ELECTRONIC PUBLISHING, VOL. 5(2), 91–96 (JUNE 1992)

Teaching electronic publishing to computer scientists

H. BROWN AND I. A. UTTING

Computing Laboratory,
University of Kent at Canterbury
Canterbury, Kent CT2 7NF, UK

SUMMARY

This paper discusses some of the issues involved in teaching electronic publishing to undergraduates specializing in computer science. It attempts to identify the significant differences between a course designed primarily for users and a course designed for specialists who may also become future developers and implementers.

KEY WORDS Computer scientists Electronic publishing Principles Design

1 INTRODUCTION

Introductory electronic publishing courses are typically designed to teach students how to use a particular system. More advanced courses may include elements of document design, and a survey of available systems and technologies with a comparison of their capabilities, but they are still normally designed to teach specific skills and so may be considered as 'service' courses.

An electronic publishing course for undergraduate computer scientists needs to cover a significantly wider range of topics. In order to be considered academically respectable it must provide an appropriate balance between theory and practice and—in addition to teaching specific skills—it should:

- identify and teach the fundamental techniques and principles used by electronic publishing systems;
- cover areas that are often carefully hidden from users (such as document structures, font information, and page description languages);
- take a broad definition of electronic publishing, covering hypermedia and active documents as well as high-quality formatting systems;
- look at existing and emerging standards.

The advent of all-electronic documents and the incorporation of active elements into these and into documents intended for paper is leading to an integration of electronic publishing and other areas of computing. For instance, documents with processable elements can be viewed as interfaces to other systems, and hypertext and user interface design overlap substantially. Computer science undergraduates are in an ideal position to appreciate the possibilities of these advances, and to understand some of the more general technical problems associated with hypermedia.

CCC 0894–3982/92/020091–06 *Received March 1992*

The sections below discuss the goals of undergraduate electronic publishing courses, and attempt to show how their content can be tailored to exploit the background and knowledge of computer science students. The discussion is based primarily on a current course at the University of Kent, but draws on experience we have gained from teaching electronic publishing in New Zealand, North America and elsewhere.

2 TEACHING ELECTRONIC PUBLISHING AT KENT

We have been teaching electronic publishing to undergraduate students for some ten years now, and to computer science undergraduates for six years. The course currently occupies one unit (one-eighth of a year's work) in the final year, and is optional for most students. It has regularly proved to be one of the most popular options.

Since the course was set up, we have attempted to combine the service and academic elements introduced above. In deciding on the balance we consider the situations the students are likely to find themselves in during their future careers. Nearly all of them will become users of electronic publishing systems, many will find themselves responsible for such users (as advisers and managers), and we hope a significant number will become developers of innovative systems. We thus attempt to combine a general 'literacy' approach with a study of principles and design decisions, and to identify and describe relevant standards.

The students have all used document preparation systems in earlier courses, and perhaps in previous careers, but they have no background in design. It is thus essential to cover the basic principles of digital typography and document design near the start of the course. Even for design, however, the skills of the computer scientists can be exploited in translating designs into the increasingly complex notations used by electronic publishing systems, and in understanding how the limitations and advantages of current and future technologies can affect design.

Given the above considerations, our electronic publishing course covers the principles of the following areas:

- imaging device technologies;
- digital typography;
- basic document design;
- paragraph and page layout (based on the TEX model[1]);
- penalty copy (tables, diagrams, mathematics, etc.);
- device independence;
- page description languages;
- structured documents and standards;
- hypertext and active documents.

Throughout the course the principles are reinforced with practical experience and assessment exercises. Time and financial constraints, as well as a recent rapid increase in student numbers (90 in 1991, 125 in 1992), mean that we cannot make this as wide-ranging as we would like. In the early stages of the course, students use troff (plus preprocessors), LATEX, and a small range of interactive systems, all running on our workstation network. At this stage they are encouraged to compare and contrast the facilities and user interfaces provided. Later on they write a PostScript[2] program and use Guide[3] and an SGML[4] parser.

3 DESIGN/LAYOUT

This is undoubtedly the area in which the average computing undergraduate student is weakest. Experience has shown that, although all of them are readers, few if any have noticed that the design of the documents they read is the product of a rational process, let alone developed an awareness of what such a process involves.

Given the disagreement between design practitioners as to what constitutes ‘good’ design, this undiscerning approach is hardly surprising. But it does mean that ‘principles of good design’ are hard to expound, ambiguous when they can be discerned, and hedged around with many caveats as to their applicability. There is no silver bullet, even on issues as basic as the preferability of ragged or flush-right margins.

Fortunately (in an argument taken from Pirsig’s *Zen and the Art of Motorcycle Maintenance* [5]) there is an amazing degree of agreement as to whether any particular example is of ‘high’ or ‘low’ quality—which can be a useful lifeline for those lost in a twisty maze of conflicting advice. This approach can be combined with physiological and cognitive arguments (such as those rehearsed in Rubinstein[6]) to which computing students are particularly susceptible.

Unfortunately design issues are also the area in which most computing staff are weakest. It can be difficult to justify spending a large large amount of course time on design, but asking colleagues in design institutions to recommend or teach a useful (and usable) subset of document design to occupy just a few hours is not likely to elicit an encouraging response. The approach we have adopted is to provide an overview of current practice, and to reinforce this with examples from as wide a spectrum of opinion as possible.

4 CHOICE OF SYSTEMS

Teaching of general principles is often supported by the use of particular systems which embody them. This is the method which we have attempted to follow in our teaching of electronic publishing: using document preparation and hypertext systems to reinforce and practice the underlying ideas being expounded in lectures.

In the area of hypertext, the choice of which particular system to use is essentially a pragmatic one; many of the available ones display appropriate behaviour at the depth to which they are likely to be investigated. In the area of paper-based systems, however, there is a more sharp delineation to be addressed.

Document preparation systems based on mark-up (such as LaTeX), while making explicit the crucial distinction between logical and physical structures of a document, are difficult for beginners to use, and provoke the frequent cry from those more interested in use than principle of ‘why use this, it’s so much easier using [my favourite word processor]’. This argument will be familiar to anyone who has attempted to introduce new concepts to those with skills in an associated (but simpler) paradigm. However, it can be effectively combatted by setting exercises involving changes to an existing large piece of work, which has the added advantage of letting the students see the use of techniques beyond the scope of the assessment.

We make relatively little use of the simpler DTP systems because they tend to obscure the logical/physical distinction and encourage students to invent design elements on a case-by-case basis as they create a document, leading to incoherent and inflexible document structures. On a more pragmatic note, using a DTP system for coursework requires that students have access to workstation or PC/Macintosh screens (of which we have dozens) for all stages of the assessment, whereas using markup-based systems most of the work can be performed from simpler terminals (of which we have hundreds).

Despite this, the choice we make must be based on the degree to which the system exhibits the principles we are teaching. The recent negotiation by UK universities of favourable educational licences for Interleaf[7] will enable us to take advantage of its object-oriented and highly structured approach to document specification, while giving students a much more attractive and easy-to-learn interface.

5 DOCUMENT AND INFORMATION STRUCTURES

Throughoutour course we place a strong emphasis on the logical structure of information and on the document and hypertext structures available to support it. Computing students are particularly strong in this area. Trees and directed graphs come naturally to them; they can understand the use of simple grammars to describe document classes and readily appreciate issues of object-oriented design, inheritance, and dangling cross-references. Similarly, when moving into areas of hypertext and active documents, they quickly understand how scripts or methods can be associated with active document elements and how these may launch other programs in order to provide database searching or other forms of specialized processing.

There is little time within our course for students to experiment with information design in any serious way, but those who also undertake their major undergraduate project in this area frequently become ambitious in experimenting with complex information structures. Possibly they are apt to become too ambitious—they rarely succumb to the disease of ‘fontitis’ but are only too apt to have bad attacks of ‘linkitis’ and ‘activitis’. This is an illustration of the well-known problem of computer scientists becoming so enthralled with their own inventions that they lose sight of the needs of users.

6 STANDARDS

At present there is little doubt about the important standards that need to be covered in an electronic publishing course. SGML and ODA[8] are essentially the only two well-established official standards in the field. Currently the course covers PostScript as the *de facto* standard for page description languages, and introduces ODA and SGML as examples of standards for structured documents and as a vehicle for illustrating how to derive multiple views of a document from a single logical description.

The picture will be a lot less clear in the near future. Many standards related to document processing have just been published or are currently going through the later stages of the ISO process on their way to becoming official standards. These can be divided into three distinct groups as follows:

- ODA-related—document application profiles (DAPs)[9] and formal specifications;
- SGML-related—DSSSL[10] for SGML document formatting and processing, SGML applications for hypermedia and music, and many miscellaneous support facilities;
- Document printing, filing and retrieval—the standard page description language (SPDL)[11], font information, and document handling in a distributed environment.

In addition, a number of hypermedia frameworks have been proposed, including HyTime[12] which is an application of SGML for hypermedia and time-based documents.

While no electronic publishing course can hope to cover the whole field, students should have some idea of the existence and relationships of all these standards. Within two years our course will probably include SPDL, a brief coverage of the ODA DAPs, and one hypermedia framework. DSSSL is a standard aimed at an important area, but including this may be a longer-term change as the current draft standard is unusable (there were 300 pages of adverse comments submitted during the recent international ballot).

7 CONCLUSION

Electronic publishing is a rapidly growing area with a multitude of different systems and techniques available. Our course makes no serious attempt to cover all areas that could reasonably come under the electronic publishing umbrella—database publishing, network information services, and (perhaps more seriously) CD-ROM publishing techniques are only given passing mentions.

In spite of the incomplete coverage, however, we believe our students acquire a good understanding of most of the important basic principles and techniques. Their background makes it easy for them to appreciate and exploit the latest advances in active documents, and to see how document standards and document handling fit into the wider computing scene. Their experience of programming and software engineering enables them to accept the need for discipline and good design, and to understand the problems of handling really large documents.

Although we cannot expect them to become typographers or graphic designers, we do believe that even in the short time available during our course we can give them what might be called an aesthetic awareness that will enable them to make reasonable decisions in their future work.

ACKNOWLEDGEMENTS

We would like to thank our colleagues, Peter Brown, Dick Jones, and Eve Wilson (who also teach on our undergraduate electronic publishing courses) for their help and ideas, and our students for keeping us on our toes.

REFERENCES

1. D. E. Knuth and M. F. Plass, ‘Breaking paragraphs into lines’, *Software—Practice and Experience*, **11**(11), 1119–1184, (1981).
2. Adobe Systems Incorporated, *PostScript Language Reference Manual*, Addison-Wesley, 1985.
3. P. J. Brown, ‘A hypertext system for Unix’, *Computing Systems*, **2**(1), 37–53, (1989).
4. ISO 8879, *Information Processing—Text and Office Systems—Standard Generalized Markup Language*, 1986.
5. R. M. Pirsig, *Zen and the Art of Motorcycle Maintenance*, William Morrow, 1974.
6. R. Rubinstein, *Digital Typography—An Introduction to Type and Composition for Computer System Design*, Addison-Wesley, 1988.
7. P. M. English, E. S. Jacobson, R. A. Morris, K. B. Mundy, S. D. Pelletier, T. A. Polluci, and H. D. Scarbro, ‘An extensible, object-oriented system for active documents’, in *EP90—Proceedings of the International Conference on Electronic Publishing, Document Manipulation and Typography*, Cambridge University Press, (1990).

8. ISO 8613, *Information Processing—Text and Office Systems—Office Document Architecture (ODA) and Interchange Format*, 1989.
9. ISO/IEC DISPs 10610–1, 11181–1, and 11182–1, *International Standardized Profile FOD11/FOD26/FOD36—Office Document Format*, 1991.
10. ISO/IEC DIS 10179, *Information Technology—Text and Office Systems—Document Style Semantics and Specification Language (DSSSL)*, 1991.
11. ISO/IEC DIS 10180, *Information Technology—Text Communication—Standard Page Description Language (SPDL)*, 1991.
12. ISO/IEC DIS 10744, *Information Technology—Hypermedia/Time-based Structuring Language (HyTime)*, 1991.