mention of them is made throughout the rest of the book. In all, the book would be undoubtedly useful as a reference book.

Examples of parallel algorithms are presented in Modula-P and Parallaxis for synchronous and asynchronous programming respectively. The interested student can obtain a free copy of the compilers and simulation systems as well as a Petri net simulation tool via ftp to experiment with. The book also includes exercises and a supplementary booklet with solutions is available from the publisher.

A. PHILIPPOU
University of Warwick

KATHY SPURR, PAUL LAYZELL, LESLIE JENNISON and NEIL RICHARDS (Eds)
Software Assistance for Business Re-Engineering. John Wiley, 1993, £26.95, 224 pp., hardbound, ISBN 0 471 94240 5

This book comprises the papers presented at the British Computer Society CASE specialist group seminar of 29th June 1993. Like all conference proceedings, it is a curate's egg. Business (Process) (Re-)engineering—its practitioners cannot agree on its syntax, let alone its semantics—takes up the baton of corporate data base and enterprise analysis. Not that B(P)(R)E is confined to IT, of course: it encompasses the entire enterprise and subsumes total quality management. It is nothing, many of the contributors tell us, if not holistic (so there's goodness for you).

Between three papers of introduction to the subject and two that address how the approach can be made to work, come six on proprietary methods and tools. However, the best introduction is probably the final paper, by Linda Hickman. The six central papers are much what we would expect: generally well written, with enough information to tempt and tease, but not enough to allow the reader any real understanding. They are, in a word, exercises in selling, some more blatant than others. One contributor lists seven selection criteria for process modelling tools and concludes: *Few process modelling tools in present use match up to these demanding requirements. The remainder of this paper discusses one tool ... which scores highly under every one of the issues discussed above.* Surprise surprise. The products range from mundane, but probably useful, tools for recording process analyses, to a process simulator that is object oriented (of course) and AI-based: just the sort of tool you would trust in a fundamental remodelling of your entire business.

The first of the selling exercises, by Julian Watts, briefly mentions the option of piecemeal process engineering—which cautious souls might favour—and gives a good account of the analysis of value streams (value-adding sequences of processes). Chris Haynes' introductory paper, the third and shortest, describes a somewhat different and rather promising value-based approach. But such nuggets are small, and few, and far between; the subject is new, its notions vague, and some of its verbiage makes one think that re-engineering has been applied to our mother tongue. Skip straight to Haynes' paper to avoid: *in the onrushing face of change ... Business Process Redesign as a concept is contained within the overall domain of change ... All this adds up to mean that the change is transformational ... New ways of working mean a change to the status quo ... rollout is the formal implementation and institutionalization of the innovation within the organizational mainstream.*

ADRIAN LARNER
De Montfort University

NADA LAVRAČ and SAŠO DŽEROSKI
Inductive Logic Programming. Ellis Horwood, 1994, £39.95, 293 pp., hardbound, ISBN 0 13 45870 8

Inductive Logic Programming (ILP) combines techniques drawn from inductive machine learning and logic programming with the principle aim of designing programs that generate (or induce) a set of general rules (clauses) from a given set of observations (facts). ILP also offers a rigorous and well-understood theoretical foundation and encouraging practical results when applied to real-world domain problems. Examples of applications include, automated knowledge acquisition inductive program synthesis and knowledge discovery in databases.

ILP splits into empirical and interactive approaches. Empirical ILP is used where numerous examples of a single concept and comprehensive background knowledge are available to the user; in contrast, interactive ILP covers cases where multiple concepts need to be acquired and where little background knowledge is available. This book concentrates on the former approach. In use empirical ILP assumes (i) a set of stipulated facts covering both positive and negative cases involving some unknown predicate in question, (ii) a description language which specifies syntactic restrictions on the target predicate, and (iii) background knowledge defining other predicates used to describe the domain and which are related to the target predicate. The goal is to find a set of clauses yielding a definition for the target predicate which accounts for the input stipulated facts and background knowledge.

The book is divided into four well-balanced parts: Part 1 introduces ILP; Part 2 is a detailed description of empirical ILP techniques and systems used; Part 3 considers the problem posed when using imperfect data; and Part 4 is an overview of ILP applications.

The authors draw upon years of research into theoretical and practical work on machine learning and qualitative modelling in an ILP framework. This integration works well in a book of this nature. As with any relatively new subject, material normally found in confer-

ence proceedings is covered, but it benefits from being woven into a self-contained book covering both introductory and advanced topics. Concepts and practical algorithms are clearly introduced and described. On the whole the book is clearly written, well-organized and sourced.

If your professional interest is purely academic, and an introductory book on ILP is sought, then this would be a good choice. If however your interests are in the practical applications of this technique outside an academic setting, then some caution is advised. Despite the book's truly excellent coverage and honest evaluation of applied ILP techniques and extant systems, you may find, as I did, that one needs to read and re-read several sections in order to clearly understand the limitations and underlying assumptions when ILP techniques are used to deal with imperfect data. In this respect, the book will require some work despite its introductory nature and intended audience.

D. A. RANDELL
Birmingham University

W. BISCHOFBERGER and G. POMBERGER
Prototyping-oriented Software Development. Springer-Verlag, 1992, 72 DM, 215 pp., hardbound, ISBN 3 540 55448 3

This book consists of two parts. Part 1 describes four different paradigms for software development and the concepts and some of the existing tools behind two of these paradigms—namely 'prototyping-oriented development' and the 'exploratory programming model'. It concludes with a short chapter on some of the issues involved in prototyping. Part 2 describes a toolset called TOPOS, developed at the universities of Zurich and Linz, based on the concepts outlined in Part 1. In this section, exploratory programming is treated more as a component of the prototyping paradigm than as an independent approach to software development. User interface prototyping, system architecture prototyping and the use of simulation in prototyping are some of the concepts implemented in TOPOS. The book concludes with a note on the experience gained through TOPOS.

From the standpoint of a practising software engineer, I find it difficult to feel enthusiastic about the book. The approaches to prototyping-oriented development are considered in an application-independent framework. Consequently, there is little room for detailed discussion on the many specific problem areas that affect software developers. For software engineers who plan, design and implement systems, it would come as a surprise that the authors have not even made a reference to the many tools that are available under MS-Windows on the PC's for prototyping. For example, the development environment under Visual Basic provides for excellent prototyping and evolutionary development through its design-time and run-time modes, 'object linking and embedding' and custom controls (which are Visual Basic objects with their own properties, events and methods). These are widely used by developers and are superior to tools like ET++ which the book describes.

The book talks about System Architecture prototyping and the use of simulation towards such efforts. Simulation is a good idea if it can be accomplished without too many tears. For large and complex systems, such prototyping itself becomes complex and expands to take up much of the resources. Project managers with schedules hanging over them often have valid reasons to see these efforts as taking the team away from the 'real problem' at hand. Additionally, a large class of problems come under the category of 'information systems'. The architecture for such systems is reasonably well understood by now and we have on-line transaction processors, relational models, report forms and the Windows' event-driven environment to provide a tested framework for such problems, leaving the developer to concentrate on prototyping the user interface and the application.

The book can be recommended for students who are interested in the software development process and in the concepts involved in prototyping-oriented software development. A section describing the kind of development environment that SMALLTALK provides helps in understanding the way TOPOS works. It would have been a good idea to give a demonstration diskette of TOPOS along with the book which the readers could install on their home or work computers to aid them in getting the best out of the book.

Except for a section on SMALLTALK, the tools described in the book are not in the mainstream experience of software professionals in industry. Professional software engineers involved in planning, designing and implementing systems would clearly note that the book is another symbol of the gulf that exists between them and the academic environment in so far as the perception of their problems is concerned.

V. RAGHUNATHAN
Freelance Consultant, Sydney, Australia

NIALL MANSFIELD
The Joy of X. Addison-Wesley, 1993, £21.95, 368 pp., softbound, ISBN 0 201 56512 9

The X Window System has long been a source of frustration to me. It is powerful, but very complex, and clarifying the relationships between the various GUIs, toolkits and libraries associated with X is not easy.

The Joy of X is a refreshing book which guides the reader gently through X, clearly explaining the concepts and utilities which form the X Window System. Niall Mansfield's style is very lucid; each section or subsection (referred to as a 'module') commences with a brief summary of its contents, and takes up a two-page spread.