



**Technical Editor:**  
**Marcin Paprzycki**  
Dept. of Computer Science and  
Statistics  
Univ. of Southern Mississippi  
Southern Station 1506  
Hattiesburg, MS 39406-1506  
m.paprzycki@usm.edu

## Book Reviews

# Structured Development of Parallel Programs

Reviewed by *Andrzej Stachurski, Warsaw University of Technology*

### Structured Development of Parallel Programs

By Susanna Pelagatti  
248 pages  
\$44.95  
Taylor & Francis  
London  
1997  
0-7484-0759-6

© 1999 Institute of Electrical and Electronics Engineers. Reprinted, with permission, from *IEEE Concurrency*.

This material is posted here with permission of the IEEE. Such permission of the IEEE does not in any way imply IEEE endorsement of this book. Internal or personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution must be obtained from the IEEE by sending a blank e-mail message to [info.pub.permissions@ieee.org](mailto:info.pub.permissions@ieee.org).

By choosing to view this document, you agree to all provisions of the copyright laws protecting it.

*Structured Development of Parallel Programs* presents a structured programming methodology for parallel computations that ensures portability, programmability, and good performance. The book's ultimate goal is to develop a suitable programming language for parallel programming and its compiler. This language is meant to deliver typical parallel constructs (skeletons) and their realizations (templates) on various architectures.

The book's first half presents a critical analysis of the state of the art of parallel software development. It also closely examines several existing approaches to parallel programming, concluding that template-based systems are the best compromise. In this approach, the programmer selects skeletons and their conversion rules, then uses them to build a program. Its performance might not match that of a low-level graph-based approach, but it is predictable and easily ensures programmability and portability.

The book's second half describes the P3L template-based methodology and its realization as the P3L language and its compiler, offering application examples. The author maintains that the template-based system gives rise to accurate performance models for the skeletons library designer as well as for the programmer. The technical and mapping details are left to the skeleton library designer, who can fully exploit specific properties of particular skeletons. The P3L methodology incorporates a small set of basic skeletons and their combination rules. Skeleton selection is based on the analysis of existing approaches. The skeletons reflect typical constructs that parallel program designers use.

The P3L methodology might be a good starting point for developing efficient high-level languages for parallel programming. It suggests how to ensure compromise between performance and portability and programmability. In any case, we should not treat it as something closed and finally established—high-level parallel programming languages continue to develop and improve.

Such high-level languages would let the programmer concentrate less on the details of the machine's architecture and more on the algorithm's design. The lack of high-level languages is one of the major obstacles hampering large, complex software projects and the development of computational algorithms. Currently, the progress of these languages is severely delayed compared to the pure parallel hardware performance. An efficient, high-level language for parallel programming available on computers with parallel processors and on clusters of machines used for distributed computations would be an important tool for people developing general theoretical and application-oriented algorithms.

This book should interest people working on parallel algorithms, but, more importantly, it should interest researchers and software engineers developing languages for parallel computations. It might also be of interest to both undergraduate and graduate computer science students because it does not require any special background. It can supplement material for courses devoted to programming languages and compilation techniques, especially for high-level parallel programming.

The difficulty of programming parallel computers in imperative languages is cited as one argument against them. [96] CDC demonstrates the Cyber 203, a predecessor of its Cyber 205 product. [97] Per Brinch Hansen describes remote procedure call (RPC) in paper on distributed processes, although he does not use that term. [253] Work begins at Indian Centre for Development of Advanced Computing (CDAC) on a transputer-based parallel machine called PARAM. [254] Thinking Machines Corporation introduces the DataVault mass storage subsystem, which uses up to 84 small disks to achieve high bandwidth and fault tolerance. [255] Inmos produces the first T800 floating-point transputer; Meiko and Parsytec begin marketing T800-based machines. The programming model targets development of families of related programs. One objective is to be able to compose programs which are near-optimal for given application instances and execution environments. Component-oriented development is motivated for parallel and distributed computations. The programming model is defined and described and illustrated with a simple example. The compilation process is briefly defined and described. Experience with one more complex application, a generalized fast multipole solver is sketched including performance data, some of which was surprising. [26] Structuring parallel and distributed programs. *Software Engineering Journal* 8(2), 73-82 (1993) CrossRef Google Scholar. 26. Structured Parallel Programming. by Michael McCool, Arch Robison, James Reinders. Main menu. [27] The Intel Parallel Computing Center at the University of Oregon has as its goal the development of an undergraduate parallel computing course to be offered each year in the Department of Computer and Information Science. However, the larger objective is to share our experiences and materials with others in the parallel computing community.