Computer Aided Assessment through Hypermedia

Andrew Peel

Computing Laboratory
The University, Canterbury, Kent CT2 7NF.
Email: A.T.Peel@ukc.ac.uk
Tel: +44 1227 823979
Fax: +44 1227 762811

Glenn Rowe

Applied Computer Studies Division, Department of Mathematics and Computer Science
University of Dundee, Dundee DD1 4HN.
Email: growe@mcs.dundee.ac.uk
Tel: +44 1382 344484
Fax: +44 1382 201604

1. Introduction

With the increase in the number of students entering Higher Education Institutions without a similar rise in teaching resources, Computer Aided Assessment (CAA) provides not a replacement for personal teaching, but a supporting aid.

If a student does not understand material presented in the traditional lecture format, then perhaps a browse through a computer based tutorial, probably written by the lecturer, from a different perspective, would shed some light on the problem. This would reduce the amount of people who rush straight to the lecturer when there is a point they don't understand.

This is one of the aims of Computer Based Learning (CBL), but CBL needs to incorporate CAA to be of any real use. The student needs to receive feedback on their performance to help them judge how well they are proceeding with the material.

This can be achieved by positioning questions at what may be vital points in the tutorial, which it is only optional to answer. A further, more formal (and compulsory) examination of the main points may be useful at the end of the tutorial.

2. Using hypermedia to construct multiple-choice items

This section discusses the implementation of two packages to demonstrate CAA through hypermedia, using PC Guide (produced by Office Workstations Ltd) and its scripting language, LOGiiX. Completed as part of TLTP Project ALTER at the University of Kent at Canterbury, the packages are called "An Introduction to Computer Aided Assessment in Higher Education", and "Constructing Multiple-choice Tests - An Introduction".

2.1. Why hypermedia?

Hypermedia packages offer a simple tool for the creation of CBL materials which are visually attractive, highly flexible, and intuitive for students to use. These packages normally also provide a scripting language that sits behind the information frames and can be used for manipulation of the data needed in CAA, as well as manipulation of the hypertext document.

CBL, at its simplest, just requires the display and navigation of teaching material. However, for student assessment we need some method of automatically marking students' answers, and of keeping track of their total score. In the packages, this is done using PC Guide's scripting language called LOGiiX. This is similar to the Pascal programming language, with added hypermedia functionality through language extensions.

2.2. PC Guide and LOGiiX

PC Guide has four types of hypermedia "buttons" (where a button is usually represented as a section of text, or part or all of an image): reference, expansion, note and command buttons, although the assessment part of the package mainly uses command buttons. A command button is hyperlinked to a section of a LOGiiX script which is executed when the button is pressed.

As can be seen in figure 1 below, all hypertext objects in PC Guide have a number of attributes. These include a user defined name field, a unique object ID, the type of the object (eg. command button), the ID of the object the button is hyperlinked to, and the target document.

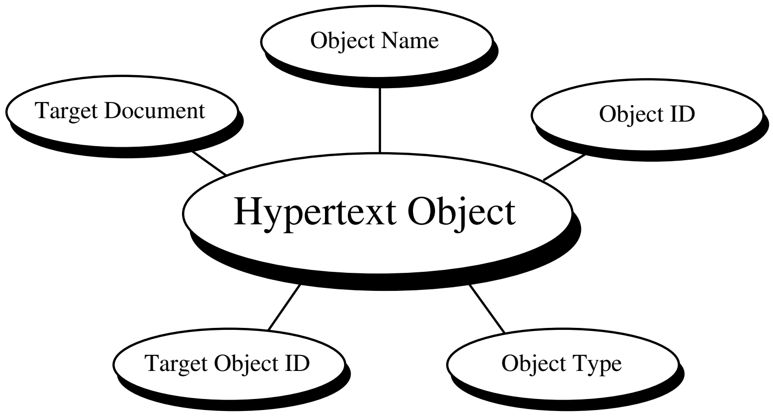

Figure 1: Some attributes of a hypertext object in PC Guide.

Each multiple-choice option is implemented as a command button pointing to a single LOGiiX script (see figure 2). The answer to each item is stored as part of the hyper-structure of that item (in the command button name field).

When one of the answer options is chosen (by clicking on it with the mouse), it executes the following script:

```
#LOGiiX
Global score;
function main()
begin
    if (GetObjectName(GetTopDocID(), ButtonID()) = "correct" then
    begin
        score := score + 1;
        Answer(0+64, "Question 1", "Correct!")
    end
    else
        Answer(0+64, "Question 1", "Wrong answer.")
end
```

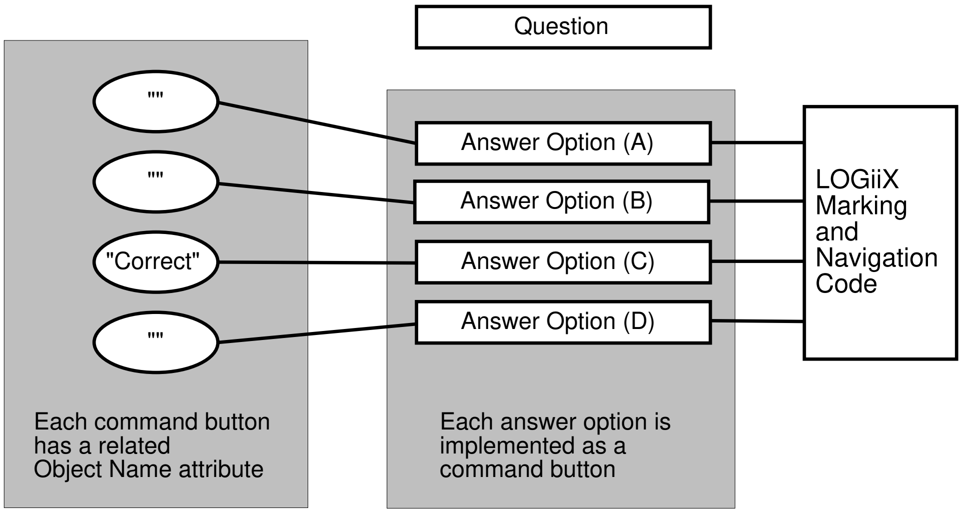

Figure 2: Model of a simple multiple-choice item using a PC Guide command button

With each answer option implemented as a command button pointing to the above script, the tutorial responds appropriately to the answer option selected by the student.

The *GetObjectName(GetTopDocID(), ButtonID())* function simply returns the value *"correct"* if the selected button is that of the correct option, or *""* if it is not. This result is then passed to the student using a standard dialogue box (instantiated using the *Answer(0+64, text)* function).

The command button name field is visible to the question's author during editing with PC Guide, but is not visible during student use with PC Guide Reader (a Guide hypertext document browser), which also keeps LOGiiX scripts hidden from the user.

The script is independent of the content of the question which means that a series of questions can be set using the same script. Each command button answer option points to this single section of LOGiiX script which handles the dialogue with the student and the scoring mechanism.

This software reuse (illustrated in figure 3) is analogous to procedure calls in high level languages, with data being passed to the procedure through command button attributes, rather than through parameters. This results in the simplified creation and maintenance of the LOGiiX scripts used in the tutorial.

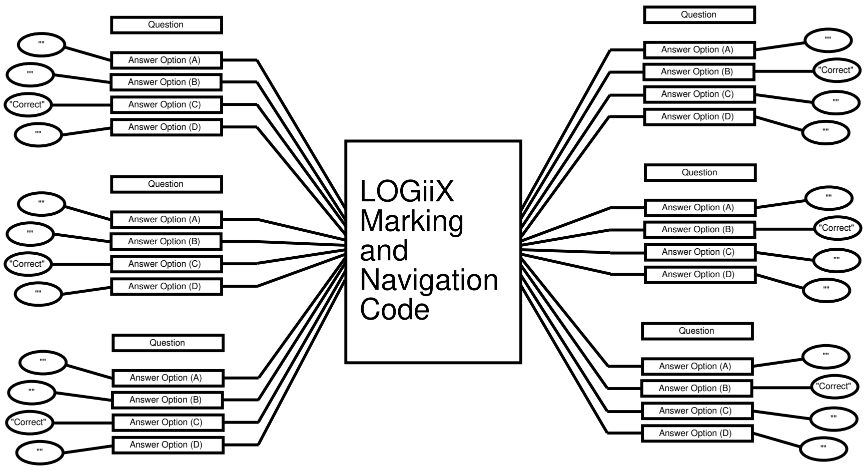

Figure 3: Software reuse using PC Guide command buttons

2.3. The Packages

So far we have only discussed simple multiple-choice items, although the packages demonstrated in the workshop (and available from the address below) contain many real examples of this type of student assessment. They also include many other different types of question, including multiple-choice using pictures, matching items using text and pictures, timed items, fill the gap items, etc.

The packages contain further examples of CAA, such as Computer Assisted Marking (where a score sheet is produced and maintained automatically by the computer), and help demonstrate the use of negative marking to discriminate between successful students and students who guess correctly during multiple-choice tests.

2.4. Summary

In the past a large amount of CAA software has been developed from scratch. Producing the packages for Project ALTER has demonstrated that hypermedia packages such as PC Guide are of sufficient flexibility that they can be used to develop Computer Aided Assessments; there's no need to write new low level application code.

It has also demonstrated that perhaps PC Guide is not flexible enough for further expansion and development in the future. The main problems in this direction lie with LOGiiX. LOGiiX has no complex data structures, making scripts more difficult to code, and cumbersome to maintain. Hence it is not ideal for building large assessments.

The obvious difficulty is that the package will only work on a PC. Perhaps this is not obvious, as it has become a fact of life for computer users that software only works on one type of machine. In the next section, we discuss how to build a CAA package that will work on any popular platform, using World Wide Web technology.

3. On-line tutorials using the World Wide Web

The World Wide Web (WWW) offers a convenient platform on which to develop Computer-Aided Assessment tutorials. This section describes a package being developed at the University of Dundee for teaching first and second year computer science using on-line notes and tutorials to replace traditional lectures. The system had not been tested on a real class at the time of the workshop at the University of Kent at Canterbury, but is being used in the second semester courses (February to May) in both first and second year computer science at the University of Dundee.

3.1. HTML forms

The hypertext language used by WWW viewers such as Mosaic and Netscape is HyperText Markup Language or HTML. Traditional HTML supports a fairly rudimentary set of text formatting options. HTML forms are an extension to basic HTML which allow users to enter information in a variety of formats, such as text fields, radio buttons (for selecting one of a number of choices), clickable image maps (where sections of an image are "hotspots" which can initiate a hyperlink), and so on. The information entered by the user is sent to a server program (which is a program separate from the viewer, and can be written in any language, although Perl and C are current favourites) which interprets the results, possibly saving information in a file and writing a response to a Netscape or Mosaic window. For the rest of this section, I shall assume that the WWW viewer being used is Netscape, although the system has been tested extensively on Mosaic as well.

Information is sent from a Netscape form object in the form of name-value pairs. Each form object has a name, which acts as a label so that the server can determine the object from which information is being sent. The name is usually hard-coded into the HTML script defining the form. The value of a form object is usually determined by what the user enters interactively. For example, in a text field with the name "field1", the user might enter some text such as "I have this terrible pain in all the diodes down my left side." This text becomes the value of the text field with name "field1".

Netscape also supports hidden fields, which are not displayed in the Netscape window but which can transmit name-value pairs to the server. This method can be used to send information along with name-value pairs from visible fields. For example, a hidden text field could be associated with the field "field1" above, with the name "hidden1" and the value "Marvin".

3.2. Question-answer tutorials

The facilities available in Netscape have been used to create tutorials in first and second year computer science. The features available so far include:

- Questions requiring a simple one-word, symbolic or numerical answer. For example, the student might be asked for the binary form of the decimal number 67. The student would answer by typing 1000111 in the text box next to the question. The name-value pair from this text box is sent to a parser program, along with a hidden name-value pair which contains the tutor's answer. The parser's job is to compare the student's answer with the value from the hidden field. In order to allow variations in the student's answer (such as one or more leading zeroes in the binary number question) it is often necessary to give the tutor's answer as a regular expression, rather than a single ASCII string. A parser written in Perl or some other language designed to handle strings is most appropriate for dealing with regular expressions.
- Multiple choice questions can be implemented using sets of radio buttons. A set of radio buttons consists of several choices, only one of which may be selected at a time. Only the name-value pair of a selected button is sent on to the parser, so the answer may be checked using the same parser as for a textual answer. The value of the name-value pair is compared to the tutor's answer (contained in a hidden field, as before). In this case, since the value submitted is hard-coded into the radio button, no regular expression is needed in the tutor's answer. Multiple choice questions in which more than one selection is required in a single answer are also supported using checkboxes.
- Rather than use radio buttons, clickable menu lists can be used. A menu in a Netscape form is a list of text items (which, if long enough, can be scrolled), each of which can be selected by clicking with the mouse. One or more items from a menu can be selected and their name-value pairs passed to the parser.
- A clickable image map provides a convenient way to implement questions where the student is required to select an object in a diagram as the answer to a question. For example, a diagram of a binary tree may be shown, and the student asked to indicate where a particular item of data would be inserted into the tree.

  An image map is constructed by creating an image file (using some standard figure drawing tool). Hotspots must be defined on this image. In the binary tree example, the node corresponding to the correct answer can be defined as a circular hotspot, and the rest of the image allocated to the default area. When the student clicks the mouse on the image, the co-ordinates of the point chosen are sent to an image map processing program which checks to see in what area of the image the selected point lies. Each region can be mapped to another Netscape page using a URL (Uniform Resource Locator). In the binary tree example, if the student selected the correct area, a Netscape page printing the message "Correct!" might appear (possibly accompanied by a happy face :-)); if the student chose some part of the image outside the correct area (somewhere in the "default" region), a message "Sorry, try again...." might appear.

Other Netscape utilities may be used for constructing other types of questions, but these examples should give the flavour of the package.

3.3. Assessment and help

A tutorial page in this package usually consists of several questions of the text field or multiple choice variety grouped together into a single form. When the student has answered all the questions in the tutorial, he/she can click on a "Submit answers" button, which sends all the name-value pairs (including the hidden fields containing the tutor's answers) to the parser. The parser marks the student's answers and prints a summary page in a Netscape window informing the student which answers were right or wrong. If desired, the tutor can encode a marking scheme in the hidden fields, although this has not been done yet.

If the student answered any question incorrectly, a help option is presented in the summary page. This option is a hotlink to the section in the course notes dealing with the topic on which the question was based. The notes for the courses on which the tutorials are based were written originally in either Latex or Word for Windows, and converted to HTML using latex2html (by Nikos Drakos of Leeds University) or cu_html (a Word to HTML converter written by the Chinese University of Hong Kong). The notes are not interactive (they contain no tutorial questions) but they do contain the complete text of the course material.

Finally, a hotlink can bring up a Netscape window showing the tutor's answers for the current tutorial.

3.4. Student records

Keeping records of student performance in such a system turned out to be something of a problem. Netscape was not originally designed as a tutorial system, so its record keeping facilities are somewhat primitive.

The answers submitted to a tutorial can be stored in a file by the parser program by simply writing the name-value fields it receives from the tutorial onto disk. However, since the parser program is owned by the tutor, not the student, it cannot determine directly the username of the person submitting the answers to the form. It would seem that, since the actual process running Netscape is owned by the student, there should be a way of transmitting the username (or address of the machine calling the tutorial page) of the student through to the parser along with the name-value pairs from the form, but we have not yet managed to discover how this can be done.

As a stopgap measure we have included a separate text field at the top of each tutorial page in which students enter their username whenever they wish their work to be recorded by the parser. Although this is a cumbersome way of doing things, it has the advantage that the student can try out the tutorial several times before actually submitting a final form for recording and viewing by the tutor.

3.5. Computer programs

Work is underway on methods of including questions requiring students to write or modify computer programs. This idea is based on the package ceilidh, a TLTP project based at the University of Nottingham.

A programming question in Netscape gives the student the option of copying a skeleton version of the program which is to be modified or supplemented by the student. For example, an outline of a Pascal program requiring the student to convert decimal numbers into binary may be given, which contains data declarations and main section with input and output statements. The actual function or procedure where the conversion from decimal to binary is done is omitted in the skeleton program.

The student copies the skeleton program to his/her own directory by selecting a hotlink. HTML allows a given file extension (such as .p for a Pascal program) to be associated with a particular action, using the MIME conventions. For example, if the link file to which a hotlink points has a .gif extension, an image viewer is spawned which loads the file and displays it for the user. The httpd server (which handles incoming WWW requests) allows local modifications to these MIME conventions so that users can attach specific actions to file extensions. Using this technique, all skeleton programs used in the Netscape tutorials could be given an extension of .skl, and the httpd server informed that files with this extension should bring up the Netscape file selector box which allows the user to save the file in his/her own directory.

Once the student has their own copy of the skeleton program, they can work on it using their favourite editor (outside Netscape). Once the program works to the student's satisfaction, it must be submitted to the Netscape tutorial program for marking. There are (at least) two ways this can be done:

1. Provide a text area in the Netscape form and copy the program into this text area using, say, the mouse to cut and paste the program from an ordinary window into the Netscape window. This is fairly easy to do for short programs, but for longer programs that cover more than a screen's height, it can be cumbersome.

2. Type in the full path name of the file in the student's directory where the program code is stored. This is somewhat easier than the previous option, but suffers from a major drawback. The file must be copied from the student's directory into the tutor's directory by the script called by submitting the form. Since data are being transferred between two sites, neither of which is the Netscape window, the WWW server has no control over this event, which means that the source and destination directories must both be on the same computer system. Using this method therefore means that the tutorial cannot be used over the WWW network.

For the initial trial of this system at the University of Dundee, we are using option 1 above: the student can either edit the code directly in the text area on the Netscape form, or else can cut and paste the finished code from another window into Netscape.

The actual evaluation of the code is still being developed, but current features include the ability to compile the program, report any compilation errors to the user through Netscape, test the output of the program (if it compiles!) against some test data provided by the tutor, and examine the source code for various desirable features, such as appropriate number and size of comments, lengths of functions and procedures, lengths of variable names, and so on. These tests are all done by analyzing the source code using Perl routines. The results of the tests are reported to student by displaying them in Netscape, and are saved in data files for later examination by the tutor.

A similar system has been developed at the University of Kent at Canterbury. Undergraduates can write Occam (a specialised parallel processing language) into a text area of an HTML form. On the click of the submit button, this code is compiled and then downloaded onto the Transputer network at UKC. Once it has been executed, the resulting output is then wrapped in HTML and sent back to the students Web browser.

3.6. Summary

Despite the drawbacks of the this system, which are due mainly to our attempts to make Netscape do things for which it was not originally intended, the use of Netscape and the World Wide Web shows promise as a tutorial system. The international nature means that a tutorial designed in one location can be accessed world wide, so that students in one city or country can take courses designed anywhere in the world. The Netscape viewer is also available on all major computer platforms, so that access is not limited to, say, PC only or UNIX only computers. The current system has been tested on UNIX using X windows, and on PCs using Windows Mosaic. At the time of writing, the tutorial has been in use in the first year class (on elementary data structures and algorithms using Pascal) and in the second year class (advanced data structures and algorithms and object oriented programming using C++) for six weeks. Initial feedback from the students has been very positive, with most of them preferring computer based learning to traditional lectures.

4. Conclusion

This paper has described two approaches to using computer-aided assessment in teaching. The PC Guide package, although producing an attractive, easy-to-use interface, is limited by its restriction to one platform (IBM PCs and clones) and by the relative simplicity of its scripting language, LOGiiX. The WWW package may not have such an attractive or flexible user interface, but is available on all systems capable of running a viewer supporting HTML forms. This currently includes all the most commonly used platforms (X Windows, Microsoft Windows, Macintosh). In addition, the answers submitted by the student may be processed by a script written in any language (such as C or Perl), so sophisticated parsing and record-keeping methods may be used.

Besides offering students an alternative to the traditional lecture-based course, the use of computer-aided assessment has many other positive features:

- Students "learn by doing" rather than solely by listening to non-interactive lectures or reading books or handouts.
- The on-line tutorials and notes are available 24 hours a day, 7 days a week so students can study them at times suitable to them.
- The use of WWW as a vehicle for Computer Aided Assessment means that courses can be taken by students at remote sites without the need to physically be at a university campus.
- Once the tutorials have been written, tutorial classes can often be conducted by student tutors (for example, honours students or postgraduate students) rather than by academic staff, thus freeing up staff for research and other duties.
- Staff at different institutions can collaborate on producing courses allowing broader input to course design.

5. Further information

Information on how to obtain "An Introduction to Computer Aided Assessment in Higher Education" and "Constructing Multiple-choice Tests - An Introduction" is available from:

Joanna Bull, Project ALTER, UCoSDA, Level Six, University House, Sheffield S10 2TN.
Email: J.Bull@sheffield.ac.uk
Tel: +44 1742 750820
Fax: +44 1742 728705

The PC Guide workshop was based on an article originally published in Issue 1 of Active Learning (December 1994), available from :

Joyce Martin, CTI Support Service, 13 Banbury Road, Oxford, OX2 6NN.
Email: ctiss@vax.ox.ac.uk
Tel: +44 1865 273273
Fax: +44 1865 273275

For further information on the WWW package in use at the University of Dundee, contact Glenn Rowe via email (preferably) or by postal mail at the addresses given at the beginning of this paper.