

COMPUTING FOR THE NEW GENERATION

How, in a world of plug and play, self-configuration, automatic updates and installations, personal profiles and information on-tap, can we convince students that there is work still to be done? *Is there work still to be done?*

Richard Hebblewhite

Stuart Cunningham

Vic Grout

Centre for Applied Internet Research
University of Wales, NEWI Wrexham

ABSTRACT

This paper notes both the technical advances in applications and communications over recent years and the effect these have had on the learning experience of computing students. The ongoing change in the nature and balance of course material is discussed, with a particular emphasis on programming. Key issues relating to perception and delivery are introduced in conclusion.

INTRODUCTION AND BACKGROUND

Advances have been made across the entire information technology spectrum in the past two to three decades. Two in particular are relevant here:

- The ubiquity of the Internet.
- The sophistication of computing applications.

Through the use of improved, largely simplified, hardware and software solutions, a typical computer user can now perform complex activities and search vast information banks in a manner that would have been considered unthinkable a single generation ago. Indeed, anyone in possession of the skills necessary to maintain, or even operate, a computer from the late seventies or early eighties, would now be considered a computing professional. As a result of this increased range of applications and the reduction in skill necessary to use them, the emphasis in most computing courses has shifted from the computer science based interest in understanding to the computer studies principles of exploitation. It is common today to find entire computing degree programmes that, apart from a grudging concession to a computer systems module or two, do little more than introduce students to a succession of packages written for them by someone else. Programming in particular suffers in two respects: firstly as just another applications environment to master but one whose relevance students struggle to appreciate; secondly, in requiring skills not to be found elsewhere in the curriculum, as an unpopular and difficult subject.

DISCUSSION

For a number of years now, programming and systems modules, once the foundation of any computing programme have been under threat. The causes are not complex, merely linked, and can be summarised as follows:

- It takes considerably less skill to achieve substantially more in modern computing. Off-the-shelf solutions abound and are readily available and easy to implement. The complexities of algorithmic design, language syntax, architectural and organisational studies are seen as unnecessary and unpleasant by many students, and some staff.
- With staff-student contact hours a scarce resource on most courses, the largely laboratory-based, time-intensive, often unpopular, developmental subjects such as programming have been ready candidates for the axe. The introduction of rapid applications development techniques and visual programming have, far from alleviating the problem, merely caused students to lose sight of underlying principles altogether and further confusion results.
- With the wealth of material available on the Internet, it is becoming increasingly difficult to find assessment methods with adequate protection against plagiarism. Programming suffers particularly in this respect. The use of cut-and-paste, for example, is easily detected and proved in a report but far less so in program code. There is also a growing attitude, for right or wrong, that much of the code that needs to be written, *has* been written, so why learn how to write more?

And of course, the process is cyclic. As the technical areas are seen as increasingly irrelevant and unpopular, they are, where possible, avoided more and taught less. The subjects become less common and the skills regarded as more specialist. The perception of computing as a subject shifts. Under constraints of lack of time and resources, quality of delivery declines and attitudes harden further.

QUESTIONS AND CHALLENGES

So what, if anything, should be done to rescue the technical areas, particularly programming, of a computing degree? Should we find ways to make these subjects more approachable, enjoyable, understandable, relevant, even simpler, or are they to be abandoned to the sanctuary of the computer science programmes, which in turn become the equivalent of the cryptic crossword in a world of word-searches? There may not be simple answers in store but these questions must be posed nonetheless:

- Will the continued erosion of technical content from many computing programmes lead to a two-tier graduate system: the builders and the users? If so, are we content to let this happen? Is it right for the majority of the computer-literate population thus educated to be devoid of technical knowledge, to become borrowers, in effect, of expertise available from elsewhere?
- If we think it necessary to support the ongoing inclusion of programming and systems subjects on courses then what can be done to make them more accessible to the majority of students? How can they be delivered in a hostile environment with originality and assessed with authenticity?

There are some tough choices ahead of us!