ELECTRONIC PUBLISHING, VOL . 5(2), 53–54 (JUNE 1992)

Editorial

Software engineering and document manipulation have strong resemblances and both of them are concerned with generating commercial products (software programs or books and magazines). Document manipulation environments are now adopting, after a 10 year delay, the very techniques that were originated for programming language design and manipulation. In the same way, the techniques used for teaching about document manipulation ought to parallel those used for teaching about programming languages. Indeed, we do find that teaching techniques developed for programming languages *are* now being adapted to teach document manipulation, but again with a 10 year delay and unfortunately repeating the same errors that were made 10 years ago.

When the first lessons were given on computer programming, the focus was only on the computer's machine language and not on broader computer science principles. Obviously, the vision of early teachers was limited by the low level of these highly specific machine codes. Similarly, the first phototypesetters were driven through highly exotic codes and their use was taught in the same way as for machine language on computers.

When languages such as FORTRAN or Cobol became commonplace then most programming courses became 'FORTRAN oriented' or 'Cobol oriented'. However an understanding of a programming language without an accompanying understanding of the underlying principles of programming leads to the delusion that you are a programmer. For example, after a three-day seminar on programming a student might be given some positive integer number, greater than one, that is known to be a perfect square and might be asked to compute its square root. One way to proceed is to take each positive integer in sequence (2, 3, etc.) and multiply it by itself. If the result turns out to be the the given number, then a solution has been discovered. But this is not the way a professional programmer would proceed.

In a similar fashion there are, today, many Desktop Publishing courses which are 'Framemaker oriented' or 'Word-*n* oriented'. These Electronic Publishing courses are typically designed to teach students only how to use a particular system. After a few day-long seminars on Word 4.0, the student can write a style-sheet and compose a page with 10 different lovely fonts. But this is not the way a professional designer would proceed.

Even though the final goal of programming is a program, to be an effective programmer a student must learn *how* programs are built. Similarly, even if the final goal of EP is paper sheets or screen images, a student must learn how the *how* the papers sheets or screen images were conceived. Program specification languages along with the fundamental principles of programming form the basis for teaching computer programming. Before we can teach electronic publishing effectively we must design page specification languages that are more accessible than DSSSL and more formal than the anecdotal know-how of designers. However we must not forget to study the traditional principles of publishing. Electronic Publishing has a lot to learn, in terms of education, from computer science history.

CCC 0894–3982/92/020053–02 *Received 1 October 1992*

Even though documents and programs have much in common, a key difference must be noted; a program is intended to be executed by a computer while a document is intended to be read by a human reader. The document's visual aspects require not only the involvement of software engineers (as in programming language environments) but also the involvement of type designers, editors, etc.

Another problem is that Electronic Publishing, Desktop Publishing and other concepts, are not well defined. Until recently, there were three different fundamental 'roads' leading to today's publishing technology. The first road originates with the chancelleries' cursive hand-writing which was then replaced with typewriting and today with word-processing. The second road — the Royal Road typographers say — comes from hot metal, phototypesetters and now lasertypesetters. Today the first road is joining the second one as evidenced by the title of a recent book, *The Mac is not a typewriter*. The third road, based on the structure of the document rather than its appearance, started with phototypesetters and then lead to structured documents, SGML, and even hypertext. This path is also converging with the others but the result is not yet well-defined. Clearly the result must be defined before it can be taught to students.

To help form this definition, we held the TEP'92 (Teaching Electronic Publishing) workshop in the spring of 1992 in conjunction with the EP92 conference at Lausanne, Switzerland. The papers in this special issue of EP-odd are selected from those presented at that workshop.

I would like to thank the authors for agreeing to permit republication of their papers and Richard Furuta for his important work as editor, especially on those papers that were not written by native English speakers.

JACQUES ANDRÉ, GUEST EDITOR