COMPUTING SCIENCE

A Self-Taught Computer Engineering Course

J.N. Coleman, D.J. Kinniment, F.P. Burns and A.M. Koelmans

TECHNICAL REPORT SERIES

No. 546 March, 1996
TECHNICAL REPORT SERIES

No. 546 March, 1996

A Self-Taught Computer Engineering Course

J.N. Coleman, D.J. Kinniment, F.P. Burns and A.M. Koelmans

Abstract

The application of computer-aided-learning as a direct replacement for, as opposed to an adjunct to lecturing, is still in its infancy in higher education. This paper examines some of the reasons for its slow uptake and then describes a project to develop courseware for a large proportion of the Electronic Engineering syllabus within several UK Higher Education Institutions. The first modules to be completed cover the area of Computer Engineering. We describe the philosophy and design of this courseware, and then report a series of tests in which the examination performance of students using it was compared with that of control groups taught in traditional lectures. The results clearly suggest that carefully-designed courseware can lead to a large reduction in teaching time, with no significant difference in learning.

© 1996 University of Newcastle upon Tyne.
Printed and published by the University of Newcastle upon Tyne,
Computing Science, Claremont Tower, Claremont Road,
Newcastle upon Tyne, NE1 7RU, England.
Bibliographical details

SELF-TAUGHT COMPUTER ENGINEERING COURSE

A Self-Taught Computer Engineering Course
[By] J.N. Coleman [and others].


(University of Newcastle upon Tyne, Computing Science, Technical Report Series, no. 546)

Added entries

UNIVERSITY OF NEWCASTLE UPON TYNE.
COLEMAN, John Nicholas

Abstract

The application of computer-aided-learning as a direct replacement for, as opposed to an adjunct to lecturing, is still in its infancy in higher education. This paper examines some of the reasons for its slow uptake and then describes a project to develop courseware for a large proportion of the Electronic Engineering syllabus within several UK Higher Education Institutions. The first modules to be completed cover the area of Computer Engineering. We describe the philosophy and design of this courseware, and then report a series of tests in which the examination performance of students using it was compared with that of control groups taught in traditional lectures. The results clearly suggest that carefully-designed courseware can lead to a large reduction in teaching time, with no significant difference in learning.

About the author

Mr. J.N. Coleman has been a Lecturer in the Department of Electrical and Electronic Engineering since 1990.

Prof. D.J. Kinniment is a Professor in the Department of Electrical and Electronic Engineering.

Mr. F.P. Burns received his PhD in Electrical and Electronic Engineering at the University of Newcastle upon Tyne in January 1992 and is currently unemployed.

Mr. A.M. Koelmans joined the Department of Computing Science as a Lecturer in March 1985, after two years as a Research Associate on the VLSI project in the Department of Electrical and Electronic Engineering.

Suggested keywords

CURRICULUM ISSUES INSTRUCTIONAL SOFTWARE MULTIMEDIA

Suggested classmarks (primary classmark underlined)
Dewey (18th): 371.39445 621.381958
U.D.C. 378.1694 681.322.02
A SELF-TAUGHT COMPUTER ENGINEERING COURSE

J. Nicholas Coleman, David J. Kinniment, Francis P. Burns,
Dept Electrical & Electronic Eng;
Albert M. Koelmans, Computing Science Dept.;
University of Newcastle upon Tyne, NE1 7RU, U.K..

Contact: Dr J.e. Coleman
Dept Electrical & Electronic Eng,
Merz Court,

The University,
Newcastle upon Tyne,
NE1 7RU,
U.K..

Tel: +44 191 222 7860
e-mail: j.n.coleman@ncl.ac.uk

Subject areas: Instructional Software, Multimedia, Curriculum Issues.
A SELF-TAUGHT COMPUTER ENGINEERING COURSE

Subject areas: Instructional Software, Multimedia, Curriculum Issues.

ABSTRACT

The application of computer-aided-learning as a direct replacement for, as opposed to an adjunct to lecturing, is still in its infancy in higher education. This paper examines some of the reasons for its slow uptake and then describes a project to develop courseware for a large proportion of the Electronic Engineering syllabus within several U.K. Higher Education Institutions. The first modules to be completed cover the area of Computer Engineering. We describe the philosophy and design of this courseware, and then report a series of tests in which the examination performance of students using it was compared with that of control groups taught in traditional lectures. The results clearly suggest that carefully-designed courseware can lead to a large reduction in teaching time, with no significant difference in learning.

1. INTRODUCTION

In recent years a great many computer-based packages have entered the higher education curriculum. Examples of interest in the field of Microprocessor teaching include register level simulators [1], and higher-level simulators such as [2]. These educational packages have in common that they are intended to augment a traditional lecture-based course, in order better to illustrate the principal theme of the syllabus. They are not, by and large, intended as a direct replacement for the lecturing.

Observers have drawn attention to the fact that the application of such 'courseware', whilst becoming increasingly accepted in secondary schools, is still in its infancy in higher education [3]. Reasons postulated include the lack of suitable courseware, lack of time and staff to develop new material, and lack of financial and administrative support to do so. Whilst these problems are obviously soluble, a number of more fundamental doubts about the desirability of this type of teaching have tended to deter university staff from making the necessary commitment to it.

Let us examine some of these objections in detail. The first concerns its effectiveness as a teaching technique. Do the students learn as well from it as from traditional lectures? There is as yet little objective evidence on this point. Related to this is the fact that students in higher education are much less tightly supervised than their high-school counterparts, and a greater degree of self-discipline is therefore required of them if self-teaching is to be a success. Is it reasonable to expect sufficient commitment from them to make the experiment work? A second issue derives from the high degree of specialisation of university-level courses, and the tight integration of the several courses comprising a particular degree. This tends to prevent the easy transfer of courseware from one institution to another, and therefore exacerbates the shortage of suitable material.

Against these objections, a number of benefits have been suggested, some indisputable and others open to question. The one certain benefit is the reduction in staff lecturing time. However, it would be hoped that the time spent by the students in their self-teaching activities would be less than that taken by the original lectures, yielding a saving in their time too. Another possible advantage is that the students may positively welcome the new teaching techniques, and feel more motivated to achieve better results.

In 1992, starting from a base in which almost no suitable material was available, a collaborative project was established between several U.K. E.E. (and more recently C.S.) departments with the aim of developing courseware for a large proportion of the identifiably common parts of the degree
syllabus. In doing so, we had an opportunity to investigate the questions posed above. Standing midway between the creative arts and natural sciences, engineering was a suitable discipline for this experiment because of its reliance on large quantities of diagrammatic information. Whilst it was not felt that CAL could ever portray the creative design concepts pervading any advanced engineering activity, it could possibly express many of the fundamental techniques, ideas and principles characterising the early stages of a degree education.

By 1994 the first piece of courseware was ready for use. It was entitled 'Computer Architecture', and during the second term of the 1993-94 academic year it was tested by dividing a class of Electrical Engineering students; half doing the courseware and half coming to the equivalent lectures. At the end of the trial, a short test was given to the entire class to see if any difference could be observed in the scores obtained by the two halves. A more detailed analysis was then made, which took into account their overall average for the year. Encouragingly, we found that there was no significant difference between the two halves, but we did find evidence to suggest that the CAL may actually be superior to the lecturing in bringing weaker students up to standard.

The remainder of this paper describes the courseware itself, the experimental procedure, and the results we obtained. It then describes the integration of this and further modules into a complete self-taught Computer Engineering course, due for use in 1996.

2. THE COURSEWARE

Within the engineering disciplines, a degree course typically includes a series of specialised lecture modules of approximately one term's duration. The modules exhibit a complex pattern of interaction; some progressing sequentially, others reinforcing concurrent modules, and others having an all-pervading influence throughout the course. It is often a matter of some difficulty for the lecturing staff of the department, as they struggle to keep abreast of a rapidly changing subject, to keep the modules within a coherent framework. The resulting courses vary quite markedly between institutions, reflecting the different perceptions of priorities at each.

From the outset then, it is unlikely that a single suite of courseware could be appropriate for all degree courses in the subject. It is also unlikely that sufficient variants could be constructed to satisfy all the requirements. The designers of the courseware sought to bypass this problem by adopting a strategy of very fine-grained modularity. The material would be divided into very small modules, each corresponding to only a few hours of lectures; very much smaller than the one-term granularity of a lecture-based course. At this level of resolution less difference is observed between the teaching at separate institutions. Where a lecturer was satisfied that part of his material was adequately covered by the courseware, he would be able to substitute this for the lectures. Where his own material differed from the courseware, he would continue lecturing as before.

We were not unduly concerned that a patchwork of lectures and courseware might develop; indeed, we would seek to augment the variation by introducing practical work and written exercises. The aim was to counteract the inherent dullness of sitting at a computer screen for long periods by frequent changes of activity. We were somewhat anxious about this point because of the minimal supervision of the students, and the high probability that, left completely unsupervised, the less well-motivated may be tempted to skip through the computer-based material with little attention.

Thus the syllabus was divided into themes, and each theme into a set of modules. The first theme to be addressed covered the area of digital electronics. This was divided into 17 modules, of which the first to be completed comprised a basic introduction to the operation of a CPU. Entitled 'Computer Architecture', the module corresponds to four hours of lectures, and was used as an experiment to evaluate the effectiveness of this type of teaching.

The material we were dealing with was detailed and factual in nature, and not intuitively appealing to many students. We employed a number of techniques to maximise the chances that the
courseware would work successfully with minimal supervision, one important factor being the use, as far as possible, of an inductive style of reasoning. Thus we proceeded by prompting the student to recall some well-understood fact, then drew an obvious conclusion about the principle we were trying to illustrate. Often we used an explicit question-and-answer format in consecutive frames, for example:

**Question:** What does a computer do?
**Answer:** It inputs information, stores it internally, processes it, then outputs the results.
**Conclusion:** So a computer must have four blocks of electronic circuitry: to input, store, process and output the information respectively. They must be connected so as to be able to pass information to each other.

This leads directly to the basic model of a computer system comprising input, memory, arithmetic unit and output; a model which is used to inculcate the concept of processing throughout the early part of this module. It is accompanied by animated displays in which the concept of a machine instruction is introduced naturally by showing data moving around the system in response to simple high-level-language commands. The student is then prompted to realise that the machine instructions themselves must have some physical realisation, and from this is developed the idea of the fetch-execute cycle, control unit, and attendant hardware items such as the instruction register and program counter.

As the complexity increases, a more tutorial style is unavoidably used to present the less intuitive aspects; for example the student is directly told how the hardware units are organised internally. In any case, a deliberate attempt is made to juxtapose contrasting styles of presentation, another commonly-used technique being example-driven.

High degrees of animation and interaction are of crucial importance to the presentation. An example is shown in fig. 1. Here the student is asked to point to the electrical signals which will be activated as a particular instruction is executed. When the correct signals are identified, the student is rewarded with a moving display of the execution of this instruction. A development package, Authorware Professional, was specially chosen for its well-developed animation facilities. It offers precise control over visual composition, timing, and trajectories of moving objects, and allows flowchart-based sequencing controlled by accurate positioning of the on-screen cursor via the mouse.

![Diagram of computer architecture](image)

**Fig. 1. 'How Instructions are Executed'**
By superimposing the factual material on an undercurrent of continuously changing pace, presentation and activity, we hoped to maintain the students' interest throughout a ninety-minute computer session, as well as we would in a lecture. In the next section we describe how we put our objectives to the test.

3. EXPERIMENTAL PROCEDURE

Broadly speaking, we have set out to answer four questions.

- Do students learn as well from the courseware as from the lectures?
- If so, then how much time is saved by the lecturer and by the students themselves?
- How much supervision is needed?
- Do the students find the work interesting?

We ran a trial of the courseware with a set of specialist Electronic Engineering first year undergraduates, doing Computer Engineering as a compulsory subject. The entire course consisted of 13 hours of lectures, of which the Computer Architecture courseware covered material equivalent to four hours of lectures. The class was divided into two halves by random allocation. At the appropriate point in the course, one half was sent to do the self-learning work, and the other continued to attend the lectures. It was emphasised to the students that this component of the course would not be the subject of any official examination, although it did serve as a background to later parts of the course which would be assessed. It was promised that if large discrepancies appeared between the two halves, remedial teaching would be offered to the disadvantaged group.

After the trial, a short multiple-choice test was given to the entire group. The test was used to determine whether any significant difference could be observed between the results obtained from the two teaching methods. We also thought it useful to a limited extent to canvass the students' opinions, and whilst we did not find it acceptable to ask for their judgement on the quality or content of the material, we did seek their views on the preferred style of presentation. From attendance records we also calculated the average time spent in each of the two teaching methods.

The two halves of the class were then reunited, and teaching continued in the lecture theatre with the next stage of the Computer Engineering course, covering Assembly Language programming. Some months later, their comprehension of the entire Computer Engineering course was examined by way of an assignment which involved writing an Assembly Language programme to solve a simple problem. The Computer Architecture work served as an essential background to this assignment, whilst not being directly the subject of it. We have again used these marks to look for a significant difference between the two groups. The analysis in this case also took into account each student's overall end-of-year average, in order to eliminate the effects of any differences in basic ability.

4. RESULTS

We begin with the results of the multiple choice test taken immediately after the end of the courseware sessions. Table 1 shows the results for each teaching method. Col. 2 gives the number of students involved; col. 3 the mean score obtained out of a total mark of 20; col. 4 the standard error of the mean; and col. 5 the time spent in hours. Column 6 gives the result of a Student's t-test, testing the hypothesis that the results for each teaching method come from the same population, i.e. that there is no difference between the effectiveness of the two methods in producing examination scores. P values below 0.05 are taken to indicate a reasonable degree of certainty that there is a significant fundamental difference between the two teaching methods. A P value above 0.05 indicates that such difference as was observed could easily have occurred by chance, and that there is therefore no reason to suspect an underlying significant difference. The
ratio of times spent, 1.47 / 3.58 hours, is 1 / 2.44 = 0.41.

The end-of-year results are as shown in table 2. One outlier was eliminated from each group. The table shows the final Computer Engineering mark, and the overall end-of-year average excluding Computer Engineering. It is evident from the results that a) the marks for Computer Engineering were higher than the overall averages and b) the students doing CAL were weaker overall than those coming to the lectures. To eliminate these inequalities a final set of results was calculated, based, for each teaching method, on the difference between each student's Computer Engineering mark and his or her overall end-of-year average excluding Computer Engineering. It is reasonable to expect that if the CAL and lecturing were equally effective teaching methods, then no significant difference would be observed between these two distributions; this, in effect, correcting the results for differences in basic ability.

In the survey of students' opinions, a majority of 80% expressed a preference for a mixture of lectures and CAL, with 63% preferring a predominance of CAL. 84% would have liked printed backup material, but only 26% wanted people at the courseware sessions to answer questions. Asked whether they found the work boring, 70% said no, 20% slightly, and 10% yes. 45% described the work as easy. When invited to suggest any additional features apart from printed backup, most respondents asked for more interaction and self-check exercises.

It is worth mentioning that a previous trial of the courseware was carried out under less well-controlled conditions in which the students were allowed to do the courseware in their own time with no record kept of their attendance. Analysis of the multiple-choice test in this case yielded a P value of 0.0017, and led us to impose a certain amount of external discipline (attendance rota and register check by a lab. assistant) in the trial reported here.

5. DISCUSSION & CONCLUSIONS

We begin with the results of the multiple choice test taken immediately after the courseware sessions, as shown in table 1. It is evident from the t-test that there is no significant difference between the exam scores produced by the two teaching methods, which gives some grounds for optimism that carefully-designed courseware can indeed be an effective replacement for lecturing.

Turning to the end-of-year results in table 2, we first observed that the students doing the CAL were weaker overall than those coming to the lectures. We therefore calculated, for each teaching method, a distribution of the difference between each student's Computer Engineering mark and his or her overall average. We then calculated the probability value of a significant difference between these two distributions. It is evident from this P value that once again there is no reason to suppose that the two teaching methods are significantly different. It must, of course, be pointed out that if a similar difference were observed with a larger sample, it would become significant. However, the difference that was observed in this sample was clearly in favour of the students doing the CAL. The Computer Engineering marks of the latter were 6.9 higher than their overall marks, whereas those of the students who came to the lectures were only 2.8 marks higher. Although more experimentation would be necessary before any strong conclusion could be reached, this does suggest that if in a larger survey a significant difference did emerge between the two teaching methods, it would probably show the CAL to be more effective than the lecturing in bringing the weaker students up to standard.

One indisputable point, however, is the large reduction in teaching time. The 4 hours of lecturer's time were eliminated completely, and that of the students reduced by a factor of about 2.5. It is possibly in this area that CAL will most prove its worth.

Our policy of mixing CAL and lecturing seems to have struck the right note with the students. We appear to have been successful in maintaining their interest (although a check on attendance was necessary) and they also seem happy not to have staff on hand at the courseware sessions to answer questions. However, the general matter of how extensively this type of teaching material
can be deployed before the students begin to lose interest is very much open to question. The survey indicated that they would prefer a predominance of courseware, but there must be some doubt as to whether this preference would be maintained once the 'novelty value' had worn off. The following section describes a new programme to test this point further.

6. FURTHER WORK

Given the apparent success of the 1994 trial, the entire 1996 Computer Engineering course has been redesigned around three CAL modules, of which the two new modules are currently in development. The old and new style course contents are summarised in Table 3. It is clear from the trial results that the students appreciate variation in presentational style and activity, and particular emphasis is placed on this aspect in the overall design of the course. More attention has been given, for example, to the inclusion of self-check sequences, an example of which is given in Fig. 2 from the module on Number Systems.

Fig. 2. 'Hexadecimal Arithmetic'

The students' preference for interaction was addressed in a different way in the Assembly Language module. This had previously been associated with 15 hours of practical work on M68000 development systems with a debugging monitor. In many Computer Science courses, this work is now done using a simulator such as those described in the references. Consideration was given to interfacing the courseware with one of these simulators, which would have eliminated some hours of familiarisation time and a great deal of hardware. However we chose to maintain the use of the real hardware because as an Electrical Engineering department, our students require a practical knowledge of microprocessor development systems in later years. Since this practical work is entirely interactive, a balance was maintained by making the associated courseware more tutorial in nature. This module is also heavily dependent on bookwork, as the courseware teaches by example only, and detailed information about the operation of each instruction has to be obtained by the students directly from reference texts.

A coursebook accompanying each module includes a statement of the objectives, a summary of key points, and further written exercises where appropriate.

Although individual contact with the students is still maintained via the practical sessions, it is evident from Table 3 that 13 hours of lecturing time have been eliminated, and replaced by 6 hours of courseware. We have, however, used some of this gain to introduce subsidiary lecture topics which were not included previously. The aim was to meet the students' preference for at least the occasional traditional lecture throughout the course, and it also has the advantage that the lecturer is allowed to concentrate on more abstract concepts whilst the students deal with basic matters themselves. This, paradoxically, seems very reminiscent of the ancient order of university teaching,
rarely seen in technical subjects and uncommon elsewhere, but which may now be about to make a welcome return.

REFERENCES


ACKNOWLEDGEMENTS

This work was undertaken as part of the Teaching and Learning Technology Programme, in collaboration with several U.K. Higher Education Institutions.

Table 1. Multiple Choice Test Results

<table>
<thead>
<tr>
<th>Teaching Method</th>
<th>No of Students</th>
<th>Mean</th>
<th>SEM</th>
<th>Time (hr)</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lectures</td>
<td>25</td>
<td>12.52</td>
<td>0.60</td>
<td>3.58</td>
<td>0.21</td>
</tr>
<tr>
<td>CAL</td>
<td>23</td>
<td>11.39</td>
<td>0.60</td>
<td>1.47</td>
<td></td>
</tr>
</tbody>
</table>

Table 2. End-of-Year Examination Results

<table>
<thead>
<tr>
<th>Teaching Method</th>
<th>No of Students</th>
<th>Comp Eng</th>
<th>Mean</th>
<th>SEM</th>
<th>Overall</th>
<th>Mean</th>
<th>SEM</th>
<th>Comp Eng - Overall</th>
<th>Mean</th>
<th>SEM</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lectures</td>
<td>22</td>
<td>63.0</td>
<td>2.0</td>
<td></td>
<td>60.2</td>
<td>2.3</td>
<td></td>
<td>2.8</td>
<td></td>
<td>2.7</td>
<td></td>
</tr>
<tr>
<td>CAL</td>
<td>24</td>
<td>57.5</td>
<td>1.6</td>
<td></td>
<td>50.6</td>
<td>2.5</td>
<td></td>
<td>6.9</td>
<td>2.3</td>
<td></td>
<td>0.25</td>
</tr>
</tbody>
</table>

Table 3. Content of Lecture vs CAL-Based Courses

<table>
<thead>
<tr>
<th>Module</th>
<th>Lecture-Based Course (hr)</th>
<th>CAL-Based Course (hr)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Lectures</td>
<td>Practicals</td>
</tr>
<tr>
<td>Introduction</td>
<td>4</td>
<td>1.5</td>
</tr>
<tr>
<td>Num. Systems</td>
<td>4</td>
<td>1.5</td>
</tr>
<tr>
<td>Computer Arch.</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>µP Applications</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Assembly Lang.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Program Style</td>
<td>13</td>
<td>6</td>
</tr>
<tr>
<td>Overall</td>
<td>28</td>
<td>24</td>
</tr>
</tbody>
</table>