1. More general aspects of a domain are examined first.
2. An aspect of a domain is not examined for an item unless the student has successfully examined for that item all other aspects of the domain on which that aspect is epistemologically dependant.
3. Once one aspect of a domain has been examined for an item, all other aspects of the domain which are epistemologically dependant upon that aspect are examined before aspects of the domain which are not.

Once a student has finished a lesson, the normal course of events will be for the course administrator to run the analysis program to obtain a detailed profile of the student's understanding of the domain. (The course administrator's ability to do this is, of course, dependant upon the final analysis program being written, which is in turn dependant upon the vagaries of research funding.) The profile which will be obtained from this program (and which can be obtained from the corresponding component of the prototype DABIS system) will describe the student's understanding of the domain in such cognitive terms as their confusions, misunderstanding and under and over-generalisations.

CONCLUSION

The DABIS project is now in its third year. The achievements of the system have not only demonstrated the points that I had hoped to show, but have far exceeded my expectations. The current system is now in use on an experimental basis at three sites around Australia. I hope that next year the system will reach a stage of development at which I am happy to make it available for general distribution.

Acknowledgements

I would like to thank the organisers and attendees of the four CALITE conferences for their encouragement and support throughout the life of this research project. The DASIS project is very much a child of the CALITE conferences having been conceived at the first CALITE conference and nurtured at subsequent ones. I also wish to thank Tom Richards without whom none of this research would have occurred. His support has been invaluable and is very greatly appreciated.