1977), and a TTL logic network simulator (Kahn and Kinniment, 1976).

In all these systems it has proved entirely successful. In addition experimental work has confirmed the potential of RCC as an extensible language, in particular in providing powerful and efficient application packages (Allison, 1976), and as a language transformation language (Lindsay, 1975).

The best overall test of RCC has been in the logic simulator, where users can describe a network, specify a set of tests to exercise it over a period of time, and then get as output monitoring and performance information. This is an independent project, with a number of people making large or small contributions to it over a period of years. The extensibility of RCC has proved useful in allowing the project leader to make an extension for general use by the group, with individuals making further extensions. The language processing facilities have made easy the task of designing and implementing the user oriented languages for network description and test specifications. The high level language has proved very useful in documentation and maintenance. A particular feature is that some of the logic components, which have to be modelled in great detail, require large routines to describe them. The modelling is done using a small set of highly descriptive primary and secondary statements tailormade to the specific problem.

There are a number of further applications of RCC currently in progress, or projected in the near future. Research is being carried out into a general purpose language that provides user definition of primitive data types using in-line code. A package is being written to generate tailor made COBOL updating programs given a (non-procedural) specification of the required operation by the user. It is hoped to write a cross compiler for a microprocessor, and to investigate the automatic transformation of working programs in various languages to and from related forms, e.g. a very compact form (for archiving) or a very descriptive form (for documentation). Of course if the language is RCC, a working program should already be highly descriptive; here an interesting possibility is to transform it into a similarly descriptive program in another natural language, e.g. French, by providing the corresponding vocabulary for identifiers and the corresponding sentence structure for primary and secondary formats.

References

Allison, L. (1976). Extensibility in a Systems Implementation Language, Ph.D. Thesis, University of Manchester, 1976.

Arden, B. W., Galler, B. A., and Graham, R. M. (1969). The MAD Definition Facility, *CACM*, Vol. 12 No. 8, pp. 432-439.

Brooker, R. A., MacCallum, I. R., Morris, D., and Rohl, J. S. (1963). The Compiler Compiler, *Annual Review in Automatic Programming*, Vol. 3, pp. 229-275.

Eissa, I. F. and Napper, R. B. E. (1976). RCCT—A Simple Extensible Systems Implementation Language, Proceedings of the International Conference of Statistics and Computer Science and Social Science, Cairo, April 1976.

Fisher, R. N. and McQuarrie, G. W. (1977). MPL1700—A High(er)-Level Microprogramming Language, *Software, Practice and Experience*, Vol. 7 No. 6, pp. 747-757.

Kahn, H. J. and Kinniment, D. J. (1976). A Design Automation System for the Teaching of Computer System Design, Proceedings of the International Conference of Statistics and Computer Science and Social Science, Cairo, April 1976.

Lindsay, H. E. (1975). The Design, Implementation and Use of the Regeneration Machinery of RCC, M.Sc. Thesis, University of Manchester, 1975.

Morris, D., Wilson, I. R., and Capon, P. C. (1970). A System Program Generator, *The Computer Journal*, Vol. 13 No. 3, pp. 248-254.

Napper, R. B. E. (1968). The need to Revise the Compiler Compiler, *IFIP Conference Proceedings*, pp. B23-B27.

Napper, R. B. E. and Fisher, R. N. (1976). ALEC—A User-Extensible Scientific Programming Language, *The Computer Journal*, Vol. 19 No. 1, pp. 25-31.

Newey, M. C. (1968). An Efficient System for User Extensible Languages, *AFIPS Proceedings* (*FJCC*), Vol. 33 No. 2, pp. 1339-1347.

Solntseff, N. and Yezerski, A. (1972). ECT—An Extensible Contractible Translator System, *Information Processing Letters*, Vol. 1, pp. 97-99.

Solntseff, N. and Yezerski, A. (1974). A Survey of Extensible Programming Languages, *Annual Review in Automatic Programming*, Vol. 7 pp. 267-307.

Standish, T. A. (1975). Extensibility in Programming Language Design, *AFIPS Proceedings*, Vol. 44, pp. 287-290.

Wegbreit, B. (1971). The ECL Programming System, *AFIPS Proceedings* (*FJCC*), Vol. 39, pp. 253-262.

Book review

Research Directions in Software Technology edited by P. Wegner, 1979; 869 pages. (*The MIT Press*, £15·00)

This is a long and ambitious book aimed at covering the complete field of software technology. It has its origins in papers presented at meetings in 1976 and 1977. The stated purpose of the book is to stimulate a dialogue between research workers and practitioners in the field of software technology. The book is divided into four parts each consisting of a set of papers by experts in the field followed by commentaries on these papers by others. This style of presentation leads to a significant amount of repetition with each author seeing the need to define terms and provide a framework for his own contribution.

The Software Engineering section includes papers by Boehm and Mills which review the tools available for managing software production introducing the software life cycle, specification languages, top down design, error days, etc. Anybody familiar with the field would find little new here and yet the level of detail is such that it provides little more than a pointer to other sources for a newcomer. The most interesting paper in this section is a frank discussion of the management of the MULTICS project describing both the successes and failures.

The Software Methodology section includes a good paper by McGowan and McHenry suggesting that the software life cycle model breaks down when a system is under continuous evolution and proposes a continuum model in its place. This same criticism is touched upon by a number of authors. This section also includes papers by Liskov and London on formal methods of program specification and verification.

The Computer Systems Methodology section has an excellent paper by Wegner on concepts and research directions in programming languages. Other papers discuss the current state of operating systems research and architectural design including software and hardware problems associated with distributed computing systems.

The final section on Applications Methodology is the shortest and perhaps least successful. It covers too wide an area—from CAD packages and data base management systems to natural language and artificial intelligence systems. It does no more than indicate the current state in a superficial manner. Even then, the passage of time has dated some of the presentations.

I found the book disappointing. The presentation is shallower than should have been possible in a book of this size. It also spends more time cataloguing the past than providing clear indications of research directions. Even so, the book has good parts. It is just a pity that over 800 pages have to be read to find them.

F. R. A. Hopgood (Didcot)