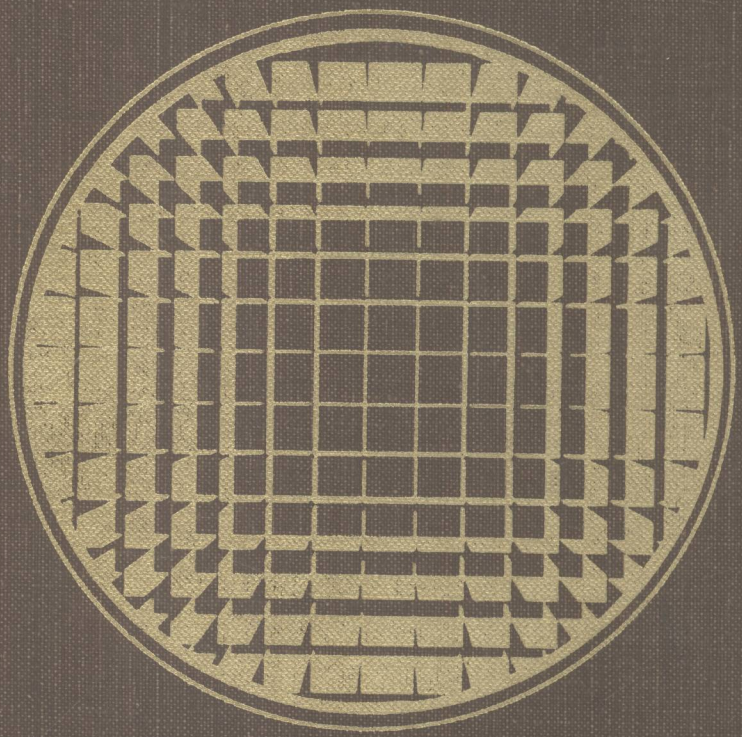
ARTIFICIAL INTELLIGENCE AND SIMULATION

Edited by
Ranjeet J. Uttamsingh
A. Martin Wildberger



Simulation Series
Volume 23
Number 4

SCS A PUBLICATION OF THE SOCIETY FOR COMPUTER SIMULATION

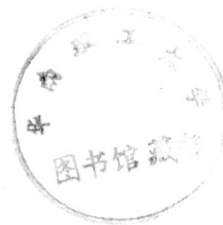
Artificial Intelligence and Simulation

Proceedings of the Simulation MultiConference on
Artificial Intelligence and Simulation
1-5 April 1991
New Orleans, Louisiana

Edited by
Ranjeet J. Uttamsingh
Synetics

A. Martin Wildberger
General Physics

Simulation Series
Volume 23
Number 4



Sponsored by
The Society for Computer Simulation (SCS)

Additional copies of the Proceedings are available from:

The Society for Computer Simulation
P. O. Box 17900
San Diego, CA 92177 U.S.A.

ISBN 0-911801-88-X

PRINTED IN THE UNITED STATES

Titles in the *Simulation Series*

Vol. 1 No. 1	*Mathematical Models of Public Systems*	George A. Bekey, Editor January 1971
Vol. 1 No. 2	*Systems and Simulation in the Service of Society*	David D. Sworder, Editor July 1971
Vol. 2 No. 1	*The Mathematics of Large-Scale Simulation*	Paul Brock, Editor June 1972
Vol. 2 No. 2	*Recent Developments in Urban Gaming*	Philip D. Patterson, Editor December 1972
Vol. 3 No. 1	*Computer Simulation in Design Applications*	Sail Ashour and Marvin M. Johnson, Editors June 1973
Vol. 3 No. 2	*Simulation Systems for Manufacturing Industries*	Marvin M. Johnson and Said Ashour, Editors December 1973
Vol. 4 No. 1	*Annotated Bibliographies of Simulation*	Tuncer I. Oren, Editor June 1974
Vol. 4 No. 2	*Spanning the Applications of Simulation*	Paul Brock, Editor December 1974
Vol. 5 No. 1	*New Directions in the Analysis of Ecological Systems: Part 1*	George S. Innis, Editor June 1975
Vol. 5 No. 2	*New Directions in the Analysis of Ecological Systems: Part 2*	George S. Innis, Editor December 1975
Vol. 6 No. 1	*Toward Real-Time Simulation (Languages, Models and Systems), Part 1*	Roy E. Crosbie and John L. Hay, Editors June 1976
Vol. 6 No. 2	*Toward Real-Time Simulation (Languages, Models and Systems), Part 2*	Roy E. Crosbie and John L. Hay, Editors December 1976
Vol. 7 No. 1	*An Overview of Simulation in Highway Transportation: Part 1*	James E. Bernard, Editor June 1977
Vol. 7 No. 2	*An Overview of Simulation in Highway Transportation: Part 2*	James E. Bernard, Editor December 1977
Vol. 8 No. 1	*Simulation in Energy Systems: Part 1*	Kenneth E. F. Watt, Editor June 1978
Vol. 8 No. 2	*Simulation of Energy Systems: Part 2*	Kenneth E. F. Watt, Editor December 1978
Vol. 9 No. 1	*Simulation in Business Planning and Decision Making*	Thomas H. Naylor, Editor July 1981
Vol. 9 No. 2	*Simulating the Environmental Impact of a Large Hydroelectric Project*	Normand Therien, Editor July 1981
Vol. 10 No. 1	*Survey of the Apllication of Simulation to Health Care*	Stephen D. Roberts and William L. England, Editors December 1981
Vol. 10 No. 2	*Computer Modeling and Simulation: Principles of Good Practice*	John McLeod, Editor June 1982
Vol. 11 No. 1	*Peripheral Array Processors*	Walter J. Karplus, Editor October 1982
Vol. 11 No. 2	*Computer Simulation in Emergency Planning*	John M. Carroll, Editor January 1983
Vol. 12 No. 1	*Lumped-Parameter Models of Hydrocarbon Reservoirs*	Ellis A. Monash, Editor March 1983
Vol. 12 No. 2	*Computer Models for Production and Inventory Control*	Haluk Bekiroglu, Editor January 1984
Vol. 13 No. 1	*Aerospace Simulation*	Monte Ung, Editor February 1984
Vol. 13 No. 2	*Simulation in Strongly Typed Languages: ADA, PASCAL, SIMULA...*	Ray Bryant and Brian W. Unger, Editors February 1984
Vol. 14 No. 1	*All About Simulators, 1984*	Vince Amico and A. Ben Clymer, Editors April 1984
Vol. 14 No. 2	*Peripheral Array Processors*	Walter J. Karplus, Editor October 1984
Vol. 15 No. 1	*Emergency Planning*	John M. Carroll, Editor January 1985
Vol. 15 No. 2	*Distributed Simulation 1985*	Paul F. Reynolds, Editor January 1985
Vol. 16 No. 1	*Simulators*	John S. Gardenier, Editor March 1985
Vol. 16 No. 2	*Aerospace Simulation II*	Monte Ung, Editor January 1986
Vol. 17 No. 1	*Intelligent Simulation Environments*	Paul A. Luker and Heimo H. Adelsberger, Editors January 1986
Vol. 17 No. 2	*Simulators III*	Bruce T. Fairchild, Editor March 1986
Vol. 18 No. 1	*AI Applied to Simulation*	E. J. H. Kerckhoffs, G. C. Vansteenkiste and B. P. Zeigler, Editors February 1986
Vol. 18 No. 2	*Multiprocessors and Array Processors*	Walter J. Karplus, Editor April 1987
Vol. 18 No. 3	*Simulation and AI*	Paul Luker and Graham Birtwistle, Editors July 1987
Vol. 18 No. 4	*Simulators IV*	Bruce T. Fairchild, Editor October 1987
Vol. 19 No. 1	*Methodology and Validation*	Osman Balci, Editor January 1988
Vol. 19 No. 2	*Aerospace Simulation III*	Monte Ung, Editor April 1988
Vol. 19 No. 3	*Distributed Simulation, 1988*	Brian Unger and David Jefferson, Editors July 1988
Vol. 19 No. 4	*Simulators V*	A. Ben Clymer and G. Vince Amico, Editors October 1988
Vol. 20 No. 1	*AI Papers, 1988*	Ranjeet J. Uttamsingh, Editor January 1989
Vol. 20 No. 2	*Simulation in Emergency Management and Technology*	Jim Sullivan and Ross T. Newkirk, Editors April 1988
Vol. 20 No. 3	*Simulation and AI, 1989*	Wade Webster, Editor April 1988
Vol. 20 No. 4	*Advances in AI and Simulation*	Ranjeet Uttamsingh and A. Martin Wildberger, Editors March 1989
Vol. 21 No. 1	*Multiprocessors and Array Processors V*	Walter J. Karplus, Editor March 1989
Vol. 21 No. 2	*Distributed Simulation, 1989*	Brian Unger and Richard Fujimoto, Editors March 1989
Vol. 21 No. 3	*Simulators VI*	Ariel Sharon and Mohammad R. Fakory, Editors March 1989
Vol. 21 No. 4	*Simulation in Business and Management*	Sal Belardo and Jay Weinroth, Editors January 1990
Vol. 22 No. 1	*Distributed Simulation*	David Nicol, Editor January 1990
Vol. 22 No. 2	*Simulators VII*	Ariel Sharon and M. R. Fakory, Editors April 1990
Vol. 22 No. 3	*AI and Simulation: Theory and Applications*	Wade Webster and Ranjeet Uttamsingh, Editors April 1990
Vol. 22 No. 4	*Simulation in Energy Systems*	W. Frisch, B. Cordier, and A. Höld, Editors October 1990
Vol. 23 No. 1	*Advances in Parallel and Distributed Simulation*	Vijay Madisetti, David Nicol, and Richard Fujimoto, Editors January 1991
Vol. 23 No. 2	*Simulation in Business and Management II*	Jay Weinroth and Joe Hilber, Editors January 1991
Vol. 23 No. 3	*Object-Oriented Simulation, 1991*	Raimund K. Ege, Editor January 1991
Vol. 23 No. 4	*Artificial Intelligence and Simulation*	Ranjeet J. Uttamsingh and A. Martin Wildberger, Editors April 1991

PREFACE

The interdisciplinary field, artificial intelligence and simulation, has a complex and intricate terrain that includes both a wide variety of applications and many deep theoretical issues. Most of these theoretical issues have been inherited from its two parent disciplines, but the unique technology produced by their combination has raised questions, and possible answers, of its own.

No one conference, or one volume of papers, could be expected to address all of these applications and issues. The papers in this volume provide more than a representative sample. They include most of the papers presented at the 1991 Artificial Intelligence and Simulation Conference, held in New Orleans, Louisiana, U.S.A., April 1-5. The applications described range over science, engineering and industry, with emphasis on manufacturing and agriculture. A number of hardware and software tools are also presented, both general purpose, and specialized to particular applications. Theoretical and practical issues in simulation modeling are discussed in the context of artificial intelligence. The representative of knowledge is of major concern in all AI work. The emphasis in this volume is on its relations to the modeling of data and the interconnections between knowledge bases and data bases.

The tutorials presented at the conference provided a balanced introduction to both theory and practice in some of the most active research and development areas within AI and simulation. Since the practical side was presented mainly through demonstrations, the theoretical aspect is more apparent in the summaries included here.

All-in-all, this volume is a fitting tribute to Dr. Ranjeet J. Uttamsingh, whose conception it was, and to whose lasting memory it is dedicated.

A. Martin Wildberger
Editor and Deputy Program Chairman
General Physics Corporation

CONTENTS

CONTENTS

ARTIFICIAL INTELLIGENCE AND SIMULATION

SOLVING SYNCHRONIZATION PROBLEMS IN RAPID SIMULATION OF A MANUFACTURING SHOP-FLOOR

John C. Peck, Roy P. Pargas, Prashant K. Khambekar, Satish K. Dharmaraj
Department of Computer Science
Clemson University
Clemson, South Carolina 29634-1906

ABSTRACT

This paper describes a solution to synchronization problems which arise in rapid near-term simulation of manufacturing. The purpose of the simulator is to provide management with a tool for use in creating and evaluating schedules as part of production planning. In order to run as accurate a simulation as possible, data describing the current status of the manufacturing plant is continually collected by a real-time shop-floor control system. This data is used to start the simulator in an initial state. Performance of the simulation with quick run-time strongly suggest a parallel implementation; T-800 INMOS transputers are used with a PC as a front-end processor. Unlike a single processor simulation, a parallel simulation gives rise to synchronization problems, deadlock and starvation. These problems are analyzed and solutions which enable an accurate, conservative simulation are presented.

INTRODUCTION

Apparel plants operate on a larger scale than manufacturing plants in many other industries. A typical apparel plant may have more than 400 direct labor (incentive) employees with perhaps 500 machines (some large plants have 1500+ employees and a correspondingly larger number of machines). Both employees and machines are capable of performing multiple operations, but only a small percentage of the total operations are required to manufacture a particular garment. Active on the shop-floor at any one time might be 100 or more production lots (orders) each consisting of perhaps 200 bundles of garment parts, each bundle consisting of five or more subassemblies, each subassembly requiring one to twenty operations. Production lots are possibly of different styles, meaning the operations and sequencing of operations are different. Bundle subassemblies flow through the manufacturing process in parallel and join (merge), as operations are completed, to produce finished garments. The correct matching of subassemblies from parent bundles is important since color shading variations will be noticeable otherwise. Production lots have due dates by which they must be shipped out. Since employees and machines have multiple, but limited, skills and capabilities, balancing these resources against required work, that is, developing an operational plan, is a major problem.

The ultimate goal in production scheduling is to improve the operation of a plant. The primary question is "How does one know if a change in an operational plan produces a better or worse schedule?" A near-term simulator of shop-floor operations of an apparel manufacturing plans has been developed to enable the manager to evaluate such a plan. High-level performance information in the form of graphics is continuously displayed. The manager can evaluate the plan and in case the performance metrics are not satisfactory, rerun the simulation with a new plan, all in a matter of minutes. Details of the simulation can be found in Pargas *et al.* 1990.

Rather than using statistical distributions to estimate crucial input information (which many simulations do) the near-term simulator uses a real-time system (Foxfire 1989) to *measure*, in advance, information such as the rate of arrival of goods to be processed, processing rates of different machines and efficiencies of employees. Thus the question of accuracy of estimates does not arise.

Since production goals, such as keeping inventory low or machine utilization high, vary from plant to plant, or indeed, from time to time in the same plant, and management styles vary from person to person, a large number of performance metrics are made available. A manager may select and observe any of the metric graphs on the front-end and optimize performance based on them. The sixty or so metrics fall into different classes: Cost, Production, Efficiency, Lateness, Labor utilization and Waiting Time (Peck *et al.* 1991).

The complexity of the application, large input data, large number of metrics and quick simulation require-

ment necessitate a parallel simulation. The implementation is done on a multiprocessor system built by Computer Systems Architects of Provo, Utah. It consists of 17 INMOS T-800 Transputers, each with 2 Mbytes of memory. The Transputers integrate a 32-bit processor, a 64-bit floating point unit and 4 Kbytes of static RAM. One of the Transputers (called the root) is connected to a standard PC front-end whereas the other 17 Transputers are networked among themselves and connected to the root.

A major design decision was to break up the major functions of the simulation among the processors available. The primary functions are input and output, execution of the simulation itself and collection and processing of performance metric data. A natural assignment of functions to processors was to assign all input/output functions (that is, inputting the data and displaying graphs) to the front-end PC, the simulation to the Transputer nodes, and the collection and processing of metric data to the root Transputer. The design is clean: the user is unaware of the Transputers, the program on the PC is independent of the number of Transputers and the simulation program on the Transputers is independent of the front-end. The Transputer nodes execute a distributed event-driven simulation and exploit the natural parallelism on the shop-floor. Distributed simulation gives rise to synchronization problems, deadlock and starvation. The solution to these problems is presented in this paper.

THE APPAREL PLANT SHOP-FLOOR

Bundles of garment parts are sewn according to their respective style (operation) flow graphs. Figure 1 shows two style flow graphs. The circles represent operations to be performed, for example, operation 17 is "hem placket" and operation 41 is "top-stitch pocket". On the shop-floor each operation has a buffer and one or more workstations are configured to perform that operation. When employees are assigned to a workstation, a bundle is extracted from the workstation's buffer, the operation is performed and then the bundle is sent to the buffer corresponding to the next operation in the style flow graph.

Bundles carry with them the expected processing time for each operation in the style graph. This time, called Standard Allowed Minutes (SAMS), is obtained from engineering time-motion studies. Employees differ in their efficiency of performing operations. Efficiency is defined as the ratio of SAMS to actual minutes and can be above or below 100%. Hence, although the SAMS value of an operation is known in advance, the actual service time depends on which employee picks up the bundle, and is only available when the processing of the bundle is simulated.

There are three types of events: *bundle events* such as the completion of an operation or the arrival of a production lot, *employee events* such as an employee signing in for work or an employee being reassigned to another workstation, and *workstation events* such as a workstation being reconfigured for a different type operation. The most common event is the completion of an operation.

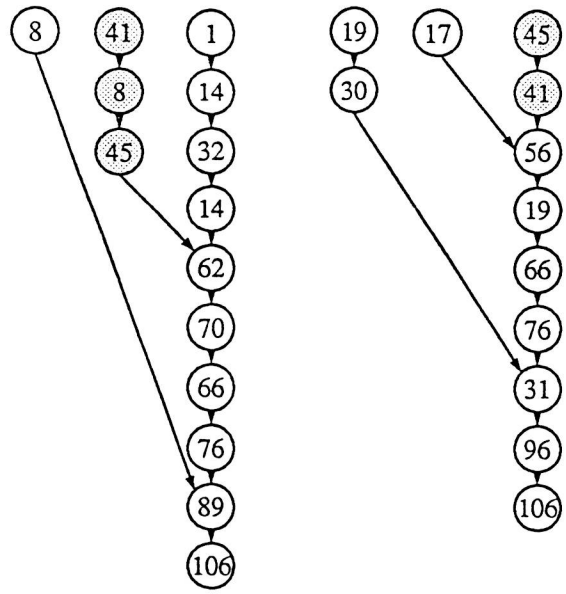
PARALLEL SIMULATION

For the simulation, operations are mapped to processors in the transputer system. A processor may simulate one or more operations. Each operation's buffer is represented by a queue and bundles are picked from the queue in order of priority and in case of equal priority, in order of time of arrival.

For correct simulation workstations configured for the same operation (and so drawing from the same buffer) must coordinate. This is most easily achieved by requiring that all workstations configured for the same operation be simulated in the same processor. Activities of all employees assigned to such workstations are also simulated in the same processor. Thus each processor has its independent queue of events and performs event-driven simulation synchronized by a logical clock. The logical clock is defined as the time of the last event processed by the processor.

A majority of researchers in distributed simulation use a logical process for each real process (e.g., workstation) in the application. Each logical process communicates with other logical processes. However as has been shown by Nicol (Nicol 1988), if the number of logical processes is greater than the number of physical processors, an overhead of context switching is experienced and efficiency is reduced. Hence in this simulation buffers, workstations and employees in one processor are handled by a single process. Since the event queue is stored in ascending time order all events within (and resultant messages from) a processor are in logical time sequence.

Transputer processors communicate with each other using messages. All messages are time-stamped, that is, they carry the logical clock time at which the sending processor issued the message. The majority of interprocessor messages are bundle messages which indicate the transfer of a bundle from one processor to a buffer in another processor for further processing according to the bundle's style flow graph. The sending processor is called the predecessor and the receiving processor



Style 1 Style 2

Figure 1: Typical Style Flow Graphs

is called the successor.

Although each style flow graph is feed-forward and has no cycles, the intersection of styles gives rise to cycles. As an example, in style 1 of Figure 1 operation 41 precedes operation 8 which precedes operation 45. In style 2 operation 45 precedes operation 41. Thus 41, 8 and 45 form a cycle of operations in the merged style graph. If such cycles were small, that is, composed of less than 8 operations (as compared to the total number of operations being in the range of 100), then operations which form cycles could all be mapped to the same processor, thus giving rise to a cycle-free interconnection among *processors*. Indeed this was the initial expectation (Khambekar and Dharmaraj 1990). However, examining the style flow graphs reveals that this is not feasible. Figure 2 shows the result of intersecting 21 style flow graphs. All operations in the big oval form a single cycle; thus 33 out of the 42 operations are involved in a cycle. Hence, cycles between processors cannot be avoided unless 75% or more of the simulation is performed in a single processor. As a means of distributing the workload, operations are mapped to processors such that the processors have equal number of operations.

SYNCHRONIZATION PROBLEMS

Since each processor has an independent event queue and logical clock, it is possible that the logical clock of a predecessor is greater than, equal to or less than the logical clock of the successor. If the logical clock of the predecessor is less than that of the successor, a message sent by the predecessor will arrive in the successor's simulated *past* and either compromise the accuracy of the simulation or incur additional overhead to undo earlier events. Accordingly, there are two approaches: *Optimistic* approaches (Jefferson 1985; Sokol *et al.* 1988) allow messages to arrive in the simulated past and in case such a message arrives, a roll-back of the simulation is initiated back to the time in the message. *Conservative* approaches prevent events from executing out of time sequence. Optimistic approaches involve keeping the state of the simulation at every step and sending anti-messages to cancel the effect of earlier non-chronological actions. Since keeping the states is memory-intensive and the amount of memory in each transputer is limited (compared to the complexity of the application) and virtual memory is not available, a conservative simulation approach was selected. Therefore, before a processor can select the next event, it must know that none of its predecessor processors will send a message with a lesser time-stamp. This restriction can cause the processor to wait for messages till the message time-stamps are equal to or greater than the time of the next event.

Two synchronization problems can occur with this conservative approach: *deadlock* and *starvation (no-*

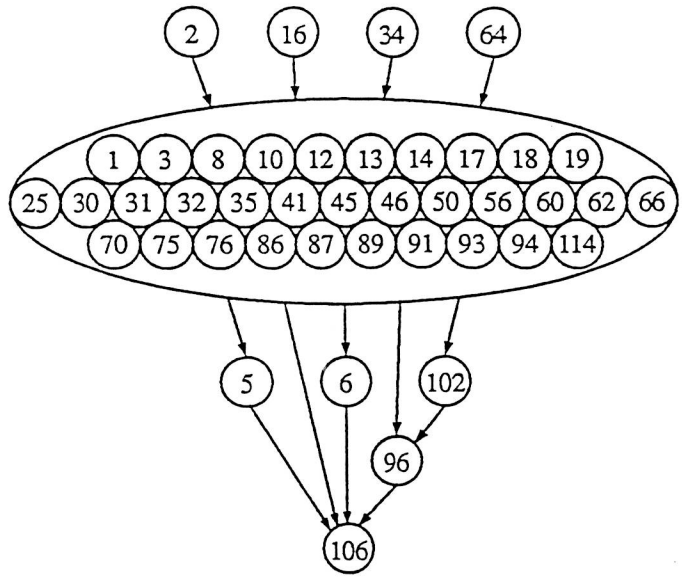


Figure 2: Composite Graph from Intersection of 21 Styles.

progress). Deadlock is the state in which a collection of processors cannot progress because they are cyclically waiting for input from one another. *Starvation* is the state in which some processors cannot progress because they are waiting for input from other processors (although the latter could be progressing normally). Deadlock arises if cycles are present in the flow graphs of the processors sending messages or if virtual "waiting" cycles may form due to finite buffers. Starvation is possible even if there are no cycles and buffers are infinite.

There are a number of approaches to solve the synchronization problems, many of which are described in an early survey by Misra (Misra 1986). In case of deadlock, Chandy and Misra (Chandy and Misra 1981) and Chandy, *et al.* (Chandy *et al.* 1983) suggest deadlock detection approaches, which let deadlock occur, detect it and then break it using collective global information. The former uses a central controller whereas the latter sends out queries for deadlock detection. However, several studies (Fujimoto 1988, Reed *et al.* 1988, Reed and Malony 88) indicate limited speed-up. These approaches do not address the starvation case.

The Null Message approach was developed independently by Bryant (Bryant 1977), Chandy-Misra (Chandy and Misra 1979a; Chandy and Misra 1979b) and Peacock *et al.* (Peacock *et al.* 1979a, Peacock *et al* 1979b). Messages which contain the current simulation time are sent from each processor to its successors so that the successors may proceed. However, the *look-ahead* provided by these null messages is limited (Peacock *et al.* 1979b, Nicol 1988) and studies show that choking of the simulation due to excessive null messages is likely (Fujimoto 1988, Reed *et al.* 1988, Reed and Maloney 88). (However, the studies were done on shared memory machines so the application of their results to distributed machines is subject to question.)

Peacock, *et al.* (Peacock *et al.* 1979a) also suggest a Blocking Table approach in which a processor blocks when the time of any of its predecessors is less than its own. However this approach involves one-to-many broadcasts to keep the tables current. Bezivin and Imbert (Bezivin and Imbert 1982) proposed a monitor approach and Christopher *et al.* (Christopher *et al.* 1982) a transaction based approach. Both involve centralized controllers which can be bottlenecks in a distributed system.

In the Appointment approach (Nicol and Reynolds 1984), processors demand appointment times from their predecessors and cannot simulate beyond the smallest of these times. The demand-driven Appointment approach has the advantage that it avoids unnecessary messages. Nicol (Nicol 1988) develops appointments further and provides greater look-ahead by doing application-dependent calculations. In case of stochastic networks, the service time is sampled probabilistically and it can be sampled even before the job arrives at the proces-

sor. Thus one can have a *future list* of events and higher appointment bounds, but its applicability is limited.

THE METHOD USED: APPOINTMENTS

In the near-term simulation described in this paper cycles between processors cannot be avoided. Starvation is also a concern since the output metrics cannot be obtained speedily if starvation occurs. Deadlock detection approaches to solving synchronization problems were deemed infeasible since they cannot prevent starvation. Null Messages were ruled out because of their limited look-ahead and probable choking. Monitor and transaction approaches were ruled out because of the possibility of bottlenecks.

Nicol and Reynolds' appointment approach seemed most promising and was adapted and enhanced for the near-term simulation. When a processor cannot make progress because of the unavailability of an event message from a predecessor, i sends a request message to that processor. The predecessor processor responds with an appointment time. Based upon the received appointment time the impeded processor may be able to process some buffered events and make progress.

The appointment time sent is not merely the current simulated time but rather is a time obtained by examining the buffers and the bundles in the processor. This application-dependent appointment time (similar to Nicol (Nicol 1988)) provides a good look-ahead.

When a request is sent by processor P_j to a predecessor processor P_i the next event time, t, to which P_j desires to advance is sent in the request. P_i, thus, only has to check its buffers for events which will complete before t. This saves P_i from having to check its entire buffer list. If an appropriate event destined for P_j is found by P_i then the time of that event is sent as an appointment. If there is no event which can complete before t, then P_i replies with a "you can proceed" message. If P_i has no pending event destined for P_j it sends a "failed" message.

P_j sends requests to those of its predecessors from whom it has no buffered messages. If the received messages are "you can proceed" messages or appointments, P_j can advance its clock to the smallest received appointment. If all the received messages are "you can proceed" then P_j can advance its clock to t. Upon receiving any "failed" messages, P_j waits for a small amount of time and restarts sending requests.

With this approach both deadlock and starvation are prevented. In terms of the design characteristics outlined by Reynolds (Reynolds 1988), this method is accurate, non-aggressive, has no risk and employs knowledge acquisition and knowledge embedding. It is non-aggressive because events are always processed in increasing time order and not on conditional knowledge. As a result the method is accurate; events *are* ultimately processed in increasing time order. There is no risk; events based on conditional knowledge are not propagated because there are no such events. Knowledge acquisition is used since the processors initiate requests for knowledge from other processors. Knowledge embedding is utilized because knowledge about the applications behavioral attributes is embedded in the simulation.

CONCLUSION

The complexity and performance requirements of the near-term simulator strongly suggest a parallel simulation. Unlike a single processor simulation, a parallel simulation gives rise to synchronization problems. These problems have been analyzed and an approach which enables an accurate, conservative simulation has been presented. The simulator has been completely implemented and real data obtained from an apparel plant is currently being processed. A pilot installation in a production apparel plant is expected to begin in summer 1991.

Future research will investigate algorithms for dynamically balancing the load across the processors of the transputer system. This balancing will be achieved by remapping operation buffers to processors with expected improvement in performance.

REFERENCES

Bezivin, J. and H. Imbert. 1982. "Adapting a Simulation Language to a Distributed Environment". In *Proceedings of the 3rd International Conference on Distributed Computing Systems*, (Ft. Lauderdale, FL), IEEE, N.Y., 596-603.

Bryant, R.E. 1977. "Simulation of Packet Communication Architecture Systems". Technical Report. MIT/LCS/TR-188, MIT, Cambridge, MA, (Nov.).

Christopher, T.; M. Evens; R.R. Gargeya; and T. Leonhardt. 1982. "Structure of a Distributed Simulation System". In *Proceedings of the 3rd International Conference on Distributed Computing Systems* (Ft. Lauderdale, Fl.), IEEE, N.Y., 584-589.

Chandy, K.M.; L.M. Haas; and J. Misra. 1983. "Distributed Deadlock Detection", *ACM Transactions on Computer Systems* 1, no. 2 (May), 144-156.

Chandy, K.M. and J. Misra. 1979a. "Distributed Sim-

ulation: A Case Study in Design and Verification of Distributed Programs", *IEEE Transactions on Software Engineering* SE-5, no. 5 (Sep.), 440-452.

Chandy, K.M. and J. Misra. 1979b. "Deadlock Absence Proofs for Networks of Communicating Processes", *Information Processing Letters* 9, no. 4 (Nov.), 185-189.

Foxfire Technologies Corporation. 1989. *Real-time Shop-Floor Control System User Manual*, Marietta, GA.

Fujimoto, R.M. 1988. "Performance Measurements of Distributed Simulation Strategies". In *Distributed Simulation 1988: Proceedings of the SCS Multiconference on Distributed Simulation* (Feb. 3-5), SCS, San Diego, CA, 14-20.

Jefferson, D.R. 1985. "Virtual Time", *ACM Transactions on Programming Languages and Systems* 7, no. 3 (Jul.), 404-425.

Khambekar, P.K. and S.K. Dharmaraj. 1990. "Approaches to Solving Synchronization Problems in Parallel Simulation of an Apparel Plant." In *Proceedings of the 1990 ACM Southeast Regional Conference* (Greenville, SC, April 18-20), ACM, N.Y., 274-281.

Misra J. 1986. "Distributed Discrete-Event Simulation", *ACM Computing Surveys* 18, no. 1 (Mar.), 39-65.

Nicol, D.M. and P.F. Reynolds. 1984. "Problem Oriented Protocol Design." In *Proceedings of the 1984 Winter Simulation Conference (16th)* (Nov. 28-30), 471-476.

Nicol, D.M. 1988. "Parallel Discrete-Event Simulation of FCFS Stochastic Queueing Networks." In *Proceedings of the ACM SIGPLAN PPEALS 1988* (Jul.), 124-137.

Pargas, R.P.; J.C. Peck; P.K. Khambekar; and S.K. Dharmaraj. 1990. "Near-term Distributed Simulation of Apparel Manufacturing." In *Proceedings of the 1990 Winter Simulation Conference* (New Orleans, LA, Dec. 9-12), SCS, San Diego, CA, 614-618.

Peck, J.C.; R.P. Pargas; P.K. Khambekar; and S.K. Dharmaraj. 1991. "Shop-Floor Performance Metrics for the Apparel Industry." Submitted to the *International Journal of Clothing Science and Technology.*

Peacock, J.K.; J.W. Wong; and E.G. Manning. 1979a. "Distributed Simulation Using a Network of Processors", *Computer Networks* 3, no. 1, 44-56.

Peacock, J.K.; J.W. Wong; and E.G. Manning. 1979b. "A Distributed Approach to Queueing Network Simulation." In *Proceedings of the 1979 Winter Simulation Conference* (San Diego, CA), IEEE, N.Y., 399-406.

Reed, D.A.; A.D. Malony; and B.D. McCredie. 1988. "Parallel Discrete Event Simulation Using Shared Memory", *IEEE Transactions on Software Engineering* SE-14, no. 4 (Apr.), 541-553.

Reed, D.A. and A.D. Malony. 1988. "Parallel Discrete Event Simulation: The Chandy-Misra Approach." In *Distributed Simulation 1988: Proceedings of the SCS Multiconference on Distributed Simulation* (Feb. 3-5), SCS, San Diego, CA, 8-13.

Reynolds, P.F. 1988. "A Spectrum of Options for Parallel Simulation." In *Proceedings of the 1988 Winter Simulation Conference*, (Dec.), 325-332.

Sokol, L.M., D.P. Briscoe; and A.P. Wieland. 1988. "MTW: A Strategy for Scheduling Discrete Simulation Events for Concurrent Execution." In *Distributed Simulation, 1988: Proceedings of the SCS Multiconference on Distributed Simulation* (Feb. 3-5), SCS, San Diego, CA, 34-42.

BIOGRAPHY

John C. Peck is a Professor of Computer Science at Clemson University. He received a B.S. in Mathematics, and M.S. and Ph.D. degrees in Computer Science, all from the University of Southwestern Louisiana. His research interests are in database systems and distributed algorithms. He is currently involved in developing support systems for manufacturing for the Defense Logistics Agency. He is also Vice-President for Research and Development of Foxfire Technologies, Inc. a company that designs real-time shop-floor control applications for the apparel industry.

KNOWLEDGE-BASED INTERACTIVE REAL-TIME CONTROL SYSTEM IN DISCRETE MANUFACTURING

†Qi-zhi Sun, #Christopher N Chrystall, †Michael M Kaye

† Business School
Portsmouth Polytechnic
Milton, Portsmouth PO4 8JF
United Kingdom

CIM Department
IBM (UK) Ltd
Havant, Hampshire PO9 1SA
United Kingdom

ABSTRACT

A prototype Knowledge-based Interactive Real-time Control System (KIRCS) is under development in IBM (UK) Ltd at the Company's Havant manufacturing site. The conceptual framework of KIRCS and the functions of each module are described in this paper.

An expert system front end has been developed to provide decision advice and/or issue control commands based on decision heuristics. The decisions are evaluated by simulation prior to implementation. Several aspects which have arisen during the development of the system are discussed.

KEYWORDS

Manufacturing, Decision-making, Real-time simulation, Production control, Knowledge-based systems

1 INTRODUCTION

The control of materials flow at shop floor level is a management problem which has been regarded as an art rather than a science. The complexity and dynamics associated with this class of problems frequently make them too difficult to be solved using mathematical algorithms (Kusiak 1990). Even the fundamentals are still under research, although attempts have been made to specify the activities involved in shop floor control and the functions which a real-time control system must perform (Ben-Arieh, et al. 1988; Swyt 1989).

Beal views Shop Floor Control (logistics) as "a variant of a standard closed loop control system" (Beal 1989). Generally speaking, the information flow of the shop floor control can be illustrated as in Figure 1.

According to (Tocher 1970), there must be four conditions under which a system may perform control tasks.

- Action times.
- A set of actions from which to choose.
- A model which can predict the future behaviour of the system.
- A criterion on which the choice of action is based by a comparison of predicted behaviour of the system with the objective.

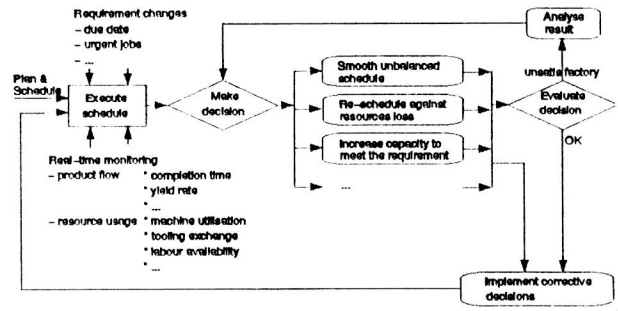


Fig. 1 Information flow of shop floor control

Present shop floor control systems generally aim to generate a schedule spanning a week or longer. Real-time control environments require such systems to run on a day by day or shift by shift basis in order to smooth the production over the week. In highly automated systems, it is possible to determine the actions on a hourly basis.

Control actions include the combination of scheduling and real-time control (Sun, Kaye and Chrystall 1990). The most commonly used method for scheduling is the use of priority sequencing rules. Real-time control actions include the alteration of process flow and adjustment of capacity. However, these options are dependent on the layout of the system and the nature of the materials handling facilities. Most of the control options, therefore, can only be derived from practice. In some production lines, the conveyors are designed in such a way that the only action is to stop the line if there is an equipment failure.

The objective of controlling a manufacturing system is usually not a single target but a trade-off between multiple criteria. For example, due dates must be achieved precisely under the Continuous Flow Manufacturing (CFM) policy, and in the meantime, resource utilisation must be maximised, cost

minimised and work-in-process kept under a certain level. Tocher claims that "the objective is logically impossible to achieve at all".

The ideal situation would be that, at one action time, the possible actions are represented in a mathematical form and a prospective action is identified by using optimising techniques. Theoretically, scheduling optimisation is possible if some objective criteria can be compromised and constraints relaxed (Kusiak 1990). However, Jagdev reviewed the commercial packages available for shop floor control applications and found that no optimising algorithms had been used due to the versatility of real problems and the complexity of algorithms (Rolstadas 1989).

Nevertheless, a forecasting model is needed. As an alternative to optimising algorithms, simulation can be adopted for modelling the future behaviour of the manufacturing systems. Simulation does not generate an optimal solution on its own, but it demonstrates the trend under a series of control policies. It is notable that nearly all of the commercial scheduling tools presently available aimed at short-term planning of shop floor materials flow make use of simulation techniques (Dwyer 1990).

It must be emphasized that the primary usage of simulation is the evaluation of a control option. The quality of the decisions derived from such mechanisms is dependent on the quantity and quality of the candidate actions (Wu and Wysk 1988).

At the Havant manufacturing site of IBM (UK) Ltd, a pilot project termed Real-time Interactive Control System (RETICS) has been carried out to investigate the feasibility of using simulation as a real-time production control tool. Subsequent attempts have been made to form a conceptual model of Knowledge-based Interactive Real-time Control System (KIRCS) with an emphasis on the use of the Expert System technology to direct the search of control actions. The conceptual model of KIRCS and the functions of each module are described below.

2 CONCEPTUAL MODEL

As with the general structure of an expert system, KIRCS consists of an inference engine, a knowledge base and a data base. In addition, a simulation module is adopted to assist in the evaluation of decision alternatives.

To convey the conceptual model of KIRCS in a formal syntax, IDEF0 (Marca and McGowan 1988) is used to define its information flow as in Figure 2. The expert system performs two functions, i.e. comparing achievement with requirement and analysing alternative actions. The simulation model forecasts the future behaviour of the system under control.

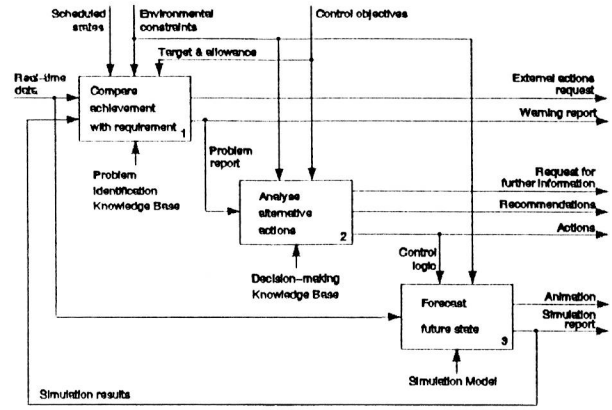


Fig. 2 KIRCS information flow chart in IDEF0

2.1 Inference Engine

KIRCS can be used manually by an end-user or driven by real-time data at a certain action time. In either case, KIRCS collects real-time data from the process and compares it with the scheduled states.

If the objective criteria are not met, the expert system evaluates if the deviation is within a boundary, beyond which the rules to be analysed in the knowledge base would not be able to cope with. When KIRCS fails to identify an action rule, it will stop and request for external actions. When the deviation from the objective criteria is detected which falls into the boundary, the expert system starts to analyse alternative actions.

The analysis of actions involves searching the decision rules until those whose conditions match the deviation are found. If no rules are matched, KIRCS stops and requests further information. The actions of all the matched rules are converted into simulation control logic.

If the objective criteria have been achieved but a key state parameter is found to be out of normal state, the expert system gives a warning report and activates the simulation model to predict the future achievement of the objective criteria. In KIRCS, only the objective criteria performance are used to direct the search of decision rules (see 2.2 Knowledge Base). Also, if all the deviations are within an allowance (see 2.3 Data Base), the system behaviour is regarded as normal and no action is taken. In this case, KIRCS stops and waits for next action time.

The simulation model is updated by real-time data and run under each control action. The simulation results are then compared with the scheduled states. If the results of at least one simulation run have achieved the objective criteria, the actions are recom-