SEYBOLD PUBLICATIONS Visit Seybold

The Seybold Report on Publishing Systems
Volume 26, Number 2 -- September 30, 1996

What has WYSIWYG done to us?

by Conrad Taylor

This is the text-only version, which allows you to download the whole article in one go, complete with endnotes.
If you'd prefer to see some pictures -- not in the Seybold-published version -- try the **alternative pages**.

FOR ELEVEN YEARS I have been an enthusiastic participant and propagandist in the DTP revolution. But now I want to reflect on how we were seduced by WYSIWYG's illusion of control and how we lowered our expectations and typographic standards and became deeply confused about who in publishing is supposed to do what.

Good typography requires a lot more than good-quality typefaces. It also requires improved composition algorithms within publishing software -- both for paper and for the Web.

As we were: galleys and hot wax

Twelve years ago, my wife Sang-usa and I shared a job as designer/artworker for a small magazine, Inside Asia, and also undertook other design jobs for voluntary organizations. In those days, the process of getting words clothed in type and ready for print followed these steps:

- We studied the text to understand the structure of ideas contained within it. In cases of ambiguity, this required checking with the writer or editor.
- Type specifications were decided by looking at type sample sheets. These showed only a few samples for each typeface; imagination was required to guess what size would look good and how much leading would be appropriate. A layout drawing might be made, and I might do copyfitting calculations to work out how to fit text into specific areas or to advise the editors that text would need to be edited down.
- I applied markup to the typescripts, writing in red ink in the margins?instructions such as "Stempel Garamond 11/13+16 picas, slightly tight letterspace, justified." Other markup took the form of the squiggles then understood by designers, typesetters and proofreaders as their common language (but no more, alas!).
- The typescript was biked to the typesetters, where a compositor typed the text into the typesetting terminal,

translating our markup into the control codes for the machine used (e.g., a Compugraphic EditWriter, an Itek Quadritek or a Mergenthaler Linotron 202 -- each machine had its own language for encoding formatting instructions).

- A roll of continuous, unpaginated setting was printed from the phototypesetting machine as a bromide -- the first time anyone had seen any representation of the typography. This printout was biked back to the studio and functioned as our "galley proof." It was carefully read to check (a) for "literals," errors caused by typing mistakes; (b) for places where the typographic specifications had not been followed; and (c) for problematic line breaks or poor hyphenation and justification.

- Meanwhile, for larger jobs a photocopy of the galley might be 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 on-screen, made amendments 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 platemakers should insert a photo; sometimes we had had photos screened as halftone bromides; these were also trimmed and glued into place.

Typographic cybernetics

In the mid-1980s, as a designer and typographer working with external typesetting services in this way, I found myself increasingly frustrated about three things:

- **Delayed visual feedback.** I wished for faster feedback about how my design decisions would turn out. From this point of view, machine typesetting was a step backward from "sliced lead." At least 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 "on the stone" by inserting or removing leading and other spacing material.

- **Poor-quality line breaks.** Early photosetting machines required the compositor to decide where to terminate each line, hyphenating manually where necessary. From the mid-1970s, machines were equipped with automatic line-break and hyphenation algorithms, increasing productivity tremendously (especially when dealing with text supplied on computer tape or disk); however, the quality of line breaks dropped. I was therefore wasting a lot of my time requesting amendments to poor-quality line breaks.

- **Inefficiency.** One could not fail to be struck by the inefficient duplication of effort: A designer marked up the typescript, then the compositor marked up the electronic file with control codes. For regular publications like Inside Asia, one could lighten the labor of design markup with rubber stamps, or devise a generic markup scheme -- that is, labeling text by its function (e.g. , "subhead") and providing the compositor with a style sheet describing the typography to apply to each textual entity. But one-off pieces required more comprehensive instructions to be written out laboriously by hand.

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 quite urgently.

Then, at an exhibition in November 1985, I saw my first Macintosh -- with Helvetica and Times recognizable on the screen. And instant feedback, albeit at low resolution and before Adobe Type Manager had come along to smooth the jaggies. This machine would give me the ability to make changes interactively and to see a design evolve in front of my eyes. Wow!

But in embracing this new tool for design, did we turn our backs on something important?

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 clear precisely how this would happen. Following the invention of word processing circa 1964, computers were increasingly used to edit texts, though their widespread use in publishing did not take off until the 1980s. Even then, there was a gulf between the software that writers and editors used to prepare texts and the machines that compositors used to typeset them. The gulf was widened by the fragmentation of the market among dozens of vendors of typesetting machines, each with its own proprietary markup 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 that would let a writer, editor or designer sit at a personal computer and control the typeset output. Five such approaches are described below; the WYSIWYG approach dominant today was only one of them.

Offline typesetting. Because vendors of typesetting machines were charging high prices for compositors' workstations, there was a market opportunity for companies that wrote programs for standard CP/M or DOS microcomputers, turning them into workstations for preparing typesetting files for Compugraphic or similar systems. But, ultimately, this path was doomed because each such system was too closely linked to the coding language of a particular make of typesetting machine.

Vendor-independent typesetting codes. A variant of this offline typesetting approach was promoted by associations of publishers keen not to be dependent on particular kinds of typesetting technology. They devised intermediate industry-standard sets of typesetting markup codes, which could be converted to vendor-specific codes at a late stage in the publishing cycle.

Tau Epsilon Chi. Donald Knuth, a Stanford professor of computing, devised his TeX typesetting language as a standard, flexible and extensible way to mark up a text file with control codes to define every aspect of the typography of a publication. TeX does not require you to use any particular machine or program to enter these codes within a text file, though it ultimately requires a TeX formatter program to digest the codes, set line breaks and hyphenations, paginate the document 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 markup. All three systems described above insert encoding in a text file to control directly how the resulting print-out will look. Such markup is called "procedural" because it describes what the typesetting system must do. An alternative approach is 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 Communications Association in the U.S., having worked on GenCode in the 1960s, joined its efforts with an 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 labors was SGML, the Standard Generalized Markup Language, and various companies built publishing systems to take SGML-encoded files and transform them for output on a particular typesetting system.

WYSIWYG. But as we know, what really took off 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 in Apple's Macintosh computer. Initial sales of the Macintosh were slow, but were rescued in 1985 by the launch of Aldus PageMaker, the first WYSIWYG typesetting and page makeup program on a personal computer. PageMaker was followed on the Macintosh by MacPublisher, ReadySetGo, Ragtime and Quark Xpress, and on the PC platform by Ventura Publisher.

WYSIWYG: Control or the Illusion of Control?

Why has WYSIWYG succeeded so spectacularly, while other typesetting approaches 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 immediate feedback. Acting on that feedback, the user then has immediate control. And people like having feedback and control.

Interactivity vs. batch processing. This high degree of interactivity contrasts with the batch processing methods of WYSIWYG's rivals. For instance, in classic implementations of TeX, the markup 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. (There are some TeX editing environments, such as Vortex for Sun workstations and Textures for the Macintosh, where a soft preview window may be put on-screen. It will update periodically, but no editing can be done in it.)

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 that such programs offer may be more illusory than real. Or perhaps it is more accurate to say that desktop publishing programs let you fiddle interactively with the details of your typography until the cows come home, but they do not let you control the default behaviors of the composition algorithms in a way that efficiently and automatically delivers the kind of quality typography that was formerly expected of trade compositors.

Lamentable H&J. Frankly, most of the H&J (hyphenation and justification) algorithms in desktop publishing programs are lamentable, particularly when compared with TeX (as pointed out recently by Stephen Edwards in the Seybold publication, *The Bulletin*). [1] TeX calculates line breaks much more carefully than a DTP program, eschewing the quick-and-dirty, one-line-at-a-time H&J algorithms of DTP. TeX assigns "penalties" or "badness" for each unfortunate thing that could happen during H&J -- for instance, stretching or squashing the word- and letterspaces (the "glue," in TeX-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 that result in the smallest total penalties. Thus a change to the last line of a paragraph in TeX can, in theory, affect where the first line breaks. Expert users may even edit the penalties tables to reprogram the algorithm.

Control over kerning tables has also been lost. In 1987, Erik Spiekermann wrote and typeset a wonderful little 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. Erik's custom editing of the Walbaum typeface's kerning and touching tables, demonstrated in the book by a before-and-after comparison, is even today beyond the capabilities of DTP.

As for automatic hyphenation, the results from DTP programs are often so bad that I know of no discerning typographer who leaves auto-hyphenation switched on. Many designers of my acquaintance prefer to set "discretionary" (soft) hyphens by hand as needed. Solving bad line breaks may also involve inserting forced line returns or nonbreaking wordspaces, and occasionally track-kerning a range of characters or words tighter or looser.

The good side of WYSIWYG. Of course, 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 several iterations through the typesetting process to get the page looking right. Nowadays a poster or brochure can be pulled together on the screen in double-quick time.

A WYSIWYG view is also valuable to designers of 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 line 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, therefore, prefer to write directly into the WYSIWYG page makeup environment, and it is how I prefer to produce such publications.

Using WYSIWYG to fix what's broken. Another reason I value WYSIWYG is that it helps me to review the end-of-line decisions 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. WYSIWYG warns me about these problems and lets me do something about them. (In fact, FrameMaker's less than perfect WYSIWYG sometimes misleads me into fixing things that aren't broken?a considerable source of irritation!)

But is this low-grade remedial work an inevitable part of publishing? Why should discerning typographers have to put up with spending so much time cleaning up the mess made by DTP? Speaking from the floor in the final plenary session of the 1995 Seybold San Francisco conference, I asked the audience of more than 2,000 how many felt that the basic H&J abilities of the DTP programs they use had not improved in the last five years of DTP. A sea of hands went up!

True, attenders at Seybold are arguably better informed than the average DTP user, so this response may not be typical.

And perhaps it is a byproduct of the democratization of typesetting that most users of any mass-market DTP program are satisfied with mediocrity, leaving the vendors and developers with little incentive to improve the typographical capabilities of their systems. But why are we seeing virtually no improvement in the automated typography of the major DTP programs? Is the future of typography really in safe hands?

The structure of control: two models

The desktop publishing programs in common use can be divided broadly into two groups. PageMaker and Xpress are representative of the first group, and Ventura Publisher and FrameMaker of the second.

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

Bogged down by minutiae. Now this is a direct and intuitive way of working. It delivers a high degree of apparent control; but for producing some kinds of publications, it can drive you crazy. Suppose that you work on a longish report, with some elements flanking the text column -- side headings, for instance, or warning icons. In PageMaker or Xpress, you cannot anchor those elements to the text so that they move down as new text is added above. Add a requirement for footnotes and cross-references, push the document through 15 revision cycles, and your DTP operator is well on the way to a psychiatric institution.

Here is a clear illustration of how a WYSIWYG paradigm certainly gives control, but also makes you responsible for the low-grade task of making sure all these bits end up in the right places. Surely it would be better to surrender that low-level control in favor of greater automation? Putting it another way: Wouldn't it be better to move control to a higher level of generality -- such as rules for how side headings behave -- and forgo the need to exert control over minutiae (unless you really want to make such local alterations)?

Markup through styles. At least these programs took an important step toward efficiency by adopting the "style sheet" system, in which the user defines a list of typographic styles, named to match with their function in the publication (e.g., "title" or "body") and associates with each style: (a) typographic attributes such as typeface, leading, alignment in the column, embedded tab markers and so on, and (b) behavioral attributes such as pagination behaviors, [2] the application of hyphenation rules, and whether such paragraphs are to be referenced in a table of contents.

The advantages of working with style sheets 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).
- Consistency of appearance and pagination behavior is guaranteed between all paragraphs of the same generic type.

In addition, using style sheets conveys two additional benefits at an organizational level:

- When a new publication is being designed, its typography can be decided efficiently by setting up styles, applying them to dummy text, then modifying the styles iteratively to fine-tune them to each other. (When a style is modified, all instances of its use are updated with the new attributes.) Thus a designer can test thoughts such as, "What if we made all subheadings 2 points larger and reduced letterspace by 2%, then put an extra 4 points of space above each one?"
- A designer can "freeze" a style sheet into a template document, then give it to 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.

DTP programs haven't always supported this method of typographic markup. 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 reset the program's defaults so that the styles palette was displayed on the screen on start-up, causing many to investigate the power of style sheets for the first time. Even so, from my experiences as a trainer and remedial "fixer" of broken DTP files, I can assure readers that many users of PageMaker and Xpress remain obstinately immune to the charms of styles!

Dtp for propeller-heads: second model. The second class of WUSIWYG desktop publishing software has a stronger orientation toward the use of generic markup techniques and rules-driven pagination. Interleaf was an early pioneer, and Ventura Publisher was the first DTP program to put styles ("tags," Ventura calls them) on the desktop. The program of this type that I use daily is FrameMaker, which runs on many kinds of computers. I use it on a Mac.

Such 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 behaviors ignored by PageMaker and Xpress, such as side headings, column balancing and the anchoring of graphics frames. They provide facilities for automatic numbering of paragraphs or figures, and create tables properly with cells, 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 such programs are not ideal for creating a quick one-time advertisement, newspaper, brochure or magazine. But they are highly suitable for the production of technical manuals, directories, product catalogs, 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 reissued in several successive versions: so-called "maintained documents."

Analyzing structure saves time. It helps to get the best out of a program like Ventura or FrameMaker if you are prepared to analyze the structure of your publication carefully, matching each element type with a style-sheet item, variable or other generic construct. It takes longer to set up a new document type this way, but you 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 ("catalogs") 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 entities. One advantage is that if a character style varies in several ways from the main text like this (from a medium serif font, to a bold sans font set slightly smaller to match the x-heights of the serif type), I speed formatting and guarantee consistency. Another advantage is that if I decide that all species names should now be changed to display in 9.5-pt. Galliard small capitals with 3% letterspace, I can apply that change globally in a matter of seconds.

So do you have to be one of those proverbial "rocket scientists" to use a publishing program like Ventura Publisher or FrameMaker? [3] No, but it demands an intellect that can see beyond the appearance of a publication to understand its structure and the ways in which it will be edited, revised and perhaps reissued in new formats. And that's not a view of publications that most graphic designers -- or secretaries graduating to DTP -- bring with them from their training. Indeed, the perceptions that are important in such endeavors are more like those of the writer or editor. Of which more later . . . .

Generic markup needs a comeback

Style sheets vs. fully generic markup. At this point, reviewing my description above of the five diverse approaches to computerized publishing at the beginning of the 1980s, you may wonder if there is any parallel between formatting text in FrameMaker using the generically named markup stored in paragraph and character catalogs, and the generic markup approach taken by SGML.

The answer is a mix of yes and no. When, in FrameMaker, I set up a character catalog style called Linnaeus, I do so principally as a convenient way of applying consistent typographic style to the names of bird species. Each catalog 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" (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 markup to define whether such entities are displayed in italics -- or perhaps in big green capital letters with purple spots, doing a jig and flashing on and off in Netscape Navigator, version 6, alpha 5.

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. [4]

Multiformat publishing is now the motivation. When the Graphic Communications Association got involved in the committee work that led to SGML, its 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 been out of business for five years. But these days, the phenomenon that is causing renewed interest in generic markup 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 online help or as a collection of pages on the World Wide Web.

As Liora Alschuler remarks 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 and Hamilton put it. Other uses include online publishing, hypertext, sophisticated search and retrieval, and platform and vendor independent transmission and storage. [5]

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 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 to view or print locally. Nicholas Negroponte, professor of media technology at MIT, speculates 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 personalized summary. This kind of paper is printed in an edition of one. [6]

If I may tame Professor Negroponte's project by excluding his exotic artificial intelligences scanning the planet on our behalf, his vision of The Daily Me nevertheless becomes easy to envisage. I like the idea of being able to subscribe to a newspaper that gives me the scientific and political news, with a special focus on southeast Asia, and suppresses all references to football and the British royal family. But I can see it working only if the text sources from which the computer compiles my edition of one have been marked up in a manner that makes their content easily computable. SGML enables this.

There are also certain reference publications that do not make sense as printed volumes. Encyclopedias and dictionaries are out of date the day you buy them, if not before. Bus, train and airline timetables -- ditto. It would be more logical for these publications to be online 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, which means, in many instances, using SGML as the data format to make these transformations easier.

Formatting for view/print on demand. 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 thing that makes these publications special 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 TV listing or a classified ad database. [7]
- The variety of means of delivery argue in favor 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 or a desktop computer with a color screen, or after it has been printed onto paper from a fax machine or a high-resolution laser printer. Each instance should look and behave differently. (You can't read 8-point Bodoni on a screen or fax, and you can't scroll or hyperlink to notes on a paper page.)

If the results are to be "magnificent" rather than "hellish," we will need to rely on intelligent typesetting and pagination on the fly -- the return of batch formatting, perhaps along the lines of how TeX does it. Without scope or time for WYSIWYG cleanup, 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 and should cause a reexamination of the benefits of both generic markup systems like SGML and batch typesetting and pagination systems like TeX.

HTML markup and Web typography. The first widespread manifestation and bellwether of the digital publishing medium Negroponte envisages is the magnificent-and-hellish World Wide Web. Now, the Hypertext Markup Language (HTML) that makes the Web possible is essentially one application of SGML, and as such has the power to be a powerful mass-market demonstration of the benefits of generic markup for publishing in the electronic age. But here, too, designers are already itching for WYSIWYG control and grabbing at every layout-oriented tag Netscape offers them. I think that this is a mistake and that the focus of pressure for improvements in Web typography should be applied elsewhere -- specifically, on improving the automated layout capabilities of browsers.

The main reason it makes sense for Web pages (or whatever succeeds them) to continue to be created using a generic markup language is that it's impossible not to be a multiformat publisher when you publish this way. You have no control over whether your publication is being viewed in color or in 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 line breaks occur. In the future, Web pages may also be viewed more on tv screens and hand-held "personal digital assistants," strengthening the arguments for a properly generic markup.

Rescuing Web typography -- from a user's perspective [8] -- requires two things:

- Upgrading browsers in the direction of giving users more power to define and apply formatting style sheets appropriate to their computer, fonts and preferences.
- Improving the H&J capabilities of browsers, and the pagination behaviors for Web pages that are printed -- for instance, keeping headings with text that follows them, and allowing widow and orphan avoidance to be defined via style sheets. [9]

WYSIWYG and People: Skill Sets, Divisions of Labor and Training

It is time to ask what WYSIWYG has done to us, the humans involved in the publishing process. What has desktop publishing done to the former divisions of labor and to the skills that 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, the four kinds of skills necessary to any publishing endeavor these days are:

- Subject expertise, such as knowledge of marine law, molecular biology, computer networks, your company's product range, etc.
- Editing and writing ability, founded on a good command of one's language but also extending into specialisms such as methods of indexing or compiling bibliographies.
- Visual sense and design skills, necessary for graphic design and typography.
- Computer and keyboard skills, including the ability to work with particular computer programs and get stuff printed out.

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

Does WYSIWYG mean one person does it all?

Many 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 pasteup artist and perhaps an illustrator, too. As I see it, the implication also (by omission) was that it was the writer, the originator of ideas, who would be displacing all these other people.

I remember one Xerox advertisement that portrayed an office inhabited by several 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 endeavor; did it also imply that the perfect operator is necessarily a person of several large talents?

Some people are equipped to do it all in DTP; many early adopters were. For me, the Mac was a liberation. I'm a writer and editor, a typographer and illustrator, an artworker and a 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 therefore greatly enhanced by the invention of this all-around communicator's Swiss Army knife.

As an active member of the Information Design Association, I can report that Information Design and technical communication are disciplines rich in people like me in this respect. Information Design, a perspective emphasizing the integration of typography with language, attracts people who work happily in either sphere (or hemisphere, perhaps).

WYSIWYG publishing software surely fulfills its promise best for those who contribute a broad swathe of verbal and 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 line or page break has no acceptable typographic fix, you go in and edit the text until it fits. Believe me, I do this on every page of every training manual I write.

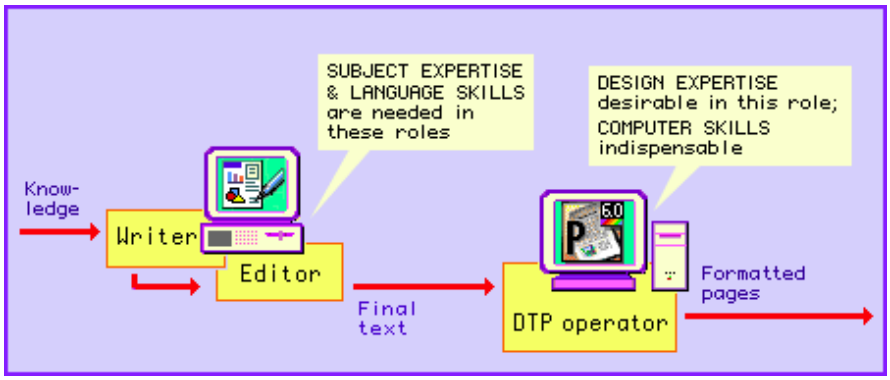
But not everyone, and not every sort of publishing activity, can work in the same way. The designer, the writer, the editor and the person who makes up the pages on-screen may all be different people, making the feedback loop necessarily more ponderous. And it also raises the thorny question of who does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

Who does what?

Few writers and editors now work with a pencil or typewriter. Most have some degree of computer skill. But their valued strengths are their knowledge of a subject and the ability to express it well in their mother tongues.

Some writers and editors are competent desktop publishers, but it is more usual for their texts to be transferred to someone else -- perhaps called a 'desktop publishing operator' -- who imports them into a WYSIWYG page makeup 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.

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



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

- Who applies the **markup**?
- Who does the **design**?
- Who takes care of **typographic quality control** by checking proofs and fixing line breaks, kerning and pagination problems?

Who does the markup? -- Before desktop publishing, markup was written on the typescript by a designer, or possibly an editor, and implemented in code by a compositor. These days, there is little room in publishing for a compositor whose sole function is text entry and the application of typesetting codes on instruction from somebody else. (In 1982-83, a lot of the typesetting I required was handled beautifully by a very meticulous Itek Quadritek operator; but she failed to adapt to the demands of desktop publishing and her next job, I regret to say, was at a supermarket checkout counter.)

Today the DTP operator applies the markup-- but also makes up pages, including 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 markup. Where texts have a complex structure and the complexity of that structure is mirrored by complex typography, the person who does the markup must have a deeper understanding of the editorial process, and probably of the subject area, too.

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 of Curtin University of Technology in Australia, trained designers handle perhaps only 2% of all pages made up using DTP. [10]

- 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 markup-- 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, markup skills and DTP-user skills together?

The designer as programmer. One solution to this impasse is to get a designer to define the look of a document, or a whole class of documents, and encapsulate those decisions within the computer program that will be used thereafter by other people to format the documents. All desktop publishing programs let you set up template documents in which page margins, master pages, paragraph styles -- perhaps also specialist character and table styles -- are predefined.

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 working with an Interleaf employee to convert the design to an Interleaf TPS template document. However, the ideal situation must surely be one in which the template designer knows the 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 mindset 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. To typeset from an SGML source file, they take it through a rule-based conversion process resulting either in a **`troff`** file [11] or a text file with embedded markup that will flow into an Xpress 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. [12]

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).

And who takes care of the details? When I worked at Inside Asia magazine and galleys came back for proofreading, 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 markup had been correctly applied, and I also focused on the quality of line breaks, 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 labor, it was the trained compositor who paid attention to such details, undistracted by

either the larger picture of the page design or the knowledge of the subject. Nowadays you have typesetting that is 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 gray patches on the page. They prefer to put their energies into page composition, choice of colors 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. It seems to me, though, that we need certain kinds of people with crossover skills and certain kinds of dialog among 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. That also means that the graphic design courses in colleges need overhauling.
- 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 better at design. Companies should be investing in more in-service training for their DTP operators.
- We need forums for discussion and training in which typography 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 toward a common understanding of the relationships among the structure of meaningful elements in a text, the way that structure is represented in typographical form, and the way 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 minimize the amount of conversion work this will require.
- Finally, faced with an exciting (terrifying?) future of documents slipping away from 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, editorial people and computer people to work together to ensure that the result will be pleasant to look at and easy to understand. And we desperately need more support from the vendors of publishing software.

Conclusion

Don't let the toolmakers off the hook

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:** Xpress (for Macintosh and Windows).
- **Corel:** Ventura Publisher (for Windows)
- **Adobe Systems:** PageMaker (for Macintosh and Windows) and FrameMaker (for Mac, Windows, Sun, HP-UX and a number of other Unix platforms).

During the last year 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 among the more thoughtful desktop publishing practitioners that we are not well served. [13] And not being listened to, either.

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 five years ago. Now that everyone has converted to desktop typesetting and there are no new worlds to conquer, upgrade fees alone may not pay for rewriting the software. If we want quality, this time we may really have to pay for it.

There does need to be a place where users of publishing software can talk to vendors, where the case for good typography can be argued, and where the point can be made vigorously that typography isn't just typefaces. I do think it's a shame that the Worldwide Publishing Consortium was a flop and must confess to a certain amount of dismay at the thought that in the excitement over the Web, publishing events and publications (Seybold ones included) will regard text composition issues as no longer deserving of attention, when there are still so many problems to be solved.

So, is what you see all you'll 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 . . . 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 the designers -- who survive and flourish in that world will be those who learn to look beyond appearances.

It's quite a challenge.

1
-- Edwards, Stephen. "H&j: Whatever Happened to Progress?" The Bulletin: Seybold News & Views on Electronic Publishing, 21 August 1996.
[Back to text]

2
-- Pagination behaviors include whether 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.
[Back to text]

3
-- Actually, some Ventura and FrameMaker users really are rocket scientists.
[Back to text]

4
-- This context-sensitive approach to the formatting of elements is also taken by the structured-editing version of FrameMaker, FrameMaker+SGML.
[Back to text]

5
-- Alschuler, Liora. ABCD . . . SGML. International Thomson Computer Press, 1995: p. 38.
[Back to text]

6
-- Nicholas Negroponte. Being Digital. New York: Knopf, 1995: pp. 152 -- 153.
[Back to text]

7
-- May I propose the term "smörgåsbord publishing"?
[Back to text]

8
-- I recognize that many designers want the ability to control the appearance of type on Web pages in the interests of adding a desired "look" and enforcing corporate identity. But my priority would be to serve the desires of the users, not the designers or publishers.
[Back to text]

9
-- As also argued for by Mark Walter in "It's time for a browser that H&Js," in the Feb. 19, 1996, issue of The Seybold Report on Desktop Publishing(Vol. 10, No. 6, p. 2).
[Back to text]

10
-- This comment was made by Swann at a conference on the teaching of typography, held at Manchester Metropolitan University in the UK on Nov. 17, 1995.
[Back to text]

11
-- **`troff`** (stands for "typesetter run-off") is a batch-formatted procedural markup 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.
[Back to text]

12
-- Alschuler, ABCD . . . SGML, 153.
[Back to text]

13
-- This resulted in an article in **IDeAs**, the newsletter of the Information Design Association, comparing these four programs and information designers' opinions of them. It may be downloaded as an Acrobat file from my Web site: http://www.ideography.co.uk
[Back to text]