The book is entertaining and interesting, but there isn't a huge amount of practical information in the book. That may or may not be a problem for you, depending on what you're interested in; not everyone wants or needs to delve deeply into typography and design. If you're just interested in an overview of typography to gain a simple awareness of what is available and how type is used, this might be a good book for you. If you're at all serious about the use of type in your work, I recommend reading more widely. There are vastly diverging points of view and perspectives on design and typography. The authors include a bibliography at the end of their book for further reading, and many bookstores have books on design and typography in their art sections. I've had good experience with the selection in university libraries as well.

There are myriad opinions about what is good, right, and true about the use of type, and it helps to get a sense of the range before you make your own decisions. I find a historical understanding of typesetting and printing helpful (usually given in the introduction or first chapter of many books on design or typography) to understand where we've been, past and present assumptions about what is readable and legible, and what's been done and what's available with design and type. Once you're done some background checking, just pay attention to the print around you: movie titles and credits, advertising, labels, books, brochures, forms, whatever you see that uses type. Develop your own list of fonts you like to use, your own tastes, stay open, and keep experimenting.

◇ Merry Obrecht Sawdey
3532 Bryant Ave So.
Apt. 112
Minneapolis, MN 55408
sawdey@denali.ee.umn.edu

Pre-publication review: *Practical SGML*

Nico Poppelier

Eric van Herwijnen, *Practical SGML*. Kluwer 1994, 284 pages (including indexes). To be published.

In my review of the first edition of *Practical SGML* by Eric van Herwijnen (*TUGboat* 13, no. 2), I praised it as 'one of the best books on SGML currently available.' It still is one of the few books on the practical application of SGML, by someone who has used SGML in practice rather extensively. The new edition has undergone significant changes with respect to the previous one. Unfortunately, they are not all changes for the good: the book still contains a lot of practical information — more than the first edition — but it is not a *better book.*

As a reference work the quality of the book has certainly improved. More material has been added, and the book has been largely re-structured. The previous edition consisted of three parts, *Getting started with SGML*, *Advanced SGML* and *SGML implementations.* The new edition has more chapters, grouped together in four parts, *Getting started*, *Writing a DTD*, *Customizing SGML* and *Special applications.* Especially the second part, about how to write a DTD (document type definition), has improved a lot, with chapters on document analysis, structure diagrams, and the various declarations one can find in a DTD. Part III, about customizing SGML, describes the SGML declaration, and SGML features such as minimization, marked section and short references. It also describes the problems that can arise with ambiguous definitions, and gives advice about how to avoid ambiguities. Under the heading of 'Special applications' (part IV) Mr. van Herwijnen discusses SGML and EDI, SGML and mathematics, and SGML and graphics. He also explains the relation between SGML and other ISO standards, such as, e.g., DSSSL and SPDL. In all the examples in the book the public-domain **SGMLs** parser is used, which makes it possible for most readers to try the examples for themselves.

On the negative side however: so much material is now contained in the book, especially in the form of figures and tables, that the book, in my opinion, is not a *pleasant-to-read* introduction to SGML any more. Another thing which I find rather distressing, at least in the pre-publication copy the author kindly sent me, is the design: the book uses too many fonts, in sometimes unharmonious combinations, the distribution of vertical space is uneven, and the placement of tables and figures leaves a lot to be desired. A possible explanation could be that this new edition of *Practical SGML* was prepared using SGML, and was formatted using Adept 5.0 from ArborText Inc. Obviously, designing a book that is comfortable to read is not the same as writing a 'FOSI', an output specification for ArborText's Adept product. I hope that the publisher will work hard on improving the layout of the book, but I have my doubts.

Of course, this says nothing about the applicability of SGML to book production, but only about the quality of available SGML tools, or the expertise of the people using these tools. That computers *are* capable of producing more readable and more attractive books is shown by a book co-authored by one of Mr. van Herwijnen's colleagues at CERN, namely *A LaTeX Companion*, by Michel Goossens, Alexander Samarin and Frank Mittelbach. But then, of course, that book was made with LaTeX!

◇ Nico Poppelier
Elsevier Science Publishers
Amsterdam
The Netherlands
Internet:
`n.poppelier@elsevier.nl`

Book review: *Literate Programming*

Christine Detig and Joachim Schrod

Donald E. Knuth, *Literate Programming*. Center for the Study of Language and Information, Lecture notes no. 27, Stanford 1991. (Distributed by the University of Chicago Press.) xvi + 386 pp., index and comprehensive bibliography.
ISBN 0-9370-7380-6 (pb), 0-9370-7381-4 (hc).

The essence of literate programming?
Say it twice!
— D. Knuth (1993)

This book is an anthology of works by Donald Knuth; it tells us the story of literate programming. It consists of an introduction, a lecture, eight articles, three book excerpts, a program, and a bibliography. John Hobby is responsible for the selection of the contents; the introduction is the only text previously unpublished.

The presented material spans almost 20 years of Knuth's work, in which literate programming developed from concerns about the quality of software description, through first ideas to categorize and improve it, to applications and experience reports based on the methods and tools he has created.

Contents

The collection starts with the *Preface* that presents Knuth's views on the relation between the different texts selected by Hobby. It shows the "red thread" of the book and gives advice on how to read this book. Besides this introduction, the only new material is a paragraph at the start of each text that presents the context of original publication.

The first text, chapter 1, is the lecture given by Knuth in 1974 when he received the *Turing Award*, the most important Computer Science award. Already at that time, Knuth had named the basic principles and motivation of literate programming:

> The chief goal of my work as educator and author is to help people learn how to write *beautiful programs.* [...]
>
> [The goals of correctness and adaptibility] are achieved when the program is easily readable and understandable to a person who knows the appropriate language. [...]
>
> Please, give us tools that are a pleasure to use, especially for our routine assignments, instead of providing something we have to fight against.

In this lecture, *Computer Programming as an Art*, Knuth argues that programming has much in common with music composition. Here we also find the reasoning behind the statement that programming is not a science, but an art. Knuth still holds the professorship for the "Art of Computer Programming" and this chapter shows us basic principles of his whole professional life.

Chapter 2 presents one of Knuth's most cited articles: *Structured Programming with* **go to** *statements* (1974). This article must be read in the context of its time: People had just started to develop programs in a systematic, structured way; the scientific community was discussing for the first time how to write long-living programs. It's written in the context of Dijkstra's famous letter "Go to statement considered harmful" and shows that the problem of unstructured programs is not based on language constructs.

Chapter 3 continues with an early effort of Knuth to present a larger piece of code in a readable and understandable way: *A structured program to generate all topological sorting arrangements* (1974). According to Knuth himself, the presentation of this article left much to be desired. He had realized that writing programs intended for critical reading means to make construction and evolution recapitulable. This requires other forms of writing and presentation than the old, machine oriented, style.