INVESTIGATION OF FAST AND HYBRID (FAST/DISCRETE-EVENT) MODELING APPROACHES FOR SIMULATION OF MANUFACTURING SYSTEMS

By

MANOJ N. DUSE

Bachelor of Engineering
University of Poona
Poona, India
1988

Master of Engineering
National Institute for Industrial Engineering
Bombay, India
1989

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
DOCTOR OF PHILOSOPHY
July, 1994

INVESTIGATION OF FAST AND HYBRID (FAST/DISCRETE-EVENT) MODELING APPROACHES FOR SIMULATION OF MANUFACTURING SYSTEMS

Thesis Approved:

Joe H. Mize

Thesis Adviser

[illegible]

Manjunath Kamath

M. P. Terrell

[illegible]

Thomas C. Collins

Dean of the Graduate College

ACKNOWLEDGMENTS

Getting an opportunity to become an active player in the Advanced Modeling Methodologies research project being pursued at the Center for Computer Integrated Manufacturing was one of the most significant events in my last three and half years of stay at this university. This dissertation is dedicated to the Center for CIM without which this dissertation would not have taken birth. Implicitly, it is dedicated to the founder of this center, Dr. Joe H. Mize and to the whole research team working on this project.

First and foremost, I would like to thank my parents, two sisters and brothers-in-law for all the encouragement and moral support they have provided to me. To my best roommate Avinash and to my true friend Gharpure, thank you for all the help you have offered me over the last three years. Special thanks to Ameya and Rashmi for making my stay in Stillwater more enjoyable.

Thanks to Dr. Mize, my adviser, for being a great source of motivation, inspiration and encouragement. My sincere appreciation to all the committee members for providing their invaluable input during this research. To Dr. Kamath, thank you for offering me all the help, guidance, and support, especially during the initial periods of my stay at this university.

I also thank the AT&T Foundation and the Oklahoma Center for Integrated Design and Manufacturing for providing significant financial support during my graduate studies. Finally, I would like to express my appreciation to all others who have contributed towards earning my Ph.D.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure Page

CHAPTER I

INTRODUCTION

Simulation of Manufacturing Systems

Discrete event simulation is a widely used tool for predicting and evaluating the performance of manufacturing systems. Simulation models can include detailed information about the system and allow representation of unique features of manufacturing systems. Computer simulation involves replication of time-variant behavior of the system as defined in the simulation model for gathering observations about the performance measures of interest. Simulation of manufacturing systems is the most preferred alternative for performance evaluation when (i) the mathematical assumptions which underlie analytic solution procedures are not satisfied and/or (ii) one is interested in assessing the transient performance of a manufacturing system rather than its steady-state behavior [Askin 1993]. The role of simulation in manufacturing can be broadly categorized into two groups viz. (a) Design/Analysis and (b) Operations Control.

Design and Analysis of Manufacturing Systems

While designing new manufacturing facilities, especially during the initial stages of the planning activities, one generally employs analytical tools to provide rough estimates of system performance measures. Taking this rough-cut evaluation of the design alternatives as input, the list of alternative designs is then narrowed down to the potentially attractive ones. These potential design alternatives are then simulated to choose the most favorable design alternative. Use of simulation during the final design stages helps to minimize the risk associated with the inability of the manufacturing system to meet the required performance criteria [Dunn 1985]. Even after implementing the chosen design alternative, one may be forced to improve the performance of the manufacturing system either because of competitive forces or changing customer demands [Suri and deTreville 1992]. In order to meet the time-based competition, one may be forced to cut down lead times and in such cases, simulation can aid in analyzing the manufacturing system to identify the bottleneck operations. Several applications of simulation in the design and analysis of manufacturing systems can be found in Heginbotham [1985].

Operations Control of Manufacturing Systems

Traditional approaches to scheduling and control of manufacturing systems include scheduling algorithms and mathematical programming applications. More recently, several researchers have identified the potential of simulation in developing shop-floor control systems. The two major applications of simulation in implementing shop-floor control systems are [Erickson et al. 1987]:

[1] Scheduling and Sequencing: The various alternatives are simulated at the beginning of some production window using actual starting conditions of the system and the alternative which best satisfies the performance measure is then implemented.

[2] Contingency Control: Contingencies like machine failures, expedited orders, raw material changes and other problems that are unavoidable regardless of good system maintenance and scheduling may not be considered explicitly while generating the schedules. In order to respond to these unforeseen occurrences in the best possible way, simulation can be used to evaluate the effectiveness of various actions such as changing the schedules, rerouting the orders, etc.

Harmonosky [1990] has proposed a framework for real-time control which uses simulation for evaluating the performance of manufacturing systems under various control decision options. This framework helps in understanding how simulation fits within the overall shop-floor control structure.

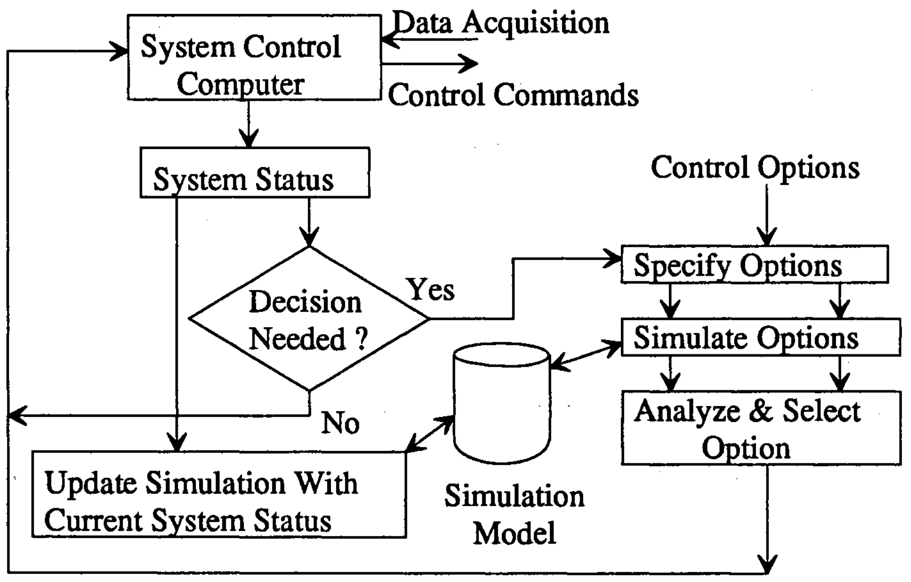


Figure 1. Framework for Real-time Control Using Simulation

(Adapted From Harmonosky [1990])

Motivation Behind This Research

Simulation models can mimic a complex, real-world manufacturing system as closely as understanding permits and needs require, but often require high computational time. This "computational time intensive" aspect of simulation has tended to limit the use of simulation to an off-line, design and analysis mode. Though execution efficiency of simulation models is not critical for design and analysis of manufacturing systems, efforts in this direction will always be welcomed. Speeding-up the simulation execution will allow us to consider more detailed models of the system and will permit evaluation of a much larger set of design alternatives. Gains in execution efficiency of simulation can even make search-based optimization more attractive for designing manufacturing systems.

In order to enable the application of simulation for real-time operations control, research efforts should be directed at improving the execution efficiency of simulation models, which is the primary motivation of this research. The importance of improving the execution speed of simulation models is reflected in the following quote from Harmonosky [1992]:

"...When using any manufacturing scheduling and control system, the amount of time it takes to make decisions will directly affect the degree to which the system is controllable in real-time. In a system that uses simulation for scheduling and control, the CPU time to perform simulation runs accounts for most of the decision making time and directly affects the ability of a manufacturing system to control its actions in real-time."

Though traditionally real-time control refers to an immediate response to some event in a system, the speed of response for decision making may actually depend upon system parameters such as magnitudes of part processing times [Harmonosky and Robohn 1991]. Ranky [1988] suggests that new schedules should be generated within 5 to 10 seconds for a medium sized FMS in a CIM environment. Even when the definition of real-time is not as demanding as above, faster execution of simulation models provides the added advantage of improving the confidence in decision making by increasing the number of replications and/or by lengthening the "look-ahead window" for simulation.

Overview of the Dissertation

The remainder of this dissertation is presented in nine chapters. Chapter II reviews the various research efforts made so far for improving the execution efficiency of simulation. The scope of this study is then defined by presenting a concise statement of the problem for this research. The literature relevant for this research is reviewed in Chapter III. Several unanswered research questions are also identified in Chapter III, a subset of which forms the basis for defining the objectives of this research. Research goals and objectives are defined in Chapter IV along with the scope and limitations of this research. The expected contributions from this research effort are also outlined in this chapter. This is followed by Chapter V which discusses the performance measures to be used, experimental scenarios, and the various phases of this research. Chapter VI deals with the modeling abstractions, conceptual frameworks, and simulation models for fast simulation of a subset of manufacturing network topologies. Chapter VII provides a brief overview of the object-oriented implementation and validation; the comparison of simulation execution times for discrete event and fast simulation is also presented in this chapter. Chapter VIII focuses on hybrid modeling whereas Chapter IX deals with the hybrid simulation issues. Chapter X is the concluding chapter that presents the research summary, its contributions, and directions for further research.

CHAPTER II

STATEMENT OF THE PROBLEM

Research efforts directed at increasing the execution efficiency of simulation models can be categorized into two groups, viz. Implementation Approaches and Modeling Approaches. What follows is a brief review of significant research accomplishments in each of the above two categories.

Implementation Approaches

These approaches attempt to speed-up the execution of simulation by concentrating on the implementation of a simulation model.

Parallel Discrete Event Simulation (PDES)

This approach involves execution of a discrete event simulation model on a parallel processor which requires partitioning of the simulation model into distinct units to be executed on different processors. The various processors need to communicate with each other in order to take care of event interdependencies. Several researchers have investigated the potential of parallel simulation for simulating manufacturing systems [Nevison 1990, Nicol 1988]. The communication overhead which reduces the gain in execution efficiency obtained by using parallel processors can become a significant problem in the case of complex manufacturing systems. Current research is directed at developing communication strategies and strategies for partitioning the model so as to minimize the communication load [Bhuskute 1993].

Improving the Data Structures Used for Event Calendars

In discrete event simulation, some mechanism must be provided for causing the events to occur at the proper simulation clock time. This is usually referred to as the Time Flow or the Time Advance Mechanism of simulation. The two common types of time advance mechanisms employed in simulation are:

Fixed time increment

Variable time increment

It has been shown by Nance [1971] that variable time increment is more efficient of the two mechanisms. In variable time increment, a list of future events is maintained in the event calendar and events are removed/executed in the order of simulation clock time. The simulation clock is incremented to the time of the next event after executing all the current events.

The implication of the data structure used for storing/removing events from the calendar for execution efficiency of simulation has been recognized by several researchers. A summary of comparative performance of various data structures and algorithms has been reported in Adam and Dogramaci [1979]. Four different simulation models were used as representative of closed queueing systems for evaluating the performance of various future event list algorithms. Three of the models used were simulations of computer and communication systems and the fourth one modeled the classical machine-repairman model. The various algorithms evaluated are listed along with their brief description in Table I. Experiments were carried out for twelve cases, seven of which belonged to the computer and communications systems category whereas the remaining five belonged to the machine-repairman model. Trace-driven simulation was used to measure the performance of each algorithm in terms of average time required to perform the insertions to and deletions from the future event list. Use of trace driven simulation helped to eliminate the overhead associated with random variate generation, gathering of statistics, and execution of events.

TABLE I

EVENT LIST MANIPULATION ALGORITHMS

ALGORITHM	DESCRIPTION
LLB	Linked linear list; search begins from back (high time end) of the list.
LLF	Linked linear list; search begins from front (low time end) of the list.
MLF	Multiple linear lists; search starts from the front of all lists.
MMF	Linear list with pointer to the middle record; search begins from the front of the half to which the new record belongs.
HEP	Heap data structure.
VAU, FRA, & HNR	All three are variations of linked linear list with special array of pointers which are used to divide the list logically into many shorter sublists; scanning starts from the back of the list.

Execution time was used as a performance measure rather than the average number of comparisons required for an insertion because it takes into account the other computational overhead incurred in implementing each of the event list manipulation algorithms. The results have shown that no data structure or algorithm is superior to all others and the performance varies with the model being simulated. The rankings of the algorithms are summarized in Table II. Though VAU algorithm has lowest average rank, it is not the best performer in all cases, the worst ranking being 4. Furthermore, it requires an upfront effort to calculate the number of sublists to be used for event manipulation and its performance may be sensitive to the value of this parameter.

TABLE II

PERFORMANCE OF EVENT LIST MANIPULATION ALGORITHMS

ALGORITHM	AVERAGE RANKING	MINIMUM RANKING	MAXIMUM RANKING
LLB	7.67	4	8
LLF	3.42	1	8
MLF	4.58	3	7
MFF	3.17	2	6
HEP	4.00	1	7
VAU	2.00	1	4
FRA	5.08	3	7
HNR	6.08	3	7

Reeves [1984] also has studied the performance of various algorithms under certain conditions and has found *ternary heaps* to be more attractive for event manipulations.

Improved Memory Management Procedures

Object-oriented simulation environments are attractive from the modeling viewpoint but have a significant drawback of slow execution speed. Attempts have been made by Beaumariage and Roberts [1991] to correct this deficiency to the extent possible. They studied the memory allocation and garbage collection policies of the Smalltalk/V environment and suggested the concept of a "recycling model".

During the execution of simulation, several objects are created and discarded from the same class hierarchies. The space allocated to discarded objects is reclaimed during the garbage collection and compacted periodically for allocating space for newly created objects. The recycling model attempts to minimize this overhead by maintaining a collection of discarded objects memory segments (rather than reclaiming via garbage collection) for each class being recycled. When a new object is to be created, this collection is checked for reuse of previously allocated memory segments. The hypothesis behind this concept is that the time required to initialize and recycle used memory segments will be less than the time used to dynamically allocate the memory and to compact the active object memory. Initial investigation of this concept has shown that savings in execution time of about 1.5% can be achieved for models of low utilization systems. Moreover, such efforts are platform-specific and extendibility of the results to other environments like Smalltalk-80 remains to be studied.

Modeling Approaches

These approaches attempt to make simulation more execution efficient by focusing on the model development (abstraction) process.

Hybrid Modeling Using Observation Based Metamodels

The behavior of the detailed model of a subsystem is observed over time and then either an analytical function is determined which describes the relationship between input variables and performance measures or a cumulative distribution function is fitted from which samples can be drawn to create metamodels of the subsystems [Pratt 1992]. Use of such analytical relationships eliminates the need for simulating the detailed models of those subsystems and, in turn, makes the simulation more execution efficient.

Hybrid Modeling Using Queueing Network Based Metamodels

Recent developments have made it possible to employ queueing networks for performance evaluation of fairly complex manufacturing systems. Queueing networks can now deal with general service/arrival time distributions, multiple customer classes and open/closed network configurations [Segal and Whitt 1989]. These recent advances in queueing networks are exploited by creating queueing network submodels of subsystems which can be solved analytically and then embedding them in simulation models of larger systems [Shantikumar and Sargent 1983].

The above two approaches have been proposed primarily for estimating the steady-state performance of the system and not for analyzing its transient behavior. In addition to this limitation, use of these approaches can only provide approximate estimates of the performance measures.

Fast Simulation

The fast simulation models developed so far [Chen and Chen 1990] are based on identifying the relationships between arrival and departure times of customers at any given node. In pure fast simulation models, one does not have to maintain a list of events as required in traditional discrete event simulation. The absence of overhead related to the time advance mechanism makes the execution of such models more efficient which is the underlying rationale for using fast simulation. Fast simulation models developed so far can handle only tandem lines with single server-finite/infinite buffer stations. Several research questions need to be addressed to broaden the applicability of this technique.

Hybrid (Fast/Discrete-Event) Simulation

Kamath [1994] has proposed a concept of a new hybrid approach to the simulation of queueing network models. The rationale for proposing such a hybrid approach is as follows:

For certain scenarios, for example, networks with state dependent routings and dynamic job priorities, the fast simulation approach may be very complex and may not be computationally superior. The hybrid simulation approach can potentially combine the execution efficiency of fast simulation with the modeling power and flexibility of the discrete event approach.

However, several research questions need to be addressed before such a hybrid approach can be employed for simulation of manufacturing systems. A more detailed discussion of this hybrid approach is deferred to Chapter III which also presents a list of several unanswered questions to be investigated for conceiving such hybrid models.

The phrase "hybrid simulation" is used in the literature to describe an approach which combines simulation with analytical models. The term hybrid, used hereafter in this dissertation, refers to a simulation approach which employs discrete event simulation for one part of the system and fast simulation for another[*] .

Problem Statement

All of the approaches, which have the potential to improve execution efficiency, should not be seen as mutually exclusive alternatives since research efforts in one direction can complement those in others. A synergistic combination of all these approaches could ultimately provide the much needed execution efficiency to simulation. Though current research efforts are mainly concentrated on improvements in individual approaches, especially in Parallel/Distributed Simulation and Metamodeling, the research community will soon realize the need for integrating the above research efforts.

The objective of this research is <u>not to</u> integrate the various research efforts <u>but to</u> make significant contribution to the evolving field of fast simulation and its use in hybrid simulation. Thus, the problem statement for this research can be summarized as follows:

To investigate the potential and the limitations of fast and hybrid simulation techniques for performance evaluation of manufacturing systems.

[*] Since two different modes of simulation are employed while simulating such hybrid models, the resulting simulation approach can also be referred to as "multi-mode simulation."

CHAPTER III

BACKGROUND OF THE STUDY

Underlying Rationale of Fast Simulation

While developing a discrete event simulation (DES) model of a system, one can select from the three world views viz. (i) Event Scheduling Approach, (ii) Process Interaction Approach, and (iii) Activity Scanning Approach [Pritsker 1986]. The following sub-sections briefly describe these three world views.

Event Scheduling Approach

Events define a set of simultaneous state changes and the model's state remains unchanged between events. Events are stored in an event list or agenda in the time order of their occurrences. Actions associated with an event are kept in modules called event routines and execution of such event routines can schedule new events (or remove events) which are again placed on the event list at the appropriate position. The simulation clock is updated to the next imminent event on the list and the event routine corresponding to that event is executed after removing that event from the list.

Process Interaction Approach

A process is a sequence of logically connected events which involve the same simulation entity (object). All the actions associated with these logically connected events are grouped into a single module called the life cycle of that simulation entity. The behavior of the system can be represented by a set of processes whose event sequences, when merged, contain all the events that occur in the system [Mitrani 1982]. An equivalent of the event list is maintained in this approach also, but the entries on the list are processes ordered according to the time of next event in their respective sequences. Each process on the list also remembers the state in which a process was last suspended from which to start when it will be taken for execution.

Activity Scanning Approach

Kreutzer [1986] describes this approach as follows:

"...An activity is a conceptual closure of some time-consuming action performed by an entity. It is typically guarded by a set of conditions under which it may start, and finishes after a specified time period. The programmer only needs to describe the conditions under which a particular event may occur. It is then up to the model execution monitor to make sure they are triggered at the right time."

The activity scanning approach is computationally expensive compared to the previous two approaches. Even with event scheduling and process interaction approaches, a lot of computational overhead is incurred while searching the event list for inserting new events at the proper position in the list. As the number of events in the system increases, this overhead also increases significantly. *If one can eliminate the need for the event list and hence, the overhead associated with it, simulation can become more computationally attractive.* The main theme of fast simulation is to achieve this by identifying the relationships between the arrival and departure times of customers at any node. The research efforts which have attempted to exploit this concept are briefly reviewed in the following sections.

Review of Previous Work

Fast Simulation Model - Tandem Line With Single Server, Infinite/Finite Buffer Stations

The following relations are identified for single server-infinite/finite buffer stations in Chen and Chen [1990] and are based on the assumptions of reliable servers, FCFS queue discipline, and a single customer class.

Let $d_{i,j}$ = departure time of jth customer from ith station

$s_{i,j}$ = service time of jth customer at ith station

The service start time (sst) of jth customer at ith station = $\max(d_{i-1,j}, d_{i,j-1})$

and $d_{i,j} = \max(d_{i-1,j}, d_{i,j-1}) + s_{i,j}$ for infinite buffers.

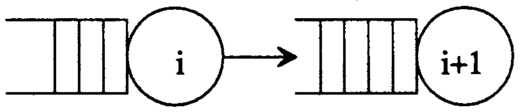


Figure 2. A Tandem Line With Finite Buffers

In the case of finite buffers, illustrated in Figure 2, two cases should be considered:

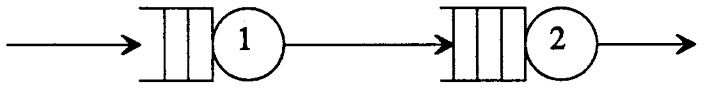
[1] Manufacturing Blocking:

In this case, if the customer at node i sees the buffer of node i+1 to be full as he completes his service, then he waits at node i until space becomes available in the buffer of node i+1.

[2] Communication Blocking:

If the customer who is to receive service at node i sees that node i is idle but buffer at node i+1 is full, then he cannot start service until space becomes available in the buffer of node i+1. Though relationships between arrival and departure times can be identified for both the types of blocking, only manufacturing type blocking is relevant for this research.

While developing fast simulation models of tandem lines with single server-finite/infinite buffer stations, Chen and Chen [1990] have adopted a "customer-by-customer" view, i.e., they focus on a particular customer and his flow along the tandem line. A particular part/customer, once taken for processing, is processed completely through the fast simulation model of a tandem line and then it departs from the last node of the tandem line. This procedure is then repeated for the next customer. Thus, when the fast simulation model takes the n^{th} customer for processing, all previous n-1 customers have been processed completely through the tandem line. This "customer-by-customer" view of a tandem line allows blocking to be taken into account. This is illustrated below using a partial tandem line shown in Figure 3.



buffer capacity = 3 (excludes the one with server)

Figure 3. Blocking Phenomenon

The n^{th} customer, after finishing the service at node 1 at time t, will be blocked if there is no space available in the input buffer of node 2. There will be space available in the input buffer of node 2 at time t, if and only if, the $n\text{-}4^{th}$ customer has departed from node 2 by time t. If the $n\text{-}4^{th}$ customer departs from node 2 at time $t1 > t$, then the n^{th} customer will be blocked at node 1 from time t to time t1. Thus, to determine the actual departure time of parts from node 1, one needs the knowledge of departure times of preceding parts from the following node. Use of "customer-by-customer" view ensures that this $n\text{-}4^{th}$ customer's departure time (from node 2) is known while determining the departure time for the n^{th} customer (from node 1). The generalized relationship between arrival and departure times is as follows:

$$d_{i,j} = \max \{ [\max(d_{i-1,j}, d_{i,j-1}) + s_{i,j}], d_{i+1,j-(b_{i+1}+1)} \}$$

where b_{i+1} is the buffer capacity of $i+1^{th}$ station excluding the one in server.

The term $d_{i+1,j}-(b_{i+1}+1)$ determines whether space will be available at the next station when the j^{th} customer finishes its service at the i^{th} station.

Thus, fast simulation of such a tandem line involves processing one customer at a time through all the nodes and collecting statistics about time-in-queue and utilization during this pass. The pseudocode presented in Figure 4 illustrates the fast simulation approach taken by Chen and Chen [1990].

```
For customer = 1 to totalNoOfCustomers (N) do
    {Generate arrival time of customer.
    For node = 1 to totalNodes (M) do
        {Determine the time at which service can start at that node.
        Generate processing time.
        Determine departure time for the customer:
            (service end time + blocking, if any)
        Collect statistics about time in queue, blocking, and utilization}
    Collect statistics about time-in-system}
Calculate performance measures like throughput, etc.
```

Figure 4. Pseudocode for Fast Simulation of a Tandem Line With Single Servers

The performance measures were calculated as follows:

$$\text{Average Utilization of node i} = \frac{1}{d_{M,N}} \sum_{j=1}^{N} s_{ij}$$

$$\text{Throughput of node i} = TP_i = \frac{N}{d_{M,N}}$$

$$\text{Average waiting time (time in queue) at node i} = W_i = \frac{1}{N} \sum_{j=1}^{N} (sst_{i,j} - d_{i-1,j})$$

$$\text{Average blocking time at node i} = \frac{1}{N} \sum_{j=1}^{N} [d_{i,j} - (sst_{i,j} + s_{i,j})]$$

$$\text{Average time in system} = \frac{1}{N} \sum_{j=1}^{N} [d_{M,j} - arrivalTime_j]$$

The savings in execution time obtained by Chen and Chen [1990] using fast simulation models of tandem lines instead of discrete event (event scheduling approach) simulation are presented in Table III.

TABLE III

PERFORMANCE OF FAST AND DISCRETE EVENT SIMULATION FOR A TANDEM LINE

Number of Stations	Execution Time (min.) Fast	Execution Time (min.) DES	Execution Time Ratio (%)	Savings Ratio (%)
10	9	16	56	44
20	18	35	51	49
30	27	58	47	53
40	36	85	42	58
50	45	116	39	61
60	53	151	35	65
70	62	190	33	67
80	71	233	30	70
90	80	295	27	73
100	90	360	25	75

All nodes have finite buffers of capacity 5. Run length of 100,000 customers
Utilization = 0.9

Figure 5 shows that run-time required by the traditional (discrete event) simulation increases "exponentially" as the number of stations (system size) is increased, but the time needed for fast simulation is only a linear function of the number of stations in the tandem line. In addition to this, Chen and Chen [1990, 1993] have made the following observations:

[1] The memory requirements for fast simulation can be significantly less than that required for discrete event simulation.

[2] The run time increases linearly with the number of customers (simulation run length) for both fast and traditional simulation.

[3] The run time needed by traditional simulation increases with average utilization but the run time for fast simulation is not influenced by the change in average utilization.

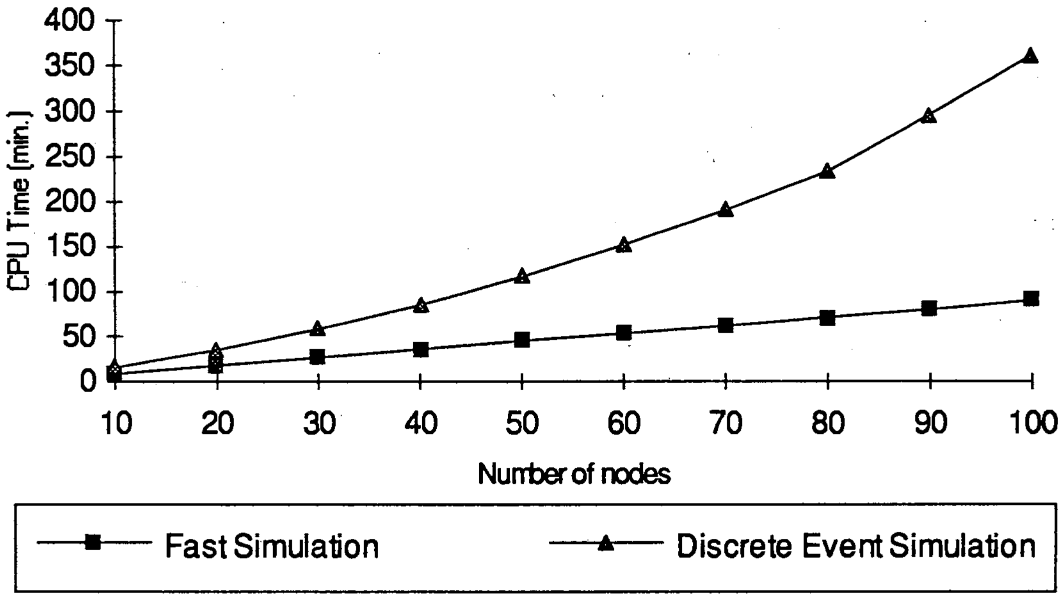


Figure 5. Comparison of Execution Time for a Tandem Line (Single Server)

For obtaining accurate performance measures of the system and for estimating confidence intervals, several replications (simulation runs) are required. This would have a multiplying effect on the execution time savings achieved by using fast simulation.

Fast Simulation Model - Tandem Line With Parallel Server, Infinite Buffer Stations

Kamath, Bhuskute, and Duse [1992] have developed and implemented a fast simulation model of a tandem line with parallel server, infinite buffer stations.

Consider a partial tandem line with parallel server station with capacity 3 (capacity = # of parallel servers at that node) as shown in Figure 6.

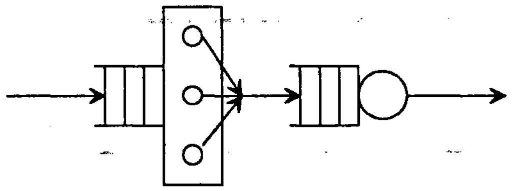


Figure 6. A Tandem Line With Parallel Servers

From the partial trace of parallel server node simulation presented in Table IV, it is seen that the order in which the customers depart from the parallel server node is not the same as the order in which they arrive at that node. This implies that one cannot adopt a "customer-by-customer" view while developing fast simulation models of tandem lines which include parallel server nodes. If we adopt this view, we will end up processing customer # 4 through the next node before customers # 5, 6, and 7; whereas actually customers # 5, 6, and 7 arrive at the next node before customer # 4.

TABLE IV

PARTIAL TRACE OF PARALLEL SERVER NODE SIMULATION

Customer ID #	Arrival time	Start of Service	Service time	Departure time Server 1	Departure time Server 2	Departure time Server 3	Departure Sequence
1	1	1	2	3			1
2	2	2	3		5		2
3	4	4	5			9	3
4	5	5	8	13			7
5	6	6	4		10		4
6	7	9	2			11	6
7	8	10	0.5		10.5		5

To determine the proper order of departure from such a parallel server node, Kamath, Bhuskute, Duse [1992] have adopted a "node-by-node" view of the tandem line, i.e., they focus on a particular node and process all the customers through that node (while updating the order of departures appropriately).

This model was validated by comparing the performance measures obtained by fast simulation with those obtained by running a discrete event simulation model. The speed-up achieved for such models was not investigated.

In order to develop fast simulation models of a tandem line which have combinations of finite buffer and parallel server nodes, there are conflicting requirements in terms of the view to be adopted. Presence of a finite buffer feature suggests the use of a "customer-by-customer" view of a tandem line, whereas presence of a parallel server node requires the use of a "node-by-node" view of a tandem line. Does it mean that this is an infeasible case for fast simulation? or can we find a "third view" which would be able to handle both the finite buffer and parallel server features?

Fast Simulation Model - Assembly Topology With Single Server, Infinite Buffer Stations

A fast simulation model for the topology shown in Figure 7 was investigated for gain in execution speed and encouraging results were obtained. In the assembly topology shown

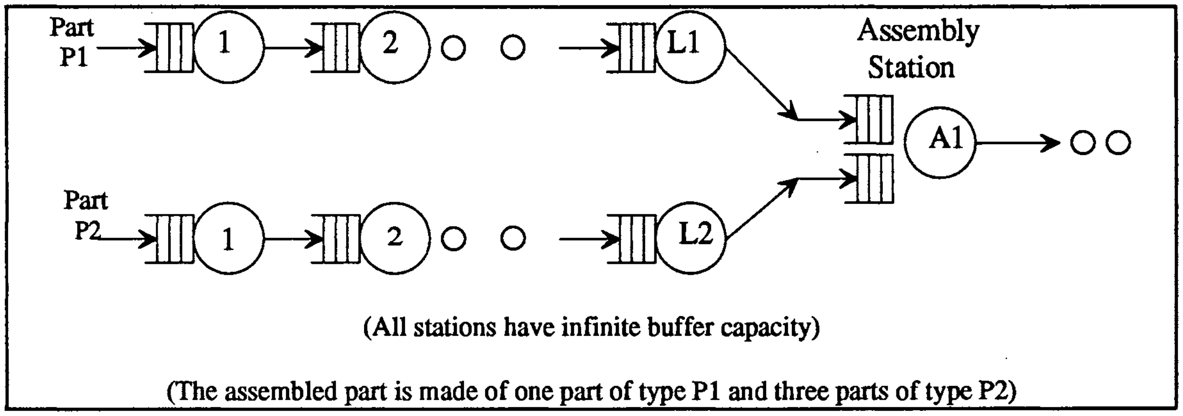

Part P1
1
2
L1
Assembly Station
A1
Part P2
1
2
L2
(All stations have infinite buffer capacity)
(The assembled part is made of one part of type P1 and three parts of type P2)

Figure 7. An Assembly Topology

in Figure 7, when a sufficient number of component parts are available and the assembly station is idle, the parts are removed from the input buffers of the assembly station in the form of an assembly kit and the kit is then assembled at that assembly station. The pseudocode for fast simulation of the above assembly topology is presented in Figure 8.

```
        [1] Process one part of type P1 through feeder line 1 using the fast simulation
model of the tandem line and store its departure time from machine L1 as $d_{L1}$.
        [2] Process three parts of type P2 through feeder line 2 and store the departure
time of third part from machine L2 as $d_{L2}$.
        [3] The earliest time at which all the required components are available at the
assembly node for next assembly is:
                max [$d_{L1}$,$d_{L2}$]
The time at which parts are removed from input buffers for next assembly is:
                max { max[$d_{L1}$,$d_{L2}$], departure time of last assembled part from the
                assembly node }
        [4] Process this assembled part through the remaining stations after the assembly
node.
        [5] This cycle of steps 1 to 4 is repeated until sufficient parts are assembled.
```

Figure 8. Pseudocode for Fast Simulation of an Assembly Topology

The above procedure can be easily generalized to handle any product structure for the assembled part. Table V shows the comparison of execution times for an assembly topology with three feeder lines. One component is required from each feeder tandem line per assembly and the assembled part is then further processed through four stations. Figure 9 shows that execution time increases "exponentially" for discrete event simulation whereas it increases only linearly for fast simulation.

TABLE V

PERFORMANCE OF FAST AND DISCRETE EVENT SIMULATION FOR AN ASSEMBLY TOPOLOGY

Total number of nodes	Execution Time (min.) Fast Simulation	Execution Time (min.) Discrete Event Simulation
35	5.17	16.1
65	9	37.25
95	10.5	66
125	13.8	102
155	17	146

Both fast and discrete event simulation models were implemented in Turbo Pascal on 386-based computer systems running at 25 MHz and equipped with numerical coprocessors.

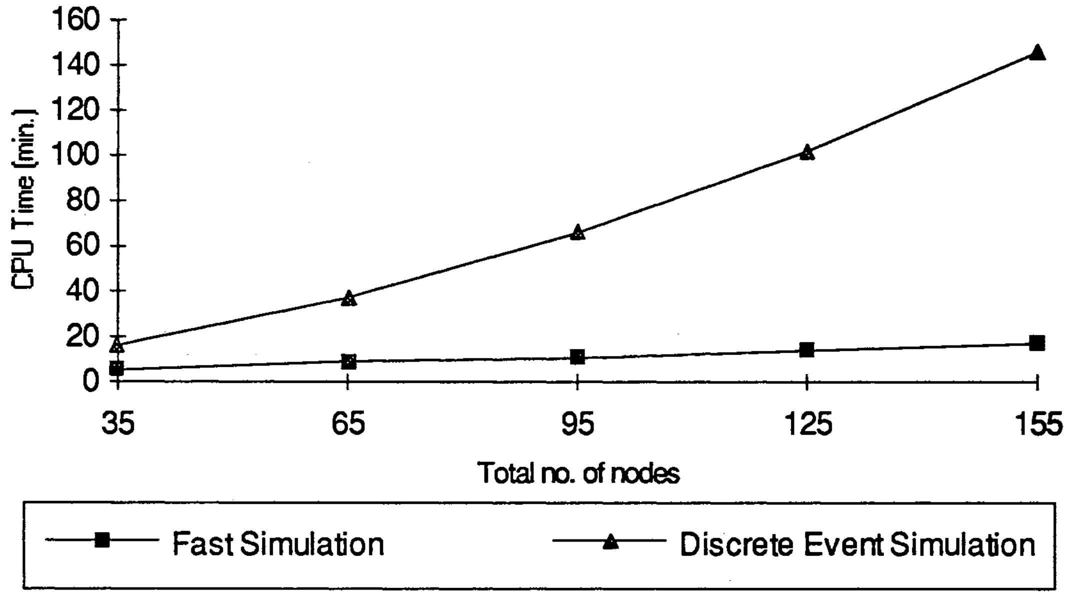


Figure 9. Comparison of Execution Time for an Assembly Topology

Fast Simulation - Unanswered Questions

1. Are any of the existing world views appropriate for fast simulation? If not, can an appropriate world view be formulated? As described earlier in this section, one can employ one of the following world views for executing discrete event simulation models:

event scheduling;

process interaction;

activity scanning.

These approaches were developed for discrete event simulation and research efforts made so far have not identified any world views for fast simulation.

2. Is the fast simulation technique capable of handling all the typical features of a manufacturing system? or When is fast simulation the most appropriate technique? Chen and Chen [1990] have demonstrated that fast simulation models can be developed for tandem lines with single server, finite/infinite buffer stations. Kamath, Bhuskute, and Duse [1992] have developed fast simulation models for tandem lines with parallel servers and infinite buffer stations. Numerous other configurations need to be studied for determining the feasibility of using the fast simulation technique.

3. If one develops fast simulation models of typical manufacturing network building blocks, then can these models be integrated to create a fast simulation model of a system which contains a combination of such network building blocks?

Hybrid Simulation

This approach to simulation gains execution efficiency by employing fast simulation models for some part of the system being simulated. Introduction of fast simulation models will tend to reduce the total number of events and the average event insertion time, thereby reducing the total simulation execution time. Following are the proposed schemes that can be used for configuration and execution of hybrid simulation models.

Simultaneous Model Execution

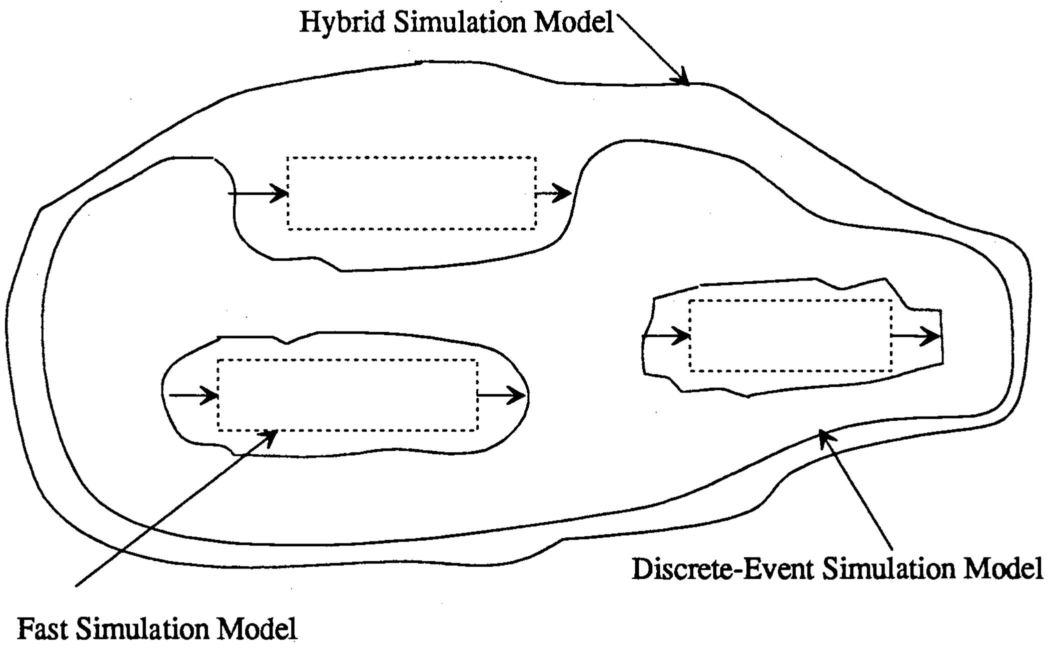


Figure 10. Simultaneous Model Execution.

In this scheme of model execution (Figure 10), the two types of models communicate by the following mechanism. The discrete event simulation (DES) model invokes the fast simulation (FS) model by passing on the part to be processed through the FS model. The FS model processes the part completely through that model and schedules an arrival at the next station which is embedded in the DES model. This method of executing the model assumes that the part, once passed on to the FS model, can be processed completely before executing the next event on the event list of the DES model. This assumption may be violated if the FS model involves features like stations with parallel server and assembly lines. Hence, configuring a hybrid model in this fashion will significantly limit the types of manufacturing configurations that can be modeled as FS models within a hybrid model.

Sequential Model Execution

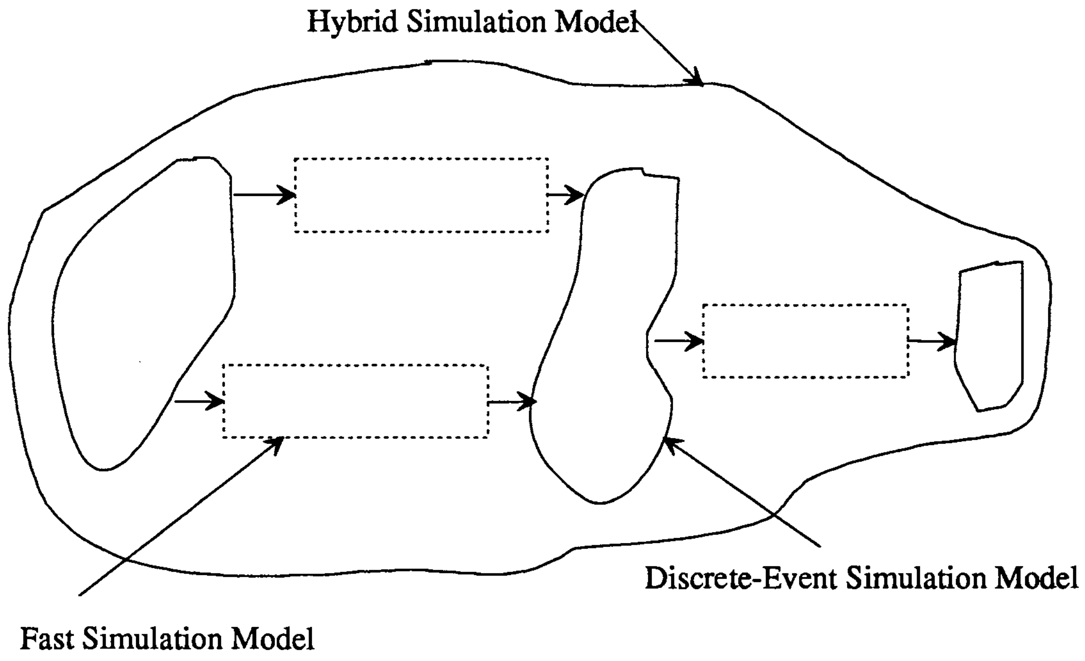


Figure 11. Sequential Model Execution

In this scheme of model execution (Figure 11), the DES and FS models are executed sequentially with each DES model having its own time advance mechanisms like event calendar and simulation clock. By employing this scheme of execution one can uncouple the FS model from the DES model which will allow a variety of manufacturing configurations to be modeled as FS models. One severe limitation of this type of hybrid model configuration is that the various DES models have to be totally independent of each other in order to allow sequential execution, i.e., the flow of material from one model to another must be unidirectional and one model must not depend on another in terms of requirement for status information. Hence, by following the above mentioned method of hybrid model configuration, one will be constrained in terms of types of manufacturing systems that can be modeled, for example, unidirectional push systems. Thus, there is a trade off between modeling flexibility and scope for increasing concentration of FS models within the hybrid model.

In order to embed certain fast simulation models inside a hybrid model, one may have to resort to the sequential model configuration scheme. However, in the sequential model execution case, the speed-up achieved by the use of FS may be offset/amplified because of the "decomposition" of the rest of the system into two or more DES models. Whether such a decomposition will offset/amplify the speed-up will be determined by the net impact of the following two factors:

[1] *Factor amplifying the speed-up*:

The average length of the event list per DES model will be reduced for some of the DES models (those which are not preceded by the FS model) and this will result in less event list manipulation overhead per event.

[2] *Factor offsetting the speed-up*:

Due to the sequential nature of the model execution, the FS model will have to initialize the event list of the downstream DES model representing arrival of all the parts exiting the FS model (in traditional DES simulation, the event list will have only one arrival event per part type rather than all the events for all the parts which enter the simulation). This will drastically increase the average event list length leading to higher event list manipulation overhead. This effect may be toned down if (i) these arrival events are stored separately from the other events which are scheduled dynamically during the run-time of that DES model, or (ii) one schedules a next arrival to the downstream DES model by reading from the departure time data provided by the preceding FS model.

Whether following such a scheme of executing a hybrid model to increase concentration of fast simulation models is attractive from the speed-up point of view or not needs to be studied in the light of the above discussion.

Hybrid Simulation - Unanswered Questions

1. Why and when would one want to create hybrid simulation models?

2. Are there any general principles or strategies that can guide the partitioning of the model into fast and discrete event segments?

3. Is one particular scheme of hybrid model execution superior to another in terms of execution efficiency?

4. Should one strive for maximum possible concentration of fast simulation models within the hybrid model?

5. Are there any implications of using hybrid models for model management and reusability?

6. Can fast simulation models be developed independently without worrying about their implications for hybrid modeling?

7. Can the speed-up to be gained be predicted based on the structure of a hybrid model? Such predictions can be used for answering trade-off questions like "Are you ready to make the approximation XYZ for creating a hybrid model if the speed-up to be achieved would be increased by so much?" This question will not be dealt with in this research and no additional assumptions will be made for creating hybrid simulation models just for the sake of speeding up the execution.

CHAPTER IV

STATEMENT OF THE RESEARCH

Research Goal

The overall goal of this research is to (i) investigate the potential and limitations of the fast simulation technique and (ii) lay the foundations for hybrid discrete event/fast simulation of manufacturing systems. In order to achieve this research goal, six research objectives have been identified.

Research Objectives

Objectives Related to the Fast Simulation Technique

OBJECTIVE 1 Identify the possible views (conceptual frameworks) that can be employed for generating fast simulation models. A subset of possible manufacturing system configurations will be investigated for identifying the set of views that can be used while developing fast simulation models. This set of views will form the input to objective 2.

OBJECTIVE 2 Determine if any one of these views has the potential to serve as the "world view" for developing fast simulation models. If a particular view identified in objective 1 is capable of handling the entire subset of manufacturing system topologies investigated, then it becomes the candidate for serving as the "world view" for fast simulation. This world view should then be evaluated in terms of its ability to handle other configurations that were not included in the subset used for this research. This research would only provide the list of candidates, if any, which have the potential to serve as the "world view".

OBJECTIVE 3 Identify manufacturing network topologies which cannot or should not be handled by the fast simulation technique. If one cannot develop fast simulation models for certain configurations, then one needs to resort to discrete event simulation. Even if one can create fast simulation models, they may not be superior in terms of their execution efficiency, in which case, discrete event simulation needs to be employed.

Objectives Related to the Hybrid Simulation Technique

OBJECTIVE 4 Determine the execution scheme that needs to be used in order to embed fast simulation models of certain manufacturing configurations within the hybrid simulation models. The type of execution scheme employed for hybrid simulation models, in turn, has implications for modeling flexibility.

OBJECTIVE 5 Demonstrate the feasibility of the hybrid simulation technique and provide a set of guidelines for identifying subsystems which are amenable to fast simulation and for creating hybrid simulation models.

OBJECTIVE 6 Test the following initial hypothesis of Diminishing Marginal Speed-up:

Given a discrete event system, if one gradually increases the "degree of hybridness" or "concentration of fast simulation models in the hybrid model" and then measures the marginal speed-up (incremental savings achieved), one may observe the following behavior:

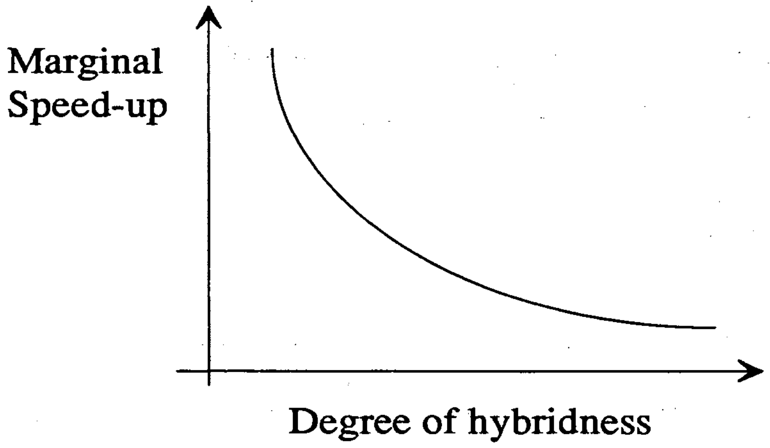


Figure 12. Diminishing Marginal Speed-up

In other words, at first, cumulative speed-up will increase at a much faster rate and then it will saturate. When operating in the saturation zone of this curve, additional efforts to increase the concentration of FS models will not provide any significant speed-up. If this hypothesis is confirmed, then one need not strive for increasing the concentration of FS models after a certain limit. This means that one can let DES handle certain features which require a sequential execution scheme if modeled as FS models and not lose significantly in terms of execution efficiency.

Research Scope and Limitations

The author's experience with the preliminary work done previously shows that the two issues viz. (i) development of fast simulation models and (ii) their use in hybrid simulation are closely interrelated. Tight coupling between these two issues implies that it would be highly ineffective to handle them in a totally independent manner. Thus, it is important to include both research domains in the agenda of this research. This restricts the investigation of fast simulation technique to a few chosen manufacturing system configurations; the actual set of configurations which will be investigated is defined in Chapter V. Numerous other types of manufacturing systems would then still remain to be investigated.

As a result of scoping the research to a limited set of configurations, this research will not be able to claim any of the views employed for developing fast simulation models as a "world view". A world view for fast simulation of manufacturing systems can probably evolve only after a fairly complete set of manufacturing systems has been investigated.

The algorithm used for event list manipulations is one of the factors which can influence the execution efficiency of discrete event simulation. This factor will be fixed at one level while comparing the performance of fast and discrete event simulation. Identifying the best data structure and algorithm for event list manipulation is not the focus of this research. Hence, the actual savings in computational time should be seen only in the context of such fixed variables.

Research Contributions

The primary contribution of this research would be the improvement in execution efficiency of simulation models. It would bring the dream of using simulation for real time control of manufacturing systems one step closer to reality. It is anticipated that the following contributions will be made to the body of modeling and simulation knowledge, while pursuing the above mentioned primary contribution:

- Realization of the potential and limitations of applicability of fast simulation technique to manufacturing systems.

- Development of fast simulation models for a subset of manufacturing system configurations.
- Proof-of-concept of hybrid (fast/discrete-event) simulation within the OSU-OOM Environment[$].
- Insights into hybrid simulation which would serve as guiding principles for the generation of hybrid simulation models.

[$] Object-Oriented Modeling Environment being developed as a part of the Advanced Modeling Methodologies research project at Oklahoma State University.

CHAPTER V

RESEARCH PLAN

Performance Measures

In the case of fast simulation, savings in simulation execution time is the primary performance measure. For hybrid simulation, performance measures can be broadly divided into two categories viz. (i) quantitative measures and (ii) qualitative measures.

Quantitative Measures

1. Simulation execution time: This is the direct measure of the execution efficiency of the simulation. The gain in execution efficiency can be measured only in the context of hardware, language used, and programming optimization.

2. Average length of event list, average number of comparisons required per insertion of new event, and total number of events scheduled: These are factors which contribute to the speed-up. These factors are absolute measures in the sense that they are independent of the hardware and language used. If one decides to rely on these measures rather than measuring execution time, then one would be guilty of neglecting other differences in execution overheads between pure DES and hybrid simulation.

Qualitative Measures

1. Difficulty/awkwardness of model generation
2. Complexity of model management
3. Ease of model modification for experimentation

Qualitative arguments will be made for and against hybrid simulation in the context of the above issues. The insight and experiences gathered during this research effort will be the main source for assessing the above mentioned performance measures.

Selection of Modeling and Simulation Environment

Several researchers have demonstrated that use of the object-oriented paradigm is much better than traditional (procedural) programming for modeling and simulation of manufacturing systems [Adiga and Gadre 1990, Beaumariage 1990, Mize et al. 1992]. There is an increasing trend towards the use of object-oriented modeling and simulation environments for the reasons of ease of modeling, higher modeling reusability, flexibility and maintainability. Hence, for this research, the OSU-OOM Environment [Bhuskute et al. 1992] will be used for both the fast and hybrid simulation. In addition to this, use of the OSU-OOM modeling environment as a test-bed for the research will also lead to enhancement of the modeling environment itself.

Research Plan

In order to satisfy the various research objectives, the research will be conducted in the following phases:

PHASE I

For identifying the potential views that can be utilized for developing FS models, the following set of manufacturing network topologies will be investigated:

1. tandem line with parallel server, infinite buffer stations
2. tandem line with parallel server, finite/infinite buffer stations
3. merge (join) and split topologies
4. assembly topology with finite buffers at the assembly station
5. tandem line with unreliable stations

If any particular view can handle all the above topologies, then it becomes the candidate for "world view". After this phase of research, objectives 1 and 2 will have been satisfied. Realization of the limitations of the fast simulation approach will occur during this stage of the research which will partially satisfy objective 3 and will also provide input for phase VI.

PHASE II

Develop fast simulation models for the above set of manufacturing topologies and validate those by creating discrete event simulation models for the same set of manufacturing topologies. Since there are no additional approximations or assumptions required for fast simulation (other than those made while creating DES models), the developed models must provide exactly the same results as those obtained by executing the DES models.

Phase III

Determine the gain in execution speed obtained by the use of fast simulation models. Since the event scheduling approach is the most execution efficient approach for discrete event simulation, the gain in execution efficiency realized by the use of fast simulation will be calculated with respect to the equivalent DES model implemented using the event scheduling approach. At the end of this research phase, one would be in a position to decide as to which manufacturing topologies should not be modeled by the fast simulation technique and thus, would contribute towards satisfaction of research objective # 3. The speed-up achieved for networks with assembly and merge configurations is mainly influenced by the speed-up obtained for tandem lines which serve as building blocks for the above configurations. Hence, the proposed experimentation deals only with tandem lines with parallel servers and tandem lines with single, unreliable servers.

Tandem Line With Parallel Server, Infinite Buffer Stations. The speed-up can potentially be influenced by the number of parallel servers at each node. To study the behavior of speed-up with respect to the above factor, experimentation is proposed in Table VI (set # 1). All the nodes have parallel servers so that the effect of modeling parallel server nodes with fast simulation can be isolated from that of modeling single server nodes. In set # 1, the average utilization of each node will be kept constant for all scenarios by adjusting the mean processing time of the parallel servers. In addition to this set of experiments, three other sets of experiments as presented in Table VI will be carried out to investigate the behavior of speed-up with respect to number of nodes, number of customers, and average utilization. In set # 4, the average utilization will be varied by varying the mean processing time of the parallel servers.

Tandem Line With Unreliable Servers. The objective is to study the behavior of speed-up with respect to different failure rates. To achieve this objective, a set of experiments will be conducted with five different failure rates keeping all other factors like number of nodes, number of customers, and repair time distribution fixed at the same level for all the scenarios.

TABLE VI

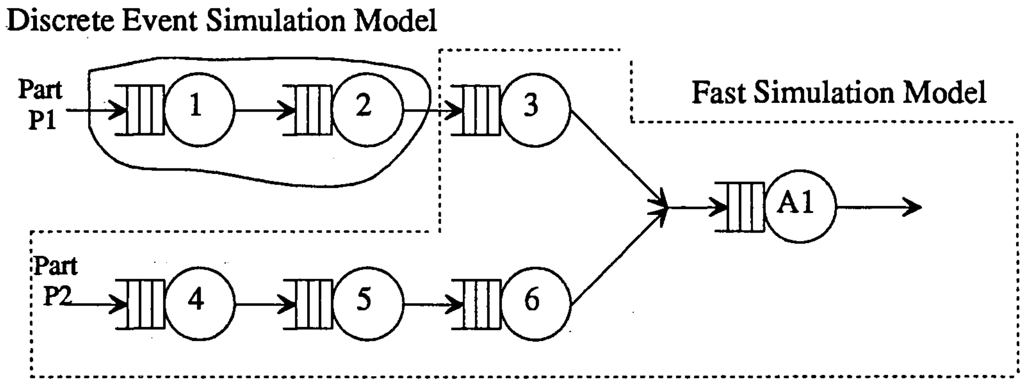
EXPERIMENTAL SCENARIOS FOR TANDEM LINE WITH PARALLEL SERVERS

Set #	No. of nodes	No. of servers	No. of customers	Average utilization
1	20	2 3 4 5 6	30,000	0.75
2	20	3	10,000 20,000 30,000 40,000 50,000	0.75
3	10 15 20 25 30	3	30,000	0.75
4	20	3	30,000	0.5 0.6 0.7 0.8 0.9

PHASE IV

Evaluate the appropriateness of views adopted for generating fast simulation models in the context of hybrid simulation. This exercise will help in identifying the implications of the view employed for developing FS models for operationalization of hybrid simulation models and will lead to fulfillment of objective # 4. The following initial thoughts about hybrid simulation would illustrate the fact that development of FS models has implications for operationalizing the hybrid approach to simulation:

Consider the scenario shown in Figure 13 in which the fast simulation model of an assembly line is embedded within the hybrid model. As described in the earlier section, fast simulation of an assembly line involves determining the departure times of the component parts from the last stations in the feeder line, i.e., machines 3 and 6 in this case. It can be seen that arrivals to machine 3, which is the first station in the feeder line, is not an external arrival and its arrival time will not be known until the appropriate event is executed in the DES model.



(All machines have infinite buffer capacity)

Figure 13. A Hybrid Modeling Scenario

One can solve this problem in two ways:

[1] Execute the two models sequentially; first run the DES model completely so that the information required for executing the FS model is made available. This approach has memory implications because one has to store all the parts departing from the DES model so that they can be introduced into the FS model. As described earlier, one also loses modeling flexibility by adopting such a sequential execution scheme.

[2] Modify the hybrid model as shown in Figure 14:

The two types of models will communicate by the following means:

DES ----> FS	activate FS model when part exits the DES model.
FS ---> DES	schedule proper events on the event list of the DES model.

The two DES models in this alternative will have a common simulation clock. Choosing this alternative means that we will not use a FS model of an assembly line in the above hybrid model even though it is possible to have one.

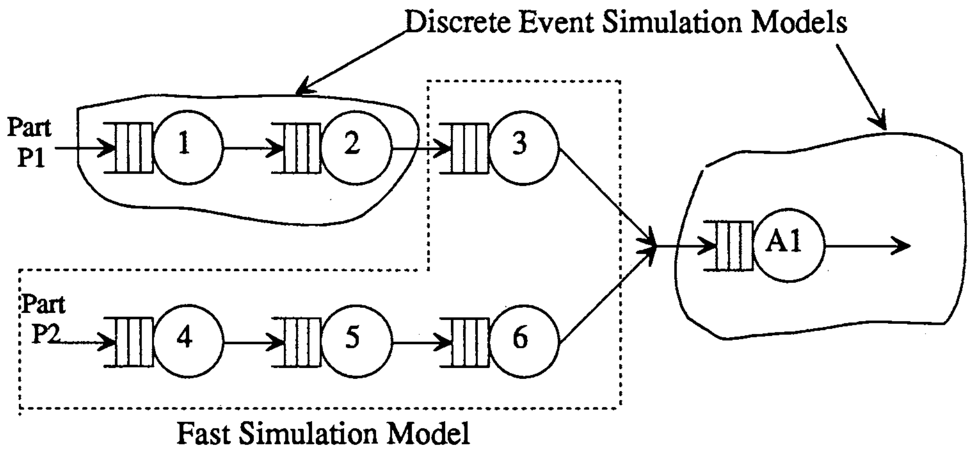


Figure 14. An Alternative Hybrid Modeling Scenario

PHASE V

Define a set of criteria for determining the need to resort to the hybrid simulation technique and for identifying the potential candidates for which FS models will be used within the hybrid simulation environment. Concepts from parallel discrete event simulation will be studied for relevance while developing such a set of criteria. Completion of this research phase will partially satisfy objective # 5. The following initial observations about the hybrid simulation models illustrate how the Parallel Discrete Event Simulation (PDES) concepts could be useful in this research phase:

Some of the problems that can arise in the case of hybrid simulation may be very similar to those with parallel discrete event simulation. The activities performed within fast simulation (though it does not maintain any simulation clock as such) can be out of sync with the main simulation clock. Preserving the causality principle, which is the key issue in PDES, seems to have significance even for hybrid simulation, especially for identifying potential candidates which can be modeled as fast simulation models.
For example, consider the system shown in Figure 15. Suppose we have developed a FS model for a tandem line which is a part of the bigger system and for some reason (may

be because of complex queue disciplines), we have decided to model the rest of the system as a DES model.

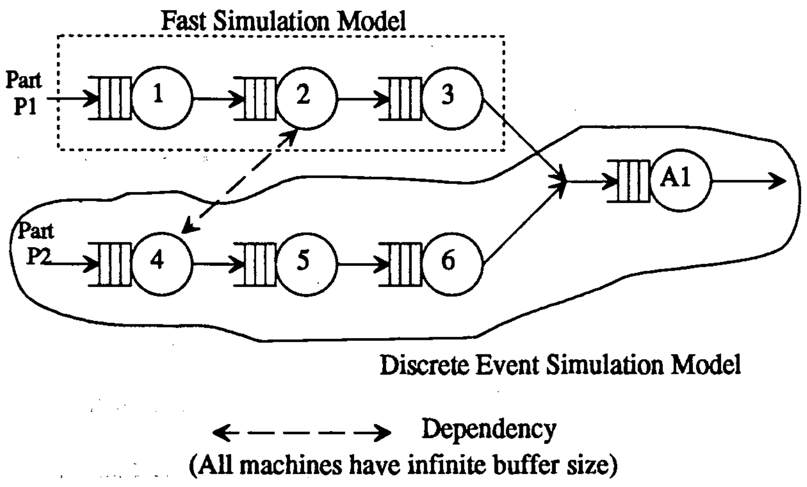

Fast Simulation Model
Part P1
1
2
3
A1
Part P2
4
5
6
Discrete Event Simulation Model
Dependency
(All machines have infinite buffer size)

Figure 15. Model Interdependency

An arrival to machine 1, say at time T1, will trigger the fast simulation model and the part will be processed through machines 1, 2, and 3 and an arrival event will be scheduled, say at time T2. This event will introduce the part into the DES model at the proper simulation clock time. Thus, the topology of the system allows creation of such a hybrid model. Now consider the case that one of the features of the system is to ask the operator at machine 2 to help the operator at machine 4 whenever failure occurs at that machine. If the failure event is to occur at machine 4 between time T1 and T2, then the arrival-event time (from FS model to DES model) T2 is no longer valid. An occurrence of a certain event in the DES model may require the undoing of some of the activities of the FS model violating the causality principle. Thus, even though the two models are to be executed on the same processor, such interactions between two heterogeneous models can be a significant problem. This example also serves as an illustration of the fact that the topology of the system is not the only factor influencing the identification of subsystems which can be modeled as FS models within a hybrid model.

PHASE VI

Make modifications to the existing OSU-OOM environment for generating and executing hybrid simulation models. Demonstrate the feasibility of a hybrid simulation technique by simulating and validating a proof-of-concept, prototype hybrid simulation model. Implementing a prototype hybrid simulation model will also provide the insight for developing guidelines for partitioning the model. Completion of this research phase would satisfy research objective # 5.

PHASE VII

Run a series of experiments to test the hypothesis of diminishing marginal speed-up. The size of the subsystem that is modeled by fast simulation is not a true indicator of the degree of hybridness. This is because not every element of the system contributes in the same proportion to the total number of events. Hence, the experiments should be designed such that all the subsets of the system which will be incrementally replaced by fast simulation models should be homogeneous in terms of their contribution to the total number of events. The manufacturing system that will be used for experimentation is depicted in Figure 16. All the assembled parts need one unit of each of the components.

In order to ensure that all the subsystems which are replaced incrementally by the fast simulation model are homogeneous in terms of their contribution to the total number of events, the following assumptions are made:

1. All part arrivals follow the same deterministic distribution for time between arrivals and processing times at each node have the same deterministic distribution.

2. Each subsytem contains the same number of nodes.

3. All buffers have infinite capacity.

Table VII shows the various scenarios of hybrid simulation that will be evaluated for testing the hypothesis of marginal speed-up.

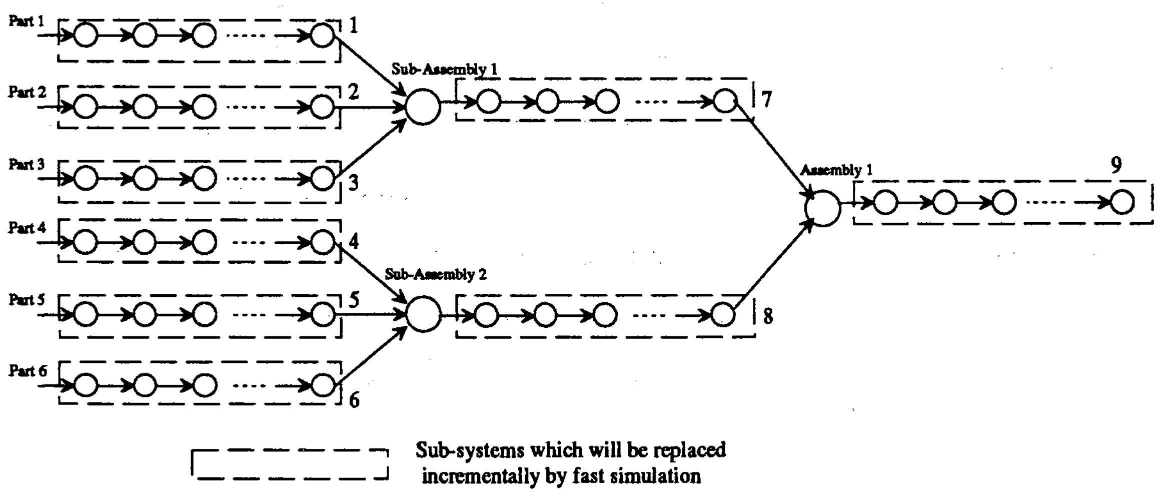


Figure 16. Experimental Manufacturing System

TABLE VII

EXPERIMENTAL SCENARIOS FOR TESTING HYPOTHESIS OF DIMINISHING MARGINAL SPEED-UP

Scenario No.	Subsystems replaced by fast simulation
1	Subsystem # 1
2	Subsystem # 1 and 2
3	Subsystem # 1 to 3
4	Subsystem # 1 to 4
5	Subsystem # 1 to 5
6	Subsystem # 1 to 6
7	Subsystem # 1 to 7
8	Subsystem # 1 to 8
9	Subsystem # 1 to 9

At the end of this research phase, objective 6 will be satisfied.

The following chapters present the outcome of the research process which was conducted within the framework of the research phases outlined above. Chapters VI and VII deal with the fast simulation technique, whereas the hybrid simulation approach is the focus of Chapters VIII and IX. The summary of the research outcomes and the contribution of this research are presented in Chapter X.

CHAPTER VI

FAST SIMULATION - CONCEPTUAL FRAMEWORKS AND MODELS

This chapter focuses on the modeling abstractions, relationships for fast simulation, conceptual frameworks which form the basis for developing fast simulation models, and the fast simulation models of various manufacturing network topologies. The topologies are presented in the order in which they were investigated which (i) portrays the evolution of various relationships and abstractions and (ii) explains the reasoning behind the need for new abstractions and conceptual frameworks. The following section defines some common nomenclature which is used hereafter in this and the following chapters.

Nomenclature

i	:	index for machine
j	:	index for part or Work Flow Item (WFI)
$ST_{i,j}$	:	Service Time of j^{th} part on i^{th} machine
b_i	:	Input buffer capacity of i^{th} machine (excluding the one with the server)
$SST_{i,j}$	:	Service Start Time of j^{th} part on i^{th} machine
		$= \max\{DT_{i-1,j}, DT_{i,j-1}\}$ [a]
$SET_{i,j}$	:	Service End Time = $SST_{i,j} + ST_{i,j}$
$DT_{i,j}$	:	Departure Time of j^{th} part on i^{th} machine

(same as $SET_{i,j}$ if input queue of downstream machine has infinite capacity)

$= \max\{SET_{i,j}, DT_{i+1,j-[b_{i+1}+1]}\}$ [b]

Split Topology

The relationships [a] and [b] defined in the previous section rely on the index of parts and hence, are not useful when dealing with topologies like split and merge. This happens because these equations assume that the sequence in which parts are processed on various machines is the same for all the machines, which is true only in the case of tandem lines with single server stations. The following example illustrates this in the context of a split topology.

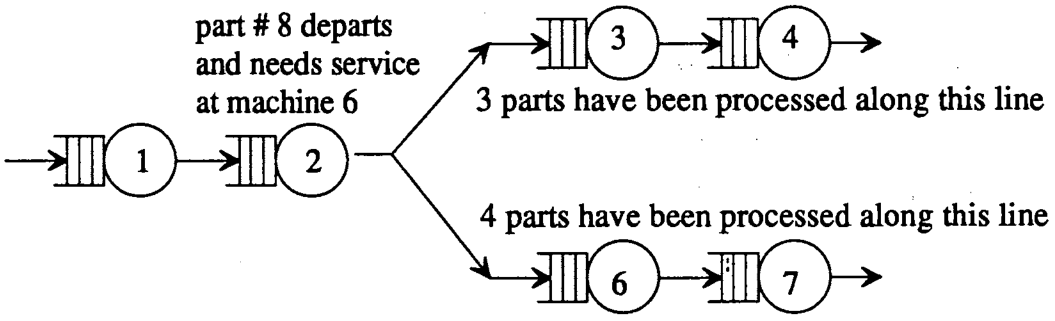

part # 8 departs and needs service at machine 6
1
2
3
4
6
7
3 parts have been processed along this line
4 parts have been processed along this line

Figure 17. A Split Topology

As seen from Figure 17, part # 8 which has finished its processing on machine 2 will be the 5th part that will be processed on machine 6. Thus, it is obvious that the index of the part loses its significance when it starts its processing along either of the two branches. Hence, it is necessary to get rid of the reliance on the part index to make such relationships general enough so that they can be employed for topologies other than a tandem line with single server stations.

Eliminating the Dependency on Part Index - Concept of Time History

The relationship [b] suggests that departure time of j^{th} part from machine i =

max [service completion time on machine i,

departure time of $j-(b_{i+1}+1)^{th}$ part from machine i+1]

Since, the real interest is in the arrival time of that part to the next machine in its routing, the jth departing part can be seen as the kth arrival to the next machine in its routing. For evaluation of blocking of the kth arrival to a particular machine, what is really required is the departure time of the k-$(b_{i+1}+1)^{th}$ departure from that machine which need not be the departure time of the j-$(b_{i+1}+1)^{th}$ part (for topologies like merge and split).

Modeling the evaluation of blocking using the above logic eliminates the dependency on the sequence in which parts are processed on various machines. This is achieved by maintaining the chronological *history of departure times* for each machine (it is more sensible to store the departure times with machines rather than with individual parts).

depTimes$_i$: Departure Time History of ith machine (maintained for last b_i+1 parts). History initialized to zero at the start of simulation; it is updated whenever departure of the part from that machine is fast simulated; it is discarded once it has been used for the evaluation of the blocking of arriving parts.

The relationship [b] then takes the following form:

$$DT_{i,j} = \max\{SET_{i,j}, \text{depTimes}_{i+1}[\text{first}]\} \qquad [d]$$

where i+1 refers to the next machine in the routing of jth part.

If this jth part is the kth arrival, then depTimes$_{i+1}$[first] provides the departure time of the k-$(b_{i+1}+1)^{th}$ departure from that machine.

The relationship [a] then takes the following form:

$$SST_{i,j} = \text{Service Start Time of j}^{th} \text{ part on i}^{th} \text{ machine}$$
$$= \max\{DT_{i-1,j}, \text{depTimes}_i[\text{last}]\} \qquad [c]$$

depTimes$_i$[last] provides the departure time of the last part processed on that machine and i-1 refers to the previous machine on that part's routing.

The relationships [c] and [d] are more general in nature and also take care of special cases such as tandem lines with single server stations. (Relationships [c] and [d] boil down to relationships [a] and [b] for this special case). The machine indices are to be interpreted according to the order of that machine in that part's routing, rather than the physical arrangement of machines.

The "customer-by-customer" view is appropriate for developing fast simulation models of split topology provided that one employs relationships [c] and [d] instead of using relationships [a] and [b].

Merge (Join) Topology

Consider the merge topology shown in Figure 18:

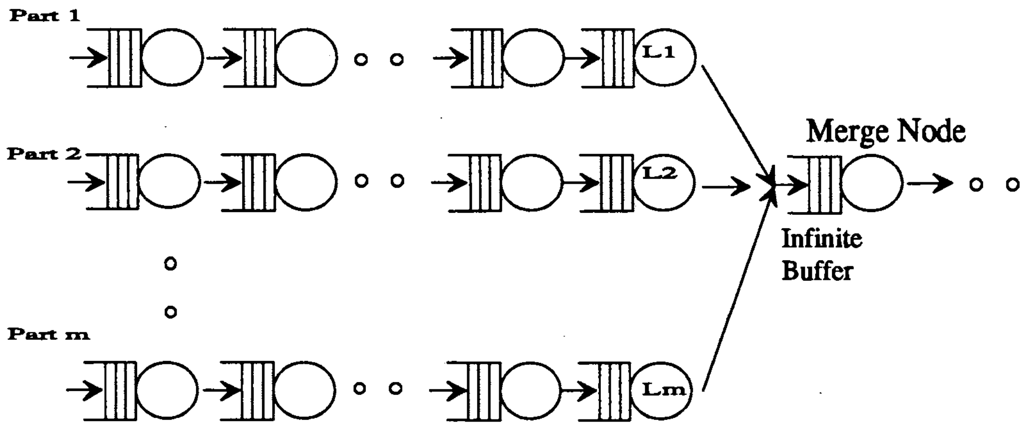


Figure 18. A Merge Topology

In the above figure, m stands for number of part types (customer classes) and L_i represents number of stations on the ith merging tandem line. In discrete event simulation, one would simulate the m different external arrival processes by scheduling initial arrival events on the calendar. The part-arrival event initiates/fires the processing of the arrived part; the part either immediately receives service or joins the queue. One can try to follow similar logic and determine the time of next arrival by comparing the values of variables representing the time of next arrival for different external arrival processes.

Assume that the first arrival is of part type 2. One fast simulates the processing of this part across the tandem line and then needs to fast simulate its processing at the merge node. But one does not know yet as to which part will arrive first at the merge node since *parts which arrive later* than this part *can reach the merge node earlier* than this part. Hence, one cannot further fast simulate the processing unless one has simulated processing of at least one part across each of the merging lines.

So, the fast simulation of this part is *temporarily abandoned* and this part is added to the list of parts maintained in the order of their departure times (from the last stations of merging lines). The *focus of fast simulation is then shifted* to the processing of next arrival until it reaches the merge node. This is repeated until all arrivals have been processed up to the stage of the merge node. Using the list of parts, one can then further fast simulate the processing across the merge node. The following points about the preceding discussion should be noted:

[1] A view similar to the "customer-by-customer" view has been employed with a difference that instead of fast simulating the processing of an arrived part in *one shot* (i.e., from the entry into the system until it exits the system), one fast simulates *only to the extent possible*. The fast simulation is carried out in two passes viz. (i) fast simulation of processing across the merging lines, and (ii) fast simulation at the merge node and thereafter.

[2] This alternative is viable if and only if the buffer at the merge node has infinite buffer capacity. If this is not so, then for determining whether parts which have completed the service at stations L_i $(i = 1, 2, ..., m)$ will be blocked or not, one needs to have the information about previous departures from the merge node. Because of the use of a two-pass approach, this information will not be available when one is fast simulating the processing across the merging lines.

The next question which arises is, what if one encounters a finite buffer at the merge node?

Merge Node With Finite Buffer

Even if one decides to abandon the fast simulation of part type 2 as one did in the infinite buffer case, one cannot determine its departure time from the L_2 station and has to abandon the fast simulation with this part still at station L_2 waiting for its "fate" (what time will it be able to depart from here?) to be figured out. Such abandoning of fast simulation will start filling the merging lines by parts with "unknown fate" and after a certain stage will prevent the progress of fast simulation. This will happen because in order to determine the time at which service can start, one needs the knowledge of departure time of previous parts and such information would not be available due to the parts which are still waiting for their fate to be decided.

Thus, it is clear that one may not want to use such arrival initiated/fired fast simulation for merge topology (especially in the case of a finite buffer at the merge node). Since determination of which part arrives first to the merge node is critical for the progress of fast simulation across the merge node, one may want to depart from such an arrival initiated approach to what will be called a "*need driven*" approach. The concept and the pseudocode for fast simulation based on this approach is illustrated in Figure 19.

```
for i = 1 to m do
        {determine $SET_i$ = service end time of next part at station $L_i$}
for "certain number of parts" do
        {minSET = min ($SET_1$, $SET_2$, ... ,$SET_m$)
        nextLine = mergingLine which has SET equal to minSET.
        determine the departure time (considering blocking) of the part corresponding to
                minSET from station $L_{nextLine}$ and process it across the merge node.
        determine $SET_{nextLine}$}
```

Figure 19. Pseudocode for Fast Simulation of a Merge Topology

The "need" refers to the need of information as to which merging lines will provide the first arrival at the merge node. One views the merge node as if it asks the feeding merging lines to process the parts as and when it requires it. Unlike the infinite buffer case, the fast simulation is carried out in one pass only but the view adapted here still differs from the "customer-by-customer" view in that the customer whose processing is next simulated across the merge node can be different from the customer whose processing was fast simulated across one of the merging lines. Hence, this view can be referred to as "*customer-by-customer-with-switching*".

Illustrative Example

Fast simulation of a merge topology based on the above framework is presented in Table VIII. The scenario used for that fast simulation is described below:

- Three tandem lines (for part types P1, P2, and P3) merge at the merge node.
- Input buffer at the merge node has a capacity of 2.
- The station downstream to the merge node has infinite capacity. Hence, at the merge node; DT = SET = SST + ST
- What happens along the tandem lines is captured only in terms of the SET on the last stations of the merging tandem lines.
- What happens beyond the merge node is not included.

Follow the trace of fast simulation from the shaded row. The last part whose processing was fast simulated across the merge node was fed by the merging line for part P1. Each of the other two lines have a part with unknown departure time (only SETs are known, 8.5 and 6 respectively). As per the need driven approach, the merging line for part P1 is asked to fast simulate the processing of the next arrival to that line. Assume that the SET of that part at the last station of the merging line is determined to be 9. *Next Part* which refers to the part corresponding to minSET is now determined to be of type

TABLE VIII

FAST SIMULATION OF A MERGE TOPOLOGY

Last Station - Merging Line (Part Type P1)		**Last Station - Merging Line (Part Type P2)**		**Last Station - Merging Line (Part Type P3)**		**Merge Node (Input Buffer Capacity = 2)**				
SET (P1)	DT (P1)	SET (P2)	DT (P2)	SET (P3)	DT (P3)	Next Part Type (min SET)	SST	ST	DT	depTimes [0,0,0]
4										
		3								
				6		P2 ~ min(4,3,6)				
			max [3,0] = 3				max [3,0] = 3	5	8	[0,0,8]
		3.5				P2 ~ min(4,3.5,6)				
			max [3.5,0] = 3.5				max [3.5,8] = 8	2	10	[0,8,10]
		8.5				P1 ~ min(4,8.5,6)				
	max [4,0] = 4						max [4,10] = 10	7	17	[8,10,17]
9						P3 ~ min(9,8.5,6)				
					max [6,8] = 8		max [8,17] = 17	3	20	[10,17,20]
				29		P2 ~ min(9,8.5,29)				
			max [8.5,10] = 10				max [10,20] = 20	1	21	[17,20,21]
		11				P1 ~ min(9,11,29)				
	max [9,17] = 17						max [17,21] = 21	4	25	[20,21,25]
33						P2 ~ min(33,11,29)				
			max [11,20] = 20				max [20,25] = 25	2	27	[21,25,27]
		30				P3 ~ min(33,30,29)				
					max [29,21] = 29		max [29,27] = 29	3	32	[25,27,32]

Note:
The fast simulation at the last station of the merging tandem line has to be abandoned even before one can determine the departure time for the part which intends to join the merge node. This is because (potentially) several other parts may try to join the queue of the merge node before it. If one disregards this fact and goes ahead to determine the departure time without ensuring that no other part can join the queue before it, then it would lead to incorrect evaluation of blocking and hence, of departure time.

P3. Fast simulation of the part along line 1 is now abandoned and the part is left at the last station with SET = 9. The focus of fast simulation is now switched to part type P3. Its departure time is calculated to be 8 as per relationship [d]. The processing of this part is then next fast simulated across the merge node using relationships [c] and [d]. The SST is determined to be 17 and DT is determined to be 20 based on the arbitrarily assumed service time of 3. The departure time history (depTimes) is then updated and the fast simulation progresses following a similar cycle.

Assembly Topology

Infinite Buffers

The concept of "simulate to the extent possible and then temporarily abandon" has to be employed for this case because the earliest start time of assembly cannot be determined until we know the arrival times of all the required components. The arrival fired/initiated approach in combination with the two-pass scheme will work only for infinite buffers but, in the case of finite buffers, this approach will again suffer from the limitations similar to those identified in the context of the merge scenario. Hence, once again we need to resort to the "need driven approach" for fast simulation. The pseudocode for fast simulation of an assembly topology with infinite buffers is presented in Figure 20 (for one example of an assembly topology, refer to Figure 7). Term n_i in the pseudocode denotes the quantity of i^{th} component required to make one assembly.

Thus, the conceptual framework of "customer-by-customer-with-switching" in which *fast simulation is continued only to the extent possible and then abandoned* also forms the basis for fast simulation of an assembly topology with infinite buffers at the assembly station.

```
[I] Fast simulate enough WFIs from each of the feeder lines and create an assembly kit
for the next assembly.
        1 to: (number of components) do: [ :i|
            a. Fast simulate $n_i$ WFIs from feeder line corresponding to ith component.
            b. Store these WFIs in a kit.]
[II] Fast simulate the assembly kit across the assembly node and thereafter.
        1. Decide earliest start time (EST) for the assembly, i.e., the time at which all the
           parts required for the assembly are available.
        2. Decide the service start time (SST) = max (EST, last departure time)
        3. Collect time in queue statistics for the WFIs in the kit.
        4. Determine the departure time for the assembly.
Repeat steps I and II until required number of assemblies have been simulated.
```

Figure 20. Pseudocode for Fast Simulation of an Assembly Topology
(Infinite Buffers at the Assembly Station)

<u>Finite Buffers</u>

The assembly topology with finite buffers looks structurally similar to the merge topology, but the following differences should be noted.

- The input buffer of an assembly station has separate queues for each type of component which goes into the assembly. Hence, in order to determine the departure time of a WFI (i.e., evaluate blocking possibility) which has finished the service at the last station of a feeder line, one does not have to be concerned about the arrivals to the assembly node from the remaining feeder lines. In the case of merge topology, due to the shared physical queue, one cannot derive the departure time from the last stations of a merging line without fast simulating all the parts that may reach the merge node earlier, across the merge node.

- The abstraction of an assembly node for developing a fast simulation model differs considerably from that of the merge node. For every part arriving at the merge node, there is a corresponding departure time history available for determining the blocking time, if any, for the parts intending to leave the merging line. In the case of an assembly node, the arriving component parts are logically destroyed (lose their identity) once they go into the assembly. Thus, the departure time history refers only to the assembled parts leaving the assembly node and not to each of the component parts that arrived. In addition to this, the assembly node at some point of time can hold more than one WFI of any component type depending upon the bill of material; which is not the case with the merge node. Also, it is realistic to assume that if a WFI of one component type is present in the queue and the assembly station is idle, it will not leave the queue until sufficient quantities of all components are available to start the assembly. Due to these factors, the departure information is not directly useful for modeling the blocking of last stations on individual feeder lines. So, an alternative relationship needs to be developed.

The whole idea behind maintaining the departure time history is to enable the determination of blocking possibility for future arrivals. In the case of tandem lines, if the $[n - (b_{i+1} + 1)]^{th}$ part has departed from the $i+1^{th}$ station before the n^{th} part finishes its service from the i^{th} station, then the n^{th} part will find a place in the input queue of the $i+1^{th}$ station and hence, will not be blocked. This decision can also be reached from the *queue removal time history* instead of *departure time history* as follows:

If the $[n - (b_{i+1})]^{th}$ part has been removed from the input queue of the $i+1^{th}$ station before the n^{th} part finishes its service from the i^{th} station, then the n^{th} part will find a place in the input queue of the $i+1^{th}$ station and hence, will not be blocked.

Thus, one needs to maintain the *queue removal time history* for each of the input buffers which will be used for evaluation of blocking of the corresponding feeder line. The reasoning for getting rid of the dependency on part index applies equally well even when the notion of queue removal time is used instead of the notion of departure time. The new versions of relationships based on the notion of queue removal time are presented below:

QRT_i : *Queue Removal Time* History of i^{th} machine (maintained for last b_i parts). History initialized to zero at the start of simulation; it is updated whenever service start time (time at which part is removed from the queue) for any part processed on that machine is fast simulated; it is discarded once it has been used for the evaluation of the blocking of arriving parts.

The relationship [b] then takes the following form:

$DT_{i,j} = \max \{SET_{i,j}, QRT_{i+1}[\text{first}]\}$ [f]

where i+1 refers to the next machine in the routing.

The departure time of the last part processed on that machine needs to be stored separately when using this relationship.

The relationship [a] then takes the following form:

$SST_{i,j} = \max \{DT_{i-1,j}, \text{time of last departure from } i^{th} \text{ machine}\}$ [e]

where i-1 refers to the previous machine in the routing.

The pseudocode for the finite buffer case needs modification to incorporate use of *queue removal time history* for evaluation of blocking and maintenance of such time history at the assembly node. The pseudocode is presented in Figure 21 (modifications are shown in italics).

In conclusion, the notion of queue removal time history is more meaningful for the development of a fast simulation model for an assembly topology, whereas the abstraction of departure time history was meaningful enough for the previous network topologies.

```
[I] Fast simulate enough WFIs from each of the feeder lines and create a kit for the next assembly.
        1 to: (number of components) do: [ :il
        a. Fast simulate $n_i$ WFIs from a feeder line corresponding to $i^{th}$ component.
The departure time from the last station of the feeder line is dictated by the queue removal time history. Since one has updated queue removal history of the $i^{th}$ input queue (for $n_i$ parts) using the SST of the last assembly, it allows one to evaluate the blocking of the next $n_i$ arrivals, even though one has not fast simulated (i.e., has not determined queue removal times) previous WFIs (at the most $n_i$-1) across assembly node.
        b. Store these WFIs in a kit.]
[II] Fast simulate the assembly kit across the assembly node and thereafter.
        1. Decide earliest start time (EST) for the assembly, i.e., the time at which all parts required for the assembly are available.
        2. Decide service start time (SST) = max (EST, departure time of last assembly)
        3. Collect time in queue statistics for the WFIs in the kit.
        4. Update the queue removal time history for all the input queues.
(queue removal time = SST, based on the assumption that parts are not removed from input queues until sufficient quantities of all the parts are available for next assembly)
        5. Determine the departure time for the assembly.
Repeat steps I and II until required number of assemblies have been simulated.
```

Figure 21. Pseudocode for Fast Simulation of an Assembly Topology
(Finite Buffers at the Assembly Station)

Illustrative Example

Fast simulation of an assembly topology is illustrated in Table IX and is based on the following scenario:

- The assembled part requires two of part type P1 and one of part type P2.

TABLE IX

FAST SIMULATION OF AN ASSEMBLY TOPOLOGY

Last station of tandem line which feeds part type P1		**Last station of tandem line which feeds part type P2**		**Assembly Station (Input Buffer Capacities : 3 for P1 and 2 for P2)**					
SET (P1)	DT (P1)	SET (P2)	DT (P2)	last DT 0	EST	SST	ST	QRT-P1 [0,0,0]	QRT-P2 [0,0]
5	max [5,0] = 5							[0,0]	
7	max [7,0] = 7							[0]	
		10	max[10,0] = 10						[0]
					max[7,10] = 10	max[10,0] = 10		[0,10,10]	[0,10]
							5		
				15					
8	max [8,0] = 8							[10,10]	
9	max [9,10] = 10							[10]	
		12	max[12,0] = 12						[10]
					max[10,12] = 12	max[12,15] = 15		[10,15,15]	[10,15]
							3		
				18					
11	max [11,10]= 11							[15,15]	
13	max [13,15]= 15							[15]	
		21	max[21,10]= 21						[15]
					max[15,21] = 21	max[21,18] = 21		[15,21,21]	[15,21]
							4		
				25					

Note:
Updating the QRT history can occur only after knowing the SST. (One updates the QRT history for each of the input buffers for n_i parts when one determines the time at which assembly can be started). Even though updating can be done only after knowing SST, the "used history" needs to be discarded once it has been used for the evaluation of blocking. Failure to do so can result in erroneous results.

- Input buffer for part type P1 has capacity 3 and that for part type P2 has capacity 2.
- What happens along the component feeder lines is captured only in terms of the SET of parts intending to leave the last stations of the feeder lines.
- What happens beyond the assembly station is not included.
- To keep this manual fast simulation simple enough, it is assumed that the downstream station after the assembly has infinite buffer capacity.

Follow the fast simulation from the row which is lightly shaded. The last departure from the assembly station has occurred at time 15. The processing of two parts of type P1 and one part of type P2 is then fast simulated. The departure times of these parts from the last stations of the feeder lines are determined to be 8, 10, and 12, respectively. The elements of queue removal time history of the assembly station (QRT-P1 & QRT-P2) which were used in the calculation of departure time are discarded. Earliest start time for the assembly kit is max(10,12)=12 and the service start time is max(12,15)=15. The component parts leave their respective queues at time 15 and this service start time is used to update the queue removal time history of the assembly station. QRT-P1 is changed from [10] to [10,15,15] and QRT-P2 is changed from [10] to [10,15]. Service time is arbitrarily assumed to be 3 and the assembled part leaves the assembly station at time 18 (lastDT of assembly station is updated) and its processing beyond the assembly station is then fast simulated.

Tandem Line - Stations With Single and Unreliable Servers

Infinite Buffers

The state transition diagram for stations with infinite buffers and single, unreliable servers based on the assumption that failures can occur even when the server is idle is depicted in Figure 22.

Nomenclature:	NFT_i	: Next Failure Time of ith machine
	TTF_i	: Time To Failure for ith machine
	RT_i	: Repair Time for ith machine

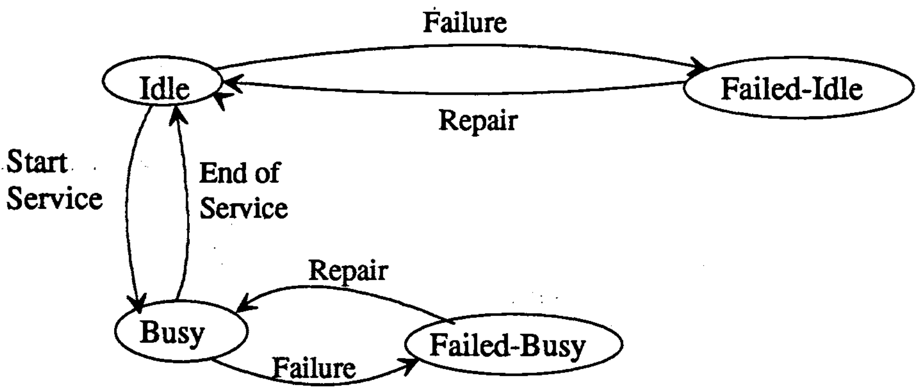


Figure 22. State Transition Diagram for Station With Failure (No Blocking)

<u>Fast Simulation Model</u>.

```
$SST_{i,j}$ = max {arrival time, last departure time} = max {$DT_{i-1,j}$, $DT_{i,j-1}$}
while ($SST_{i,j} > NFT_i$)
    { $SST_{i,j}$ = max [$SST_{i,j}$, ($NFT_i + RT_i$)]   *start of service may get delayed*
    $NFT_i = NFT_i + RT_i + TTF_i$ }
```

The above while block fast simulates "idle --> failed-idle --> idle" transitions (if any) in order to determine the earliest time at which service can start. Executing this block is equivalent to simulating a failure and a repair event. Setting the time at which next failure will occur is like scheduling a next-failure-event. *This block is not required, if one assumes that a machine cannot fail when it is idle.*

```
$SET_{i,j} = SST_{i,j} + ST_{i,j}$
while ($SET_{i,j} > NFT_i$)
    { $SET_{i,j} = SET_{i,j} + RT_i$   *service completion delayed by repair time*
    $NFT_i = NFT_i + RT_i + TTF_i$ }
$DT_{i,j} = SET_{i,j}$   (since no blocking)
```

Illustrative Example. Figure 23, which portrays the processing of WFI # 5, will help in understanding the logic of fast simulation. The following initial values are assumed: NFT = 12 time of last departure = 5 arrival time of WFI # 5 = 30

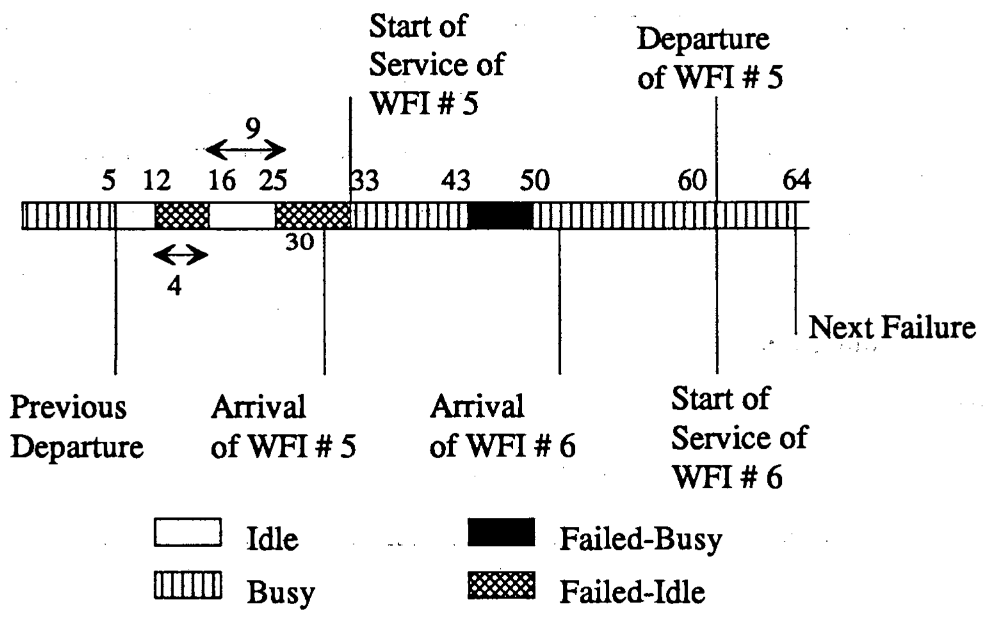


Figure 23. Station With Unreliable Server (Infinite Buffers)

SST = max { 30, 5} = 30.

SST is greater than NFT which implies that a failure will take place which can delay the start of processing and hence, "while loop" is executed (assume RT=4; TTF=9).

SST = max { SST, (NFT + RT) } = max { 30, (12 + 4) } = 30.

NFT = NFT + RT + TTF = 12 + 4 + 9 = 25

SST is still greater than NFT and hence, "while loop" is executed once again (assume RT=8 and TTF=10).

SST = max { SST, (NFT + RT) } = max { 30, (25 + 8) } = 33.

NFT = NFT + RT + TTF = 25 + 8 + 10 = 43

Now, SST is less than NFT and hence "while loop" will not be executed.

SET = SST + ST = 33 + 20 = 53.

SET is greater than NFT and hence, "while loop" is executed (assume RT=7; TTF=14).

SET = SET + RT = 53 + 7 = 60.

NFT = NFT + RT + TTF = 43 + 7 + 14 = 64.

SET is now less than NFT and hence, "while loop" will not be executed.

DT = SET = 60.

Thus, processing of WFI # 5 is completed and this will set the starting conditions (last departure = 60 and NFT = 64) for processing of the next WFI. Failures may not be so frequent in reality and hence, the "while block" may not be executed during processing of every WFI.

Finite Buffers

Assumptions: 1. Machine can fail when it is idle.

2. Machine can unblock itself even when it is failed.

The state transition diagram for stations with finite buffers and single, unreliable servers based on the above assumptions is shown in Figure 24.

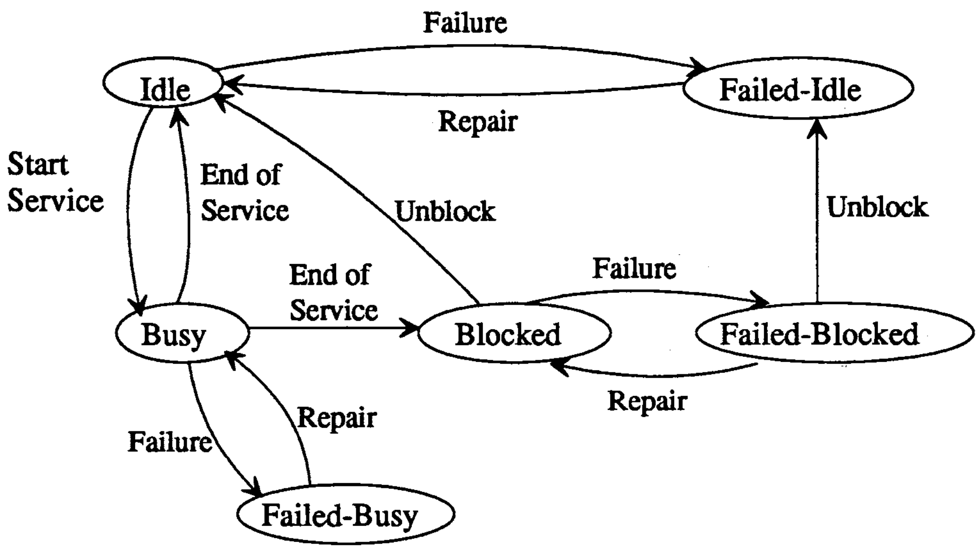


Figure 24. State Transition Diagram for Station With Failure and Blocking

Based on these assumptions, the following relationships are identified for fast simulating the processing of a WFI at such a station.

<u>Fast Simulation Model</u>.

```
$SST_{i,j} = \max \{DT_{i-1,j}, DT_{i,j-1}\}$
while ($SST_{i,j} > NFT_i$)
    {$SST_{i,j} = \max [SST_{i,j}, (NFT_i + RT_i)]$
    $NFT_i = NFT_i + RT_i + TTF_i$}
```

The above while block fast simulates the following transitions (if any)

(i) "idle --> failed-idle --> idle"

(ii) "blocked --> failed-blocked --> blocked --> idle"

(iii) "blocked --> failed-blocked --> failed-idle --> idle"

in order to determine the earliest time at which service can start.

```
$SET_{i,j} = SST_{i,j} + ST_{i,j}$
while ($SET_{i,j} > NFT_i$)
    {$SET_{i,j} = SET_{i,j} + RT_i$
    $NFT_i = NFT_i + RT_i + TTF_i$}
$DT_{i,j} = \max \{ SET_{i,j}, DT_{i+1,j-[b_{i+1}+1]} \}$.
```

This will not work !

We need another while block to take care of failures, if we work under the assumption that a machine cannot unblock itself when it is failed. This block will then fast simulate transition # (ii); whereas transition # (iii) will be invalid under this assumption.

The following section explains why the above "blocking evaluation relationship" based on the departure times may not work for stations with single, unreliable servers.

Why the above blocking evaluation relationship may not work. Consider the following trace of simulation for a tandem line shown in Figure 25:

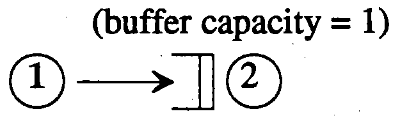


Figure 25. A Tandem Line

1. nth part departed from machine 2 at time 15 and the queue was empty at that time;
2. machine 2 failed at time 15.5;
3. n+1th part arrived at machine 2 at time 16;
4. machine 2 was repaired at time 18 & the n+1th part started its service on machine 2 at time 18;
5. n+1th part departed from machine 2 at time 23;
6. n+2nd part finished its service on machine 1 at time 17.5;

In *discrete event simulation*, the n+2nd part will be *blocked* by machine 2 since its input queue will be full (n+1th part is still in the queue at time 17.5) and will depart at time 18 when the machine is repaired. In *fast simulation*, departure time of the jth part from machine i = max [service completion time on machine i,

departure time of j-(b_{i+1}+1)th part from machine i+1]

Thus, departure time of n+2nd part from machine 1 =

max [service completion time on machine 1,

departure time of n+2-(1+1) part from machine 2] =

max [17.5, 15] = 17.5 (*not blocked*)

An error occurs in the determination of departure time!

The above relationship for calculation of departure time is based on the assumption that if the machine is not busy actively processing any part and a part is waiting to receive service at that machine, then the part will leave the queue and occupy the server no matter what the status is of that machine. This inherent (implicit) assumption that needs to be made for using departure time history for evaluation of blocking is nowhere brought out explicitly in any of the previous work on fast simulation. The logic of determining departure time based on departure time history can fail in the case of unreliable servers with finite buffer capacity because the above assumption may get violated (machine can be in failed-idle state when the part arrives and the service for that part can be started only after the machine is repaired). The transitions:

idle --> failed-idle --> idle (with queue empty when the machine fails and failure occurs before the arrival of the next part)

blocked ---> failed-blocked ---> failed-idle --> idle

violate the above assumption; whereas the transition "busy --> failed-busy --> busy" does not lead to violation of the above assumption.

What follows is an illustration of how "blocked --> failed-blocked --> failed-idle --> idle" transition can also lead to violation of the above mentioned assumption.

1. n^{th} part on machine 2 is blocked by a downstream machine at time 30;
2. machine 2 fails at time 32;
3. machine 2 gets unblocked at time 34 (can get unblocked even if failed) and n^{th} part departs at time 34;
4. machine is repaired at time 40;
5. queue is not empty at time 40;

Thus, even if the n^{th} part departs from machine 2 at time 34, the next part cannot start its service until time 40 and leads to a violation of the assumption.

It should be noted that these violations occur because one allowed failures of idle machines, unblocking of machines in the failed state, and parts to stay in the queue even when there is a space available with the server. The most obvious (intuitive) solution to this problem is to modify the departure time history to take care of the above cases in the following way:

Whenever the above transitions occur, overwrite the departure time history by the time at which the machine gets repaired (the time history maintained is no longer the pure departure time history) and use that for evaluation of blocking.

In order to find a "better and cleaner" way to achieve this, the usefulness of the notion of *queue removal times* was evaluated for this case. Fortunately, as it turned out, if one relies on the abstraction of queue removal times then one does not have to be constrained by the implicit assumption (described above) which was a prerequisite for using departure time history for evaluation of blocking. The relationship to be used for evaluation of blocking (when one employs the concept of queue removal times) is presented below:

$$DT_{i,j} = \max \{ SET_{i,j}, QRT_{i+1,j-[b_{i+1}]} \} \qquad [g]$$

Now, reconsider the first trace of simulation. In fast simulation,

departure time of j^{th} part from machine i =

max [service completion time on machine i,

queue removal time of $[j-(b_{i+1})]^{th}$ part from input queue of machine i+1]

Thus, departure time of $n+2^{nd}$ part from machine 1 =

max [service completion time on machine 1,

queue removal time of $[n+2-(1)]^{th}$ part from input queue of machine 2] =

max [17.5, 18] = 18 (*blocked*)

Correct determination of departure time!

In conclusion, the notion of *queue removal time history* is a simple but very powerful concept (much more general than the notion of *departure time history*) and should be adopted as a *generic modeling abstraction for fast simulation*. Recall that the notion of queue removal times was used effectively also for the fast simulation of assembly topology with finite capacity of input buffers at the assembly station.

Tandem Line - Stations With Parallel Servers

In the case of a tandem line with single server and finite buffer stations, a customer-by-customer view was employed for fast simulation. On the other hand, in the case of a tandem line with parallel servers and infinite buffer stations, a node-by-node view was employed to take care of the fact that the order in which customers depart from the parallel server node can be different from the order in which they arrive at that node. It was pointed out in Chapter III that the simultaneous presence of these two features (viz. parallel server and finite buffer) impose conflicting requirements in terms of the view to be adopted for fast simulation. Following is the proposed conceptual framework for fast simulation of such a tandem line.

[1] Focus on the first part and fast simulate its processing through the tandem line until it first encounters a parallel server node. One cannot further fast simulate the processing of this part because parts arriving later can depart from the parallel server earlier than this part. Hence, one has to abandon the fast simulation after *partially* fast simulating the processing at the parallel server. The phrase *partially* is used since only service end time (SET) can be determined and determining the actual departure time needs the evaluation of blocking possibility which can be done only after the parts which will depart earlier than this part are fast simulated across the parallel server node. Thus, one has to fast simulate only to the extent possible and then abandon the fast simulation. When a parallel server is "filled" with n such abandoned parts, where n = capacity of the parallel server, one will be in a position to determine which part will depart first from the parallel server (until then, one needs to loop through n such passes). This departing part may further encounter a parallel server node, in which case one may again have to abandon the fast simulation at that station. After the last parallel server node in the tandem line, the processing of the part will be fast simulated until it leaves the tandem

line. The phase of fast simulation until the first part leaves the tandem line can be referred to as a cranking phase. (The fast simulation is running but does not churn out any part or departure from the tandem line until a certain period of time has passed).

[2] After the cranking phase, all the parallel server stations are filled with abandoned parts to the level of n-1. Fast simulation of one additional arriving part will be enough to trigger the departure from all the parallel server nodes. Since there is one part with each of the servers, the one with the minimum SET will depart first and any further arrivals cannot overtake/depart earlier than this part. Once the departure of one of the n abandoned parts is fast simulated, the parallel server station will again be left with n-1 abandoned parts and the cycle repeats. The server from which departure takes place becomes the candidate for processing the next arrival. It is to be noted that, the "customer-by-customer-with-switching" view has been employed for this fast simulation where switching occurs at multiple stages, potentially one at every encounter with a parallel server node.

Note: The case of tandem line with parallel server and infinite buffers can also be handled by the above framework. Relationship [f], which does not depend on the part index and hence on the sequence in which parts are processed on various machines, was used in this topology. Queue removal time abstraction was used effectively for this topology also.

Illustrative Example

Fast simulation of a tandem line with a parallel server station based on the above framework is illustrated in Table X. Following are the terms used in that table:

TABLE X

FAST SIMULATION OF A TANDEM LINE WITH A PARALLEL SERVER STATION

Upstream Station			**Parallel Server Station** (Input Buffer Capacity = 4, Number of Servers = 3)											**Downstream Station** (Input Buffer Capacity = 2)		
PID	SET	DT	SID	SST	QRT	ST	SERVER 1		SERVER 2		SERVER 3		SPID (SID)	SST	QRT	lastDT= SST+ST
							SET	last DT	SET	last DT	SET	last DT				
					[0,0,0,0]		0	0	0	0	0	0			[0,0]	0
1	5	max[5,0] = 5	1	max[5,0] = 5	[0,0,0,5]	3	8									
2	6	max[6,0] = 6	2	max[6,0] = 6	[0,0,5,6]	4			***10***							
3	7	max[7,0] = 7	3	max[7,0] = 7	[0,5,6,7]	0.5					7.5		3 (3)			
												max[7.5,0] = 7.5		max[7.5,0] = 7.5	[0,7.5]	7.5+5.5 = 13
4	8.5	max[8.5,0] = 8.5	3	max[8.5,7.5] = 8.5	[5,6,7,8.5]	1					***9.5***		1 (1)			
								max[8,0] = 8						max[8,13] = 13	[7.5,13]	13+2 = 15
5	11	max[11,5] = 11	1	max[11,8] = 11	[6,7,8.5,11]	3	***14***						4 (3)			
												max[9.5,7.5] = 9.5		max[9.5,15] = 15	[13,15]	15+1 = 16
6	11.5	max[11.5,6] = 11.5	3	max[11.5,9.5] = 11.5	[7,8.5,11,11,5]	1.5					13		2 (2)			
										max[10,13]= 13				max[13,16] = 16	[15,16]	16+0.5 = 16.5

SID	:	ID of the Server which will provide the service (*Next server to which work will be assigned is the one which became free first*)
PID	:	Part ID
SPID	:	PID of part to which the focus of fast simulation is switched (corresponds to the one which has minimum SET)

The scenario for fast simulation depicted in Table X is described below:

- Capacity of the input queue of the parallel server station is 4.
- The parallel server station has three servers.
- The station downstream to parallel server station has input queue capacity of 2.
- What happens before the parallel server station is captured only in terms of the SET of parts at the upstream station.
- The downstream station is never blocked.

Follow the fast simulation from the row which is lightly shaded. Part (PID = 5) completes its service at upstream station at time 11 and then the QRT history of the parallel server station is used for evaluation of blocking. The part departs at time max(11,5) = 11 and is assigned to server 1 which had become free first. This part starts its service at time max(11,8) = 11. The QRT history of the parallel server station is updated and the service start time is determined to be 14 (arbitrary service time of 3 is assumed). The fast simulation of this part is then abandoned at this station. The parallel server station is now filled with 3 abandoned parts (with SETs of 10, 9.5, and 14) and hence, a departure from this station can be triggered. SPID is then determined to be 4 as it has the minimum SET of 9.5. This part (PID = 4) was abandoned before at server 3. Departure time of this part from server 3 is then determined to be max(9.5,7.5) = 9.5 using the QRT history of the downstream station. This part starts its service at downstream station at time max(9.5,15) = 15 and then the QRT history of the downstream machine is updated using this service start time. This part departs at time 16 (arbitrary service time of 1 assumed) whose processing across the remaining tandem line will be further fast simulated. The fast simulation will then proceed by focusing on the next arrival and will follow a similar cycle.

This chapter has formed the basis for developing fast simulation models of various manufacturing network topologies. The implementation of fast simulation models based on the relationships, modeling abstractions, and conceptual frameworks presented in this chapter is one of the subjects of Chapter VII. Chapter VII also deals with the validation of fast simulation models using equivalent discrete event simulation models. The results of various experiments conducted to test the speed-up obtained by the use of fast simulation approach are also discussed in the next chapter.

CHAPTER VII

FAST SIMULATION - IMPLEMENTATION, VALIDATION AND EXECUTION EFFICIENCY

Implementation

The fast simulation models for all the manufacturing network topologies were implemented in an object-oriented language viz. ObjectWorks/Smalltalk-80 R4.0. Various classes implemented for creating fast simulation models are shown in Table XI.

TABLE XI

CLASSES IMPLEMENTED FOR FAST SIMULATION

Class	Brief Description
Station	Abstract class which models common characteristics of fast and discrete event simulation resources
FSWorkStation	A single server workstation which uses fast simulation approach to process parts and can have either reliable or unreliable server
FSAssemblyStation	An assembly station which uses fast simulation approach and maintains QRT history for each of component parts
FSServer	Models an individual server of a parallel server station
FSParServerStation	A parallel server station; all the servers at this station share a common queue removal time history
TimeHistory	This construct provides for maintaining the departure or queue removal time history of fast simulation resources
TimeRecord	Individual time field in the time history

The fast simulation models were validated by running equivalent discrete event simulation models. Table XII shows the various classes implemented for developing and executing discrete event simulation models.

TABLE XII

CLASSES IMPLEMENTED FOR DISCRETE EVENT SIMULATION

Class	Brief Description
Station	Abstract class which models common characteristics of fast and discrete event simulation resources
WorkStation	A single server workstation used in discrete event simulation which can handle failures and blocking
AssemblyStation	A dedicated assembly station used in discrete event simulation which has one input buffer for each of the component parts
Server	A server which is part of a parallel server station
ParServerStation	A parallel server station used in discrete event simulation; each server is fed from a common input queue
Buffer	An input buffer of a machine which has infinite capacity
FiniteBuffer	An input buffer with a finite capacity
MultipleBuffer	Models input buffer of an assembly station
Event	Holds event time and a pointer to the event routine
EventList	A doubly linked list of events in the order of event time
Simulation	Holds resources, work flow generators, event list; drives the simulation by executing next event and coordinates statistics collection at the end of simulation

Other "utility classes" which were used in both fast and discrete event simulation are listed in Table XIII. Many of the classes described in the above tables were taken from the OSU - OOM class library [Beaumariage 1990, Bhuskute et al. 1992] and were modified as required to meet the specific needs of this research.

TABLE XIII

OTHER UTILITY CLASSES

Class	Brief Description
Operation	Represents a processing task on a particular machine
Routing	An ordered collection of operations which models the flow of a part
WorkFlowItem	A part which visits various machines for processing as per its routing; instances of this class hold information such as queue entry time and system arrival time which is used in calculation of performance measures.
WorkFlowItemGenerator	Generates arrival of parts using the specified probability distribution for inter-arrival time
RandomNew	Uniform random number generator
ProbabilityDistributions	Provide various types of random variates
StatisticsCollection Classes	Collect and summarize observations for generating statistics for various types of performance measures

Validation of Fast Simulation Models

The fast simulation models of each topology described in the previous chapter were validated by constructing and simulating equivalent discrete event models. The following performance measures were used for the purpose of validation:

1. Utilization
2. Time In Queue
3. Blocking Time
4. Time In System
5. Throughput

Summary statistics of the above measures used for validation included the following values:

1. Mean
2. Variation

3. Minimum Observation

4. Maximum Observation

5. Number of Observations

Since no approximations were made while developing fast simulation models, the results of fast simulation matched exactly with those of discrete event simulation. However, in order to get results which match exactly with those of discrete event simulation, one cannot employ a common random number generator. The following section explains this factor in detail.

Need for Using "Dedicated Random Number Generator" for Each Source of Randomness

The sequence in which random variates are generated in discrete event simulation is different from that of fast simulation. For example, while simulating a tandem line one possible sequence of random variates generated could be:

time between arrival

service time at machine 1

time between arrival

service time at machine 2

and so on

In fast simulation, when one is using a "customer-by-customer" view for simulating a tandem line with single server stations, the sequence of random variates generated would be as follows:

time between arrival

service time at machine 1

service time at machine 2

•

•

service time at machine n

time between arrival

service time at machine 1, and so on

Because of this inherent difference in the sequence in which random variates are generated, it is not possible for fast simulation to give results which are exactly the same as those obtained by discrete event simulation when one is using a single random number generator for all the variates. Hence, in order to obtain results by fast simulation which are exactly the same as discrete event simulation, multiple random number generators were used. Each source of randomness, i.e., each random variate was allotted a dedicated random number generator while ensuring that each generator in fast simulation had an initial seed value the same as that used in discrete event simulation.

Another alternative to get around this problem could be to eliminate all the sources of randomness from simulation for the purpose of validation. But rather than imposing this restriction of deterministic values on the validation process, the above mentioned solution was employed for the validation of fast simulation models.

Execution Performance of Fast and Discrete Event Simulation

All the simulations were executed on a SUN SPARCstation IPC. The execution speed (in all cases) is the mean of four observations. This was required to average out the inherent variation in CPU time (caused by the variation in initial memory occupancy status and background garbage collection). In order to reduce the overheads of generating random numbers/random variates (which is incurred in both fast and discrete event simulation), all the processing times were assumed to be deterministic. The following tables and graphs present the savings achieved by the use of fast simulation models and also show the trend in savings with respect to certain parameters. It was also ensured that the two simulation approaches were equivalent in terms of the type of statistics they collect so that neither of the two simulation approaches was penalized or favored. The set of experiments designed for testing the execution speed-up was defined in Chapter 5 (Research Phase III).

TABLE XIV

PERFORMANCE OF FAST AND DISCRETE EVENT SIMULATION FOR A TANDEM LINE WITH UNRELIABLE SERVERS

Time Between Failures	Execution Time (sec.) Fast Simulation	Execution Time (sec.) Discrete Event Simulation	Savings (%)
260	177	332	46.7
200	179	336	46.7
140	179	341	47.6
80	181	347	47.9
20	185	404	54.3

Number of stations (all with single server) = 20   Processing time = 2.5   Repair Time = 2.0

Number of parts = 20,000   Mean time between arrivals = 3.0   Infinite Buffers

Savings are defined as: (Execution time for DES - Execution time for FS)/Execution time for DES

- It can be seen from Table XIV that the use of fast simulation for tandem lines is attractive even after including the feature of unreliable servers.

TABLE XV

PERFORMANCE OF FAST AND DISCRETE EVENT SIMULATION FOR A TANDEM LINE WITH PARALLEL SERVER STATIONS (EFFECT OF NUMBER OF SERVERS)

Number of Servers	Execution Time (sec.) Fast Simulation	Execution Time (sec.) Discrete Event Simulation	Savings (%)
2	381	722	47.3
3	414	848	51.1
4	427	976	56.3
5	443	1115	60.3
6	457	1219	62.5

Number of stations = 20   Utilization = 0.75

Number of parts = 30,000   Mean time between arrivals = 4.0

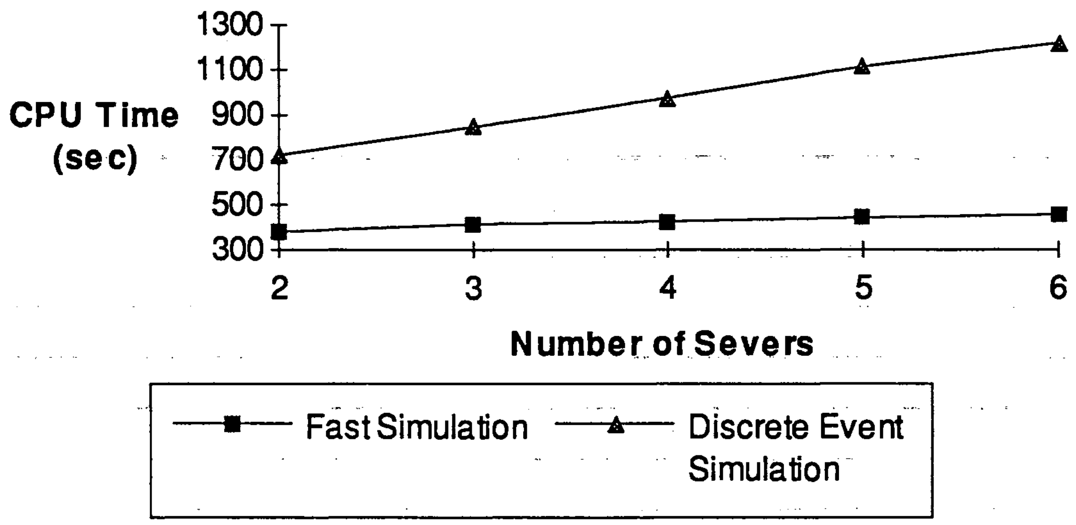


Figure 26. Comparison of Execution Times as a Function of Number of Servers (Tandem Line With Parallel Server Stations)

- Table XV shows that use of the fast simulation technique is execution efficient even for simulating tandem lines with parallel server stations.
- The execution time for both fast and discrete event simulation increases with the number of servers but the CPU time for discrete event simulation increases at a faster rate as compared to the fast simulation (Figure 26). Thus, the greater the number of servers at the parallel server station, the higher the savings achieved by employing fast simulation.
- The increase in execution time for fast simulation can be attributed to the "switching task". The higher the number of servers, the higher the time required to determine the part to which the focus of fast simulation needs be shifted. The increase in CPU time for discrete event simulation can be attributed to the fact that the average length of the event list increases with the increase in number of servers at the parallel server station.

TABLE XVI

PERFORMANCE OF FAST AND DISCRETE EVENT SIMULATION FOR A TANDEM LINE WITH PARALLEL SERVER STATIONS (EFFECT OF RUN LENGTH)

Number of Parts	Execution Time (sec.) Fast Simulation	Execution Time (sec.) Discrete Event Simulation	Savings (%)
10,000	135	273	50.6
20,000	272	545	50.3
30,000	409	815	49.8
40,000	546	1113	51.0
50,000	687	1386	50.4

Number of stations = 20	Utilization = 0.75
Number of servers = 3	Mean time between arrivals = 4.0

- The execution time increases linearly with the number of parts processed for both fast and discrete event simulation; the rate of increase for discrete event being higher than that for fast simulation (Figure 27).

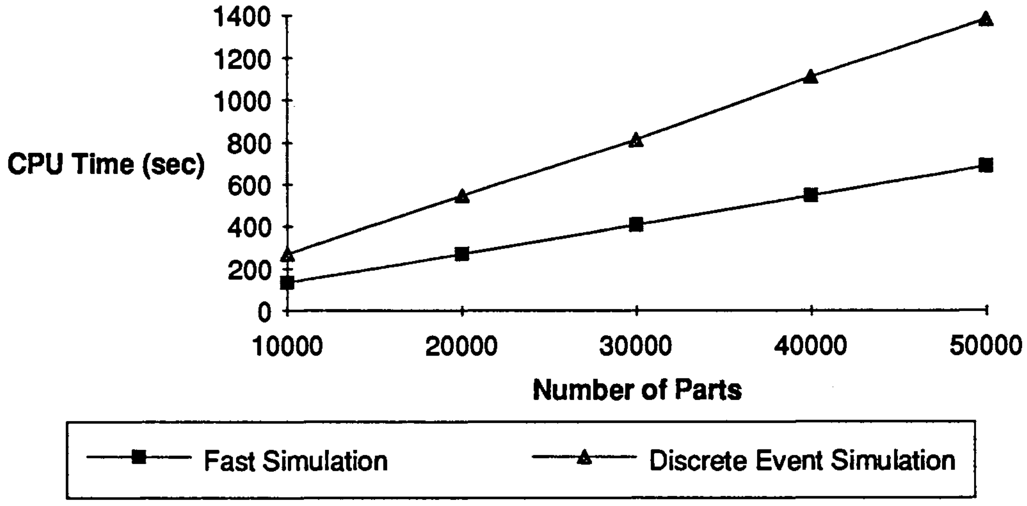


Figure 27. Comparison of Execution Times as a Function of Number of Parts (Tandem Line With Parallel Server Stations)

TABLE XVII

PERFORMANCE OF FAST AND DISCRETE EVENT SIMULATION FOR A TANDEM LINE WITH PARALLEL SERVER STATIONS (EFFECT OF NUMBER OF STATIONS)

Number of Stations	Execution Time (sec.) Fast Simulation	Execution Time (sec.) Discrete Event Simulation	Savings (%)
10	236	377	37.5
15	307	604	49.1
20	386	845	54.3
25	458	1212	62.2
30	540	1714	68.5

Number of parts = 30,000 Utilization = 0.75

Number of servers = 3 Mean time between arrivals = 4.0

- The execution time required for discrete event simulation increases drastically with the increase in number of stations (Figure 28). This is also reflected in the increase in percentage savings achieved with the increase in number of stations.

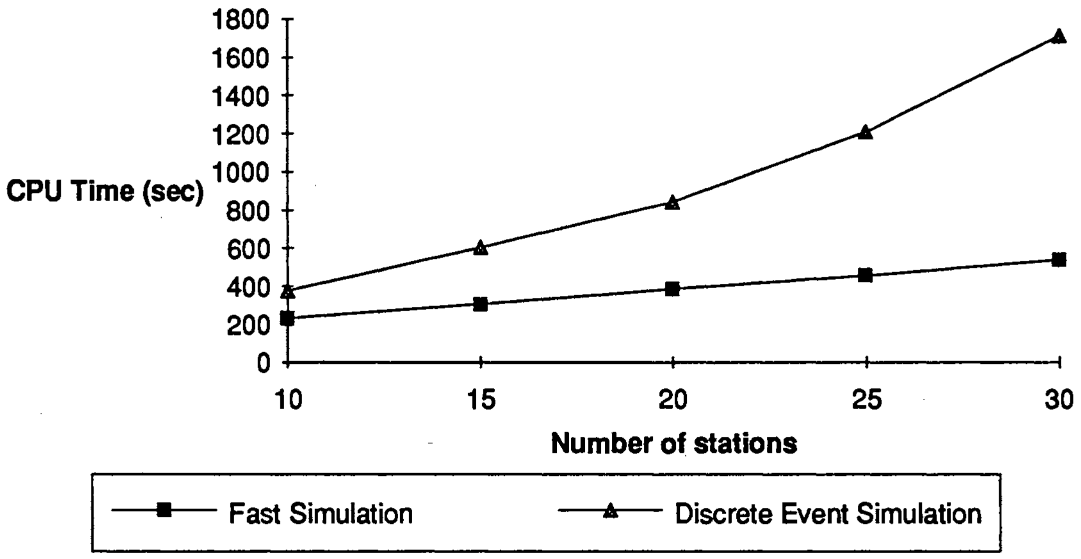


Figure 28. Comparison of Execution Times as a Function of Number of Stations (Tandem Line With Parallel Server Stations)

TABLE XVIII

PERFORMANCE OF FAST AND DISCRETE EVENT SIMULATION FOR A TANDEM LINE WITH PARALLEL SERVER STATIONS (EFFECT OF UTILIZATION)

Utilization	Execution Time (sec.) Fast Simulation	Execution Time (sec.) Discrete Event Simulation	Savings (%)
0.5	323	675	52.1
0.6	315	729	56.8
0.7	316	781	59.6
0.8	317	827	61.7
0.9	314	874	64.0

Number of stations = 20 Mean time between arrivals = 4.0 Number of parts = 30,000

Number of servers = 3

- The execution speed for fast simulation is not affected by the utilization level; whereas the execution time for discrete event simulation increases with the utilization (Figure 29). Thus, higher savings can be achieved by employing fast simulation when stations have higher utilization. Similar observations were made by Chen and Chen [1990] for tandem lines with single server stations.

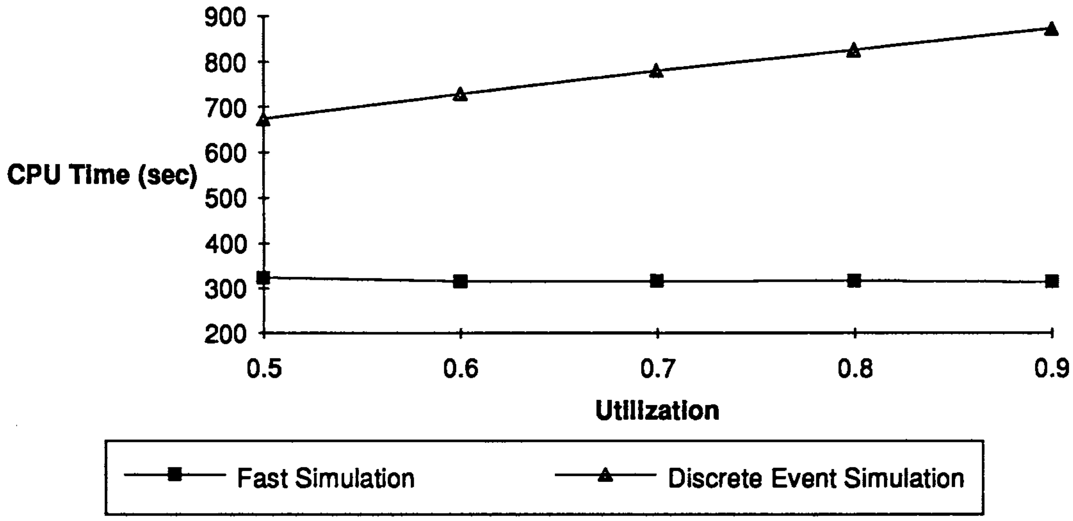


Figure 29. Comparison of Execution Times as a Function of Station Utilization (Tandem Line With Parallel Server Stations)

Thus, it can be concluded that the fast simulation models do have the potential to provide significant execution speed-up for the cases considered above.

The following chapter deals primarily with three topics viz. (i) identification of implications of fast simulation models for hybrid modeling, (ii) evaluation of appropriateness of *synchronization solutions employed in PDES* for hybrid modeling, and (iii) guidelines for partitioning the model into discrete event and fast simulation segments. Chapters VI and VII, in which issues of conceptual frameworks, models, and attractiveness of fast simulation from the viewpoint of execution efficiency were discussed, have set the stage for discussion in Chapter VIII.

CHAPTER VIII

HYBRID MODELING

Hybrid simulation models are the models in which some parts of the system are represented by discrete event simulation (DES) models whereas other parts are represented by fast simulation (FS) models. Identifying guidelines for partitioning the system model into the two different types of simulation models is important for developing hybrid simulation models and is the topic of the first section of this chapter.

Model Partitioning Guidelines for Configuration of Hybrid Models

If one views the manufacturing system as a network in which nodes represent the machines and arcs represent the flow of parts, then one can define the model partitioning (into FS and DES model segments) in terms of the cuts across the arcs of the network. Every cut will form an interface between the two different types of model segments. Hence, the guidelines for model partitioning can be described by defining the "valid cuts".

Sequential Model Execution

Since the model segments are executed sequentially, any given cut will be valid only if the input buffer of the machine that corresponds to the downstream node of the arc across which the cut is being taken (i.e., machine B in Figure 30) has infinite capacity. When parts intend to leave a particular model segment, say MS1, after finishing service from machine A, one cannot evaluate the possibility of blocking by downstream machine B since one has not yet simulated the model segment that follows MS1 (i.e., model segment MS2) and hence, the restriction of infinite buffer capacity. The same is true in the case of FS --> DES interface.

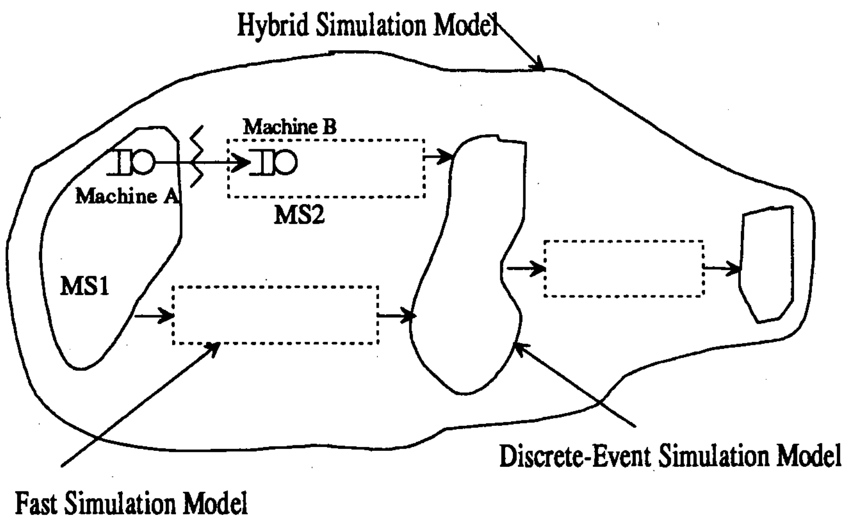


Figure 30. Hybrid Model Configured for Sequential Execution Scheme

Simultaneous Model Execution

DES --> FS Interface. When a particular part P1 intends to leave the DES model and enter the FS model, one needs to evaluate the possibility of blocking by the downstream machine (which is a part of the FS model). In this execution scheme, the processing of the parts that enter the FS model is fast simulated as and when the parts enter the FS model. This means that one has fast simulated the processing of all parts which entered the FS model previous to this part, P1. In other words, one can obtain the information from the FS model that is required to evaluate the blocking of parts intending to leave the DES model. This implies that, under this type of execution scheme, a cut that forms the DES --> FS interface is valid even if the input buffer of the machine corresponding to the downstream node of the arc (arc across which the cut is being taken) has finite capacity.

<u>FS --> DES Interface</u>. When a part enters the FS model (say at time 50), its processing is fast simulated and then it has to leave the FS model after getting processed on machine A as shown in Figure 31. One determines the service end time of this part on machine A (assume it to be 76) and then for determining the departure time, one has to evaluate the possibility of blocking by downstream machine B, which is a part of the DES model. The DES model might not have progressed up to the time corresponding to the service end time of that part (in fact, DES-time would still be 50).

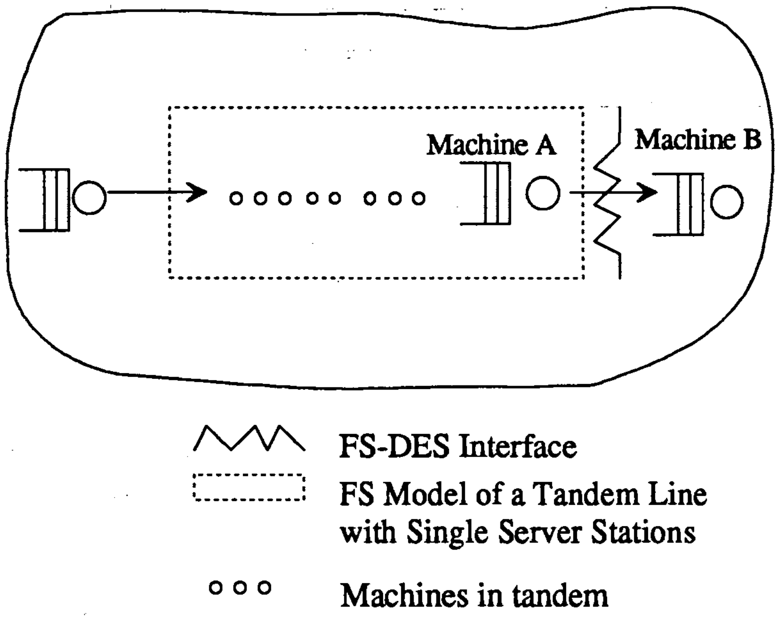


Figure 31. Hybrid Model Configured for Simultaneous Model Execution

Thus, one has no knowledge about the state of that downstream machine (machine B) at time 76 when the part intends to enter the DES model and hence, one cannot yet determine the time at which it would be able to join the input queue of machine B and also cannot schedule a corresponding event on the DES event calendar.

(This event, hereafter referred to as an "exit event", represents the exit of the part from the FS model). In such cases, one has to leave the part in the FS model with "unknown fate". Accumulation of such parts (parts with unknown fate) in the FS model can "clog/prevent" further progress of the FS model. But if the input buffer of the downstream machine B has infinite capacity, the question of blocking does not arise. The departure time of the part is equal to the service end time and hence, one does not need any knowledge of the state of the downstream machine and can schedule the "exit event" on the event calendar with event time equal to 76. In conclusion, a cut that forms the FS --> DES interface is always valid irrespective of the input buffer capacity of the machine corresponding to the downstream node of the arc (arc across which the cut is being taken).

However, dealing with the clogging phenomenon which is encountered when the downstream node of the arc (arc across which cut is taken to form the FS-DES interface) has a finite buffer imposes additional computational overhead. Hence, model partitioning based on such a cut may not always lead to the expected execution efficiency. The detailed description of the clogging process and the operational solution for dealing with clogging in the context of hybrid modeling is deferred to Chapter IX.

Relevance of Concepts From PDES

In PDES (Parallel Discrete Event Simulation), each submodel implemented on an individual processor has its own simulation clock and the synchronization between simulation clocks of various processors is achieved by establishing an appropriate communication protocol between various processors (submodels). Use of proper communication between various processors can avoid the problems of violating the causality principle in the case of PDES. Even though a FS model is like a "submodel being executed on a different processor", all the elements within the same FS model can potentially progress to different simulation times as shown in Figure 32.

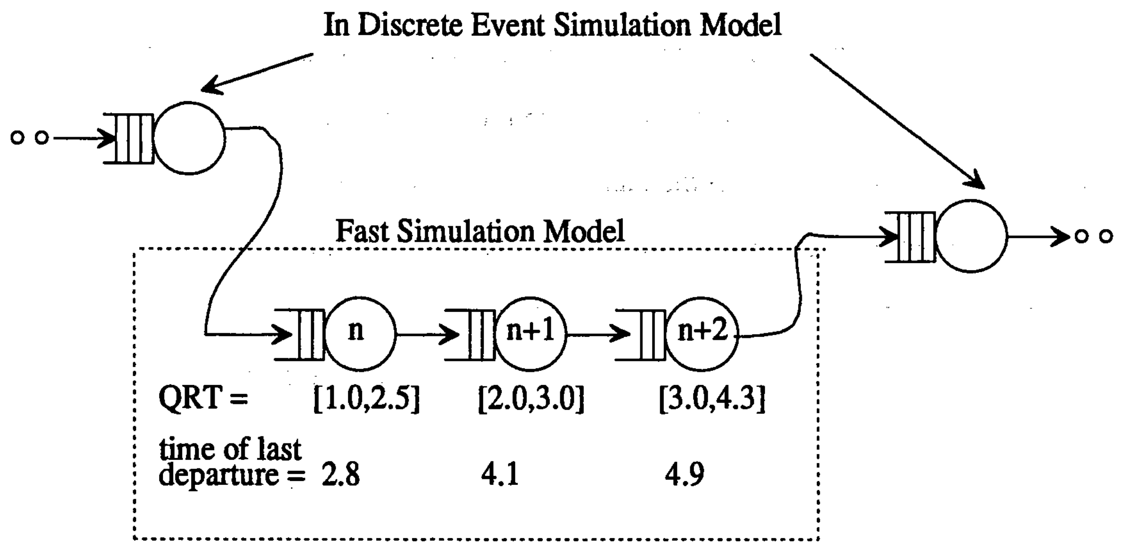

In Discrete Event Simulation Model
Fast Simulation Model
n
n+1
n+2
QRT = [1.0,2.5] [2.0,3.0] [3.0,4.3]
time of last departure = 2.8 4.1 4.9

Figure 32. A Snapshot of an Embedded Fast Simulation Model

As seen from Figure 32, machine n has progressed up to simulation time 2.8, whereas machine n+2 has reached simulation time of 4.9 and thereby has passed ahead of machine n in simulation time. Thus, there is no global simulation clock for the fast simulation model and this aspect of FS model is very different from the submodel being executed on a separate processor in PDES. Due to this lack of notion of a global simulation clock in FS, keeping the FS totally in sync with DES does not appear to be a viable alternative. Thus, though the problems in PDES and Hybrid Simulation are similar to a certain extent, the solutions are not transferrable.

Given that the FS models can go ahead of the DES model in simulation time, one should not only follow the guidelines for partitioning the model into DES and FS segments, but also ensure that the FS models are isolated from the rest of the DES model. In other words, FS models should be free from any interdependencies on the DES model.

For example, no control logic in the DES model should affect the flow of parts in the FS model.

Implications of Fast Simulation Models for Hybrid Modeling

Fast simulation models were developed in Chapter VI for a certain set of manufacturing network topologies. It was also realized in Chapter VII that these fast simulation models do really have the potential for improving the execution efficiency of simulation. In order to effectively embed previously developed fast simulation models within the hybrid models, one needs to revisit them and identify their implications for configuring and executing the hybrid models.

Tandem Line With Parallel Server Stations

Consider the hybrid model shown in Figure 33 that embeds the fast simulation model of a tandem line which has one parallel server station (three servers). Assume that two parts, p1 and p2, enter the fast simulation model (after an end of service event is executed at some station in the DES model) at DES-time 1 and 11 respectively. Further, assume that one has abandoned the fast simulation of these two parts at the parallel server station with service end times determined as 15 and 25. One cannot determine the time at which they will depart from the parallel server station until one fast simulates the processing of additional parts that enter the FS model (evaluation of blocking cannot be done correctly since one does not yet know as to which part will depart first from the parallel server station). Hence, there are two parts which have entered the fast simulation model but for which there are no "exit events" (from FS model back into the DES model) on the event calendar. Since the DES-time does not advance when one fast simulates the part within the FS Model, the DES-clock is still at time 11. Assume that when the DES-clock advances to time 38, a third part, p3, enters the fast simulation model. One fast simulates its processing along the tandem line until it reaches the parallel server station where one abandons its fast simulation after determining its service completion time to be, say, 53.

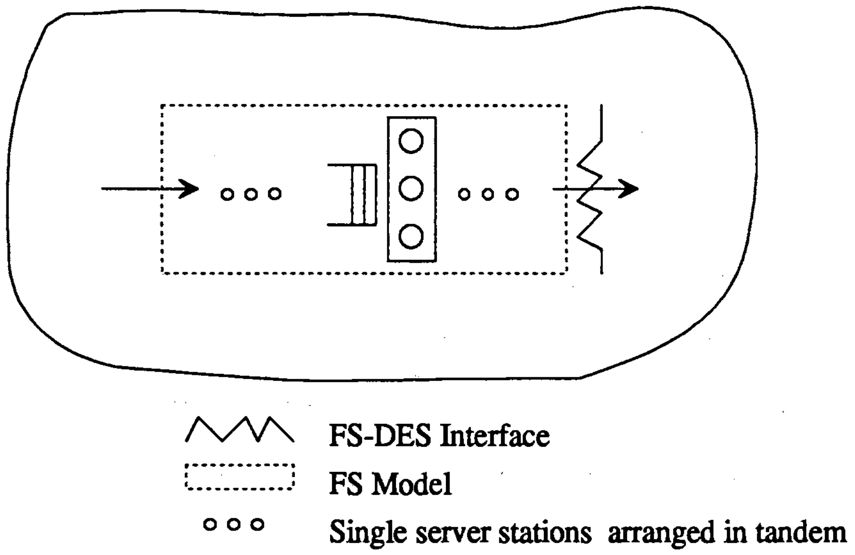


Figure 33. A Hybrid Model Which Embeds a Fast Simulation Model of a Tandem Line With a Parallel Server Station

At this stage, one is in a position to decide as to which of these three parts will depart first from the parallel server station. It turns out that part p1 will depart first and hence, its processing is fast simulated across the remaining tandem line and it is determined that this part will leave the fast simulation model (of the tandem line) at time, say, 19. To represent its entry back into the DES model, one needs to schedule an "exit event" on the DES event calendar to be executed at time 19. But, by now, the DES-clock has already advanced to time 38. This leads to the problem of "delayed events" (delayed in terms of the time at which they are scheduled on the event calendar).

Had the FS model consisted of a tandem line with all single server stations, one could have fast simulated the processing of a part which entered the FS model across the complete tandem line instantaneously. The word instantaneously means that the "exit event" can be scheduled at the same time as when the part entered the FS model. In other words, the DES-clock does not progress during the entry and exit of the part into and from the FS model. Hence, one can embed the FS model of the tandem line with single server stations within the hybrid model without encountering the problem of "delayed events".

One can solve this problem of "delayed events" by any one of the following:

(i) employing, if possible, the sequential execution scheme for the hybrid model;

(ii) excluding that segment of the tandem line which contains the parallel server station from the FS model and modeling that segment as a DES model;

(iii) establishing some way of communication between the FS and DES models. Two kinds of communication can be established between the two models:

(a) Broadcast the "simulation time advance" message to all such fast simulation models so that they can take appropriate actions. For example, in the above scenario, the fast simulation of part p1 can be restarted (i.e., evaluation of blocking can be done) when DES-time is to advance beyond time 15. This is because if the DES-time is to advance beyond 15, then no part could enter the FS model later and still depart from the parallel server station before part p1. Such broadcasting of the "simulation time advance" messages to fast simulation models at every time-advance could hamper the execution gain to be achieved by the use of fast simulation.

(b) Insert the events corresponding to service end time of the parts abandoned at the parallel server station into the event list of the DES model. In the above scenario, at DES-time 15, there would be two events on the event list of the DES model, one corresponding to part p1 (event time 15) and the other corresponding to part p2 (event time 25). The objective of inserting such events is to restart the fast simulation of parts abandoned at the parallel server station at appropriate DES-time. Before the DES-time is advanced beyond time 15, one such event would be encountered which would result in restarting the fast simulation of part p1 from the parallel server station (i.e., evaluation of blocking can now be done). This is valid because if the DES-time is to advance beyond 15, then no part could enter the FS model later and still depart from the parallel server station before part p1 departs. This form of communication would be effective only when one is dealing with fast simulation models of tandem lines with very few parallel server stations. If all the stations of a tandem line have parallel servers, then employing fast simulation models for such tandem lines may not lead to any savings in execution time.

Assembly and Merge Topology

Fast simulation models for these topologies were developed based on the "need-driven approach" as described in Chapter VI. It was pointed out in Chapter V that embedding fast simulation models of such topologies within the hybrid model would necessitate the use of a sequential model execution scheme. If the use of sequential model execution is not feasible, one has to exclude the assembly and merge nodes from the fast simulation model and model those as parts of the discrete event simulation model. Another alternative would be to stick to the arrival driven/fired approach and deal with the clogging process that would arise as a result of employing this approach.

Embedding the other topologies viz.

1. Tandem lines with single server stations
    finite or infinite buffers
    reliable or unreliable servers
2. Split topology

within the hybrid model configured for simultaneous model execution does not lead to any such problems.

The next chapter deals with the implementation of "interface interactions" that arise during the execution of hybrid simulation models. The interface interactions represent the communication between two different types of simulation models. Chapter IX also presents the experimental results for hybrid simulation which support the hypothesis of diminishing marginal speed-up.

CHAPTER IX

HYBRID SIMULATION

Classes developed for implementing fast simulation models and for validating those models using discrete event simulation were reused to implement the hybrid simulation. No additional classes were required to implement the hybrid simulation approach. Either additional behavior was added to the already existing classes or changes were made to some of the methods to model the interactions between the two types of the simulation models. The following sections describe the need for such interaction and explain how it is achieved in the context of hybrid simulation.

DES-FS Interface Interaction

Consider Figure 34 in which an end of service event is executed to represent part # 9 finishing its service at machine 2. Thus, part # 9 intends to depart from machine 2 at time 1.4 in order to receive service at machine 3 which is a part of the fast simulation model. The snapshot of the status of the discrete event model and the time history of the fast simulation model are also shown in Figure 34. Machine 2 (in DES model) communicates with machine 3 (in FS model) in order to determine if part # 9 *would have found space* in the input queue of machine 3 at this simulation time. Though machine 3 does not have the notion of status, it will respond to this query by looking into its queue removal time history. Not only can machine 3 respond to this query, but it will also be able to figure out the "end of blocking" time for machine 2, if blocking had occurred (i.e., if this part *would not have found place* in the input queue of machine 3).

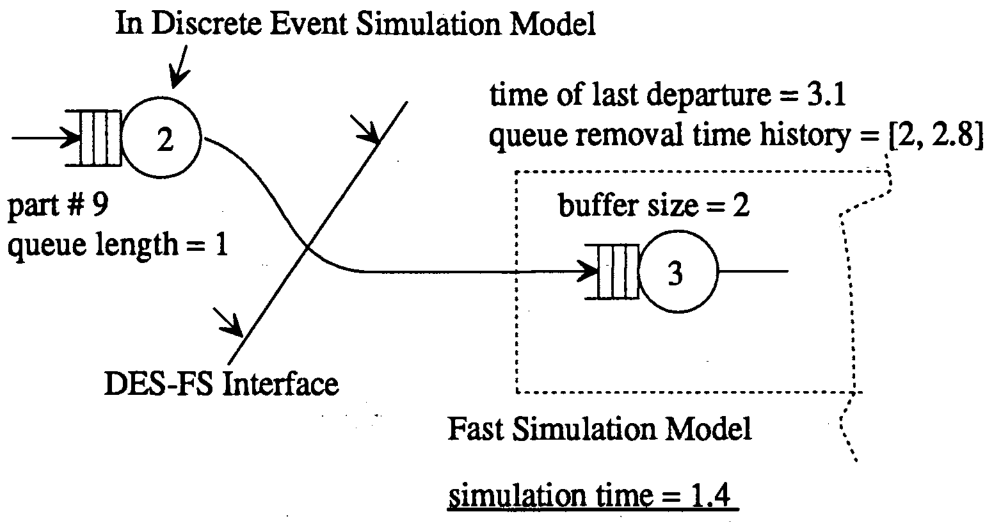

In Discrete Event Simulation Model
2
part # 9
queue length = 1
DES-FS Interface
time of last departure = 3.1
queue removal time history = [2, 2.8]
buffer size = 2
3
Fast Simulation Model
simulation time = 1.4

Figure 34. DES-FS Interface Interaction

The pseudocode which represents the logic used by machine 3 for responding to the above queries is presented in Figure 35.

```
hasInputSpaceForPart: aPart
    return true if QRT[first] of this machine is less than or equal to arrival time of
aPart  to this machine (service end time at upstream machine)

endOfBlockingTime
    return QRT[first] and discard this piece from the time history
```

Figure 35. DES-FS Interface Interaction Logic

This interaction is very different from that between two machines which are in a DES model. In the latter case, the machine getting blocked informs the blocking machine so that the blocking machine could unblock the blocked machine whenever its input queue can accept additional parts. Thus, the end of blocking is a conditional event in the sense that its time is not known in advance when the blocking begins. However, in hybrid simulation, a blocked machine has a′ priori knowledge of the time at which blocking will end and hence, can schedule a deterministic "end of blocking event". The end of blocking event is needed to reset the status of the blocked machine and to allow the blocked part to enter the FS model.

In this example, machine 3 cannot accept part # 9 at this simulation time because QRT[first] = 2 is greater than arrivalTime = 1.4 and also provides the information that this part will be blocked until time equal to QRT[first] = 2. This end of blocking time is used for scheduling an "end of blocking" event on the event list of the hybrid simulation. This event will be executed by machine 2 at simulation time equal to 2. After the execution of this event, machine 2 will be unblocked and part # 9 will be allowed to enter the fast simulation model.

FS-DES Interface Interaction

This type of interaction needs to be handled separately depending upon the buffer capacity of the DES resource at the FS-DES interface. The following two sub-sections deal with this interaction for the cases of infinite and finite buffers.

Infinite Buffer Case

As described earlier in Chapter VIII, the concept of an "exit event" was used for modeling the interaction between the FS and DES models. This is elaborated in more detail using the scenario portrayed in Figure 36. Part # 5 has entered the FS model and its processing has been fast simulated. It was determined that this part will finish its service at machine 2 at time 3.4. This part intends to leave the FS model and eventually enter the DES model to receive service at machine 3. Since machine 3 has an infinite buffer, part # 5 will definitely be able to join its input queue at time 3.4.

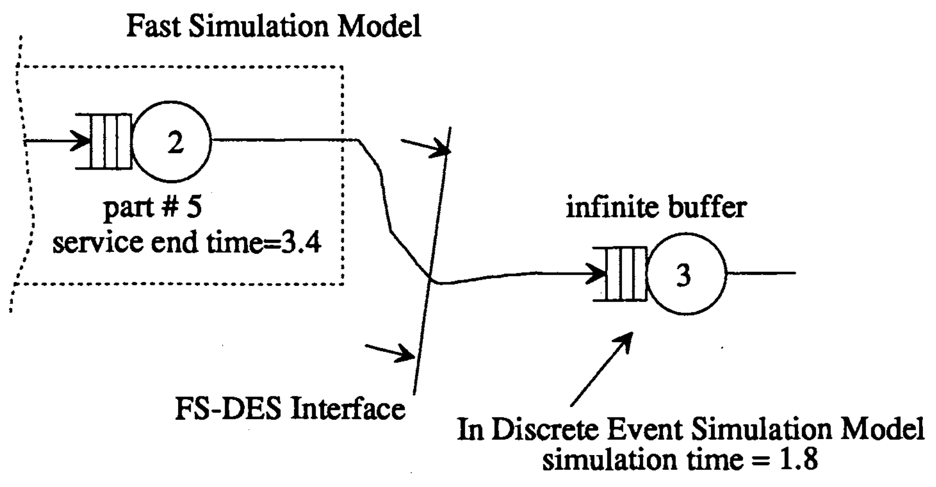


Figure 36. FS-DES Interface Interaction

The DES model has progressed only up to time 1.8 and the status of machine 3 (part # 5 joining the input queue) should only be changed at simulation time of 3.4. This is achieved by scheduling an exit event on the event list to be executed at simulation time of 3.4. The event routine corresponding to this event is to be executed by machine 3 and will simply result in accepting the part into its input queue.

Finite Buffer Case

It was mentioned in Chapter VIII that the FS-DES interface interaction in the finite buffer case can lead to the clogging of fast simulation. This section is aimed at describing the clogging process and the way to handle it in the hybrid simulation.

The Clogging Phenomenon. In order to understand why a finite buffer at the FS-DES interface leads to clogging, consider the hybrid modeling scenario depicted in Figure 37. Machines 1 and 5 are in the discrete event simulation model; whereas machines 2, 3, and 4 are part of the fast simulation model. In addition to machines 1 and 5, the discrete event segment of the hybrid model could potentially embed many other parts of the system.

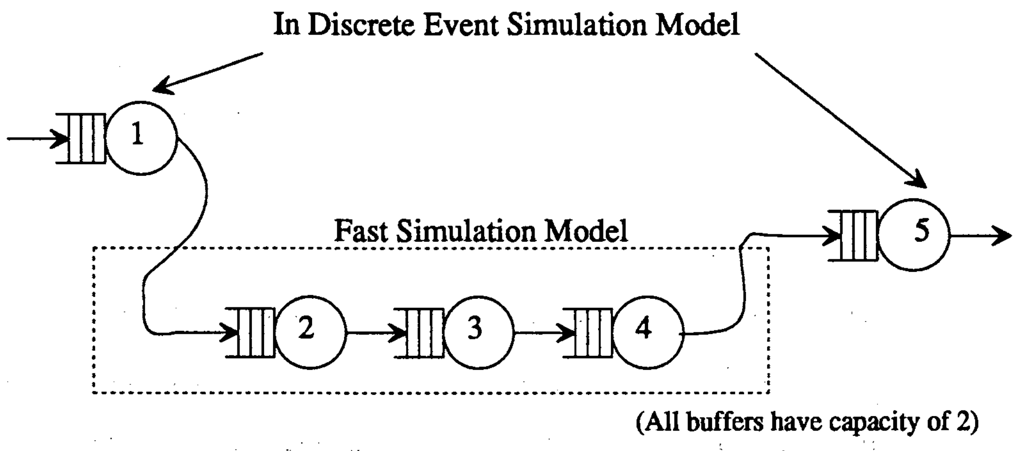

In Discrete Event Simulation Model
Fast Simulation Model
1
2
3
4
5
(All buffers have capacity of 2)

Figure 37. A Hybrid Simulation Scenario

A partial trace of a manual simulation for this scenario is shown in Table XIX. The legend for understanding this trace of simulation is presented in Figure 38.

Event Details	:	Event Type, Event Time, (machine #, part #)
		Event in italics indicates the "next-event"
Status of machine in DES model	:	{queue length, busy(b) or idle(i)}
History of machine in FS model	:	{[queue removal times], last departure time }

Figure 38. Legend for Understanding the Trace of Hybrid Simulation

At the start of the simulation, the event list had only one arrival event (arrival of part # 1 at machine 1) to be executed at simulation time 0.0. Execution of this event leads to an EndOfService (EOS) event for part # 1 at machine 1 (arbitrary service time of 1.0 assumed). An arrival event representing the arrival of the next part at time 0.5 is also scheduled on the event list. The snapshot of the hybrid simulation at time 0 (after execution of current events) is shown in the column corresponding to t=0.0. The time history of all the machines embedded in the fast simulation model is still in the initialized state. At time 0.5, another arrival event is executed as a result of which the second part joins the queue of machine 1. The snapshot after executing this event is shown in column t=0.5. The next event on the calendar is an end of service event to be executed at time

TABLE XIX

A TRACE OF HYBRID SIMULATION

t = 0.0	**t = 0.5**	**t = 1.0**	**t = 1.5**
Arr 0.5 (1,#2)	Arr 1.5 (1,#3)	*Arr 1.5 (1,#3)*	Arr 3.0 (1,#4)
EOS 1.0 (1,#1)	*EOS 1.0 (1,#1)*	EOS 2.5 (1,#2)	*EOS 2.5 (1,#2)*
		Unclog 4.0 (4)	Unclog 4.0 (4)
Machine 1 {0,b}	Machine 1 {1,b}	Machine 1 {0,b}	Machine 1 {1,b}
Machine 5 {0,i}		Machine 5 {0,i}	
Machine 2 {[0.0,0.0],0.0}		Machine 2 {[0.0,1.0],2.0}	
Machine 3 {[0.0,0.0],0.0}		Machine 3 {[0.0,2.0],3.0}	
Machine 4 {[0.0,0.0],0.0}		Machine 4 {[0.0,3.0],#1?}	
		#1 set=4, depTime=?	
t = 2.5	**t = 2.9**	**t = 3.0**	**t = 3.1**
Arr 3.0 (1,#4)	*Arr 3.0 (1,#4)*	*Arr 3.1 (1,#5)*	Arr 3.2 (1,#6)
EOS 2.9 (1,#3)	Unclog 4.0 (4)	EOS 3.15 (1,#4)	*EOS 3.15 (1,#4)*
Unclog 4.0 (4)		Unclog 4.0 (4)	Unclog 4.0 (4)
Machine 1 {0,b}	Machine 1 {0,i}	Machine 1 {0,b}	Machine 1 {1,b}
Machine 5 {0,i}	Machine 5 {0,i}		
Machine 2 {[1.0,2.5],2.8}	Machine 2 {[2.5,2.9],3.2}		
Machine 3 {[2.0,3.0],4.1}	Machine 3 {[3.0,4.1],4.5}		
Machine 4 {[3.0,#2?],#1?}	Machine4 {[#2?,#3?],#1?}		
#1 set=4, depTime=?	#1 set=4, depTime=?		
#2 qjt=4.1, qrt=sst=?	#2 qjt=4.1, qrt=sst=?		
	#3 qjt=4.5, qrt=sst=?		
t = 3.15	**t = 3.2**	**t = 3.3**	**t = 3.5**
Arr 3.2 (1,#6)	*Arr 3.3 (1,#7)*	Arr 4.2 (1,#8)	Arr 4.2 (1,#8)
EOS 3.5 (1,#5)	EOS 3.5 (1,#5)	*EOS 3.5 (1,#5)*	EOS 4.6 (1,#6)
Unclog 4.0 (4)	Unclog 4.0 (4)	Unclog 4.0 (4)	*Unclog 4.0 (4)*
Machine 1 {0,b}	Machine 1 {1,b}	Machine 1 {2,b}	Machine 1 {1,b}
Machine 5 {0,i}			Machine 5 {0,i}
Machine 2 {[2.9,3.2],4.0}			Machine 2 {[3.2,4.0],4.2}
Machine 3 {[4.1,4.5],#4?}			Machine 3 {[4.5,#5?],#4?}
#4 set=5.1, depTime=?			#4 set=5.1, depTime=?
			#5 qjt=4.2, qrt=sst=?
Machine4 {[#2?,#3?],#1?}			Machine4 {[#2?,#3?],#1?}
#1 set=4, depTime=?			#1 set=4, depTime=?
#2 qjt=4.1, qrt=sst=?			#2 qjt=4.1, qrt=sst=?
#3 qjt=4.5, qrt=sst=?			#3 qjt=4.5, qrt=sst=?

Table XIX (Continued)

t = 4.0 (step i)	**(step ii)**	**(step iii)**	**(step iv)**
Arr 4.2 (1,#8)	Arr 4.2 (1,#8)	Arr 4.2 (1,#8)	*Arr 4.2 (1,#8)*
EOS 4.6 (1,#6)	EOS 4.6 (1,#6)	EOS 4.6 (1,#6)	EOS 4.6 (1,#6)
EOS 6.1 (5,#1)	EOS 6.1 (5,#1)	EOS 6.1 (5,#1)	EOS 6.1 (5,#1)
	Unclog 4.4 (4)	Unclog 4.4 (4)	Unclog 4.4 (4)
Machine 1 {1,b}	Machine 1 {1,b}	Machine 1 {1,b}	Machine 1 {1,b}
Machine 5 {0,i}	Machine 5 {0,b}	Machine 5 {0,b}	Machine 5 {0,b}
Machine 2 {[3.2,4.0],4.2}	Machine 2 {[3.2,4.0],4.2}	Machine 2 {[3.2,4.0],4.2}	Machine 2 {[3.2,4.0],4.2}
Machine 3 {[4.5,#5?],#4?}	Machine 3 {[4.5,#5?],#4?}	Machine 3 *{[4.5,#5?],5.1}*	Machine 3 *{[4.5,5.1],#5?}*
#4 set=5.1, depTime=?	#4 set=5.1, depTime=?	*#4 set=5.1, depTime=5.1*	*#5 qjt=4.2, qrt=sst=5.1 set=5.7, depTime=?*
#5 qjt=4.2, qrt=sst=?	#5 qjt=4.2, qrt=sst=?	#5 qjt=4.2, qrt=sst=?	
Machine 4 *{[#2?,#3?],4.0}*	Machine 4 *{[4.1,#3?],#2?}*	Machine4 {[#3?,#4?],#2?}	Machine4 {[#3?,#4?],#2?}
#1 set=4, depTime=4	*#2 qjt=4.1, qrt=sst=4.1 set=4.4, depTime=?*	#2 set=4.4, depTime=?	#2 set=4.4, depTime=?
#2 qjt=4.1, qrt=sst=?	#3 qjt=4.5, qrt=sst=?	#3 qjt=4.5, qrt=sst=?	#3 qjt=4.5, qrt=sst=?
#3 qjt=4.5, qrt=sst=?		*#4 qjt=5.1, qrt=sst=?*	#4 qjt=5.1, qrt=sst=?

equal to 1.0. Part # 1 can depart from machine 1 (no blocking) and enters the fast simulation model. Part # 2 starts its service at machine 1 leading to the scheduling of an end of service event to be executed at time 2.5 (service time of 1.5 assumed). Flow of part # 1 across machines 2, 3, and 4 is fast simulated as per the relationships [e] and [f] defined in Chapter VI and the time history of all the machines in the fast simulation model is correspondingly updated (arbitrary values of service times were assumed). This new time history can be seen from the snapshot of the hybrid simulation presented in the column corresponding to t=1.0. However, at machine 4 we can determine only its service end time (SET=4.0) and not the departure time. This happens because the necessary status information (i.e., queue size of machine 5 at time equal to 4.0) is not available to the fast simulation model at this simulation time. Thus, one needs to abandon this part at machine 4 in the fast simulation model with "unknown fate". The unknown piece of time history in the fast simulation model is shown by question marks.

Backpropagation of Clogging. As seen from the above example, first machine 4 gets clogged when part #1 cannot exit from the FS model back into the DES model because the DES and hence machine 5 has not yet progressed up to simulation time t=4.0. As the hybrid simulation progresses, an arrival event is executed at time 1.5 and part # 3 joins the queue of machine 1. The status of machine 1 at this time is shown in column t=1.5. An end of service event is executed at time 2.5 and part # 2 departs at the same time from machine 1 (no blocking) and its flow across machines 2, 3, and 4 gets fast simulated. The updated status of machine 1 and updated time history of machines 2, 3, and 4 can be seen from the column corresponding to time 2.5. However, at machine 4, the QRT or SST of part #2 cannot be determined since the time of the last departure from machine 4 is not yet known (only the queue join time, QJT, is known to be 4.1). Thus, the progress of fast simulation is again prevented at machine 4 and part # 2 is also left with unknown fate at machine 4 in the fast simulation model. Similarly, part # 3 enters the fast simulation model at time 2.9 and is left with unknown fate at machine 4 due to the inability to determine its QRT or SST. After executing two arrival events at time 3.0 and 3.1, an end of service event is executed at time 3.15 when part # 4 enters the fast simulation model and its flow across machines 2 and 3 gets fast simulated. However, departure time of part #4 from machine 3 cannot be determined because doing so requires knowledge of the QRT history of machine 4 (specifically QRT of part #2) which is not yet known. Thus, the "clogging phenomenon" further backpropagates from machine 4 to machine 3. As the hybrid simulation progresses, two arrivals occur at time 3.2 and 3.3 and then, at time 3.5, part # 5 finishes its service at machine 1 and enters the fast simulation model. However, at machine 3, the QRT or SST of part #5 cannot be determined since the time of last departure from machine 3 is not yet known. Thus, the progress of fast simulation is again prevented at machine 3 and part # 5 is left with unknown fate at machine 3 in the fast simulation model. Such clogging can potentially backpropagate until it completely clogs machine 2.

Dealing With Clogging. As seen from the above discussion, one can manage to keep up the progress of hybrid simulation even though the clogging occurs within the fast simulation model. What is needed is some mechanism which will ensure that the fast simulation model is unclogged as soon as the required piece of information is available. To achieve this, the concept of an "Unclog Event" is proposed. Such an unclog event is put on the event list when DES cannot provide the required status information to the FS model. Such an unclog event was put on the event list at time 1.0 when fast simulation of part # 1 at machine 4 required the status information of machine 5, which is a part of the DES model. This unclog event is to be executed by machine 4 and is shown in the column corresponding to time t=1.0. Unclog event time represents the time at which required status information would be available to the FS model. For example, the unclog event in the above hybrid simulation scenario is scheduled to be executed at time 4.0

because that is the time when DES would provide the required status information to the clogged fast simulation model. The unclogging process is initiated as a result of executing such an unclog event. The following section throws more light on how this unclogging process works.

The Unclogging Process. The unclog event is to be executed by the machine which scheduled it. Since the clogging started at machine 4 and then backpropagated, the unclogging should start at machine 4 and then a backward pass should be initiated to give all the upstream machines a chance to unclog themselves based on the newly available time history of downstream machines. When the unclog event is executed at time 4.0, machine 5 may accept part # 1 which has finished its service at machine 4 or it may block machine 4 depending upon its status at that time. In general, execution of the unclog event will lead to one of the following two cases:

[1] Machine in the FS model at the FS-DES interface will be blocked and the exact departure time of the blocked part still will not be known. Thus, execution of the unclog event does not provide any additional knowledge about the "unknown piece of time history" and hence, the unclogging process cannot yet be initiated.

[2] No blocking is encountered and the exact departure time of the part which was left with unknown fate (with the machine at the FS-DES interface) can now be determined. Availability of this new piece of time history implies that the unclogging process can be triggered at this simulation time.

The code for the routine (method) to be implemented for executing the unclog event is shown in Figure 39. In the hybrid simulation example being considered, part # 1 will be accepted by machine 5 (queue is not full) and the time of last departure from machine 4 is determined to be 4.0. The updated time history at this stage is shown in the column corresponding to t=4.0 (step i). This situation belongs to case [2] described above and hence, will result in initiation of the unclogging process. Machine 4 has two parts (parts #

2 and 3) with unknown QRT. QRT time of part # 2 is determined to be 4.1 and its service end time is calculated (SET=4.4). However, its departure time cannot yet be decided and hence, at this stage, yet another unclog event with event time 4.4 is scheduled on the event list.

```
unclogSelfAndBackPass
self isInCloggedStatus
ifTrue:
        [mac := next station in the routing of WFI@ left with unknown departTime (unknownDepWFI).
        mac isDESResource
        ifTrue:
                [ this machine is at the FS-DES interface
                (mac hasInputSpaceFor: unknownDepWFI)
                ifTrue:
                        [ case (ii) - can trigger unclogging
                        lastDepartTime := currentSimulationTime.
                        mac provideServiceTo: unknownDepWFI.
                        qOfUnknownQRT isEmpty
                        ifFalse:
                                [ some wfi was left with this machine with unknownQRT
                                aWFI := qOfUnknownQRT removeFirst
                                self provideServiceTo: aWFI.
                                give upstream machines a chance to unclog themselves
                                aWFI upStreamStn unclogSelfAndBackPass]]
                ifFalse:
                        [ case (i) can not unclog yet
                        Inform mac that it is blocking this machine]]
        ifFalse:
                [ this machine is followed by another FSWorkStation
                Determine the departure time of unknownDepWFI using relation for blocking.
                Collect blocking statistics and update lastDepTime
                mac provideServiceTo: unknownDepWFI.
                qOfUnknownQRT isEmpty
                ifFalse:
                        [aWFI := qOfUnknownQRT removeFirst.
                        self provideServiceTo: aWFI.
                        aWFI upStreamStn unclogSelfAndBackPass]]
ifFalse:
        [ this machine is not in clogged status
        Do nothing]
                                                        @ WFI (Work Flow Item) is synonymous with Part
```

Figure 39. Smalltalk-80 code for the Unclog Event Routine

The snapshot of the simulation at this stage is shown in the column titled "step ii". The control of the unclogging process is then backpassed to machine 3 and thus, it is given a chance to unclog itself. Machine 3 then interacts with machine 4, which is also in the FS model, to determine the departure time of part # 4 using the new QRT history of machine 4. The departure time is then calculated to be 5.1 using the standard fast simulation relationships. The processing of part # 4 at machine 4 is then fast simulated but is immediately abandoned because the QRT time cannot yet be determined. This happens due to lack of knowledge of the time of last departure from machine 4 and thus, part # 4 is then left with unknown QRT at machine 4. The status of the hybrid simulation is depicted in the column titled "step iii". Machine 3 then starts fast simulating the processing of part # 5 and determines its QRT to be 5.1 and service end time to be 5.7. The departure time of this part from machine 3 cannot be determined because of the unknown piece of information in QRT history of machine 4. Thus, part # 5 is left with unknown departure time at machine 3. Machine 2 is then given a chance to unclog itself but it is not in the clogged state and hence, does nothing. The final snapshot of the simulation at the end of the unclogging process is shown in the column titled "step iv". The backpass of the unclogging process ends here and the hybrid simulation proceeds further by executing the next event on the event list.

In case (i) where the blocking occurs, the FSWorkStation at the interface would be unblocked by the downstream DESWorkStation at some future time. At that simulation time, the FSWorkStation at the interface would execute the same unclogSelfAndBackPass routine to initiate the unclogging process.

In this hybrid simulation example, the clogging could have backpropagated right up to machine 2 and the QRT time history of that machine would have had all unknown elements. In such a case, it would disable machine 1 to schedule the "end of blocking event". Machine 1 also, in this sense, could get clogged. During the unclogging process, control of the unclogging process would also be backpassed to machine 1 giving it a chance to schedule the "end of blocking event" using the newly available QRT history of machine 2. The code for this is shown in Figure 40.

```
unclogSelfAndBackPass
If this machine was blocked without scheduling the endOfBlocking event, then
schedule it with the newly available information. No backpassing required.
        isClogged
                ifTrue:
                        [mac := nextStation on the routing of blocked part.
                        Determine end of blocking time by interacting with mac.
                        Schedule the "end of blocking event"
                        Set status to unclogged]
                ifFalse:
                        [nothing to be unclogged
                        unclogging process terminates here]
```

Figure 40. Unclogging the DESWorkStation

Thus, "unclog event" provides a mechanism for taking care of the finite buffer interface interaction between the FS and DES models. It allows one to take a cut along the arc whose downstream DESResource has a finite buffer.

Hypothesis of Diminishing Marginal Speed-up

An initial hypothesis of diminishing marginal speed-up was formulated in Chapter IV and is reproduced below:

"Given a discrete event system, if one gradually increases the *degree of hybridness* or *concentration of fast simulation models in the hybrid model* and then measures the marginal speed-up (incremental savings) achieved, one may observe that this marginal speed-up decreases as the degree of hybridness increases".

A series of experiments based on the experimental manufacturing system defined in Figure 16 (Chapter V) was conducted for hybrid simulation. The results of this hybrid simulation were validated by an equivalent pure discrete event simulation model. The experimental results are presented in Table XX which support the initial hypothesis. The marginal speed-up or incremental saving is defined as the difference between execution time for "nth degree of hybridness" and that for "n+1th degree of hybridness". The values for simulation execution times are averages of ten observations.

TABLE XX

EXPERIMENTAL RESULTS SUPPORTING THE HYPOTHESIS OF DIMINISHING MARGINAL SPEED-UP

Degree of Hybridness	Subsystems Replaced by Fast Simulation Model	Simulation Execution Time (Seconds)	Incremental Savings (Marginal Speed-up)
0	none	1523.1	-
1	1	1393.5	129.6
2	1 and 2	1285.7	107.8
3	1 to 3	1201.9	83.8
4	1 to 4	1115.6	86.3
5	1 to 5	1041.5	74.1
6	1 to 6	966.4	75.1
7	1 to 7	909.8	56.6
8	1 to 8	856.8	53
9	1 to 9	800.9	55.9

Number of stations per subsystem = 5 Time between arrivals for each part type = Deterministic 5.5
Number of arrivals for each part type = 20000 Service time at each station = Deterministic 5.2

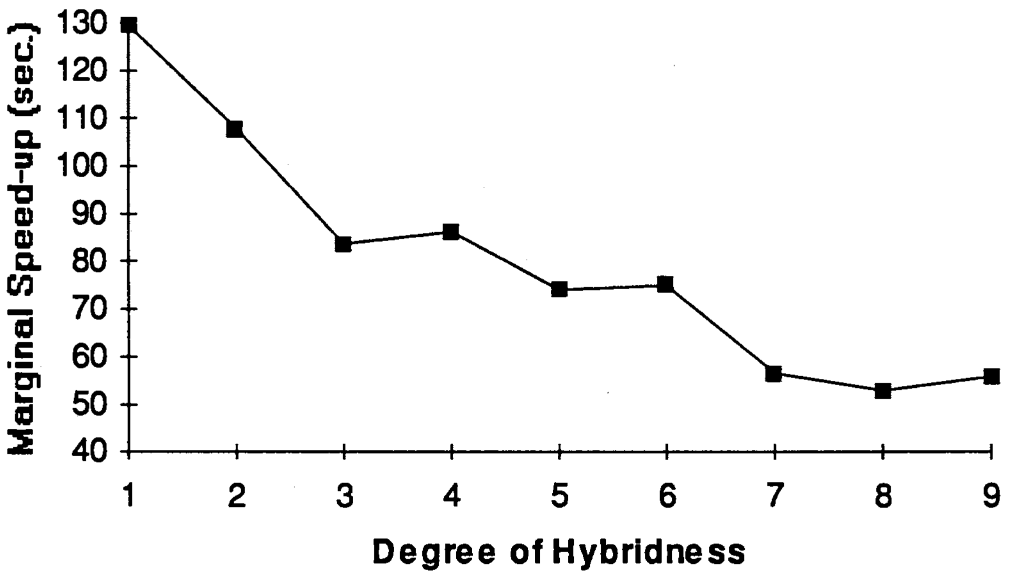


Figure 41. Marginal Speed-up vs. Degree of Hybridness

It can be seen from Figure 41 that, overall, there is a decreasing trend in marginal speed-up with respect to an increase in the degree of hybridness.

Some Qualitative Remarks About Hybrid Simulation

The hybrid simulation approach introduces additional complexities during the processes of model development, scenario modification, and model maintenance. The following sections explain how these additional complexities arise during the above three processes.

Model Development Process

While creating hybrid simulation models, one has to deal with the following phases which are not encountered during the generation of pure discrete event models:

1. Deciding as to which segments of the model should be handled by fast simulation;
2. Modeling interactions between the two different types of models.

These additional activities make the hybrid model development process more difficult and complex as compared to the development of pure discrete event models. Another reason why the model development process appears more complex is the current status of hybrid simulation research which is still in its budding stage. As greater insight is obtained, one would be in a position to develop some knowledge base for making some of the decisions involved in the model development process. Moreover, if the simulation product vendors see the potential of this approach and provide "automatic model generation facility" (similar to developing simulation code from the network model specification in SLAM System), then most of the complexities would be eliminated, thereby freeing the user from such responsibilities.

Scenario Modification Process

The scenario modification process also appears to be a complex one, primarily because of the following factors:

1. Identifying the implications of modifying some part of the model for model segmentation;
2. Reconfiguration of the model due to the changes in the scenario.

The following example will help in understanding the above mentioned factors. Consider the hybrid model shown in Figure 42, in which machine 4 has an operator who exclusively attends to that machine.

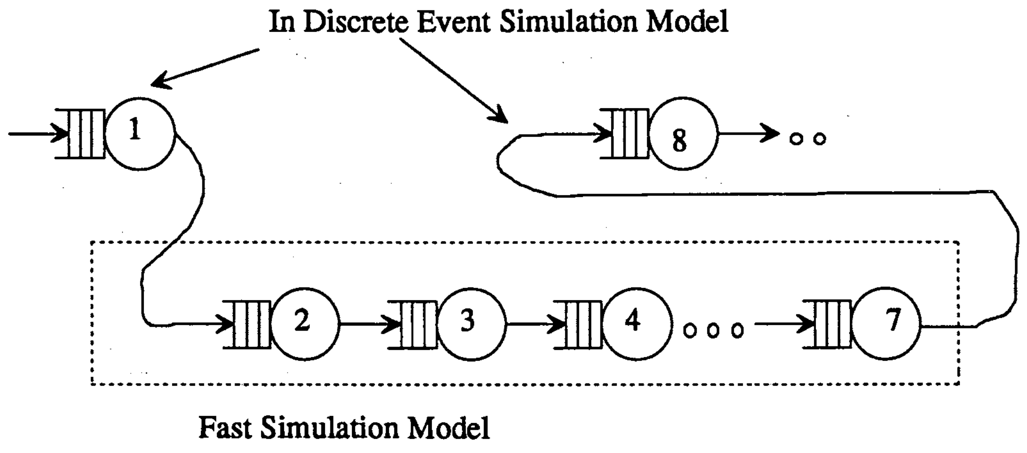


Figure 42. A Scenario Modification Example for a Hybrid Simulation Model

As a part of the scenario modification, it is proposed to depart from the dedicated operator policy and the same operator is now assigned the responsibility of catering to the needs of machines 4 and 8. The implication of this scenario modification is that the original model segmentation is no longer valid and that machine 4 can no longer be part of the fast simulation model but needs to be included in the discrete event simulation model. The changed model segmentation scheme implies that the hybrid model needs to be reconfigured to take account of the new segmentation scheme. Once again, these complexities would be alleviated by the availability of a rule-base and automatic model generators.

Model Maintenance Process

Finally, the task of model maintenance needs to be evaluated in the context of the hybrid approach. After the model has been used, it needs to be maintained to keep it in step with the evolution of the real system. The same factors which were attributed the responsibility of making the scenario modification more complex, once again come into the picture when one is dealing with the model maintenance task. Maintaining the model to reflect the changes in the real system is very similar to the scenario modification process - the only difference being that model modification in the context of the scenario modification process is initiated as a result of conceiving a "what-if" change; whereas model modification in the context of the model maintenance process is initiated as a result of a change in the real system.

Thus, when one adopts the hybrid approach to simulation for improving execution efficiency, one needs to deal with these added complexities. When modeling and simulation environments evolve and include facilities like those mentioned earlier, these processes of model development, scenario modification, and model maintenance would become more manageable.

CHAPTER X

RESEARCH SUMMARY, CONTRIBUTIONS, AND FUTURE RESEARCH

Chapters VI through IX have presented the detailed outcomes of this research. The first section of this chapter is intended to summarize the research results and to link the research outcomes with the research objectives defined in Chapter IV. The contributions of this research to the field of Industrial Engineering, specifically to the "Modeling and Simulation - Body of Knowledge", are also identified in this chapter. This chapter ends with the section which defines a possible agenda for future research.

Research Summary

Six research objectives were defined in Chapter IV which served as the driving force for this research. The following sub-sections present the highlights of the research outcomes and identify their contributions to the various research objectives.

Modeling Abstractions, Conceptual Frameworks, and Relationships

The first objective of this research was to identify the possible conceptual frameworks that can be employed for developing fast simulation models. A subset of manufacturing network topologies was identified in Chapter V and was investigated for generating fast simulation models. During this investigation, which was the subject of Chapter VI, it was realized that "customer-by-customer" and "node-by-node" conceptual frameworks have limited applicability; the "node-by-node" framework being more restrictive of the two. The "customer-by-customer" framework was useful for split topology and tandem lines with single, unreliable servers. A new conceptual framework viz. "customer-by-customer-with-switching" was identified during the investigation of the remaining topologies. The "customer-by-customer" framework can be seen as a special case of the above framework where no *switching* (shifting the focus of fast simulation) is required. The "customer-by-customer-with-switching" framework which is based on the *simulate to the extent possible and then abandon* concept emerged as the one which has the potential for serving as the "world view" for fast simulation. This also led to the satisfaction of the second research objective which was to identify candidates for serving as the world view for fast simulation. Even though identifying new modeling abstractions and relationships for fast simulation was not explicitly stated as a research objective, the following results were also accomplished (these can be thought of as the by-products of the investigation process described in Chapter VI):

- A new modeling abstraction of *Queue Removal Time* was identified and it was seen that it is a more generic abstraction for fast simulation than the abstraction of departure time.
- The concept of *Time History* provided more flexibility to the fast simulation technique by eliminating the dependence on part index.
- New relationships were defined based on the "queue removal time history" for developing fast simulation models of various topologies.

The above contributions, along with the new conceptual frameworks, played crucial roles in extending the applicability of the fast simulation technique to other topologies.

Fast Simulation - Validation and Execution Efficiency

With the power of newly developed abstractions, conceptual frameworks, and relationships, it was possible to develop fast simulation models for all the topologies which were included in the scope of this research. Significant speed-up was achieved in all cases by the use of fast simulation; detailed results were presented in Chapter VII. Development of fast simulation models, their validation using pure discrete event simulation models, and testing the execution efficiency led to the accomplishment of the third research objective which was to identify any topologies which cannot or should not be modeled by the fast simulation approach.

Implications of Fast Simulation Models for Hybrid Modeling

The fast simulation models were revisited in Chapter VIII with the aim of identifying the implications of embedding them within the hybrid models. These implications, if any, were defined in terms of the execution scheme that needs to be employed for enabling the embedding of fast simulation models of various topologies within the hybrid models. This phase of the research satisfied the fourth research objective.

Hybrid Modeling - Partitioning Guidelines and Proof-of-Concept

In Chapter VIII, guidelines were proposed for partitioning the model into FS and DES segments. The communication that takes place between the two types of model segments (i.e., interactions at the interface of FS and DES models) was elaborated in Chapter IX. A prototype hybrid model was configured for the manufacturing system described in Figure 16 and the hybrid simulation results were validated by running an equivalent pure DES model. This served the purpose of demonstrating the feasibility of the hybrid simulation approach on a proof-of-concept basis. Thus, the fifth research objective was also satisfied.

Hypothesis of Diminishing Marginal Speed-up

The initial hypothesis was stated in Chapter IV, to be validated as a part of the sixth research objective. A series of experiments was conducted in the context of an experimental manufacturing system defined in Figure 16 and encouraging results were obtained which supported the initial hypothesis of diminishing marginal speed-up. The actual results of this experimentation were presented in Chapter IX. This contributed towards the satisfaction of the sixth research objective.

Research Contributions

Simulation is being widely employed for the design and analysis of manufacturing systems in an off-line mode. However, application of simulation for an on-line control of manufacturing systems is still prohibitive due to the high execution time requirements. Improving execution efficiency of simulation is the main concern for enabling use of simulation in an on-line mode and is the primary contribution of this research. It should be noted that several other research efforts in the areas of integrated modeling frameworks, knowledge representation for manufacturing domain, and acquisition/elicitation of control knowledge are also contributing towards realizing the goal of simulation-based on-line control of manufacturing systems. The specific contributions of this research to the "Modeling and Simulation - Body of Knowledge" are as follows:

- New modeling abstractions, relationships, and conceptual frameworks were developed for generating fast simulation models of topologies other than tandem lines with single server stations. This contribution single handedly gives significant power to the fast simulation technique.

- Attractiveness of the fast simulation approach from the viewpoint of savings in execution time was established for certain types of manufacturing topologies.
- Feasibility and viability of the hybrid simulation approach was demonstrated on a proof-of-concept basis by generating and validating hybrid simulation models of a prototype manufacturing system.
- Insights were provided for configuring and executing hybrid simulation models of manufacturing systems.
- This research has laid the foundation for further research in the area of fast simulation and its use within hybrid simulation.

Even though this research was conducted in the context of manufacturing systems, several contributions are applicable or at least adaptable for the simulation of other types of systems such as information systems.

Future Research

Several unanswered questions were identified in Chapter III and only a partial set was chosen for inclusion in the scope of this research. These remaining unanswered questions, along with the offshoots of this research process, can set one possible agenda for future research. The following are some of the significant issues that deserve inclusion in this future research agenda:

Conceptual Frameworks

The conceptual framework of "customer-by-customer-with-switching", in which customers are fast simulated to the extent possible and then abandoned, was identified as the one which was meaningful for the whole set of manufacturing topologies investigated in this research. In order to realize the potential of this framework, one should investigate more features like batch nodes, queue disciplines other than First Come First

Served and examine the applicability of this framework. If this framework alone is inadequate to handle such features, then new, more general conceptual frameworks need to be developed for fast simulation.

Fast Simulation

The fast simulation approach is limited by the type of statistics it can generate without significantly losing its execution efficiency. Further research should be directed at identifying the potential of fast simulation in terms of its ability to provide other types of statistics such as queue length distributions or any state information such as queue length at a particular simulation time without sacrificing the speed-up in execution time.

Approximate Hybrid Simulation Models

An arbitrary, self-imposed constraint of *no additional assumptions should be made for generating hybrid simulation models* was adhered to during the entire scope of this research. It is, by all means, conceivable to slacken this constraint for the purpose of hybrid modeling with the intentions of achieving greater savings in execution time [Hunt 1994]. Investigation along this line of approach can potentially lead to several other research questions. Developing trade-offs between loss of accuracy and additional savings in execution time can be one possible outcome of such research efforts.

Predicting Speed-up

In order to make trade-off decisions brought out above, one should be in a position to predict the speed-up that can be achieved for a particular hybrid simulation model. Further research efforts should also be directed towards predicting the speed-up based on the structure of the hybrid simulation model.

Integration of Various Approaches

Synergistic integration of various approaches such as Parallel Discrete Event Simulation, Fast Simulation, and Metamodeling can potentially provide the required execution efficiency to simulation. Directing efforts towards achieving such integration of various approaches can further give rise to several other research questions.

BIBLIOGRAPHY

Adam, N.R. and A. Dogramaci. 1979. Current Issues in Computer Simulation. New York, NY: Academic Press, Inc.

Adiga, S. and M. Gadre. 1990. "Object-Oriented Software Modeling of a Flexible Manufacturing System." Journal of Intelligent and Robotics Systems 3: 147-165.

Askin, R.G. and C.R. Standridge. 1993. Modeling and Analysis of Manufacturing Systems. New York, NY: John Wiley & Sons, Inc.

Beaumariage, T.G. 1990. "Investigation of an Object-Oriented Modeling Environment for the Generation of Simulation Models." Ph.D. dissertation, Oklahoma State University.

Beaumariage, T.G. and C.A. Roberts. 1991. "Object-Oriented Modeling: Attempts at Improving Model Execution Speed." In Proceedings of the SCS Multiconference on Object-Oriented Simulation 1991, edited by R.K. Ege, 93-99.

Bhuskute, H. 1993. "Application of Parallel Processing for Object-Oriented Discrete Event Simulation of Manufacturing Systems." Ph.D. dissertation, Oklahoma State University.

Bhuskute, H., M. Duse, J. Gharpure, D. Pratt, M. Kamath, and J. Mize. 1992. "Design and Implementation of a Highly Reusable Modeling and Simulation Framework for Discrete Part Manufacturing Systems." In Proceedings of the 1992 Winter Simulation Conference, edited by J.J. Swain, D. Goldman, R.C. Crain, and J.R. Wilson, 680-688.

Chen, L. and C. Chen. 1990. "A Fast Simulation Approach for Tandem Queueing Systems." In Proceedings of the 1990 Winter Simulation Conference, edited by O. Balci, R. Sadowski, and R. Nance, 539-546.

Chen, L. and C. Chen. 1993. "A Fast Simulator for Tandem Queueing Systems." Computers and Industrial Engineering 24, no. 2: 267-280.

Dunn, P.L.C. 1985. "Risk Avoidance by Independent Simulation." In Proceedings of the First International Conference on Simulation In Manufacturing, edited by W.B. Heginbotham, 23-36.

Erickson, C., T. Miles, and A. Vandenberge. 1987. "Simulation, Animation and Shop-Floor Control." In Proceedings of the 1987 Winter Simulation Conference, edited by A. Thesen, H. Grant, and W. Kelton, 649-653.

Harmonosky, C.M. 1990. "Implementation Issues: Using Simulation for Real-Time Scheduling, Control, and Monitoring." In Proceedings of the 1990 Winter Simulation Conference, edited by O. Balci, R. Sadowski, and R. Nance, 595-598.

Harmonosky, C.M. and S.F. Robohn. 1991. "Real-Time Scheduling in Computer Integrated Manufacturing: A Review of Recent Research." International Journal of Computer Integrated Manufacturing 4, no. 6: 331-340.

Harmonosky, C.M. 1992. "Investigating Simulation for Real-Time Control in Computer Integrated Manufacturing." In Proceedings of the 1992 NSF Design and Manufacturing Systems Conference: 857-860.

Heginbotham, W.B. 1985. Proceedings of the First International Conference on Simulation in Manufacturing. North Holland.

Hunt, C. 1994. "Fast Simulation of Open Queueing Systems." Masters thesis, University of Oklahoma.

Kamath, M. 1994. "Recent Developments in Modeling and Performance Analysis Tools for Manufacturing Systems." To appear in Computer Control of Flexible Manufacturing Systems, edited by S. Joshi and J. Smith, Chapman Hall, London.

Kamath, M., H. Bhuskute, and M. Duse. 1992. "Fast Simulation Techniques for Queueing Networks." Center for Computer Integrated Manufacturing, Working Paper Series: CIM-WPS-92-MK1. Oklahoma State University.

Kreutzer, W. 1986. System Simulation: Programming Styles and Languages, Addison-Wesley Publishing Company, 41.

Mitrani, I. 1982. Simulation Techniques for Discrete Event Systems, Cambridge University Press, 20.

Mize, J.H., H. Bhuskute, D. Pratt, and M. Kamath. 1992. "Modeling of Integrated Manufacturing Systems Using an Object-Oriented Approach." IIE Transactions 24, no. 3: 14-26.

Nance, R.E. 1971. "On Time Flow Mechanisms for Discrete Systems Simulation." Management Science 18, no. 1: 59-73.

Nevison, C. 1990. "Parallel Simulation of Manufacturing Systems: Structural Factors." In Proceedings of the SCS Multiconference on Distributed Simulation 1990, edited by D. Nicol, 17-22.

Nicol, D. 1988. "Parallel Discrete Event Simulation of FCFS Stochastic Queueing Networks." In Proceedings of the 1988 Winter Simulation Conference, edited by M. Abrams, P. Haigh, and J. Comfort, 124-137.

Pratt, D.B. 1992. "Development of a Methodology for Hybrid Metamodeling of Hierarchical Manufacturing Systems Within a Simulation Framework." Ph.D. dissertation, Oklahoma State University.

Pritsker, A.A.B. 1986. Introduction to Simulation and SLAM II. 3rd ed. New York, NY: Halsted Press.

Ranky, P.G. 1988. "A Real-Time, Rule-Based FMS Operation and Control Strategy in CIM Environment-Part I." International Journal of Computer Integrated Manufacturing 1, no. 1: 55-72.

Reeves, C.M. 1984. "Complexity Analyses of Event Set Algorithms." The Computer Journal 27, no. 1: 72-79.

Segal, M. and W. Whitt. 1989. "A Queueing Network Analyzer for Manufacturing." In Proceedings of the Eighth International Teletraffic Congress: 1146-1152. North Holland.

Shantikumar, J.G. and R.G. Sargent. 1983. "A Unifying View of Hybrid Simulation/Analytic Models and Modeling." Operations Research 31, no. 6: 1030-1052.

Suri, R. and S. deTreville. 1992. "Rapid Modeling: The Use of Queueing Models to Support Time-Based Competitive Manufacturing." In Proceedings of the German/U.S. Conference on Recent Developments in Operations Research, edited by G. Fandel.

2

VITA

Manoj N. Duse

Candidate for the Degree of

Doctor of Philosophy

Thesis: INVESTIGATION OF FAST AND HYBRID (FAST/DISCRETE-EVENT) MODELING APPROACHES FOR SIMULATION OF MANUFACTURING SYSTEMS

Major Field: Industrial Engineering and Management

Biographical:

Personal Data: Born in Poona, India, on November 24, 1967, the son of Narayan P. and Nalini N. Duse.

Education: Graduated from S. P. College, Poona, India in May 1984; received Bachelor of Engineering degree in Mechanical Engineering from University of Poona, Poona, India in May 1988; received Master of Engineering degree in Industrial Engineering from National Institute for Industrial Engineering (NITIE), Bombay, India in December 1989. Completed the requirements for the Doctor of Philosophy degree at Oklahoma State University in July 1994.

Experience: Distribution Executive, Asian Paints (India) Ltd., Bombay, India, from February 1990 to December 1990; Research Assistant, Center for Computer Integrated Manufacturing, School of Industrial Engineering and Management, Oklahoma State University, from March 1991 to June 1994.

Professional Memberships: Institute of Industrial Engineers, Operations Research Society of America, Tau Beta Pi, Alpha Pi Mu.