The making of CourseMaker, a web-based shell program which can be set up by the teacher to run online courses

Dee Pratt
Durban Institute of Technology

ABOUT THE AUTHOR:
Dee Pratt taught high school English for 22 years, joined the Communication Department at the Durban Institute of Technology (formerly Technikon Natal) in 1990, and is now a Senior Lecturer and Online Faculty Mentor for Arts. She has a master's degree in Applied Linguistics, and is currently working on a Ph.D. in CAI/Written composition.

ABSTRACT: CourseMaker is an HTML shell program which was developed by the presenter as part of a Ph.D. research project on CAI/written composition, but which can be used for a variety of other instructional purposes. CourseMaker contains many of the elements of the traditional classroom translated into the electronic medium, and can be set up by teachers to run a variety of courses in either academic or non-formal subjects, along with any instructions, lesson materials or notes they may wish to include. It has features such as lesson links and pop-up boxes which make it possible to layer and cross-link teaching materials and resources either on CourseMaker itself or the Internet. CourseMaker also provides for input by students, who can continue with a course at any stage or level, and can choose which course or lesson to access as needed. Setting up courses on CourseMaker does not require knowledge of computer programming: courses can be set up by the teacher to suit different academic contexts, purposes and student target groups. CourseMaker is not a commercial product but research output which is thought to have educational potential when used either as or in conjunction with a web-based learning programme.

INTRODUCTION

CourseMaker is a prototype program for creating short courses and lessons. It can be installed on a server or website, and can be used either as a stand-alone application or operated from within the framework of a web-based administrative program such as WebCT. It requires very little training to use, and is capable of infinite permutations and variations. Although the menu (which is generated automatically as the lesson elements are added) is linear, the teacher can set up the course so that the student experiences learning either in a structured or open-ended way: it can also be set up so that the students themselves can choose whether to work in a structured or random

fashion. This paper will look at the design, creation and application of CourseMaker, using as an example of its application a short briefing lesson on oral presentations set up to work from within a WebCT 4.0 course, *Comm. Skills Online for Chemical Engineering*.

PROGRAM DESIGN

The program which eventually came to be renamed "CourseMaker" is something of an inspired mistake. Thanks to the efforts of a creative young programmer, Taliesin Sissons, the program turned out to be an ingenious and relatively low-cost course or lesson creator, which was intended for use as a writing tutor. The programmer was keen to create a WebCT clone: the resulting application is neither an administrative shell program, such as WebCT, nor a writing tutor program (which requires a more sophisticated and versatile help menu in stages, working in tandem with Word), but something of a hybrid. As a novice courseware designer, I am as much to blame as the programmer, however, because I was attempting to create a "one version fits all uses" program, and the end result was to sacrifice basic functionality for versatility. CourseMaker is a "do-it-yourself" program with a vengeance: the teacher creates the program out of the building blocks (or program units) supplied by the programmer – there *is* no program until one is constructed by the teacher. In this sense, CourseMaker is not even a shell program: it is a program construction kit. Because of this design feature, CourseMaker is extremely versatile in its application, but works best with short courses. As a low-cost HTML application (ASP is used to access data-base information) its operation with even medium-length courses is slow, and it eventually starts to look (and work) like a course made of Lego, as the menu becomes as long as the list of contents, with all subsets showing instead of having fold-up options. These problems could have been ironed out, but CourseMaker was rejected as a potential writing tutor program because it did not fulfil three basic specifications: (1) it could not be made portable without sacrificing another key specification (student data input and retrieval); (2) it could not be made to work in tandem with Word without using reduced window settings; and (3) students could not quickly retrieve (or display) key data which they had fed in, and which they would need to refer to while composing. In CourseMaker, student data can be saved, but is hard to locate quickly amongst the teacher-generated text: this is not a problem in the case of

retrieval in short lessons or courses, but is a serious shortcoming in the case of the protean and swiftly-changing processes involved in written composition.

Green (2000) comments on the lack of theoretical underpinnings for CAI writing courses. This was not the case for this CAI application, which was intended to exemplify a theoretical framework of composition developed in the course of my Ph.D. research. I had storyboarded my writing tutor program as a series of five stages, which constituted a discourse management strategy (Condon & Cech, 1999) for composing. This would have been easier to program than CourseMaker, and thus a more sophisticated tutor program version could have been produced. However, there was a theoretical issue which suggested the "teacher-builds-the-program" approach used in CourseMaker. I was satisfied that the discourse management strategy developed in my master's thesis in 1986 was open-ended enough to fit into any context in which composing might take place, because the strategy took into consideration key aspects of discourse, namely, the contextual, ideational, communicative, social and reflexive aspects. These can be viewed as key aspects of not only composing, but also knowledge construction, as knowledge is constructed in discourse (see the account given in the OLC in-house papers, 2002). The discourse management strategy affords the learner writer infinite flexibility in composing (see Figure 1 below) and acknowledges the complexity and recursivity of composing which is stressed by the teachers and researchers who have made a study of composing processes (notably Raimes, 1985, and Zamel, 1985). However, as there are an infinite number of ways in which knowledge construction can be interpreted by teachers as well as by students, I wanted to include yet another layer of flexibility in the tutor program, so that teachers could key in their own preferred strategy for knowledge construction, and not necessarily the discourse management strategy which initially formed the template for the program. The additional level of flexibility, which is reflected in the left-hand section of Figure 1, allows for both socio-cultural and cognitive constructivist perspectives to be accommodated within the program design (Vanderstraeten & Biesta, 1996), provided that the teacher creates a course structure which can be interpreted flexibly by the student.

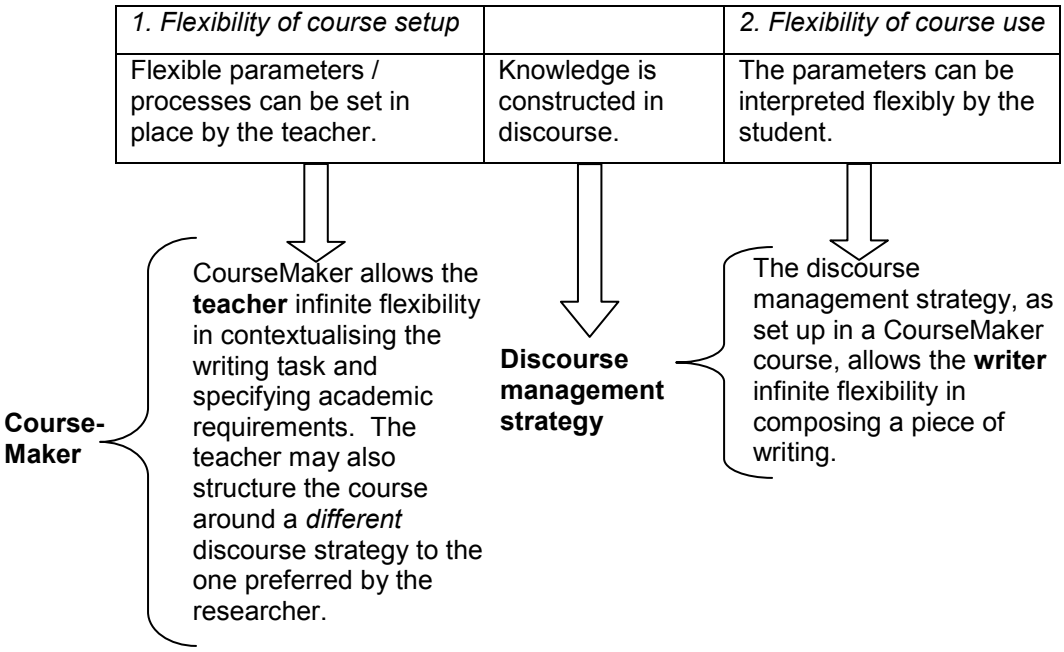


Figure 1: *The two levels of flexibility provided by the CourseMaker program*

In retrospect, and in terms of my own specific research focus and program design, this emphasis on open-ended course structure was a mistake. The discourse strategy which I had developed as a template for instruction in written composition already allowed the writer infinite flexibility in composing: what the writing tutor program needed was to be easily *customised* by the teacher to suit different contexts and purposes. However, although the open-ended options for course structure in the program design made the program too unwieldy for a writing tutor, they resulted in a versatile course maker program.

THE CREATION OF COURSEMAKER

The storyboard on which CourseMaker was based had been written in three weeks, and the program took shape in collaboration with the programmer over a period of a few months. During this period we established that the writing tutor was a kind of course. In retrospect this was a serious tactical error: while it contains lessons on composing, the writing tutor program is more properly a sophisticated help menu. If one considers the function of most help menus, they can be seen to work very much in the same way as a coach or tutor. The logical consequence of viewing the writing tutor as a type of composition course, moreover, one which must be flexible enough to fit the different contexts in which writing occurs, was to construct the program as a

course maker. This led to a consideration of the various ways in which courses could be conceptualised. A schematic was drawn up in collaboration with the programmer to illustrate how a teacher might set up a learning programme, and the concepts in the schematic were defined in the simplest way possible (see the Annexure for the schematic and definitions). As the programmer was not an educational researcher or teacher, lay terms had to be used in discussions: this led to a very practical rendition of the concept of a "course". A course in programming terms had no existence, but was just a set of lessons. Lessons themselves did, however, have a basis in programming, as they were made up of various programmed "lesson elements", which were basically shells to contain texts, pictures, questions, routines, and so on. The fact that courses and lessons were rendered in simplistic terms does not mean, however, that CourseMaker can be used only for transmission type teaching, or that subtleties or complexities cannot be built into a course by the teacher. The main difficulty at this stage was translating teaching/learning routines into programming routines. For example, a teacher might want to group lessons so that they could be seen to be conceptually related by students. This meant that, while a lesson group might be programmed in exactly the same way as individual lessons, the programmer had to make a visual distinction between the two in the program screen display for a teacher setting up a course.

The programmer produced the basis for the CourseMaker program in little more than a month. However, it took six more months to finish a working version of the program, which was then set up on a local browser on my home computer, and ultimately, on the Online Learning Centre server so that my Ph.D. supervisors could view the program. It took me 30 – 40 hours to key in the version of the writing tutor (a high school composition tutor) depicted on the original storyboard. At that stage it became apparent that the CourseMaker program in HTML format on a browser would not handle more than short courses or short routines. I had been told by the programmer that I would be able to format text on FrontPage: this made for professionally formatted lessons, but put too much of a strain on program memory. Towards the end of keying in the writing tutor course, the program hung for two or more minutes on each lesson save, and the version on the DIT server crashed repeatedly each time I reached the end of a course, erasing the last third of the lessons each time. The program was out of action for six months on the server, and the only

version I had was on a local browser on my home computer. I eventually gave up the idea of using CourseMaker for the writing tutor, as it was clear that the program did not possess the functionality described in the specifications, and, while the program concept was creative, the programmer appeared to have painted himself into a corner with a cumbersome directory type program structure than did not allow any preambles or introductions to lessons without upsetting the hierarchy of the menu. The fact that I was unable to demonstrate or pilot the program easily from my home computer illustrated the importance of the portability specification, which had not been dealt with satisfactorily.

THE APPLICATION OF COURSEMAKER

By April 2003 the program was again up and running on the OLC server, however, and I considered how it might be used for short lesson routines in online Communication courses set up on WebCT: experimentation showed that CourseMaker worked very well when added as a URL to a WebCT course. This application of CourseMaker is illustrated below by showing how it can be used to set up and run a short briefing lesson on oral presentations intended to work from within a WebCT4 course, *Comm. Skills Online for Chemical Engineering* (see Figure 2).

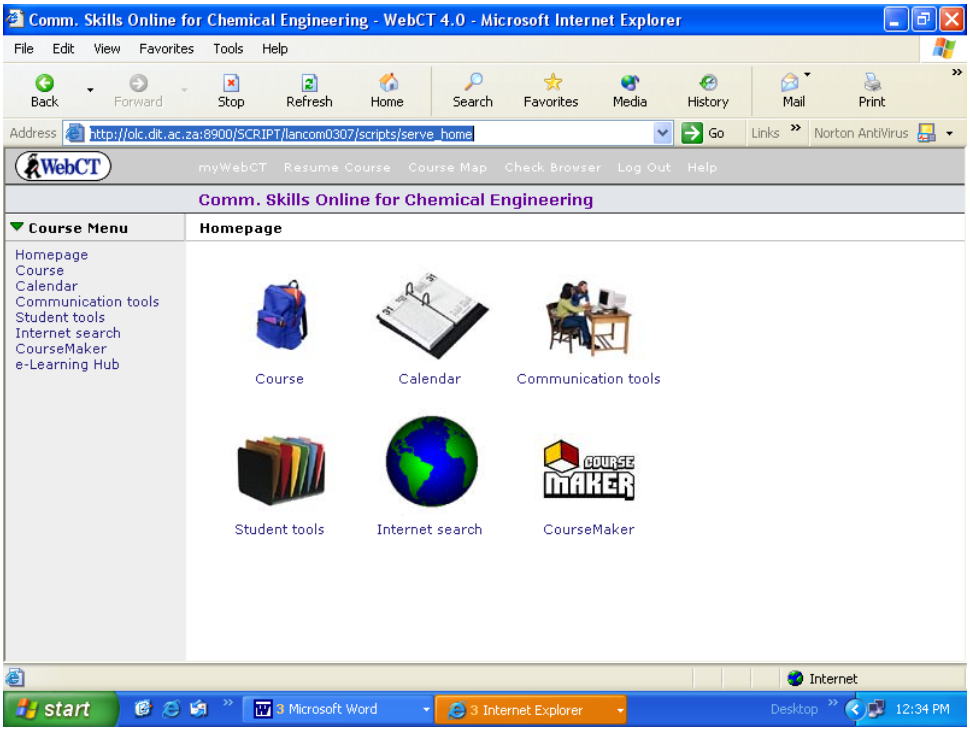


Figure 2: *CourseMaker operating from within WebCT 4.0*

The student group can be presumed to have covered the section on oral presentations already in face-to-face instruction: the outcome of the briefing session is for students to run through the main points on their own in preparation for the talks.

The setting up of the course by the teacher

The teacher sets up the course on CourseMaker as follows. After clicking on the CourseMaker icon on the WebCT homepage, the teacher selects the option "Teacher Mode", which leads to the options on the right hand side of Figure 3.

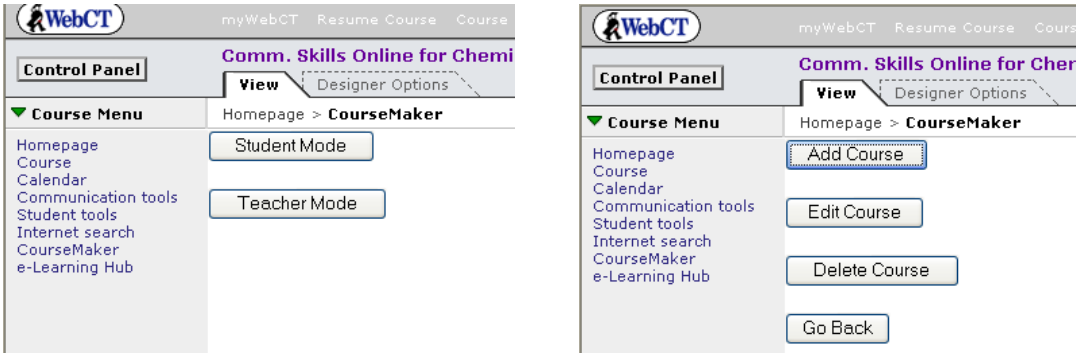


Figure 3: *Selecting "Teacher Mode" to create a course with the "Add Course" button*

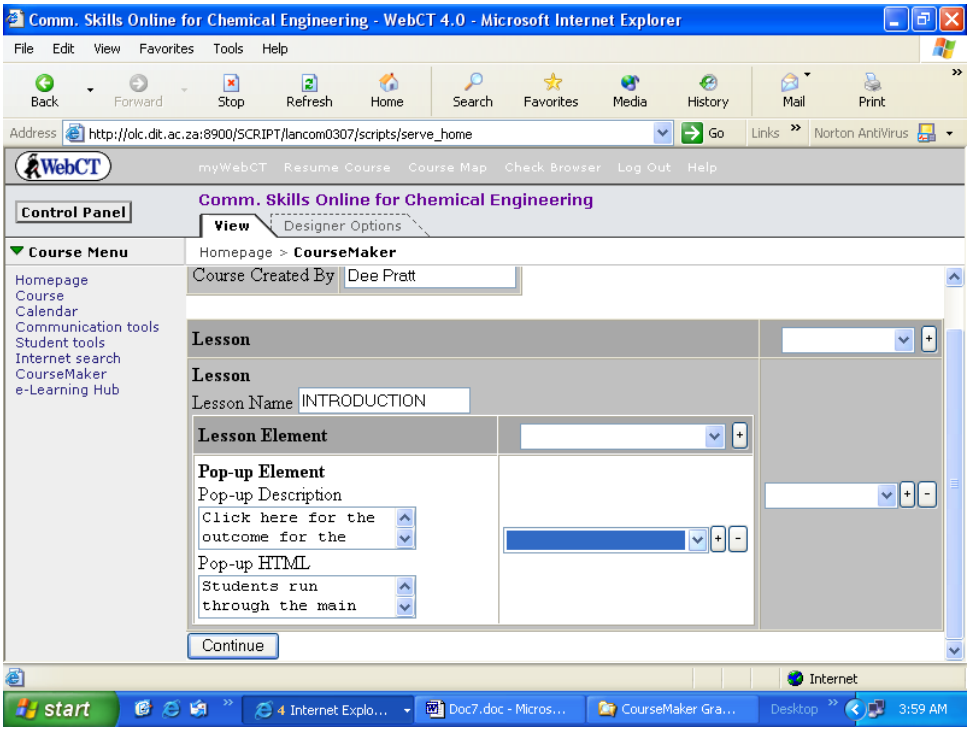


Figure 4: *A course in the process of being created on the template provided by CourseMaker*

The teacher then uses the template provided by the program (see Figure 4) to create the course block by block, adding lessons, which are constructed out of lesson elements, such as text, for teacher instructions and information, and text or memo boxes for student input (student input is saved and is recoverable later). As the course is built up, a drop down menu is generated which will allow students access to the various lessons, as shown in Figure 5.

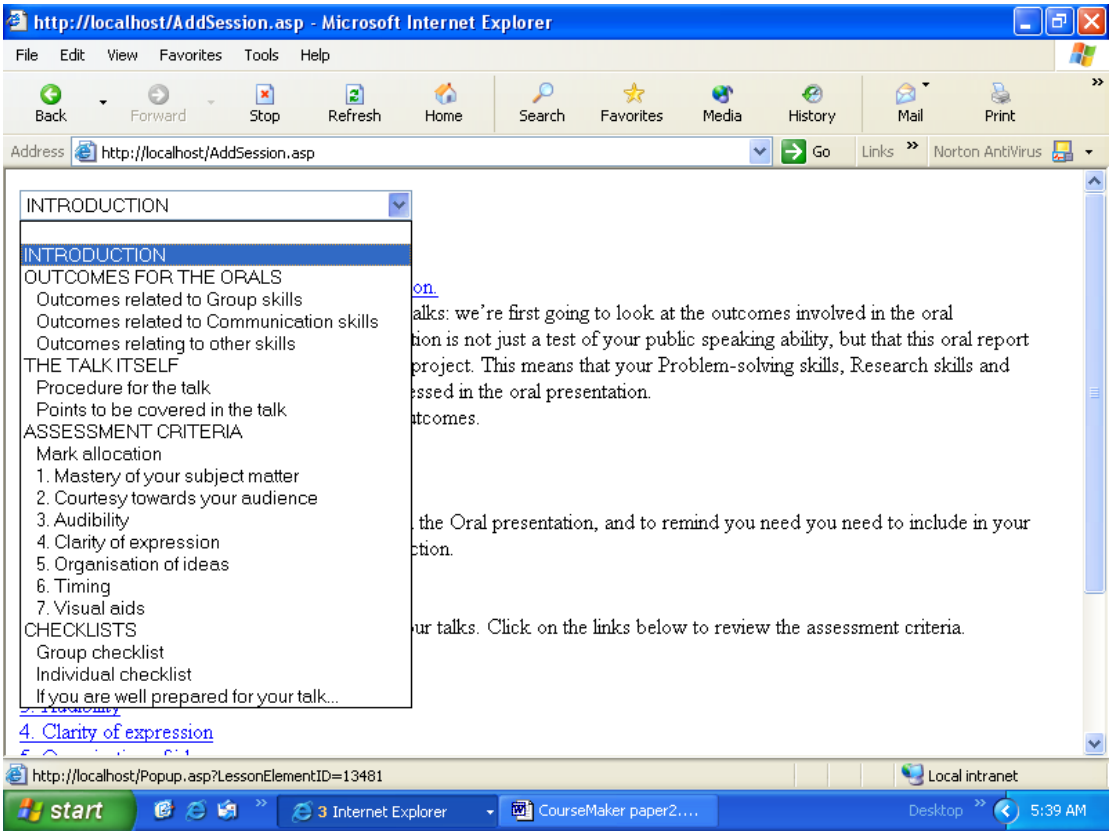


Figure 5: *A lesson and menu as it would appear to students in CourseMaker*

A useful lesson element is the "lesson link", which can be inserted into lessons to link them directly to other lessons without going via the menu: this speeds up movement through the course, and reflects a non-linear connection between course elements. Text can either be typed straight into the template lesson element boxes, cut and pasted from Word, or formatted first on FrontPage, and then copied across. While the last-mentioned lends a more polished appearance to a course, at present this unfortunately takes up too much memory for smooth operation, and the program tends to hang when the course is being saved, sometimes even erasing the last few lessons.

Student use of the course

Students access the course through the "Student Mode" shown in Figure 3, and are then given the option of starting a session, continuing a session or deleting a session (Figure 6), which leads to a display of available courses from which to choose, as shown on the right-hand side of Figure 6. This means that students can either start a course from scratch, or continue a course: the latter option enables students to review data which was input in previous sessions. The student then works through the course using the drop down menu (see Figure 5) or lesson links, which would appear as hypertext links.

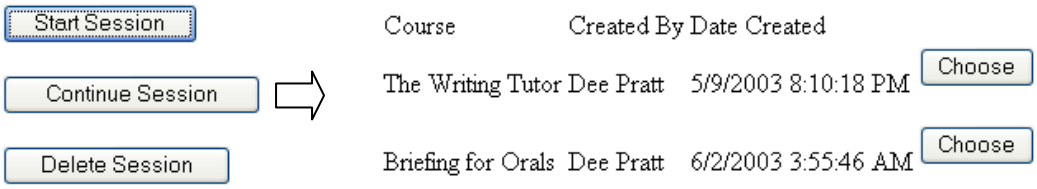


Figure 6: *Student options when accessing a course*

Features such as pop-up boxes and memo boxes make the lessons more interactive and stimulating than plain text notes or lessons, and provision has been made in the program for the addition of sound effects and music, graphics and video, although these first need to be saved on the web page or server where the CourseMaker is located, which is a problem when access to an institutional server is barred or locked.

CONCLUSION

While the program itself was not considered to be a feasible option for my Ph.D. CAI application, the making of CourseMaker proved to be an invaluable experience in learning about the design and production of educational software. In particular, it highlighted the difficulties involved in effectively communicating the envisaged working of the program successfully to a programmer, and the importance of translating basic functionality into key specifications for programming. In spite of the fact that it is not polished or completely finished, CourseMaker can be seen to have potential for setting up short interactive lessons or courses on the Internet, particularly since it can be used by teachers with advanced computer literacy but no knowledge of computer programming. The program is to be piloted on WebCT courses, where it is

hoped that some of the problems can be ironed out, for example, its slow operation when used with FrontPage texts, and the problem of adding sounds and graphics when these first have to be published to an institutional server or website with barred access. Readers who would like a live demonstration of CourseMaker should contact the writer at deep@dit.ac.za.

REFERENCES

Condon, S.L. & Cech, C.G. (1999) Discourse management in three modalities. http://www.ucs.louisiana.edu/~slc6859/cmcpaper.htm

Greene, D. (2000) A design model for beginner-level computer-mediated EFL writing. ***Computer Assisted Language Learning*** 13(3):239-252.

Pratt, D.D. (1990) The process approach to writing. In K. Chick (ed.) ***Searching for relevance***. Durban: South African Applied Linguistics Association.

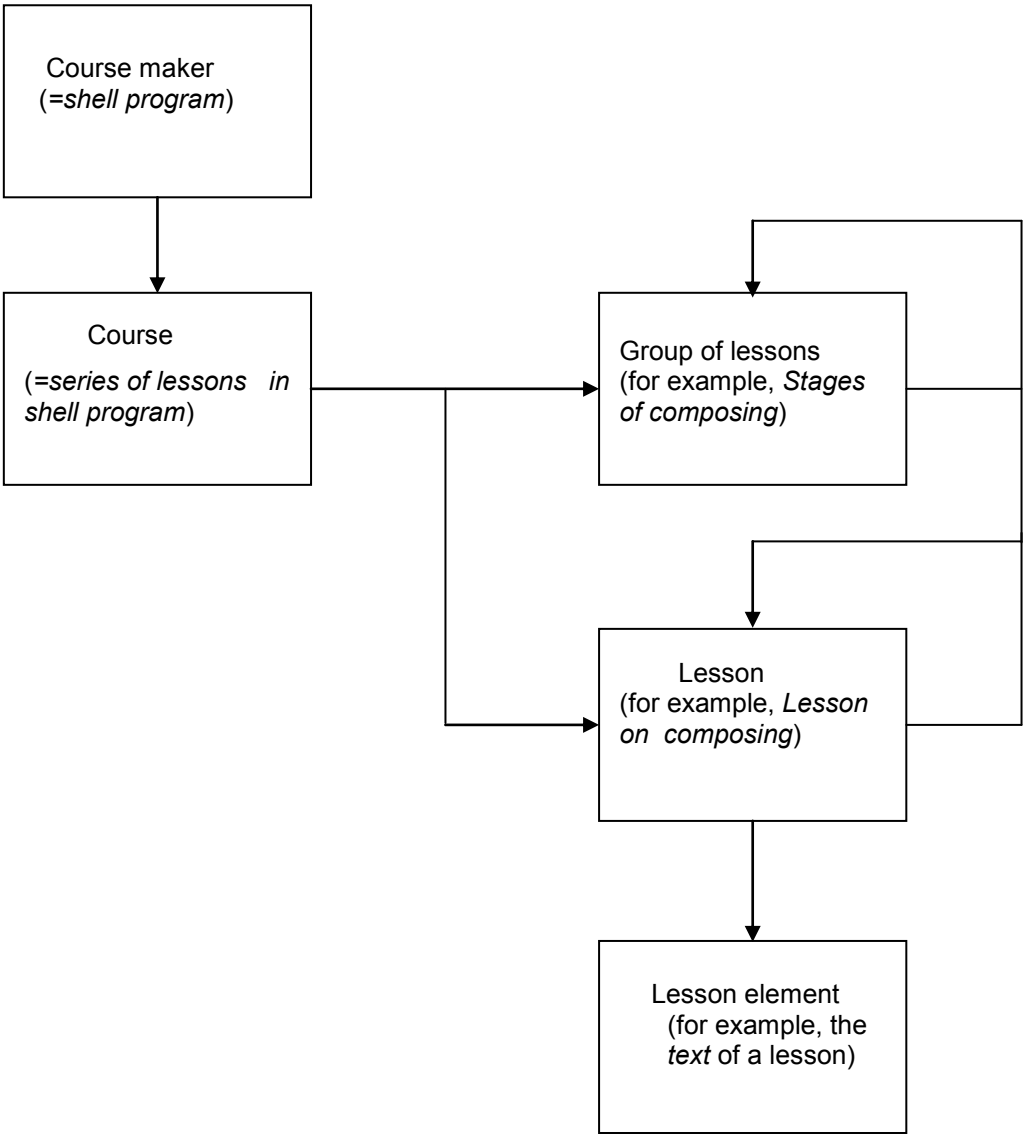
Pratt, D.D. (2002) From dummies to dissertations: an account of three virtual courses designed to facilitate the development of advanced academic and computer literacy in a master's degree programme. http://olc.dit.ac.za/bio/inhousepapers.htm

Raimes, A. (1985) What unskilled ESL students do as they write: a classroom study of composing. ***TESOL Quarterly 19(2)***:229-256.

Vanderstraeten, R. & Biesta, G. (1996) Constructivism, Educational research and John Dewey. http://www.bu.edu/wcp/Papers/Amer/AmerVand.htm

Zamel, V. (1985) Responding to student writing. ***TESOL Quarterly 19(1)***:79-101.

ANNEXURE: SCHEMATIC OF THE PROGRAM STRUCTURE



The above schematic of the program was worked out so that the programmer could have some insight into how the teacher sets up a learning programme, and so that the teacher could have some insight into how the computer program was structured. The programmer could then set up the data input sections of the program so that they made sense to the teacher. The names of the different sections are familiar ones to teachers in terms of course and lesson planning, and the hierarchical order of the various sections of a course are similar to what would be found in a typical syllabus list or text book index. The input sections for different levels and elements were to be set out and colour coded so as to make the input of data easier [this still needs to be done]. The schematic demonstrates to the teacher how the various sections of a

course can be sequenced or linked in the computer program. This is *not* a schematic of the computer program however, merely a construct intended to make teaching clearer to the programmer, and programming clearer to the teacher.

Glossary

Course maker: The course maker is the program template which provides the shell for the composition (or other) course.

Course: The course is an abstract concept only referring to the series of lessons contained by and run within the program template. The program template may be used for different courses with different content and functions.

Lesson: The course consists of a number of lessons which can be accessed sequentially or in random order as needed by the learner.

Groups of lessons: Some lessons are displayed as groups for teaching purposes. The grouping is an abstract concept only, and is not reflected in the programming, only in the display. The grouping of lessons is intended to assist the teacher to enter data, and may assist the learner by showing certain sequences or combination of lessons as meaningful, for example, the various stages of composing.

Lesson elements: As in the classroom, lessons consist of a number of different elements, for example, texts, diagrams, pictures, examples, references to further texts, and opportunities for the learners to ask questions and get feedback. Lesson elements are the various items which the teacher puts together to form lessons, and can be seen to correspond with the activities and teaching materials which teachers use to facilitate learning in actual classrooms. The main difference is that in the virtual classroom students can access elements of the lesson as needed, and can rapidly access multiple resources in ways which are not feasible or possible in the actual classroom. So far we have not identified any lesson elements which are specific to the virtual classroom and are not found (or their equivalents) in actual classrooms: this may change. An explanation of the various lesson elements is given below.

Video element: Video clips can be inserted into the program to give lessons.

Picture element: Pictures can be inserted to illustrate lessons.

Sound element: Sounds can be inserted for various reasons, for example:

- to provide the sound track of a lesson to accompany a video clip;
- to emphasise a point (right answer/wrong answer);
- to create a mood;
- to back up or emphasise visual instructions;
- to create a sense of fun.

Text element: Text elements are the main source of input, providing lesson instructions, subject content and materials, and correspond to the teacher's voice in the actual classroom.

Text box element: This provides space for student input or responses in lessons. As with teacher instructions or materials, student input can be accessed at any time by the student as needed.

Memo box element: As with the text box element, the memo box element provides space for student input or responses in lessons, but at greater length.

Check list element: This gives the students the opportunity to check whether various options have been/have not been performed.

Radio list element: As with the check list element, but appearing as a radio list.

Lesson link element: This allows students to access further texts (information or answers) from current texts, and has replaced the proposed help function in the list of options available, as help has been found always to take the form of further explanatory texts. The lesson link element always forms a link to another lesson element or part of it, and cannot be inserted before the lesson element to which it refers has itself been inserted.

Pop-up element: This is similar to the lesson link element, but provides the students with short answers rather than further long texts (or other elements); it is easier for the teacher to type in short answers in pop-up boxes rather than create a number of short text elements and link them up.

Drop down list element: This provides students with a list of options from which to choose.

Branching tree element: This tool provides students with a branching tree list (similar to a directory) to illustrate hierarchical relationships in data or processes.