



What has WYSIWYG done to us?

Conrad Taylor

DTP—the best thing since sliced lead?

In the last decade, desktop publishing has revolutionised how printed material is designed and typeset. Comparisons are often made with the original publishing revolution brought about by Gutenberg's invention of movable type.

I have been an enthusiastic participant and propagandist in the DTP revolution. But today, while considering what we have won, I want to reflect on how we have been seduced by WYSIWYG's illusion of control, missed some opportunities, lowered our expectations and become deeply confused about who does what.

Design in the era of galleys and hot wax

Eleven years ago, when I started teaching for Popular Communication, my wife and I shared a job as designer/artworkers for a small magazine, *Inside Asia*. We also undertook other design jobs for voluntary organisations.

In those days, the process of getting words clothed in type and put on artwork ready for print typically followed these steps:

- We studied the text to understand the structure of the ideas contained within it. In cases of ambiguity, this required checking with the originator of the text (a writer or editor).
- Type specifications were decided by looking at **type sample sheets**. These showed only a few sizes and leadings for each typeface; so imagination was required to guess what would look good.
- A **layout drawing** might be made—the blueprint for the printed piece. We might have to do **copyfitting calculations** to work out how to fit text into specific areas, or to advise the editors that the text would not fit, and would need to be edited down.
- **Comprehensive mark-up** was applied to the typescripts. I wrote instructions to the typesetters in red ink in the margins—instructions such as 'Stempel Garamond $^{11}/_{13} \times 16$ picas, slightly tight letterspace, justified'. Other mark-up took the form of those squiggles understood by designers, typesetters and proofreaders as their common language.
- The typescript was biked to the typesetters, where a compositor typed the text into the typesetting terminal, translating our mark-up into **control codes** for the machinery being used (e.g. Compugraphic Editwriter or MCS, VariTyper, Quadritek or Linotron 202—each machine had its own way of encoding formatting instructions).

- A roll of continuous, unpaginated setting was printed out from the phototypesetting machine as a **bromide**. This would be the first time that anyone had seen any representation of the typography.
- This print-out was biked over to the studio, and functioned as our ‘**galley proof**’. We made two photocopies. One was carefully proof-read to check [a] for ‘literals’, errors caused by typing mistakes; [b] problematic linebreaks or poor hyphenation/justification; and [c] places where the typographic specifications had not been followed. As this was the first time we had seen the setting, we might change our mind about some of the specifications and ask for them to be altered.
- Meanwhile, the second photocopy was pasted up onto layout sheets, to figure out pagination and column breaks.
- The corrected proof was biked to the typesetters, who called the job back up on screen, made amendments to the file and ran out another bromide. When this came back to the studio, it was checked again. If we were lucky, we would not need to ask for yet more corrections!
- Finally, artwork was assembled by waxing the back of the bromide, cutting it apart with scalpels and fixing it to the artboard. Sometimes we would draw a keyline box to show where the litho platemakers should insert a photo; sometimes we had had photos screened as halftone bromides; these were also trimmed and glued into place.

Typographic cybernetics

As a designer/typographer working with external typesetting services, I found myself increasingly frustrated about three things:

Visual feedback: I wished for faster feedback about how design decisions would turn out. From this point of view, machine typesetting was a step *backwards* from ‘sliced lead’. When setting foundry type by hand one had an immediate sense of how the page was building, a proof could be pulled quickly on the proofing press, and the design could be tinkered with directly by inserting or removing leading and other spacing material.

Linebreak decisions: Early photosetting machines required the operator to make a decision about where to terminate each line, hyphenating where necessary. From the mid-1970s, machines were equipped with automatic linebreak and hyphenation algorithms, which increased the compositor’s productivity tremendously (especially when dealing with text supplied on computer tape or disk); but the quality of linebreaks dropped. I spent a lot of my time requesting amendments to poor-quality linebreaks.

Inefficiency: One could not fail to be struck by the inefficiency of the process, especially the duplication of effort: a designer marked up the typescript manually, and the compositor marked up the electronic file afterwards with control codes. For structured publications like *Inside Asia* it was possible to lighten the labour of mark-up either with rubber stamps, or by devising *generic* mark-up—that is, labelling text according to its function (e.g. ‘subhead’) and providing the compositor with a stylesheet describing the typography to apply to each category.

The issues here are what I call ‘design cybernetics’—cybernetics being the science of control, communication and feedback. I wanted more feedback, I wanted more control, and I wanted it *now*.

And then, at the Repro 85 exhibition at Olympia in November 1985, I saw my first Macintosh. With Helvetica and Times on the screen. They called it WYSIWYG—*What You See Is What You Get*. Instant feedback, albeit at low resolution. The ability to make changes interactively and see a design evolve in front of your eyes. Wow!

But how did this magic come to be? And what was the price we paid?

Computers and typesetting in the early eighties: a crossroads

It was inevitable that personal computers and typesetting would get mixed up with each other, but at the beginning of the 1980s it was not quite clear how this would happen.

Following the invention of word-processing (in about 1964), computers were increasingly used to process text, though their widespread use by writers and editors did not take off until the 1980s. Even then, there was a gulf between the kind of applications which writers and editors used to prepare texts, and the machines which compositors used to typeset them. The gulf was widened by the fragmentation of the typesetting market amongst dozens of vendors of typesetting machines, each with its own proprietary mark-up language and sometimes even its own disk format.

From the late 1970s, adventurous publishers and computer scientists sought to bridge that gap through a variety of solutions which would let a writer, editor or designer sit at a personal computer and introduce codes to control typeset output. Five such approaches are described below; the WYSIWYG approach which is dominant today was only one of them.

Alternative solutions **Off-line typesetting:** Vendors of typesetting machines were charging high prices for compositors’ workstations, and this created a market opportunity. Companies wrote programs for CP/M or DOS microcomputers, which let these cheap machines prepare files for downloading to a Compugraphic or similar system. Many typesetting firms embraced this system. Ultimately, however, this path was doomed, because each such system was too closely linked to the coding system of a particular make of typesetting machine.

Vendor-independent typesetting codes: A variant on this off-line typesetting approach was favoured by associations of publishers, who were keen not to be dependent on a particular kind of typesetter). They devised intermediate industry-standard sets of typesetting mark-up codes which could be converted to vendor-specific codes at a late stage in the publishing cycle. (E.g. GenCode in the USA, ASPIC in Britain.)

Tau Epsilon Chi: Donald Knuth, a Stanford professor of computing, devised his **T_EX** typesetting language as a standard, flexible and extensible way to mark up a text file with control codes defining every aspect of the typography of a publication. T_EX does not require you to use any particular machine or program to enter these codes, though it ultimately requires a T_EX formatter program to digest the codes, set linebreaks and hyphenations, put columns in place on each page and create printer control files (for instance,

CORA for Linotype, or PostScript). \TeX remains popular in academic publishing, largely because it is very good at mathematical typesetting.

Generic mark-up: All three of the preceding systems I have described insert typographic encoding in a text file to control directly how the resulting print-out will look. Such mark-up is sometimes called ‘**procedural**’ because it describes what the typesetting system must do. An alternative approach was to encode the structure of a text in a highly **generic** way, for instance to identify a certain portion of the text as a *Level 2 heading*, or a *cross-reference*, or the *name of a journal*, or *strong emphasis*.

The Graphic Communication Association in the USA, having put effort into GenCode in the 1960s, came together with a IBM text-processing project team under the auspices of the American National Standards Institute committee on Computer Languages for the Processing of Text. The result of their labours was **SGML**, the **Standard Generalized Mark-up Language**, and various companies built publishing systems to take SGML-encoded files and transform them for output on a particular typesetting system.¹

WYSIWYG: What really took off, however, was the typesetting system we describe as WYSIWYG or desktop publishing. This approach owes its origins to research into workstation technology pioneered by Xerox at its Palo Alto Research Center, later picked up by Interleaf (for its Technical Publishing Software, running on Unix workstations) and more popularly by Apple with its Macintosh computer, which made its debut in January 1984.

Initial sales of the Macintosh computer were slow, but Apple was rescued in 1985 by the launch of Aldus PageMaker, the first WYSIWYG typesetting and page make-up program on a personal computer. PageMaker was followed on the Macintosh by MacPublisher, ReadySetGo, RagTime and QuarkXPress, and on the PC platform by Ventura Publisher. DTP had arrived.

Batch versus interactive

Why has WYSIWYG succeeded so spectacularly, while the other typesetting approaches I have described have languished? I think WYSIWYG’s main appeal is that it appears to offer its users *superior cybernetics*—i.e. *feedback and control*. To the extent that you can trust its authenticity, the screen gives you immediate feedback. And acting on that feedback, the user then has immediate control. (People like having feedback and control.)

This high degree of interactivity contrasts with the **batch processing** methods of all of WYSIWYG’s rivals. For instance, in classic implementations of \TeX , the mark-up is processed for output all in one go: only then does the program figure out line breaks and page breaks in the process of generating the DVI page-description file.² (There are some \TeX editing environments such as Vor \TeX for Sun workstations and \TeX tures for Macintosh where a soft preview window may be put on screen, and this will update periodically; but no editing can be done in this preview window.)

1. We shall return to the topic of SGML on page 10 and again on page 16.
 2. The DVI ‘Device Independent’ file is a representation of the page geometry which will then be translated to a printer control language such as PCL5 or PostScript.

Control, or the illusion of control?

It is worth remarking in this context that while WYSIWYG may have won the hearts and minds of designers through ‘superior cybernetics’, the degree of control which such programs offer may be more illusory than real. Examples:

- In 1987, Erik Spiekermann wrote and typeset a wonderful book called *Rhyme and Reason: a typographic novel*. It was set on a Berthold Diatronic photosetter with glass font matrices and a command-line interface. Not only did Erik not need WYSIWYG to produce this fine piece—his custom editing of the Walbaum typeface’s kerning and touching tables is, even today, beyond the capabilities of DTP.
- One reason T_EX takes so much time to calculate linebreaks is that it does so much more carefully than a DTP program; or, putting it another way, T_EX is not forced to use the quick-and-dirty algorithms for hyphenation and justification (H&J) which DTP programs *must* use to speed up screen redraw during editing. DTP programs make a decision on how to break the current line, without re-examining decisions made for previous lines. T_EX in contrast assigns ‘penalties’ or ‘badness’ for each unfortunate thing that could happen during H&J, for instance stretching or squashing the word- and letter-spaces (the ‘glue’, as it is called in T_EX-speak) or breaking words at various points—and then applies a dynamic testing process to find *for the paragraph as a whole* the H&J decisions which result in the smallest total ‘penalties’ awarded by the system against that paragraph. Thus a change to the last line of a paragraph in T_EX may affect where the first line breaks—an astonishingly sophisticated algorithm which rarely generates an objectionable linebreak.³

The delights and distractions of WYSIWYG

Who gains from WYSIWYG?

There are clearly areas of publishing where the enhanced design cybernetics of WYSIWYG are all to the good. This is particularly the case for short, one-time design-intensive publications where the precise spatial relationship of type and picture elements is critical for aesthetic reasons, such as advertisements, brochures, posters and consumer magazines. In the past these required careful layout planning, and either a great deal of hand-work or several iterations through the typesetting process to get the page looking just right. Nowadays a poster or brochure can be pulled together on the screen in front of you in double-quick time.

A WYSIWYG view is also valuable to designers of various information products such as training and procedures manuals, user guides and business forms. In such products, the arrangements of words on a page is a way of reinforcing their meaning, so information designers take care not to set confusing linebreaks or page-breaks, and may need illustrations to be positioned in a precise relationship both to the accompanying text and to captions. I know that there are many technical authors and information designers who prefer to write directly into the WYSIWYG page make-up environment, as I am doing right now.

3. The ‘penalties’ for each class of H&J behaviour can be edited to tune the way in which the algorithm is executed. This gives the user a high degree of control at the system level while affording less easy control than DTP at the level of the individual line.

Remedial WYSIWYG to fix what's broken

One of the reasons I value WYSIWYG is that it helps me to review the end-of-line decisions that have been made for me by the DTP program I'm using; and, on the rare occasions when I set type in justified columns, the distortions of wordspace and letterspace it has introduced.

If you permit automatic hyphenation, the results from DTP programs are so bad that I know of no discerning typographer who leaves auto-hyphenation switched on. Many designers of my acquaintance prefer to set 'discretionary' hyphens by hand as needed. Solving bad linebreaks may also involve inserting forced line returns and occasionally track-kerning a range of characters or words tighter or looser. We would not be able to do this so efficiently if we did not have a reliable WYSIWYG view.

However, we may be justified in asking why so much remedial work is necessary. At the 1995 Seybold San Francisco conference on electronic publishing, in the final plenary session, I spoke from the floor, asking the audience of more than 2,000 how many felt that the basic hyphenation-and-justification performance of the DTP programs they used had not improved in the last five years of DTP. A sea of hands went up. There is scope for a great deal of improvement in the software to address this aspect of typography.⁴

On the other hand, we cannot expect all linebreaks to be generated automatically; there will always be occasions when we will want to 'tweak' the result. An example is the sideheading above. When typed, that sideheading broke naturally thus: **Remedial WYSIWYG to /fix what's broken.** I amended this by setting the wordspace between 'to' and 'fix' as a hard space which *cannot* break over line endings. I did that because it reinforces the structure of the semantic content better, and also because it makes more balanced line lengths and so looks better. That was a human judgement which I could not expect the computer to make for me.

Retaining the worst of 'lick-and-stick'

In the days when you made up pages at the drawing board using a scalpel and adhesive wax (unless you had a rubber cement habit), you lined up blocks of text and other page elements by eye. If you got it wrong, or if a few lines of extra type were added requiring the other elements on the page to be moved around, you swore quietly, reached for the lighter fluid, and prised the bits of bromide off the artboard to be re-waxed and stuck down elsewhere.

PageMaker brought to the desktop the same ability to work with detached blocks of text. We even got a digital simulation of the non-printing guidelines we once drew in light blue pencil to help line things up. It is a direct and intuitive way of working; it delivers a high degree of apparent control; and it can drive you crazy.

Suppose that you are working on a longish report, with some elements which flank the text column and align with certain parts of the text (e.g. side headings, and icons which signal special text elements such as warnings.) In PageMaker and in QuarkXPress, you cannot lock those elements in place

4. Attendees at Seybold are arguably better informed than the average DTP user, so this response may not be typical. And perhaps it is a by-product of the democratisation of typesetting that most users of any mass-market DTP program will be satisfied with mediocre performance, leaving the vendors and developers of these programs with little incentive to directing engineering resources to improving typography.

beside the text so that they will move down as new text is added above. Here is a clear case where the WYSIWYG paradigm certainly gives you control, but also makes you responsible for the low-grade task of making sure all the bits line up which need to line up. Surely it would be better to surrender that low-level control in favour of a greater degree of automation?

Putting it another way: wouldn't it be better to move control to a higher level of generality—such as setting rules for how sideheadings behave—and forgo the need to exert control over minutiae (unless you really want to make such local alterations)?

DTP and levels of control: a range of approaches

Desktop publishing programs in common use today can be divided broadly into two groups. Programs such as **PageMaker** and **QuarkXPress** take a relatively unstructured approach to the control of page make-up. Here the model of control is very hands-on: the user interacts with a WYSIWYG representation of the page, frequently intervening at a direct level of control over individual publication elements to define typography, add rules, set alignment and control pagination.

PageMaker and QuarkXPress are the clear leaders in the desktop publishing market.⁵ Obviously these programs have 'done something right' in the eyes of customers; I believe their appeal is due to the satisfying experience of interactivity with the page-image, and minute control over it. Indeed, the marketing competition between these programs has been fought on such ground as whether you can define type size in 1/1000th of a point and define a profile for how text wraps around an irregularly-shaped graphic.

A little less Manual, a little more Style

Nevertheless, both these programs took an important step towards efficiency by adopting a '**Stylesheet**' system for applying regularly used typographic formats. The user defines a list of Styles, named to match with their function in the publication (e.g. 'Title' or 'Body Text'), and associates with each Style [a] **typographic attributes** such as typeface, leading, alignment in the column, embedded tab markers and so on; and [b] **behavioural attributes** such as pagination behaviours,⁶ application of hyphenation rules, and whether such paragraphs would be referenced in a Table of Contents.

The advantages of working with Stylesheets in desktop publishing are fourfold. Two of these apply to every user:

- Typographic formatting is faster in almost all cases (except for a few adverts in which typography varies a lot); and
- Consistency of appearance and pagination behaviour is guaranteed between all paragraphs of the same generic type.

5. QuarkXPress tends to dominate in advertising agencies and periodical/newspaper publishing, largely because it solved the problem of handling colour separations before PageMaker did. PageMaker tends to dominate in the corporate publishing market, and also on the Windows platform, where Quark is still not well established.

6. Pagination behaviors would include whether the lines in the paragraph are allowed to break over page boundaries, and if so with what controls over 'widow and orphan' lines; whether the paragraph should stay with the one that precedes or follows it; and whether it should start a new column or page.

In addition, using Stylesheets conveys two additional benefits at an organisational level:

- When a new publication is in the process of being designed, its typography can be decided much more efficiently by setting up Styles, applying them to dummy text, then modifying the Styles to fine-tune them to each other. (When a Style is modified, all instances of it are updated with the new attributes.) Thus a designer can test thoughts such as ‘What if we made all subheadings 2 pts larger and reduced letterspace by 2%, then put an extra 4 pts space above each one?’
- The designer can ‘freeze’ a stylesheet into a template document, then give it to a less typographically skilled operators to apply to texts. By these means one designer can define the look of the published output of a whole department, or even a whole corporation.

Style: some folks just ain’t got it

DTP programs haven’t always supported this ‘generic’ method of typographic mark-up. It is a shock to remember that PageMaker 1.1 required each and every paragraph—such as every instance of a subheading—to have its type and paragraph specifications applied individually on a one-by-one basis.

PageMaker did not introduce Styles until version 2.0, and even then few people took advantage of them. One reason perhaps was that PageMaker 2.0’s Styles Palette was not displayed on the screen on start-up, so few users discovered that this new feature had been introduced. In a later version, Aldus re-set the program’s defaults so that the Styles Palette *was* displayed on the screen on start-up, causing many to investigate Styles for the first time.

Even so, many users of PageMaker and QuarkXPress remain obstinately immune to the blandishments of Styles. There is a certain class of DTP software user for whom interacting with cascades of linked dialogue boxes to set up a new Style or modify an existing one is as pleasurable as filling out a tax return; they just won’t do it!⁷

Publishing programs for propeller-heads?

Nevertheless, there is another class of WYSIWYG desktop publishing software with a stronger orientation towards the use of generic mark-up techniques and rules-driven pagination. **Ventura Publisher** was one of the first DTP programs—indeed, the very first for IBM compatibles. **FrameMaker** is a similar program originally available only on Unix workstations, but latterly on Macintosh and Windows computers too.⁸

These programs are harder to learn and less spontaneous to use. Their interface and structures encourage the use of continuous text flows and style-sheets. They automate certain pagination behaviours ignored by PageMaker and QuarkXPress, such as sideheadings, column balancing, and the anchoring of graphics frames. They provide facilities for automatic numbering of paragraphs or figures, and create tables properly with cells,

7. I am not infrequently asked to examine or ‘fix’ DTP files. It is common to find that Styles have not been used, and that each paragraph been formatted on a one-by-one basis. It is also common to find many separate text blocks slapped down on the page, rendering editing and repagination difficult. These users typically put their faith in ‘auto-leading’ and auto-hyphenation, and their paragraphs are separated by two carriage-returns...

8. Ventura Publisher was created by Ventura Software; for a while it was owned by Xerox and now by Corel Corporation. Frame Technology has now been acquired by Adobe Systems.

cell rules and shading, straddles and the like. They make it easy to gather chapter files into books with consecutive page numbers, support dynamic cross-references within and between chapters, and automate the compilation of correctly-referenced tables of contents, tables of figures and indexes.

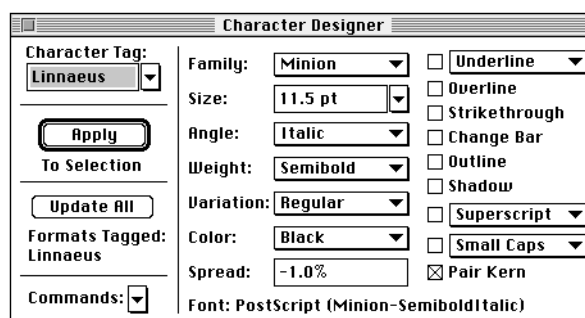
Clearly these are not the kind of program with which one would create a quick one-time advertisement, newspaper, brochure or magazine. But they are highly suitable for the production of technical manuals, directories, product catalogues, books, reports and the like—long documents with lots of structure and cross-referencing, which often grow in a series of revision cycles. Many such documents are also revised and re-issued in several successive versions: so-called ‘maintained documents’.

It helps to get the best out of a program like Ventura or FrameMaker if you are prepared to analyse the structure of your publication carefully, matching each element type with a stylesheet item, variable or other generic construct. It takes longer to set up a new document type this way than it would with PageMaker or QuarkXPress, but you will save time in the long run by not having to interact with document elements at a low level.

This approach is particularly encouraged by FrameMaker, which has styles (‘catalogues’) not only at the paragraph level, but also at the level of words or characters. For example, I may use italics *like this* for emphasis, or to mark *Moby Dick* as the name of a publication or *Cepphus grylle* at the Latin name for the Black Guillemot. In most programs, you would just select the words in question and apply the command ‘Italic’. Working in FrameMaker, I set up separate character styles for these different kinds of text entity. One advantage is that if a character style varies in several ways from the main text **like this** (from Minion 11.5 pt Semibold, to Frutiger 10 pt Black), I speed formatting and guarantee consistency. Another advantage is that should I decide that all species names should display in 12 pt Minion Expert small capitals with 3% letterspace, I can apply that change globally in a matter of seconds.



This document’s Character Catalogue, as described on this page. The styles ‘Italics’, ‘Linnaeus’ and ‘Publication’ all happen to look the same in this instance of the publication.



FrameMaker’s Character Designer, in which formats are defined which can be applied to text elements below the paragraph level.

After such elements have been ‘tagged’ with a named character format, all such instances in a document can be issued with revised formats simultaneously.

So do you have to be one of those proverbial ‘rocket scientists’ to use a publishing program like Ventura or FrameMaker?⁹ Not quite; but it demands a type of intellect which sees beyond the appearance of a publication to understand its deep structure and the ways in which it will be edited, revised and perhaps re-issued in new formats. And that’s not a view of publications which most graphic designers—or secretaries graduating to DTP—bring with them from their training. Indeed, the perceptions which are important in such endeavours are more like those of the writer or editor. Of which more later...

9. Actually, some Ventura and FrameMaker users really *are* rocket scientists.

Why generic mark-up is likely to make a comeback

Stylesheets vs. *fully* generic mark-up

At this point you might review my description of the five diverse approaches to computerised publishing at the beginning of the 1980s (see page 3). You may wonder whether there is any parallel between my practice of formatting text in FrameMaker by applying the generic mark-up stored in Paragraph and Character Catalogues, and the generic mark-up approach taken by SGML, the Standard Generalized Mark-up Language.

The answer is a mix of Yes and No. When in FrameMaker I set up a character catalogue style called *Linnaeus*, I do so principally as a convenient way of applying consistent typographic style to the names of bird species. Each catalogue entry always defines some formatting. In fact I may find myself in the annoying situation of having to set up three distinct character styles for Latin species names: one for where they appear in text, a second for where they appear in a heading, and a third for where they appear in small type in the index. After all, the typography will be different in each location.

In contrast, when adding SGML encoding, the process of inserting tags thus: `<LIN>Sula bassana</LIN>` is purely to define the text string ‘Sula bassana’ (it’s a gannet, in case you wondered) as the content of an SGML entity of type ‘Linnaeus’. Absolutely no attempt is made *at the stage of adding mark-up* to define whether such entities are displayed in italics—or in big green capital letters with purple spots, flashing slowly on and off on a computer screen.

Furthermore, the tagging remains exactly the same regardless of the context in which the entity makes an appearance, though I would certainly want the rule-driven process of formatting `<LIN>Sula bassana</LIN>` for output to take account of the context—and generate different appearances for each case.¹⁰

Multiformat publishing is now the motivation

When the Graphic Communications Association got involved in the committee work that led to SGML, their principal motivation was to avoid the publisher’s nightmare of having the text of a book trapped in the encoding scheme for an ‘Acme VariTron’ typesetting system, when Acme Inc. had gone out of business five years ago. But these days, the phenomenon which is causing renewed interest in generic mark-up is the prospect of **multiformat publishing**: taking the text of a book originally written for paper publishing, and ‘repurposing’ it as a CD-ROM, as on-line help, or as a collection of pages on the World Wide Web.

As Liora Alschuler puts it in her recent book *ABCD...SGML*:

When text is tagged according to its structure and meaning, it has many lives, including, but not limited to, a beautiful life in print. “Appearance is one possible use,” is how typesetters and SGML experts Adams & Hamilton put it. Other uses include online publishing, hypertext, sophisticated search and retrieval, and platform and vendor independent transmission and storage.¹¹

10. This context-sensitive approach to the formatting of elements is also taken by the structured-editing version of FrameMaker—**FrameMaker+SGML**.

11. *ABCD...SGML* by Liora Alschuler; International Thomson Computer Press, 1995. ISBN 1-850-32197-3. Page 38.

The Web as a demo SGML application

There is now a phenomenal amount of interest in the Internet, and the principal reason is because of the World Wide Web. What is ironic is that **HTML**, the **Hypertext Mark-up Language** which makes the Web possible, is essentially one *application* of the Standard Generalized Mark-up Language, SGML. As such it is a powerful mass-market demonstration of the benefits of generic mark-up for publishing in the electronic age.

What is meant by ‘an **SGML application**?’ In essence it means *an application of SGML’s principles of document mark-up to a class of documents*, following a set of rules which define all of the data types (the mark-up) permitted in that class of documents. These rules are contained in a statement called a **Document Type Definition**, or DTD.

HTML, the mark-up system which at a minimum defines the information structure of Web pages, links them to inline graphics and allows them to contain hypertext links, is a limited SGML application¹²—one with such a small set of permitted tags that they hardly need to be defined in a DTD, because every Web browser program by definition already recognises them.

One reason why Web pages have to be created as text files with generic mark-up is that it’s impossible *not* to be a multiformat publisher when you publish on the Web. You have no control over whether your publication is being viewed in colour or black-and-white, you do not know what fonts your page is being viewed in, and since you do not know how wide the browser window has been set, you cannot predict where linebreaks occur.

View on demand, print on demand

I believe that paper publishing in the form we know it today, where the printing press churns out thousands or millions of copies of identical printed items for mass consumption, will be around for quite some time. But I also believe more publishers will be transmitting electronic files to a wide variety of local viewing and printing environments, and readers will go trawling the networks for electronic documents which they view or print locally. Nicolas Negroponte, Professor of Media Technology at MIT, considers that newspapers may evolve into this form:

Imagine an electronic newspaper delivered to your home as bits... The interface solution is likely to call upon mankind’s years of experience with headlining and layout, typographic landmarks, images, and a host of techniques to assist browsing. Done well, this is likely to be a magnificent news medium. Done badly, it will be hell.

...[B]eing digital will change the economic model of news selections, make your interests play a bigger role, and, in fact, use pieces from the cutting-room floor that did not make the cut on popular demand... Imagine a future in which your interface agent can read every newswire and newspaper and catch every TV and radio broadcast on the planet, and then construct a personalised summary. This kind of paper is printed in an edition of one.¹³

12. Here I’m being simplistic for the sake of brevity. HTML has gone through several stages of evolution; earlier browsers can’t understand later mark-up types such as forms, tables and equations. HTML has always included some non-generic mark-up such as ‘Bold’. And one company, Netscape, is ‘polluting’ the generic nature of HTML by popularising a large set of non-standard tags which allow a page designer to control formatting directly—provided the reader views them with Netscape’s own browser, Navigator.

13. *Being Digital*, by Nicholas Negroponte. Knopf, New York, 1995. ISBN 0-679-43919-6. Pages 152-153

If I may be allowed to tame Professor Negroponte's project, excluding exotic artificial intelligences scanning the airwaves on our behalf, his vision of *The Daily Me* nevertheless becomes quite easy to envisage. I like the idea of being able to subscribe to a newspaper which gives me the scientific and political news, and suppresses all references to football and Pamela Anderson. But I can only see it working if the text sources from which the computer makes its selection on my behalf¹⁴ have been marked up in a manner which makes their content easily computable. SGML enables this.

There are also certain reference publications which do not make sense as printed volumes. Encyclopædias and dictionaries are out of date the day you buy them, if not before. It would be more logical for these publications to be on line where they can be kept up to date, perhaps charged to readers on a pay-per-view basis.

These are just the consumer applications. Already in the fast-growing world of specialist scientific, technical, medical, financial and legal publishing of scholarly journals and reference works, the leading publishers are facing up to the task of publishing online and on CD-ROM as well as on paper.

**Typesetting on the fly:
batch formatting again**

Negroponte is right to point out that good quality layout and typography will be essential to make print-on-demand and view-on-demand publications acceptable. The trouble is that they may never have been typeset before you request them to appear before your eyes, for two reasons:

- Some documents won't exist until you ask for them, such as custom collations from a *What's On* listing or a Classified Ads database.¹⁵
- The variety of means of delivery argue in favour of the 'late binding' of typography to semantic structure, to suit the characteristics of the particular delivery medium. The document may be viewed on a pocket computer with a monochrome LCD screen, a desktop computer with a colour screen, or after it has been printed onto A4 or US Letter paper from a high-resolution laser printer. Each instance should look and behave differently (you can't read 9 pt Jenson on screen, and you can't scroll or hyperlink to notes on a paper page).

If the results are to be 'magnificent' rather than 'hell', we will need to rely on intelligent typesetting and pagination on the fly—the return of batch formatting, along the lines of how T_EX does it (see pages 3–5). Without scope for WYSIWYG clean-up, we'll need browser and print-on-demand technologies to incorporate far more impressive hyphenation, justification and pagination algorithms than those with which we have been limping along in desktop publishing.

In short, I believe that the demands of multiformat publishing are nibbling at the edges of WYSIWYG's ten-year hegemony over the publishing industry, causing a re-examination of the benefits both of generic mark-up systems like SGML, and batch typesetting and pagination systems like T_EX.

14. When you subscribe to such a newspaper, you would fill out an extensive form to describe your interests—ideally, an on-line one which you can amend as your interests change.

15. May I propose the term 'smörgåsbord publishing'?

The implications for PEOPLE: divisions of labour, skill sets and training

Finally, it is time to ask what WYSIWYG has done to *us*, to the humans involved in the publishing process. What has desktop publishing done to the former divisions of labour, and the skills which were bound up with them? Have publishing operations—be they corporate or commercial—adjusted well to the new ways of working that WYSIWYG tools have brought?

Four skill sets for modern publishing

Any publishing project has **content**; content is conveyed by **language**; language is presented in typographic **form**; and the publication in that form is brought to a wider audience by some method of **production**. Thus four kinds of skill necessary to any publishing endeavour these days are:

- **subject expertise**, such as knowledge of marine law... molecular biology... computer networks... the company's product range, &c;
- **editing/writing ability**, which is founded on a good command of one's language but may also extend into specialisms such as methods of indexing or compiling bibliographies;
- **visual sense**, necessary for graphic design and typography; and
- **computer and keyboard skills**, including the ability to work with particular computer programs and get stuff printed out.

In addition, where publications are to be litho printed, someone in the operation needs to know how to work with outside suppliers; and some projects will require the contributions of photographers or illustrators.

Does WYSIWYG mean one person does it all?

Several early advertisements for desktop publishing implied that anyone could now sit down at a computer and take over the roles formerly played by a graphic designer, a typesetter, a paste-up artist and perhaps an illustrator too. As I see it, the implication (by omission) was that it was *the writer*, the originator of ideas, who would be displacing all these other people.

One Xerox advert portrayed an office inhabited by several exact clones of Leonardo da Vinci, each seated at a Xerox Documenter workstation. One Leonardo was setting type, another drawing a diagram... and so on. This advertised DTP as a tool for every creative endeavour; did it also imply that the perfect operator is necessarily a person of several large talents?

Some people *are* equipped to do it all; many early adopters of the technology were. For me, the DTP workstation was a liberation. I'm a writer and editor, typographer and illustrator, artworker and photographer. I have always been used to taking on projects in which my contribution includes writing, editing and design. My ability to make a living and help clients was greatly enhanced by the invention of this all-round communicator's Swiss Army knife.

Information Design is a discipline rich in people like me in this respect, because its a perspective which emphasises the integration of typography with language, and attracts people who work happily in either sphere (or hemisphere, perhaps).

WYSIWYG publishing software surely fulfils its promise best for those who contribute a broad swathe of verbo-visual skills. If you are both designer *and* compositor, and what you see on screen doesn't agree with your design sense, you can make some remedial change. If you have editorial powers too, and a bad linebreak or page break has no acceptable typographic fix, you go in and edit the text until it fits. And believe me, I'm doing this right now as I write.

But not everyone, and not every sort of publishing activity, can work in the same way. The designer, the writer, the editor and the person who makes up the pages on screen may all be different people, making the feedback loop necessarily more ponderous.

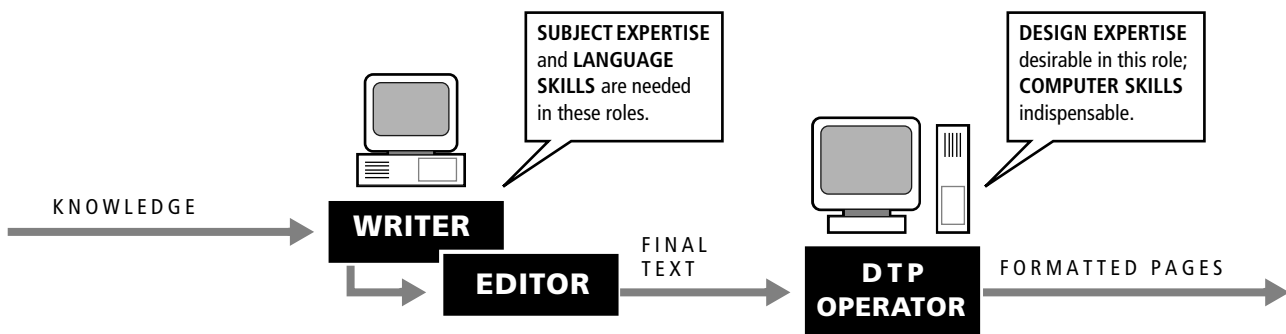
And it also raises the thorny question of who does what.

Jobs, skills and responsibilities

In practice these days few **writers and editors** work with a pencil or a much-loved Remington typewriter. Most have some degree of computer skill, however basic. But their valued strengths are their knowledge of a subject, and the ability to express it in good English.

Though some writers and editors are competent desktop publishers, it is more usual for their texts (in some word-processing format) to be handed to someone else—perhaps called a '**desktop publishing operator**'—who imports them into a WYSIWYG page make-up program and formats them for publication. That operator's skills may not be strong in the areas of subject expertise and linguistic ability; but he or she is likely to be a competent computer user, and hopefully has some graphic design ability.¹⁶

This diagram shows how skills are usually thought of as being distributed between the jobs:



Designer interventions

But the subject is not so simple; the complication arises when we consider exactly what the **designer** is in the new model of publishing. Let's look more carefully at diverse answers to these three questions:

- Who applies the **MARK-UP**?
- Who does the **DESIGN**?
- Who takes care of **TYPOGRAPHIC QUALITY CONTROL** by checking proofs and fixing linebreaks, kerning and pagination problems?

16. Managers who believe that sitting someone down in front of a PageMaker screen turns them into a designer are, of course, wrong. But sending those people for design training, such as Popular Communication's courses, may help.

Who does the mark-up? Before desktop publishing, mark-up was written on the typescript by a designer, or possibly an editor, and implemented in code by a compositor. These days, there is no room in publishing for a compositor whose sole function is accurate entry of text and the application of typesetting codes on instruction from somebody else.¹⁷

Today the DTP operator applies the mark-up—but also makes up pages, including the placement of tints, illustrations and photographs. This approach is fine [a] for small projects with a simple typographic structure and [b] where the DTP operator can communicate clearly with writers and editors to resolve ambiguities about what formats get applied to what text.

However, there are publishing environments in which desktop publishing skills alone—or even DTP skills plus design sense—are not enough to qualify someone to do mark-up. Where texts have a complex structure, and the complexity of that structure is mirrored by complex typography, the person who does the mark-up *must* have a deeper understanding of the editorial process, and probably the subject area.

This is why it is increasingly common for technical authors to be found sitting at DTP workstations. It is too great a communication burden for an author to have to explain to a DTP operator where to apply which of three levels of heading, which text elements get what special formatting, and where markers must be set for index entries, cross-references and footnotes. It's quicker to do it yourself; so the author becomes a DTP operator too.

Who does the design? The assumption in the era of desktop publishing is that the DTP operator *is* the designer. There are many cases where this is true, and just as there is no longer a role for a compositor who can't make up pages, there is scarcely any role for a designer who can't drive a mouse. Graphic design these days is implemented with computers, and that's that. But...

- Many graduates of graphic design courses, even those who studied in the last few years, have really only a superficial knowledge of how desktop publishing works and probably have no formal training in it. They are therefore not likely to make very good DTP operators.
- Anyway, according to an estimate by Professor Cal Swann, trained designers handle perhaps only 2% of all pages made up using DTP.¹⁸
- Maybe that is just as well. Many designers, however strong they may be in visual skills and a 'broad stroke' approach to design, don't pay a great deal of attention to detail, can't type fast enough and can't spell.
- In any case, as indicated above, there are plenty of situations in which the people who should apply the mark-up—and therefore do the DTP—are the subject experts and writers. The problem is that these people probably do not have design training.

So how do we bring design skills, mark-up skills and DTP-user skills together?

17. In 1982–83, a lot of the typesetting I required was handled by a very meticulous QuadriTek operator who failed to adapt to the demands of desktop publishing. Her next job, I regret to say, was at a supermarket check-out.

18. Comment made by Cal Swann at a conference on the teaching of typography held at Manchester Metropolitan University on 17 November 1995.

The designer as programmer

There is one solution to this impasse, which is to employ the skills of a designer to define the look of a document, or a whole class of documents, and encapsulate that person's decisions within the computer program which will be used thereafter by other people to format the documents.

This is not to say that graphic designers must learn how to write brand new DTP programs in C++. All of the existing programs let you set up **template documents** in which page margins, master pages, paragraph styles—perhaps also specialist character and table styles—have been pre-defined.

It isn't even necessary for the designer to know all the ins and outs of the DTP program. I once successfully set up template documents for British Rail by prototyping them in PageMaker and then, after BR committee approval, working with an Interleaf employee to convert the design to an Interleaf TPS template document on a Sun SparcStation. However, the ideal situation must surely be one in which the template designer knows the publishing software well enough to automate as many aspects of the design as possible.

The more structured the publishing environment, the greater the power it puts in the hands of a template designer—which is ironic, because designers who have come of age in a WYSIWYG environment find the structured-publishing mind-set hard to adjust to.

However, other designers experience this way of working as a liberation. The experience of Toronto book typesetters Adams & Hamilton, who use SGML as input, gives pause for thought. We have already seen how Adams and Hamilton regard print as only one possible use for SGML-encoded text (see page 10). To typeset from an SGML source file, they take it through a rule-based conversion process resulting either in a **troff** file,¹⁹ or a text file with embedded mark-up that will flow into a QuarkXPress document, picking up correct paragraph styles and other typography on the way.

The SGML file remains the master file. Once printed, the file with the output markup becomes a throwaway file. Other designers "think the Quark file is the product; it is, in fact, designer-added value." ... Hamilton and Adams clearly get the bigger picture—information is more than its appearance—but have sacrificed nothing as craftspeople along the way.

According to Kate Hamilton, SGML makes typesetting into the process that it was always supposed to be—the pure application of design. Without SGML, Hamilton and Adams claim that typesetters and designers allocate more time to clearing up exactly what that squiggle means and whether they have found every instance of a nested bullet than they allocate to the business taught in school as typesetting and design.²⁰

If you think Kate Hamilton makes it sound simpler than it is, you could be right—she wrote the SGML-to-Quark conversion routines herself using **mawk**, a Unix utility; not what you expect from your average graphic designer. But elsewhere this combination of programming and graphic design would be accomplished by teamwork; and more off-the-peg software is now available for such conversions (e.g. SoftQuad's SGML Enabler for Quark).

19. **Troff** (stands for 'typesetter run-off') is a batch-formatted procedural mark-up encoding language. Adams and Hamilton print about 90% of their books with troff, and use Quark only for books where formatting is more complex and will require WYSIWYG tweaking.

20. Liora Alschuler [Ibid]. Page 153.

And who takes care of the details?

When I worked at *Inside Asia* magazine, and galleys came back for proof-reading, they were read and corrected by three people—the two editors, who concentrated on making sure that the spellings and punctuation had been entered correctly, and myself. My role as designer was to check that the mark-up had been correctly applied, and I also focussed on the quality of linebreaks, hyphenation and justification.

Who checks these details in today’s typesetting environment? Judging by the low quality of a lot of typesetting, from the most prestigious magazines to the documents prepared in corporate offices, the answer appears to be—nobody.

Under the former divisions of labour, it was the trained compositor who paid attention to such details, undistracted by either the larger picture of the page design or knowledge of the subject. Nowadays you have typesetting done either by subject experts who are unaware of the finer points of typography—or by designers for whom text is just that boring necessity, the grey patches on the page; they prefer to put their energies into page composition, choice of colours, and showy handling of pictures and tints.

Specialists and polymaths

I have no easy answers about how we are going to put all the pieces back together again; but it seems to me that we need certain kinds of people with crossover skills, and certain kinds of dialogue between specialists:

- We need more graphic designers who really understand how to drive publishing software—either to apply their combined skills to the design of complex, design-intensive publications, or to put together template documents for others to use.
- From the other direction, we need DTP operators to increase their knowledge of typography and design—at the very least to take care of the details where publication quality is so often compromised, but also with the ambition of becoming good document designers.
- We need forums for discussion—and also for training—in which graphic design is approached from a rational perspective as an aspect of rhetoric and a tool for the advancement of communication, rather than as an art-form. Most design courses and design associations currently tend to the art-form ‘cultural’ view of what design is about.
- We need designers and editors to work towards a common understanding of the relationship between the structure of meaningful elements in a text, how that structure is represented in typographical form, and how it is encoded inside a computer system—bearing in mind that more and more texts will find themselves repurposed for completely new media and that it is sensible to minimise the amount of conversion work this will require.
- And, faced with an exciting (terrifying?) future of documents slipping off the certainties of the printed page into electronic environments—where they will be formatted on demand on a wide variety of output devices—we need our design-people, editor-people and computer-people to work together to ensure that the result will be pleasant to look at and easy to understand.

Time to take the toolmakers to task?

Ten years ago there were perhaps a dozen vendors of typesetting systems. Today there are at least as many vendors of imagesetting *equipment*, but on the software front the typesetting market is now dominated by the products of just three companies:

- **Quark Inc.**—*QuarkXPress* (for Macintosh and Windows)
- **Corel Corporation**—*Ventura Publisher* (for Windows)
- **Adobe Systems Inc.**—*PageMaker* (for Macintosh and Windows) and *FrameMaker* (for Mac, Windows, Sun, HP-UX and a number of other Unix platforms).

In recent months I have been asking designers, and information designers in particular, what they think about the quality and capabilities of these tools. There is a broad feeling amongst the more thoughtful desktop publishing practitioners that we are not being well served.

Particularly common are grumbles about the low quality of hyphenation and justification algorithms. Poor H-&-J is a hallmark of the desktop publishing era; designers with high typographic standards are spending absurd amounts of time inserting hard spaces, forced line returns and discretionary hyphens. It turns out that the ‘superior design cybernetics’ of the WYSIWYG model of typesetting has not worked in our favour, because it has led to us having to clean up after the mess left by programmers.

Somehow, people who care about good graphic design have to find a voice, and a way of communicating with the people who make tools for us. But we will have to be realistic: the market is not the same as it was just five years ago. Now that the whole world has converted to desktop typesetting and there are no new worlds to conquer, these companies’ return on investment as a result of product enhancement is bound to be constrained. (That is, upgrade fees alone may not pay for rewriting the software.) If we want quality, this time we may really have to pay for it.

So is What You See *all* you’re going to get?

My final thought is—that evidently more thought is required. Everyone in the publishing process needs to be far more analytical about the way publishing is done, particularly in relation to what will happen to publishing in the future. Increasingly the slogan *What You See Is What You Get* is being challenged by new forms of information storage, transmission and delivery.

What You See... Well, that will soon be only the visible tip of an information iceberg. Under the surface, information will need to take on a lot more *structure*, because it is at the structural level of hyperlinks and computable language that value will increasingly be added to information. What you get will be more than you see. The publishers and designers who survive in that world will be those who learn to look beyond appearances.

It’s quite a challenge. ■

Thanks to these people who shared their ideas with me while I was writing this paper:

Susan Cato
 Malcolm Clark
 Polly Pattison
 James Souttar
 Jane Teather
 Jan V. White