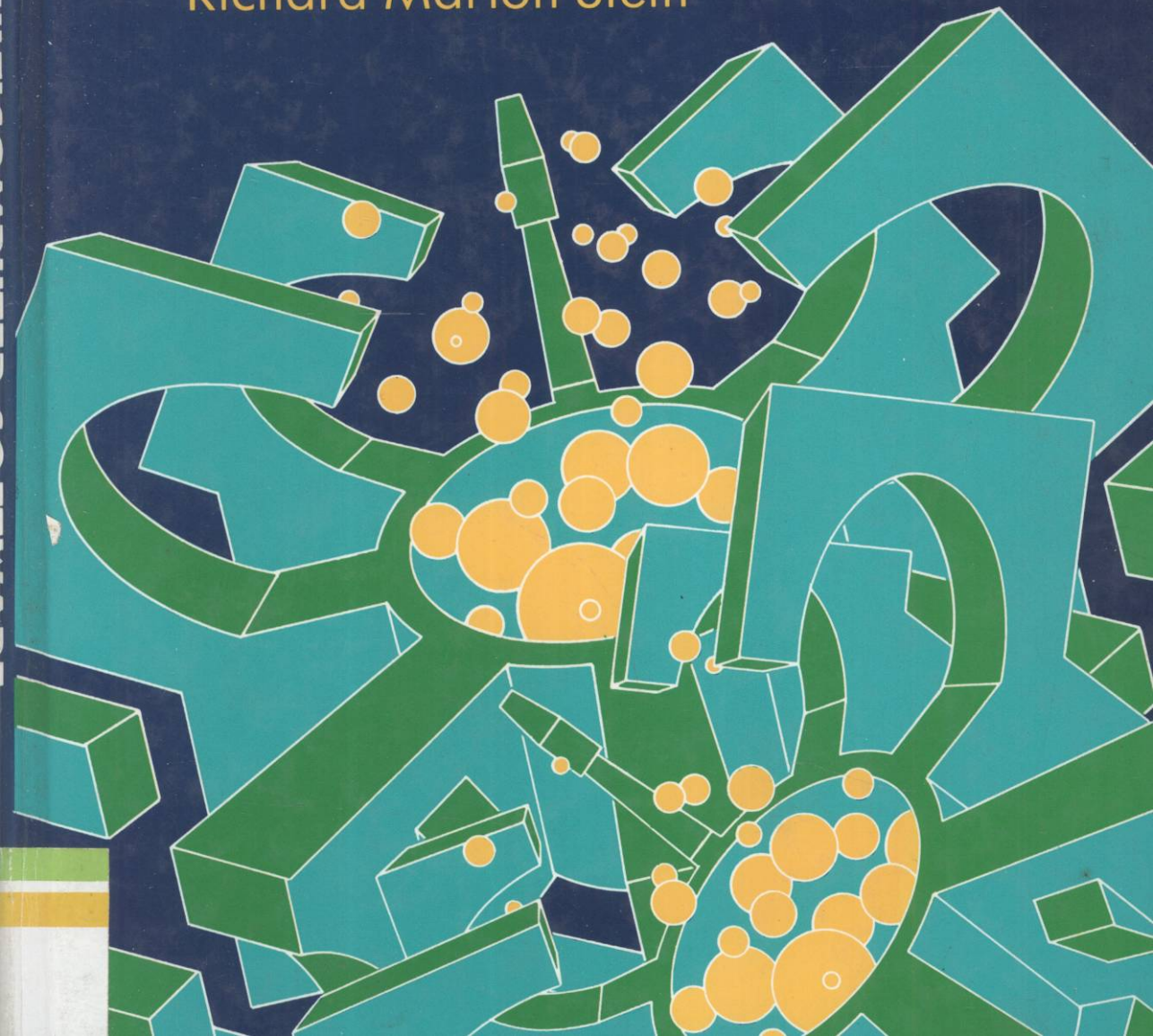
ELLIS HORWOOD SERIES IN
COMPUTERS AND THEIR APPLICATIONS

REAL-TIME MULTICOMPUTER SOFTWARE SYSTEMS

Richard Marlon Stein



REAL-TIME MULTICOMPUTER SOFTWARE SYSTEMS

RICHARD MARLON STEIN
Santa Clara, California, USA

ELLIS HORWOOD
NEW YORK LONDON TORONTO SYDNEY TOKYO SINGAPORE

First published in 1992 by
ELLIS HORWOOD LIMITED
Market Cross House, Cooper Street,
Chichester, West Sussex, PO19 1EB, England

A division of
Simon & Schuster International Group
A Paramount Communications Company

Printed and bound in Great Britain
by Bookcraft, Midsomer Norton

British Library Cataloguing in Publication Data

A Catalogue Record for this book is available from the British Library

ISBN 0–13–770777–0

Library of Congress Cataloging-in-Publication Data

Available from the publishers

REAL-TIME MULTICOMPUTER SOFTWARE SYSTEMS

ELLIS HORWOOD SERIES IN COMPUTERS AND THEIR APPLICATIONS

Series Editor: IAN CHIVERS, Senior Analyst, The Computer Centre, King's College, London, and formerly Senior Programmer and Analyst, Imperial College of Science and Technology, University of London

Abramsky, S. & Hankin, C.J.	**ABSTRACT INTERPRETATION OF DECLARATIVE LANGUAGES**
Alexander, H.	**FORMALLY-BASED TOOLS AND TECHNIQUES FOR HUMAN–COMPUTER DIALOGUES**
Anderson, J.	**MODEL-BASED COMPUTER VISION: A Synthesised Approach**
Atherton, R.	**STRUCTURED PROGRAMMING WITH COMAL**
Baeza-Yates, R.A.	**TEXT SEARCHING ALGORITHMS**
Bailey, R.	**FUNCTIONAL PROGRAMMING WITH HOPE**
Barrett, R., Ramsay, A. & Sloman, A.	**POP-11**
Beardon, C., Lumsden, D. & Holmes, G.	**NATURAL LANGUAGE AND COMPUTATIONAL LINGUISTICS: An Introduction**
Berztiss, A.	**PROGRAMMING WITH GENERATORS**
Bharath, R.	**COMPUTERS AND GRAPH THEORY: Representing Knowledge for Processing by Computers**
Bishop, P.	**FIFTH GENERATION COMPUTERS**
Brierley, B. & Kemble, I.	**COMPUTERS AS A TOOL IN LANGUAGE TEACHING**
Britton, C.	**THE DATABASE PROBLEM: A Practitioner's Guide**
Bullinger, H.-J. & Gunzenhauser, H.	**SOFTWARE ERGONOMICS**
Burns, A.	**NEW INFORMATION TECHNOLOGY**
Carberry, J.C.	**COBOL**
de Carlini, U. & Villano, U.	**TRANSPUTERS AND PARALLEL ARCHITECTURES**
Chivers, I.D. & Sleighthome, J.	**INTERACTIVE FORTRAN 77: A Hands on Approach 2nd Edition**
Clark, M.W.	**PC-PORTABLE FORTRAN**
Clark, M.W.	**TEX**
Cockshott, W.P.	**A COMPILER WRITER'S TOOLBOX: How to Implement Interactive Compilers for PCs with Turbo Pascal**
Cockshott, W.P.	**PS-ALGOL IMPLEMENTATIONS: Applications in Persistent Object-Oriented Programming**
Colomb, R.M.	**IMPLEMENTING PERSISTENT PROLOG: Large, Dynamic, Shared Procedures in Prolog**
Cooper, M.	**VISUAL OCCLUSION AND THE INTERPRETATION OF AMBIGUOUS PICTURES**
Cope, T	**COMPUTING USING BASIC**
Curth, M.A. & Edelmann, H.	**APL**
Dahlstrand, I.	**SOFTWARE PORTABILITY AND STANDARDS**
Dah Ming Chiu, & Sudama, R.	**NETWORK MONITORING EXPLAINED: Design and Application**
Dandamudi, S.P.	**HIERARCHICAL HYPERCUBE MULTICOMPUTER INTERCONNECTION NETWORKS**
Dongarra, J., Duff, I., Gaffney, P., & McKee, S.	**VECTOR AND PARALLEL COMPUTING**
Drop, R.	**WORKING WITH dBASE LANGUAGES**
Dunne, P.E.	**COMPUTABILITY THEORY: Concepts and Applications**
Eastlake, J.J.	**A STRUCTURED APPROACH TO COMPUTER STRATEGY**
Eisenbach, S.	**FUNCTIONAL PROGRAMMING**
Ellis, D.	**MEDICAL COMPUTING AND APPLICATIONS**
Ennals, J.R.	**ARTIFICIAL INTELLIGENCE**
Ennals, J.R., *et al.*	**INFORMATION TECHNOLOGY AND EDUCATION**
Fillipic, B.	**PROLOG USER'S HANDBOOK**
Ford, N.	**COMPUTER PROGRAMMING LANGUAGES**
Ford, N.J., Ford, J.M., Holman, D.F. & Woodroffe, M.R.	**COMPUTERS AND COMPUTER APPLICATIONS: An Introduction for the 1990s**
Ford, N. & Ford, J.	**INTRODUCING FORMAL METHODS: A Less Mathematical Approach**
Gray, P.M.D.	**LOGIC, ALGEBRA AND DATABASES**
Grill, E.	**RELATIONAL DATABASES**
Grune, D. & Jacobs, C.J.H.	**PARSING TECHNIQUES: A Practical Guide**
Guariso, G. & Werthner, H.	**ENVIRONMENTAL DECISION SUPPORT SYSTEMS**
Harland, D.M.	**CONCURRENCY AND PROGRAMMING LANGUAGES**
Harland, D.M.	**POLYMORPHIC PROGRAMMING LANGUAGES**
Harland, D.M.	**REKURSIV**
Henshall, J. & Shaw, S.	**OSI EXPLAINED, 2nd Edition**
Hepburn, P.H.	**FURTHER PROGRAMMING IN PROLOG**
Hepburn, P.H.	**PROGRAMMING IN MICRO-PROLOG MADE SIMPLE**
Hill, I.D. & Meek, B.L.	**PROGRAMMING LANGUAGE STANDARDISATION**
Hirschheim, R., Smithson, S. & Whitehouse, D.	**MICROCOMPUTERS AND THE HUMANITIES: Survey and Recommendations**
Hutchins, W.J.	**MACHINE TRANSLATION**
Hutchison, D.	**FUNDAMENTALS OF COMPUTER LOGIC**
Hutchison, D. & Silvester, P.	**COMPUTER LOGIC**
Johnstone, A.	**LATEX CONCISELY**
Kirkwood, J.	**HIGH PERFORMANCE RELATIONAL DATABASE DESIGN**

Series continued at back of book

ELLIS HORWOOD SERIES IN COMPUTERS AND THEIR APPLICATIONS

Series Editor: IAN CHIVERS, Senior Analyst, The Computer Centre, King's College, London, and formerly Senior Programmer and Analyst, Imperial College of Science and Technology, University of London

Koopman, P.	**STACK COMPUTERS**
Kenning, M.-M. & Kenning, M.J.	**COMPUTERS AND LANGUAGE LEARNING: Current Theory and Practice**
Koskimies, K. & Paakki, J.	**AUTOMATING LANGUAGE IMPLEMENTATION**
Koster, C.H.A.	**TOP-DOWN PROGRAMMING WITH ELAN**
Last, R.	**ARTIFICIAL INTELLIGENCE TECHNIQUES IN LANGUAGE LEARNING**
Lester, C.	**A PRACTICAL APPROACH TO DATA STRUCTURES**
Lucas, R.	**DATABASE APPLICATIONS USING PROLOG**
Lucas, A.	**DESKTOP PUBLISHING**
Maddix, F.	**HUMAN–COMPUTER INTERACTION: Theory and Practice**
Maddix, F. & Morgan, G.	**SYSTEMS SOFTWARE**
Matthews, J.J.	**FORTH**
Michalewicz, Z.	**STATISTICAL AND SCIENTIFIC DATABASES**
Millington, D.	**SYSTEMS ANALYSIS AND DESIGN FOR COMPUTER APPLICATIONS**
Moseley, L.G., Sharp, J.A. & Salenieks, P.	**PASCAL IN PRACTICE**
Moylan, P.	**ASSEMBLY LANGUAGE FOR ENGINEERS**
Narayanan, A. & Sharkey, N.E.	**AN INTRODUCTION TO LISP**
Parrington, N. & Roper, M.	**UNDERSTANDING SOFTWARE TESTING**
Paterson, A.	**OFFICE SYSTEMS**
Phillips, C. & Cornelius, B.J.	**COMPUTATIONAL NUMERICAL METHODS**
Rahtz, S.P.Q.	**INFORMATION TECHNOLOGY IN THE HUMANITIES**
Ramsden, E.	**MICROCOMPUTERS IN EDUCATION 2**
Rubin, T.	**USER INTERFACE DESIGN FOR COMPUTER SYSTEMS**
Rudd, A.S.	**PRACTICAL USAGE OF ISPF DIALOG MANAGER**
Rudd, A.S.	**PRACTICAL USAGE OF REXX**
Rudd, A.S.	**IMPLEMENTING PRACTICAL DB2 APPLICATIONS**
Rudd, A.S.	**IMPLEMENTING PRACTICAL DATABASE MANAGER APPLICATIONS**
Salomon, D.	**ASSEMBLERS AND LOADERS**
de Saram, H.	**PROGRAMMING IN MICRO-PROLOG**
Savic, D.	**OBJECT-ORIENTED PROGRAMMING WITH SMALLTALK/V**
Schirmer, C.	**PROGRAMMING IN C FOR UNIX**
Schofield, C.F.	**OPTIMIZING FORTRAN PROGRAMS**
Semmens, L. & Allen, P.	**Z FOR SOFTWARE ENGINEERS**
Sharp, J.A.	**DATA FLOW COMPUTING**
Sherif, M.A.	**DATABASE PROJECTS**
Smith, J.M. & Stutely, R.	**SGML**
Späth, H.	**CLUSTER DISSECTION AND ANALYSIS**
Stein, R.	**REAL-TIME MULTICOMPUTER SOFTWARE SYSTEMS**
Teunissen, W.J. & van den Bos, J.	**3D INTERACTIVE COMPUTER GRAPHICS**
Tizzard, K.	**C FOR PROFESSIONAL PROGRAMMERS, 2nd Edition**
Tsuji, T.	**OPTIMIZING SCHEMES FOR STRUCTURED PROGRAMMING LANGUAGE PROCESSORS**
Wexler, J.	**CONCURRENT PROGRAMMING IN OCCAM 2**
Whiddett, R.J.	**CONCURRENT PROGRAMMING FOR SOFTWARE ENGINEERS**
Whiddett, R.J., Berry, R.E., Blair, G.S., Hurley, P.N., Nicol, P.J. & Muir, S.J.	**UNIX**
Xu, Duan-Zheng	**COMPUTER ANALYSIS OF SEQUENTIAL MEDICAL TRIALS**
Yannakoudakis, E.J. & Hutton, P.J.	**SPEECH SYNTHESIS AND RECOGNITION SYSTEMS**
Zech, R.	**FORTH FOR THE PROFESSIONAL**

ELLIS HORWOOD SERIES IN COMPUTER COMMUNICATIONS AND NETWORKING

Series Editor: R.J. DEASINGTON, Principal Consultant, PA Consulting Group, Edinburgh, UK

Currie, W.S.	**LANS EXPLAINED**
Chiu, Dah Ming & Sudama, Ram	**NETWORK MONITORING EXPLAINED**
Deasington, R.J.	**A PRACTICAL GUIDE TO COMPUTER COMMUNICATIONS AND NETWORKING, 2nd Edition**
Deasington, R.J.	**X.25 EXPLAINED, 2nd Edition**
Henshall, J. & Shaw, S.	**OSI EXPLAINED, 2nd Edition**
Kauffels, F.-J.	**PRACTICAL LANS ANALYSED**
Kauffels, F.-J.	**PRACTICAL NETWORKS ANALYSED**
Kauffels, F.-J.	**UNDERSTANDING DATA COMMUNICATIONS**
Muftic, S.	**SECURITY MECHANISMS FOR COMPUTER NETWORKS**

Table of Contents

Dedication

To the loving memory of my fraternal and paternal grandparents: Edwin Morris and Beatrice Stalk, and David Nathaniel and Francis Hilda Stein.

To those who know what is not known.

Foreword

This text is a comfortable introduction to a complex and fascinating subject. The author's style is smooth and readable. His emphasis on software safety and the ethical use of multicomputer systems is well founded, and cannot be understated. This text affords a cognizant examination -- a snapshot in time really -- of an advancing technology from an active practitioner of multicomputer systems engineering. The technical descriptions and discussions are pertinent and colloquial. Anyone who is interested in becoming a practicing multicomputer software engineer with a bent for fast or real-time systems should peruse this text for a responsible and conscientious perspective of a continuously evolving scalable technology.

David L. Fielding
President, North American Transputer Users Group
Cornell University
Cornell, NY USA
January, 1992

Preface

As a vehicle for simulation and investigation, message-passing parallel computers -- multicomputer systems -- represent the most cost-effective problem solving tools yet invented. The scalable software built for these machines can provide insight and vision, but it can also imbue a nation with a strategic capability. Scalable multicomputer simulations can at once inform and enlighten, or destroy and pervert any purpose or conceivable idea created by mankind. Software applications are harder to construct in this realm of computation, and designing them requires unique skills and tools. The requisite engineering discipline must be acquired, practiced, and refined before effective and responsible use of a multicomputer platform can take place.

This text posits approaches to the solution of multicomputer software engineering problems. Among the questions explored by this text are: For what purpose have multicomputer systems been invented, and how are they applicable to the scientific and engineering problems which confront our society? What activities are necessary to prepare a proposal for a project based on a multicomputer system? What methods are available to estimate software engineering costs and schedule for a massively parallel simulation? How is software safety analysis used to prevent annoying or life-threatening mishaps from arising during simulation execution? How are real-time simulations engineered? What practices and skills are necessary to design and build a multicomputer simulation? How is a load balance realized? What methods are known for synchronizing a real-time multicomputer simulation? Answers to these questions are explored in the chapters of this book.

While the technical issues surrounding multicomputer software systems engineering are very important, they pale in comparison to the idea of a responsible multicomputer software engineering discipline. This state of mind is embraced and practiced by professional software engineers, and is expressed as a sincere concern for the societal implications of a finished edifice. Recognizing the likelihood of success or disaster hinges on the sensitive intuition of engineering judgement, a valuable characteristic of responsible practitioners. In contrast, the pure intellectual satisfaction of the engineering process is always visible, easier to appreciate and reward.

The end result of almost all engineering activities is a finished product which someone will use. The product may find a place in a home or a space shuttle. In either case, the consumer accepts that the product is safe, and will not harm the operator, or damage external equipment when used. The multicomputer software

engineering discipline espoused in this text attempts to heighten the engineer's awareness of the principal steps involved while engineering a real-time multicomputer simulation. A real-time computer system requires disciplined software engineering skills to correctly design and build. Not only must the software correctly function in an algorithmic sense, but the results produced by the software must be temporally correct. A rare blend of skills are required to construct predictably correct real-time simulation software. Real-time multicomputer simulations are especially demanding in this respect.

With few exceptions, a real-time multicomputer simulation represents the technical pinnacle of software engineering accomplishment. The temporal coordination of tens, hundreds, or thousands of separate processing entities can be accomplished and harnessed for the benefit of mankind. Like so many soldiers on parade in perfect cadence, or the soothing instrumental notes and melodies of an orchestral arrangement, their actions and operations must be precisely controlled, regulated, and directed for an effective result to develop. Alternatively, the same platform can be driven with criminal intent, producing chaos and catastrophe for an overtly evil or amoral purpose. Multicomputer systems are not toys; they are seriously powerful instruments that can at once possess the destructive potential of a strategic weapon, or the humanitarian capacity of penicillin. The multicomputer software engineer must commit to a moral obligation and a code of ethics that restricts their practice to the safety and betterment of society.

Technology is changing at an exponentially scalable rate. The complexity produced from this accelerated evolutionary process is far beyond any individual's or any organization's capacity to recognize, control, or accurately contemplate. The societal and global ramifications resulting from this sustained and compounded introduction are profound. If technology obeyed Darwinian natural selection and survival of the fittest, the overpopulation of so many digital electronic devices would have died off by now, just like overpopulation of a species eventually exhausts a habitat's food supply, and equilibrium is restored through death by starvation. But technology is exempt from Darwinian law.

For the promulgation of software engineering safety awareness, and to promote recognition of the implicit strategic nature of multicomputer systems, I have attempted to combine and communicate my collectively acquired industrial, professional, and independent experience with both real-time simulation and multicomputer systems to you, the reader. Several months previous, I relished the thought of the technical challenge to describe a rapidly emerging field of research and investigation. But during the course of my background investigation, information assemblage, and revisions to the text, I developed a disquieting notion about this arena. This book is a distillation of my experience using multicomputer systems (chiefly Inmos transputer[1] boards which I purchased way back in 1985), and a recognition of the

---

[1] The transputer is a very practical device and affords an inexpensive avenue to start multicomputer investigations. My sincerest effort to avoid discussion of specific hardware products is represented here; I do not perceive this enthusiasm as an endorsement.

important and increasingly visible role they serve.

A democratic society has many rules and structures, such as the laws established by the United States Bill of Rights, to protect the citizenry and prevent societal breakdown. Technology does not obey the Constitution, but the ideas and freedom of expression conveyed through technology are principally sponsored by it. As a vehicle for expression, multicomputer technology wields immense power, and can be exploited for any purpose. With no legal or physical controls to regulate the advance of an unchecked specie, what will be the result? I do not know this answer, but it is very easy to assume to the worst. I hope that this text will serve as a baseline for the emerging practitioners of real-time multicomputer software systems engineering to consider the consequences of their work before undertaking design with impunity. I importune you to do so with caution and care.[2]

This text contains two principal sections. Part 1, Concepts and Practices, presents background material on multicomputer systems, project planning and preparation, software metrics, an introduction to real-time computer systems, and software safety. This collection of 5 chapters provides a foundation for software engineering practice and discipline. Part 2, Multicomputer Methods, supplies practical design information for scalable software engineering activities. Chapters that discuss software design, load balancing, and synchronization for multicomputer simulations are supplied. Each chapter is largely self-contained and decoupled from the others. They may be read in any order, depending on the individual's expertise and strength. Some background in software engineering is necessary and assumed. This book is not intended for the novice or freshman. However, it is targeted at the practicing professional software engineer with an eye toward real-world situations and experience.

Chapter 1 discusses multicomputer systems as a potential mechanism for yielding solutions to complex scientific, engineering, and bio-medical problems identified by the United States Federal High Performance Computing Program. Other nations, such as Japan, or the European Economic Community, have enacted similar programs to ensure a measure of technologically-derived economic security in the future. A brief discussion is given as to why multicomputers can address complex simulation issues, and are ideally suited to deliver cost-effective scalable solutions.

Chapter 2 discusses project planning and preparation. If one perceives the need to create a scalable multicomputer simulation solution, how is the project justified, a plan prepared and supported? The chapter is based on the presentation and discussion of a sample proposal written to the format solicited by the United States' Defense Advanced Research Projects Agency (DARPA).

Chapter 3 discusses software metrics. The COnstructive COst MOdel (COCOMO) developed by Barry W. Boehm is used as a platform to illustrate how a software project is estimated to quantify cost and schedule. Software engineering is more of an art than a science. Many factors affect the software lifecycle, the stages of planning, development, and maintenance that describe the useful lifetime of a software system from creation through retirement. Organizing an accurate estimate

---

[2] The author is painfully aware of this super-hypocritical argument -- a dilemma which scientists and engineers often confront.

of the engineering costs is far harder for a software product than, say, building an automobile or a toaster oven.

Two varieties of software are known for multicomputer simulations: data parallel and control parallel. On one hand, data parallel simulations are simply replicated instantiations of one (typically) simple process or calculation. This is true for many numerically intensive computations, like weather forecasting, finite element structure problems, or image processing; the same computation is conducted for each data element. In contrast, a control parallel simulation relies on a highly articulated process structure where many unique processes are combined to synthesize a simulated representation of a natural or man-made phenomenon. For each control parallel process, the argument is made to justify a separate engineering and cost estimate for lifecycle purposes. Control parallel simulations require multiple applications of software metric estimation to accurately cost.

Chapter 4 presents a capsulized introduction to real-time system simulation. Special attention is given to the predictability aspects of real-time systems. A real-time system must produce algorithmically or algebraically correct results which are also temporally correct. A real-time simulation is often a critical component of a larger system, such as an airplane, satellite, or an automobile. If the real-time system is not predictable under all circumstances, serious consequences can result in the event of system failure. The chapter outlines essential information on executive control structure, engineering tool requirements which are often found in real-time environments, and other common properties of this engineering discipline. The discussion is intended to impress an understanding and appreciation of real-time systems for their complexity, and give insight into some internal aspects of their implementation.

Software safety is the topic of Chapter 5. When good software goes bad, terrible things can happen which should not ever occur. Four examples of software failure are given. They have been reprinted from the pages of widely published newspapers, magazines, and journals. The strong and inseparable computer dependency seen in Western culture weaves at once a life-threatening and life-sustaining vein through our lives. Building software for use by others carries an implicit trust and ubiquity that is shattered each time the telephone does not work, or the ATM will not dispense money.

This chapter provides an overview of current software safety techniques and practice. Safety analysis techniques like Petri nets and software fault trees are introduced to show what kinds of methods are available to detect software faults before a failure occurs. While software safety has a terrific impact from the user's perspective, the engineering and design aspects are far more difficult to manage in a cost-effective way. Over 60% of all errors introduced into software arise from poor requirements definition or ambiguity. A discussion of formal specification methods is presented which argues for their adoption. If the requirement is poorly understood, it is likely that the final software implementation may also generate spurious functionality which can lead to disaster.

In Part 2, beginning with Chapter 6, multicomputer software design practice is discussed with an eye toward important design and implementation issues. To build multicomputer software, a different state of mind is necessary, one which is alien to

sequential software engineering practices. A method of software design is presented which is derived from a process structure graph analysis representation of simulation. This is a familiar concept to anyone who has studied Hoare's Communicating Sequential Processes (CSP), or experienced the Occam model of programming on the Inmos transputer. The text explains how to construct this logically concurrent representation, simulate it within a comfortable environment to verify determinism, and then transform it to a physically concurrent multicomputer system. An example problem is provided which illustrates the concurrent thought process required to successfully carry out these operations.

A multicomputer runs most efficiently when all computation elements are equally loaded with data. The notion of load balance is introduced and examined in Chapter 7 along with a discussion of several techniques for realizing one. A partial taxonomy is created of the known and widely practiced load balancing methods. One technique is posited for real-time multicomputer simulations, while the majority are efficacious for purely static or quasi-dynamic problems.

The issue of temporal regulation and synchronization of the multicomputer simulation is the focus of Chapter 8. The distributed nature of multicomputer architecture poses unique problems for the temporal coordination of ten, one hundred, or several thousand computation elements, since each has it own clock source. A discussion of a synchronization algorithm is given that is applicable to homogenous multicomputer platforms.

Chapter 9 supplies a brief discussion of emerging trends and advanced technology likely to replace the existing generation of multicomputers. The parallel random access machine (PRAM) may well become the first general purpose parallel processing architecture, where the obstacles of load balance, topology, and data decomposition disappear from the multicomputer simulation scene.

Acknowledgments

I give my sincerest thanks to the staff at Ellis Horwood for their patience, prodding, and encouragement during the preparation of this manuscript. I am also grateful to Mr. Philip Presser, a long-time friend, mentor, and associate. I have been very fortunate to enjoy, experience, and absorb his thoughtful and oracle-like wisdom, patience, discipline, and insightful recognition of engineering methodology. Finally, and most importantly, I acknowledge the love and support from my family who have cared, nurtured, and raised their son, brother, nephew, and cousin with endearment and kindness.