ENGINEERS AND COMPUTER SCIENCE

D. Herbison-Evans

Technical Report No. 127
January 1978

BASSER DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF SYDNEY
NEW SOUTH WALES 2006
AUSTRALIA
ENGINEERS AND COMPUTER SCIENCE

D. Herbison-Evans, M.I.E.E. (Chartered Engineer)
Lecturer, Computer Science Department
University of Sydney

Abstract

The article discusses the distortions of computer science courses at Sydney University which seem to be based on the important Computing uses of twenty years ago namely Engineering applications. These are manifest as biases towards Numerical Methods and the Fortran language. In an attempt to reduce the impact of Engineering applications on the general courses, a Fortran course specifically for Engineers has been taught for the last three years at Sydney. This is described. Another big shift in computer uses is envisaged in the near future from business to recreation. The concomitant changes in Computer Science courses are discussed.

In the beginning, engineers designed computers. For ten years, engineers dominated the use of computers. Then the business world discovered computers too. The business world is bigger than the engineering world, so ever since then, for the past twenty years, business uses have dominated computing applications.

A University is a very conservative affair. The Computer Science courses at Sydney University are still dominated by the demands of engineers. These are manifest in two ways:

(1) Fortran: As Latin is to Theology, so is Fortran to Engineering. It is a dead lingua franca. Most Engineering programs are written in Fortran. This has lead to pressure to have Fortran taught as the first language, so that engineers who attend only the first year of our course sequence will be appropriately trained. It is still the first language taught at Sydney University, over a decade after COBOL became the main language in use outside, and a decade after the publication of ALGOL 68.

(2) Numerical Methods: Most Engineering computations are numerical. The largest applications are typically concerned with either structural or flow rate calculations. The former demand the solution of simultaneous equations, the latter the solution of differential equations. To teach an understanding of the limitations of computers in solving these problems takes currently 72 lectures on numerical methods at Sydney University. This takes students to an introduction to the condition number of a matrix and to the distinction between the three types of second order partial differential equations. The total number of undergraduate lectures is 288, so that approximately ¼ of the course is directed to the engineering demands.
A more insidious aspect of this bias is that in order to teach this numerical work, a mathematical prerequisite is imposed on each of the sequence of courses. These prerequisites are totally inappropriate for people entering the world of commerce, and serve mainly to deter such people from taking Computer Science at Sydney.

The mathematical prerequisites are rationalised as being a necessary training in symbolic manipulation. My own feeling is that Latin, as a finite language with a formal syntax, is the best prerequisite for Computer Science, closely followed by Music, Chemistry and also Mathematics.

**Engineering Withdrawal**

Sydney University has begun the process of engineering withdrawal. For the past three years, I have been giving a tailored course on Fortran to second year engineers. Regretably, many engineers still take our regular courses so that the Fortran and numerical pressures are still there and indeed irresistible. Fortran is still the first language, but 36 of the numerical lectures are now optional. The maths prerequisites remain.

The Engineering Fortran course is very practical: 18 lectures and 36 practical hours. The lecture series follows the syllabus of A.H.J. Sale and has the following features:

(a) introduction of DO loops before IF's and GOTO's, to encourage the view of a loop as a conceptual unit;

(b) relegation of Arithmetic IF, DIMENSION, P format factors and one line function statements, to the last lecture (on 'Archaic features') and a ban on their use in exercises;

(c) relegation of COMMON, EQUIVALENCE, Computed and Assigned GOTO's, and BLOCKDATA statements, to the penultimate lecture (on 'Dangerous Powerful features');

(d) sorting is mentioned only once;

(e) BNF occupies only ¼ of a lecture;

(f) error avoidance and debugging take one lecture.

The exercises are mainly numerical e.g., finding the primary real root of a cubic, interpolating in a two dimensional table, finding the minimum of a function, plotting a function, integrating a function.

Special emphasis is placed on responsible programming, and the dangers of the casual use of a powerful machine. Most exercises include a data set which causes a machine error or an incorrect answer. The attitude which I try to instil is that where engineering decisions are concerned, it is not enough to get an answer; it must be the correct answer.
Special emphasis is also placed on documentation, with the simple mnemonic that it should answer all the interrogatives: who, when, where, what, why, how.

An engineer must be prepared to put his name on his work, and his work should be of such a standard that he is proud to put his name on it. This is the 'who'. Students are encouraged to describe the problem and the method of solution in English before writing a program, as they know English better than Fortran. This is the answer to 'why' in the documentation. All variables must be briefly defined in English. This is the answer to 'what'. The format of the input and the limitations of the program are the 'how'. The installation and comments on portability are the 'where'.

I think that it is important that the course be given from the Computer Science Department, as pure Engineers do not yet have the experience to teach this disciplined style of programming.

The Near Future

As Engineers get more experience and as Computer Science Department staff levels rise to values appropriate for their student numbers, then it will become possible to separate not only the Fortran but also the Numerical Methods from the main stream of Computer Science. Indeed, it may be appropriate to consider streaming Computer Science courses not into A/B or Pass/Honours, but into Numerical/Commercial in the near future.

Already at Sydney, the Computer Science courses include their own introduction to discrete mathematics. The mathematics of Computer Science itself is not introduced at school or in Mathematics Department courses, so the need of mathematical prerequisites will disappear. Then computer science will be an academic discipline in its own right.

Further Future

As computer science departments adjust in the near future to the computer usage of twenty years ago, a new computer revolution is already underway. Large scale integration is dropping the price of microprocessors below that of a light switch. The general public will have computers in the home before Computer Science departments have sloughed their engineers.

In looking at the future of computers, we should bear in mind the error of Mr Edison's assumption that the main use of his acoustic recording device would be in the business office: The main use of computers in five years time is likely to be recreation. (How much more effective is it to make a car into a hot rod by reprogramming its micro than by polishing its exhaust valves.)

By then, students entering computer science courses may be expected:

(a) to know Basic; and

(b) to have a grounding in Computer Architecture.
The main problem in computing will then be not in how to do a computation, but in how to present its results in the most human understandable form. These changes may necessitate some further alterations to Computer Science course syllabus.