The first book starts with an extensive background on the history of HCI and user interface design. It is questionable whether this is really warranted, especially at the level of detail at which it is presented. Most readers will not be too interested with the various name and acronym changes of the HCI special interest groups in '*the early days*'. That the author met Doug Englebart (best known as inventor of the mouse) and found him a nice unpretentious bloke is an example of a questionable inclusion. Another thing that is noticeable at this stage is Treu's passion for formal definition and acronyms. It seems that in this book, if there is an acronym for something then it is defined and used as such, even if it occurs only once. Additionally, every single operative word is formally defined—even when the definition is identical in meaning to that which any pocket dictionary would give. Opening the book completely at random as I write gives me a definition of the word 'context'. I know of no one that does not understand this word in the sense it is given: is this padding or does the author really feel a need to make sure his readers know what this means? Trying to get into this book is not easy, and this is entirely down to the fact that it reads like a set of old lecture notes. Flow and continuity is poor and at some stages you really feel like putting it down, even in the middle of the more interesting portions.

If you persevere, however, you will not be entirely disappointed. The books contain some interesting subjects. One thing that a reader will notice is that they are well researched. There are further references everywhere (although at some times this proves to be a further obstruction to smooth flow) and the bibliography in the first book alone runs to some 12 pages. The second volume gives the number of publications selected and referenced for the two book set as 'over 425', and goes on to briefly discuss trends in those works. This would supposedly cover the bulk of significant published material on the subject in its short history. As such, anyone looking for a starting point for research or directed reading in this area could do far worst than look at Treu.

If you survive the early chapters and acquire sufficient resistance to the style issue, you go on to find that the books do indeed offer some worthwhile insights into the field. As I have said, Treu does not concern himself with issues of layout, but more at the feature level of interaction. How to analyse what constitutes a necessary inclusion in an interface design and what will produce a good interface between software and people is the name of the game, even if it is in a somewhat formal style. You might go as far as to describe these, indeed most HCI texts, as attempts to make order from chaos—taking the highly subjective issue of interface design and giving it a more formal, logical, structure. Do they succeed? Well, this itself is a subjective question, but I think that they could certainly have done a worse job.

As a reference tool, the books are good, but more as a guide on where to look than a definitive source of answers. How does it stand as a textbook? I would say 'on a somewhat less stable footing'. It is not easy to read Treu, and certainly many of those at the undergraduate level will struggle to find the willpower to read the books from cover to cover, or indeed to finish them at all. The numerous exercises in the book are best avoided as presented. As something to provoke thought on each chapter they are fine at the mental level. I pity anyone called upon to answer them textually though—potentially a very tedious process, and one that could not be accomplished without repeated back-references. If used as such, the only purpose served by the exercises is to make the reader *read* the book, only benefiting those who would otherwise not so do. I would say that the books could be used for a very specialized course—'only those *very* seriously interested need attent'. Additionally, a complete HCI course would need a sizeable amount of supplementary material to cover the absence of aesthetic and cognitive-based considerations on the more 'human' side of the subject.

Who will want to read this book? Basically, anyone looking for a place to start, or interested in all sides of the subject. Those strictly into implementation and programming are advised that they may get ideas from this book, but not direct help, and if you want to know how to explicitly lay out a screen, try looking elsewhere. Mr Treu does, after all, advocate going out and asking the advice of some 'experts' in the relevant field as a vital step in designing an interface.

COLIN ISAAC
Warwick University

KEE YONG LIM & JOHN LONG
The MUSE Method for Usability Engineering. Cambridge University Press. 1994. ISBN 0 521 47494 9 £29.95 330pp. hardbound.

MUSE is an acronym for Method for USability Engineering and this book is about the design process in human–computer interaction. Lim and Long argue very powerfully for a human factor design cycle which is complementary to existing methods of structured design for the functional aspects of a computer system. Unlike some sets of HCI design guidelines the MUSE methodology attempts to express design in procedural rather than in declarative terms and this approach lends itself to the integration with structured software design methods.

The contribution of human factors to system development is characterized as '*too little, to late*'. The book focuses on the development of a method for integrating HCI into the early stages of the system development life cycle, where errors are the most expensive to correct. This is in contrast to some other techniques which tend to raise HCI issues only towards the end of the life cycle. In other words, MUSE has great strength in the system specification stage, while other HCI design techniques tend to focus on the system evaluation stage of the life cycle. Other approaches to HCI design are not necessarily excluded, but they are considered to be seriously incomplete.

The MUSE methodology is concerned primarily with the development of large systems, typically those involving multidisciplinary design teams. In the context of small projects with a single designer the use of MUSE might be regarded as overkill, but even here the well documented design solution which MUSE facilitates may still be appropriate.

The book is aimed at software designers with some understanding of existing HCI principles and techniques. It is essential reading for any HCI specialist who is interested in the formal integration of human factors into the system development life cycle. It is also highly recommended for software engineers who use structured methods. The well-established methodology which the authors have selected to illustrate MUSE is JSD. The authors are attempting to assimilate human factors considerations into JSD, rather than to extend JSD to encompass human factors. Thus the desirable characteristics of JSD and the well-established JSD techniques are not affected by the authors' enhancements. This work is not the first to attempt this synthesis of methodologies, but is does seem to be the most complete so far.

Does the method work? The reviewer has not implemented any large systems during the review period, so the method has not been put to any practical test. However, there is a great deal of good sense in the MUSE approach. It attempts to find a sensible balance along the general–particular scale, i.e. avoiding both under- and overdesign. It attempts also to give a complete coverage of HCI issues in the design of systems. The authors succeed to quite a large degree in their aims of capitalizing on well-established techniques, and promoting communication between HCI designers and more traditional software designers.

The book includes a useful case study of the application of the MUSE method for network security management at University College London Computer Centre, and this provides a good illustration of the power of the methodology.

Both HCI academics and HCI practitioners will find this book thought provoking and genuinely useful. Most will not agree with everything written here: that is the price paid for evincing a distinctive point of view and attempting something of genuine practical value. Some may be dismayed that their favourite HCI sub-topic is not mentioned here, but the book is not intended as a general HCI text. However, all should enjoy the lucidity of the text, the well structured development and the explicit expression of the key ideas. The attention to detail is thorough. The diagrammatic notations are generally clear enough for use by non-HCI specialists and sophisticated users. Anybody already familiar with JSD will find this book doubly useful, but readers who do not use JSD need not be deterred.

J. ROSBOTTOM
University of Portsmouth
Department of Information Science

DERMOT BROWNE
STUDIO—Structured User-interface Design for Interaction Optimisation. Prentice-Hall. 1994. ISBN 0 13 014721 4 £22.95. 291pp. softbound.

The book addresses a neglected area, that of user interface (UI) design. The content affords the user interface the respect it deserves, given that so much resource is spent on coding and maintaining it. Browne quite rightly points out that the UI is often of as an afterthought or something that is tacked on after the other parts of the system have been designed. The idea of treating UI developments as a separate activity from systems development is a good one, particularly for client–server applications.

The introduction provides a good 'route map' for navigating through both the book and the STUDIO method. Relevant anecdotes and a continuous case study of financial dealing software are used to good effect throughout the book to communicate the importance of a systematic but user-centred approach to design.

The book challenges the complacency of those IT professionals who assert that to achieve a quality user interface requires only that '*style guidelines are followed unerringly*'. Browne uses anecdotal evidence to show that if rules are followed slavishly without understanding, they are likely to be applied inappropriately. His systematic treatment of user requirements analysis as a crucial precursor to the generation of user interface designs reveals the true complexity of user needs and behaviour. The case for thorough documentation and understanding not provided by conventional systems analysis is well made.

By example, rather than exhortation, STUDIO sets about showing readers (it is hoped, software developers), the procedures and benefits of a user-centred approach to UI design. This is a most valuable contribution. The reader, after exposure to the processes involved in thorough exploration of user requirements, is shown how to the generate user interface options, to use prototyping and impact analysis to reject inappropriate options and finally how to firm up and develop the relevant ones. The handover from user-interface designers to other developers also receives due attention, one more filling a significant gap in available guidance.

While STUDIO has a great deal to offer it also has some unfortunate limitations. For instance, the introductory example based on the HOLMES system for the Home Office is perhaps not as convincing as it