Order Number 1346746

Performance evaluation of manufacturing systems using stochastic activity networks

Shah, Hemal Vinodchandra, M.S.

The University of Arizona, 1991

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

PERFORMANCE EVALUATION OF MANUFACTURING SYSTEMS USING STOCHASTIC ACTIVITY NETWORKS

by

Hemal Vinodchandra Shah

A Thesis Submitted to the Faculty of the

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

In Partial Fulfillment of the Requirements

For the Degree of

MASTER OF SCIENCE

WITH A MAJOR IN ELECTRICAL ENGINEERING

In the Graduate College

THE UNIVERSITY OF ARIZONA

1991

STATEMENT BY AUTHOR

This thesis has been submitted in partial fulfillment of requirements for an advanced degree at The University of Arizona and is deposited in the University Library to be made available to borrowers under rules of the library.

Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgment of the source is made. Requests for permission for extended quotation from or reproduction of thesis manuscript in whole or in part may be granted by the head of the major department or the Dean of the Graduate College when in his or her judgment the proposed use of the material is in the interests of scholarship. In all other instances, however, permission must be obtained from the author.

SIGNED: H. V. Shah

APPROVAL BY THESIS DIRECTOR

This thesis has been approved on the date shown below:

William H. Sanders
Assistant Professor of
Electrical and Computer Engineering

December 18, 1991
Date

To my parents

ACKNOWLEDGMENTS

It is obvious that any merit this thesis may have is ultimately due to many people who helped and supported me during this endeavor. First of all, I would like to thank Dr. William H. Sanders, my thesis advisor, for inspiring and guiding me throughout my research work. I am also grateful to my Master's committee members, Dr. Jerzy Rozenblit and Dr. William P. Delaney, for reviewing my thesis.

I'm thankful to the Intel Corporation for their financial support which made this work possible. I'm also thankful to Ms. Rose Mary Owen and Ms. Debbie Orr for their help. For their comments and advice I am indebted to many people, particularly Ranjit Deshmukh, Balwant Singh Lall and Rohit Tandon in reviewing the earlier versions of my thesis.

I would like to thank my family, friends and colleagues for their moral support and inspiration in fulfilling my work together. I must acknowledge my great debt to my parents and sister who understood and encouraged me in my efforts to pursue graduate studies inspite of my being so far away from them. Finally, I'm grateful to God for making all this work possible.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

PERFORMANCE EVALUATION OF MANUFACTURING SYSTEMS USING STOCHASTIC ACTIVITY NETWORKS

Hemal Vinodchandra Shah, M.S.

The University of Arizona, 1991

Director: Dr. William H. Sanders

In this thesis, Stochastic Activity Networks (SANs), which are an extension to the Petri Nets, are used for performance evaluation of manufacturing systems. Using our formalism, a manufacturing system is hierarchically represented in three different layers: the manufacturing flow layer, the control layer and the network layer. *SAN* models are constructed for each of these layers. To simplify the understanding of the manufacturing flow, a new graphical representation, the Manufacturing Flow Network (MFN) has been developed. Conversion of *MFN* into *SAN* models simplifies the modeling of manufacturing flow layer. When *MFN* at the product level is very complex, a decomposition technique is applied to reduce complexity of the model under specific conditions. The accuracy of this technique is shown for specific conditions. Finally, a performance evaluation of a sample manufacturing system is shown, using the simulation for solution of the model. Performance variables of interest such as machine utilization, machine availability and operation queue length are discussed.

CHAPTER 1

Introduction

Every manufacturing system exhibits similar characteristics, though differing in detail. The similarities include products and facilities which are used to produce products, such as machines, operations, handling devices, storage locations, pallets, fixtures, tools and so on. Specifically, a flexible manufacturing cell consists of a machine tool, a workpiece store, workpiece, tool-change handling devices, automatic control and supervision subsystems [47]. A *Flexible Manufacturing System* (FMS) contains several automated machine tools of the universal or special type, and/or flexible manufacturing cells, and, if necessary, further manual or automated workstations and machine controllers. According to International Institution for Production Engineering Research (CIRP), a FMS is an automated manufacturing system which is capable, with the minimum of manual intervention, of producing any of a range or family of products. There are three broad application areas where FMS could be used. The first is the introduction of new products. The second is capacity expansion and the third is modernization [24].

1.1 Background

A sophisticated automation in both the discrete parts and process environments of the manufacturing system has been relatively common since the late 1960s. Manufacturing systems are becoming more complex as technology is evolving and this puts limit on the measurement and testing used for manufacturing system evaluation. From a production point of view, identification

of the system bottlenecks and the utilization of resources are the important features. During design, construction and modification phases of the manufacturing system, performance modeling helps in knowing how well the system will work, together with valuable information about the work in progress, production rates, and the impact of equipment failure etc.

Advantages of employing models as a design tool for a manufacturing system are:

1. The design of the manufacturing system can be verified prior to its construction and the design alternatives can be identified.
2. Modeling reduces measurement and testing costs of the manufacturing system.
3. Performance evaluation of the designed manufacturing system can be done before its construction, e.g evaluation of equipment utilization, system capacity.
4. It is easier to test modifications in the design of the manufacturing system after installation, via modeling, rather than changing the system itself.
5. It can be ensured that there are no basic flaws in the manufacturing system design.
6. The bottlenecks in the system can be identified.
7. Operating strategies for work scheduling and job sequencing can be developed.

1.2 Modeling Approaches in Manufacturing Systems

Two main types of modeling approaches; *analysis* and *simulation* are used for performance evaluation of manufacturing systems. For a simple manufacturing system, the analytic approach provide an accurate and efficient solution of performance variables. But with improvements in technology, manufacturing systems are becoming complex, thus making analytical modeling difficult. Hence, for performance modeling of manufacturing systems, computer simulation is often preferred over the analytic approach.

1.2.1 Analytic Approach

The use of queuing networks has been investigated for evaluating the performance of FMS [3, 8, 12, 15, 17, 41, 43]. More specifically, open queueing network models were used by many researchers for modeling manufacturing systems [17, 41]. Shanthikumar and Buzacott [41] analyzed an open queueing network model of dynamic job shops with general service times and first come first served or shortest processing time service discipline. They proposed an approximate decomposition approach and compared accuracy of analytical results to simulation results. Dubois [17] measured various performance indices of a flexible manufacturing system using open queueing network model with a limited amount of inprocess customers. By comparing analytical results of the model to simulation results, he estimated the amount of error introduced by assumptions of his mathematical model.

Others have attempted to use closed queueing networks to model FMS [8, 15, 43]. Using such a model, Dallery [15] derived the performance of FMS in isolation, but he did not explicitly model the parts of FMS. Bruno and Morisio used closed queueing network analysis to solve continuous problems, such as, production scheduling [8]. A closed queueing network model was also used by Solberg and Nof [43] to analyze flow control in alternative manufacturing configurations. For different layout configurations, they showed the preferred choice under appropriate conditions. They developed a method, called CAN-Q (Computer Analysis of Networks of Queues), to model the workflow in manufacturing systems.

Mean-value analysis has also been used, in the context of manufacturing systems, to derive approximate algorithms for queueing networks with general servers and a first-in, first-out service rule. In particular, heuristic methods based on mean-value analysis were used by Cavaille and Dubois [10] for performance evaluation of FMS. Furthermore, based on mean-value analysis, Hildebrant [23] developed a method for scheduling parts that considered machine failures. Suri and Hildebrant [45] developed a tool called mean value analysis of queues (MVAQ) to model FMSs.

They showed applicability of this tool in obtaining part production rates, machine utilization and average work-in-process. Chow et al. [12] developed the Research Queueing Package (RESQ) to construct and solve models of systems with contention of jobs which are served by many resources.

To avoid the restriction imposed by queueing network models, Balbo, Bruell, and Ghanta [3] used a combined approach based on queueing networks and generalized stochastic Petri nets (GSPN). They integrated product form queueing networks and GSPN. In their approach, those parts of models that can be analyzed in isolation as standard product form queueing networks were identified, and then the structure that controlled the load of submodels (which they represented using GSPN) was modeled. They used a decomposition technique to compute the solution. However, exponential growth in the state space of the GSPN model was a drawback in their models.

From the work done by researchers in modeling manufacturing systems using analytic approaches following advantages and disadvantages of this approach can be concluded:

1. *Advantages:*

   (a) With queueing network models, estimation of performance variables, such as throughputs and utilizations, is fairly robust under restrictive assumptions.

   (b) An analytic approach is usually faster than simulation when it is applicable.

2. *Disadvantages:*

   (a) Analytic models do not typically incorporate features such as blocking, breakdowns, scheduling policies for machines, synchronizations, finite buffers, divergence and convergence of parts.

   (b) Assumptions made for queueing network models are sometimes unrealistic. For example, service time distributions are typically assumed to be exponential which is quite untrue because machine timings can be deterministic.

(c) It is very hard to model the detailed operation of a manufacturing system using analytical models.

1.2.2 Simulation Approach

Due to the complexity of manufacturing systems, and the restrictions just mentioned, simulation is most widely used evaluative method. Simulation methods typically take one of two approaches: the use of Petri nets, or the use of knowledge-based systems. Using both approaches, researchers have tried to study and analyze the workflow and control structures in manufacturing systems.

Many researchers concentrated on modeling the workflow in manufacturing system [1, 2, 4, 7, 16, 18, 25, 27, 30, 48, 49, 50]. For example, Acaccia et al. developed program *XS-SIFIP*, which was structured as a knowledge based system, for implementation of a shop-floor part transportation system [1]. They concentrated on transport strategy problems. In modeling flexible manufacturing systems, Dotan and Ben-Arieh used the concurrent logic programming approach which allowed them to model concurrency and synchronization in FMS [16]. Furthermore, Bruno and Morisio used rule based programming for production scheduling [8]. They made case studies of FMS as an event based and a strategic scheduler. In both the case studies they used production system (PS) language *OPS5* which translates Prot nets (which are an extension to Petri nets).

A graphical language, based on Prot nets was applied to the simulation of a manufacturing cell by Bruno et al. [7]. Dubois used Petri net models to describe and analyze production processes [18]. Kamath and Viswanadham showed the usefulness of Petri nets in modeling FMSs [27]. Martinez et al. used High Level Petri Nets (HLPN) for modeling FMS [30]. They introduced time and stochastic hypothesis in HLPN. They did not, however, consider the breakdowns and faults in FMS, but their assumptions about FMS were quite reasonable. A large group is working on the performance evaluation of FMS using Generalized Stochastic Petri Nets (GSPNs) [2, 4].

Modeling of complex FMS has been done by Bālbo, Chiola, Franceschinis and Molinar Roet [4]. They compared their results with the queueing network models and concluded that the queueing network models cannot guarantee that the parts flowing through the system will always chose the most convenient path. Al-jaar and Desrochers chose a couple of cases of manufacturing systems and modeled them using GSPN [2]. They discussed the decomposition and aggregation techniques to deal with a large number of states of the model. They, also proved the validity of assuming the timing distribution to be exponential, and compared it with the deterministic models.

Use of simulation technique in controlling FMS was shown by many researchers [6, 14, 22, 26, 31]. Beck and Krogh modeled the structure of discrete-level control of manufacturing systems by Modified Petri Nets (MPN). Adaptive Petri Nets (APN) were used by Corbeel et al. for describing control and operative part of a process in a manufacturing system [14]. However, it was difficult for them to analyze the dynamic behavior of the model. Hasegawa, Takata, Temmyo and Matsuka proposed Layered Petri Net (LPN) for controlling FMS [22]. They modeled manufacturing cell operations, but did not provide simulation results. Jafari used Adaptive Petri Net (APN) to model the controller of manufacturing system [26]. He represented the controller in detail, did its recovery analysis and showed how scheduling policies affect recovery scheme. The building of the expert system for diagnosis and maintenance in FMS was illustrated by Milacic et al. [31].

In modeling manufacturing systems, combined approaches making use of Petri nets and knowledge based system have been used [9, 37]. In particular, Camurri and Franchi introduced a hybrid approach for the design of the coordination and real-time scheduling subsystems of a FMS, based on both an extension of *PNs*, Structured Timed Color Petri Nets (STCPNs) and object-oriented knowledge representation paradigms. Sahraoui et al. used jointly Petri nets (for detection and handling purposes) and a knowledge based approach (for diagnosis and decision) for monitoring purposes in FMS. Villa showed that hybrid control architecture, knowledge-based, and analytical can be used for planning/control of FMS [46]. He suggested two ideas using the decomposition

of overall Production Planning and Control (PPC) problem into a hierarchy of sub-problems. Colored Petri nets were used by Kasturia et al. for real time control of multilevel manufacturing systems [28]. A graph-theoretic approach for task assignment and load balancing of autonomous vehicles in a FMS was presented by Chen et al. [11].

In all these techniques, a hierarchical approach was often used. In this approach, a manufacturing system is modeled in seperate layers, such that each layer can communicate with adjacent layers. Each layer of the manufacturing system does distinct well-defined functions. Kasturia et al. divided a manufacturing system in three levels: plant level, cell level, and workstation level [28]. Sahraoui adopted a five level structuring: planning, scheduling, task dispatching, coordination (cell and/or workstation) and local control for FMS [37]. Operation hierarchy was used by Wedekind and Zoerntlein in representing FMS control system [48] to ease the understanding of functionality of the system.

So simulation has served as the most widely used evaluative method for manufacturing systems. In general, simulation tends to have following advantages and disadvantages:

1. *Advantages:*

   (a) Simulation models can be made as accurate as desired.

   (b) It is possible to incorporate features such as synchronizations, scheduling policies for machines, blocking, breakdowns etc.

   (c) Using simulation, performance evaluation of a complex manufacturing system can be done without omitting important details.

2. *Disadvantages:*

   (a) The simulation model of manufacturing system must be programmed using a particular syntax relevant to some programming language. If this syntax is not properly understood by the customer then it is hard for him/her to judge that the model's logic

reflected the specified manufacturing system requirements. Recently, this disadvantage has been overcome to some extent by the development of 'user-friendly' software tools.

(b) Most modelers are unlikely to be experts in designing the manufacturing system. Therefore, regardless of the sophistication of the model, there is a high probability that the model may not be an accurate representation of a real system. But, this can be avoided by taking precautions and having a good interaction between the modelers and manufacturing system designers.

(c) Simulation of complex models is computationally expensive. This time can be reduced by developing some efficient programming techniques for simulation.

1.3 Research Objectives

As discussed in the previous section, many researchers are working in the area of performance evaluation of manufacturing systems. There are some problems which can be inferred from previous work done by other researchers:

1. It is very hard to build a general hierarchical model of a manufacturing system which can represent, within a single model, manufacturing flow, control, and communication aspects of a manufacturing system.

2. In analytical models, the exponential assumption is often unrealistic. For example, processing times are usually deterministic.

3. When a manufacturing system model becomes very complex, the simulation time required becomes large to produce statistically significant results.

4. There are many high level representations of manufacturing systems but they do not provide automatic conversion to solvable models.

The main focus of this study is on solving the above mentioned problems. To do this, we use stochastic activity networks (SANs), a stochastic extension to Petri nets which provides an elegant and powerful modeling tool for manufacturing systems [38]. The various features and applications of SANs are discussed in detail in succeeding chapters. SANs can be used to describe a model graphically, which reduces the difficulty in understanding complex systems. When simulation is used, the SANs allow more realistic timing distributions for actions in manufacturing system. Thus, models can be more realistic. Finally, to speed the solution time, reduced base model techniques can be used [19, 35].

Specifically, we will:

1. Build models of manufacturing systems that realistically represent the flow, control, and information transfer (network) aspects of operation in a single model.
2. Develop a high-level representation of manufacturing flow within the manufacturing system.
3. Develop algorithms for conversions of this high-level representation to SANs.
4. Develop SAN models for control and information flow activities of manufacturing systems.
5. Investigate a decomposition technique for models to increase the efficiency of their solution and make their representation easier. The accuracy of this technique will be studied using an example.
6. Study an example manufacturing system using the developed techniques.

More specifically, chapter 2 provides a brief description of Petri nets and stochastic activity networks. Simple illustrative models are discussed and the execution of SANs and Petri nets is explained.

Chapter 3 presents a new graphical representation, the Manufacturing Flow Network (MFN), developed for manufacturing flow representation. An example is presented in which a manufacturing system is represented using MFNs. Conversion of MFN into SAN models is discussed by taking various examples.

Chapter 4 explains SAN models for the control and network layer of manufacturing systems. Various models for the control layer are explained. A CSMA/CD network model is also discussed in detail.

Chapter 5 provides formal definitions, conditions, and analysis of MFNs. A formal mathematical definition of MFN is given. The conditions for connecting elements of a MFN are also discussed. A calculation of average arrival rates in SAN model representation of MFN members is given. A decomposition technique is also discussed to decompose MFNs at operation level under specific conditions, and a hypothetical system is studied to investigate the accuracy of the techniques.

Chapter 6 gives an example study of a manufacturing system. The usefulness of the techniques developed in this study is illustrated. Several performance variables are discussed and their values are obtained.

Chapter 7 summarizes the results and techniques developed in this study and recommends areas for further possible research.

In the appendices, SAN models for non-decomposed and decomposed example MFNs are shown. Steady-state simulation results and error tables for various performance variables are also presented.

CHAPTER 2

Stochastic Activity Networks

2.1 Petri Nets

2.1.1 Petri Net Structure

Petri nets were originally described in C. Petri's dissertation [33]. A Petri net structure consists of places and transitions [32]. Directed arcs (arrows), connect the places and the transitions, with some arcs directed from the places to the transitions and other arcs directed from the transitions to the places. An arc directed from a place to a transition defines the place to be an input of the transition. Multiple inputs from a place to a transition are indicated by multiple arcs from the place to the transition. An output place is indicated by an arc from the transition to the place. Again, multiple outputs to a particular place are represented by multiple arcs. A marking of a Petri net model is an assignment of tokens to the places of the net. Tokens are assigned to, and can be thought to reside in the places of a Petri net. The number and position may change during the execution of a Petri net. The tokens are used to define the state of a Petri net.

2.1.2 Execution Rules for Petri Nets

The execution of a Petri net is controlled by the number and distribution of tokens in the net. The tokens reside in places and control the execution of the transitions of the net. A Petri net executes by firing transitions. A transition fires by removing tokens from its input places and creating new tokens which are distributed to its output places.

A transition may fire if it is enabled. A transition is enabled if each of its input places has at least as many tokens as arcs from the place to the transition. Multiple tokens are needed for multiple input arcs. The tokens in the input places, which enable transition, are its enabling tokens. A transition fires by removing all of its enabling tokens from its input places, and then depositing into each of its output places one token for each arc from the transition to the place. Multiple tokens are produced for multiple output arcs.

2.1.3 Manufacturing System Example Using Petri Nets

A simple machine shop example of Petri net is shown in figure 2.1. The conditions for the system are;

- The machine is waiting,
- An order has arrived and is waiting,
- The machine is working on the order,
- The order is complete.

The events would be;

- An order arrives,
- The machine starts processing the order,
- The machine finishes the order,
- The order is sent for delivery.

Each transition and place have been labeled with the corresponding event or condition.

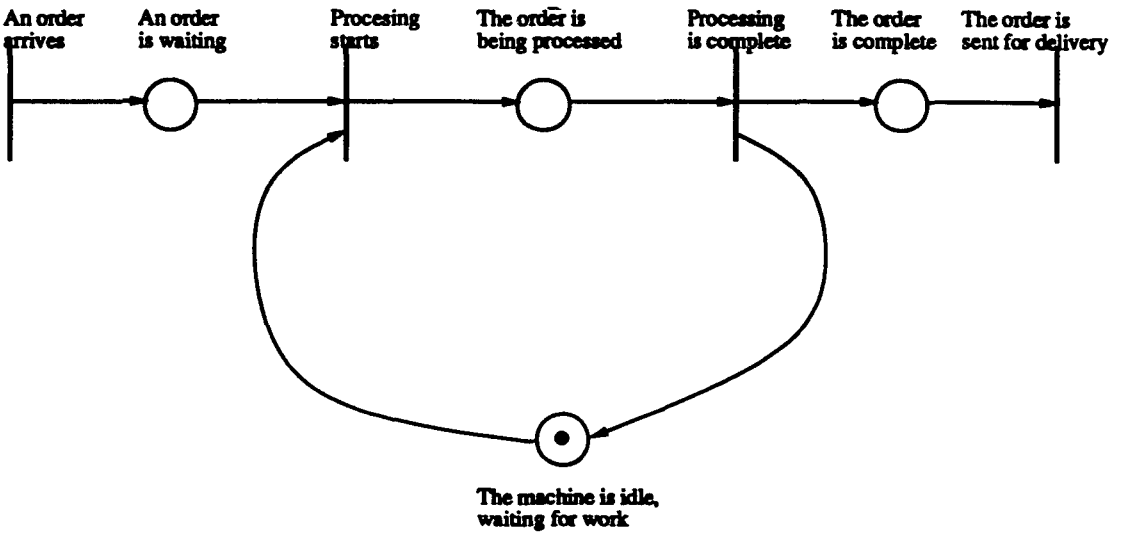


Figure 2.1: A Petri Net Model of a Simple Manufacturing System

2.1.4 Drawbacks of Petri Nets

Petri nets are a powerful representation technique, but still have a few drawbacks when used for performance modeling. In particular:

1. It is not clear which transition should fire if more than one transition is enabled in a marking.
2. There is no means of clearly modeling delays corresponding to actions.
3. It is not always possible to depict probabilistic events.
4. It is very difficult to represent complex scheduling policies for machines, and complicated control actions.
5. The size of a Petri net model can be quite large, even for simple systems.

Stochastic activity networks (SANs) [38], which are an extension to stochastic Petri nets, overcome the above limitations. For example, by using timed "activities" in a SAN model, we can

model delays in the system. Furthermore, using recently developed reduced based model construction techniques [39] developed for SANs, models can be replicated and/or joined together to form a composed model of the system. This technique aids in hierarchical modeling of complex manufacturing systems; and makes their solution more efficient.

2.2 Stochastic Activity Networks

2.2.1 Primitives of SANs

Stochastic activity networks (SANs) [38] are used to provide a compact and efficient graphical representation of a system. Primitives of SANs can be briefly described as follows:

- *Activities* are analogous to transitions in Petri nets. These are of two types: *timed* and *instantaneous*. Timed activities represent operations, tasks or events, and they have *activity time distribution function* associated with them. A *reactivation function* is provided for restarting activities that have been activated. Timed activities are drawn as hollow bars. On the other hand, instantaneous activities represent the system activities which complete in a negligible amount of time. They are drawn as filled bars. Each activity has a positive number of cases associated with it. Each case denotes a possible action which can take place upon the completion of the activity. A *probability distribution function* is associated with each set of cases.

- Places and tokens are as in Petri nets. Places hold tokens, which control the execution of the activities of the net. The number of tokens residing in a place is called the *marking* of the place. The *marking* of a *SAN* is the collective marking of all the places in the SAN. Places are depicted as circles while tokens are drawn as small dots.

- *Input gates:* Input gates connect places to activities and are depicted as triangles. An input gate has a *predicate* (enabling function) and a *function*. Input gate predicates and functions

are defined over the markings of places which are connected to the gate. When a predicate is true, the activity connected to the input gate is enabled. Upon completion of an activity which is connected to an input gate, the input gate function is executed, which may or may not result in the change of markings of the places connected to that input gate.

- *Output gates:* Output gates connect activities to places and are depicted as triangles. An output gate has a function, but no predicate.

- *Arcs:* Arcs connect places to activities and gates, and gates to activities. An inhibitor arc is a special type of gate that enables the activity when there are zero tokens in the place, and an identity function.

2.2.2 Execution of SANs

The execution of stochastic activity networks is discussed in detail in several publications including that by Sanders [38]. Informally, though, SANs execute in time through completions of activities, resulting in changes in markings. More specifically, an activity is chosen to *complete* in the current marking based on the relative priority among activities (instantaneous activities have priority over timed activities), and the activity time distributions of *enabled* activities. An activity is enabled when all of its input places have at least one token (if any input place is connected to the activity by inhibitor arc then it should have no tokens residing in it.), and all of its input gates have true predicate (i.e., they *hold*). The time required for an activity to complete is determined by its activity time distribution. A case of an activity chosen to complete is then selected based on the probability distribution for that set of cases. The next marking of the network is then obtained by executing the input gates connected to the input of the activity chosen and the output gates connected to the chosen case. This procedure is repeated by considering the activities enabled in the new marking.

Stochastic activity networks can be solved both analytically, and by using simulation, depending on system characteristics. For a given SAN model, if all activity time distributions are exponential and activities are reactivated often enough to ensure that their rates depend only on the current state, it can be solved analytically . In this case, stochastic processes exist which can be used to obtain analytic solutions for a wide class of variables characterizing both activity and marking related behavior. If this is not the case, simulation can be used to evaluate the system behavior.

For applications involving realistic systems, model construction and solution technique require machine implementation. Both, the complexity of the construction procedures and the typical sizes of the resulting base models make this a necessity. To fulfill this need an extensive software package, called *UltraSAN* [40], has been developed specifically for the construction and solution of SAN-based performability models.

To illustrate the construction and execution of SANs, consider figure 2.2. This figure shows a SAN for a simple manufacturing system where only one machine is used, and jobs are processed at the machine. The machine may fail and be repaired.

In this model, the timed activity *job_arrival* represents the arrival of jobs to the machine, and the place *job_queue* represents the queue of jobs at the machine. The marking of place *job_queue* represents the number of jobs waiting for the machine. The finiteness of the queue is represented by the input gate *queue_size*, which specifies (via its predicate) that activity *job_arrival* is enabled only when the number of jobs in the queue is less than the system's capacity.

Input gate *access* checks availability of the machine. If, the machine is not in use (i.e., the marking of *job_in_process* is 0) and has not failed (the marking of *machine* is 1) then timed activity *job_start* is enabled. After completion of this activity, the marking of the place *job_in_process* is set to 1 to indicate start of job processing at the machine. Continuing, timed activity *job_time* represents processing time required for a job, and input gate *machine_working* checks whether the machine is working or not. Once the job is started, if the machine fails, timed activity *job_time* is

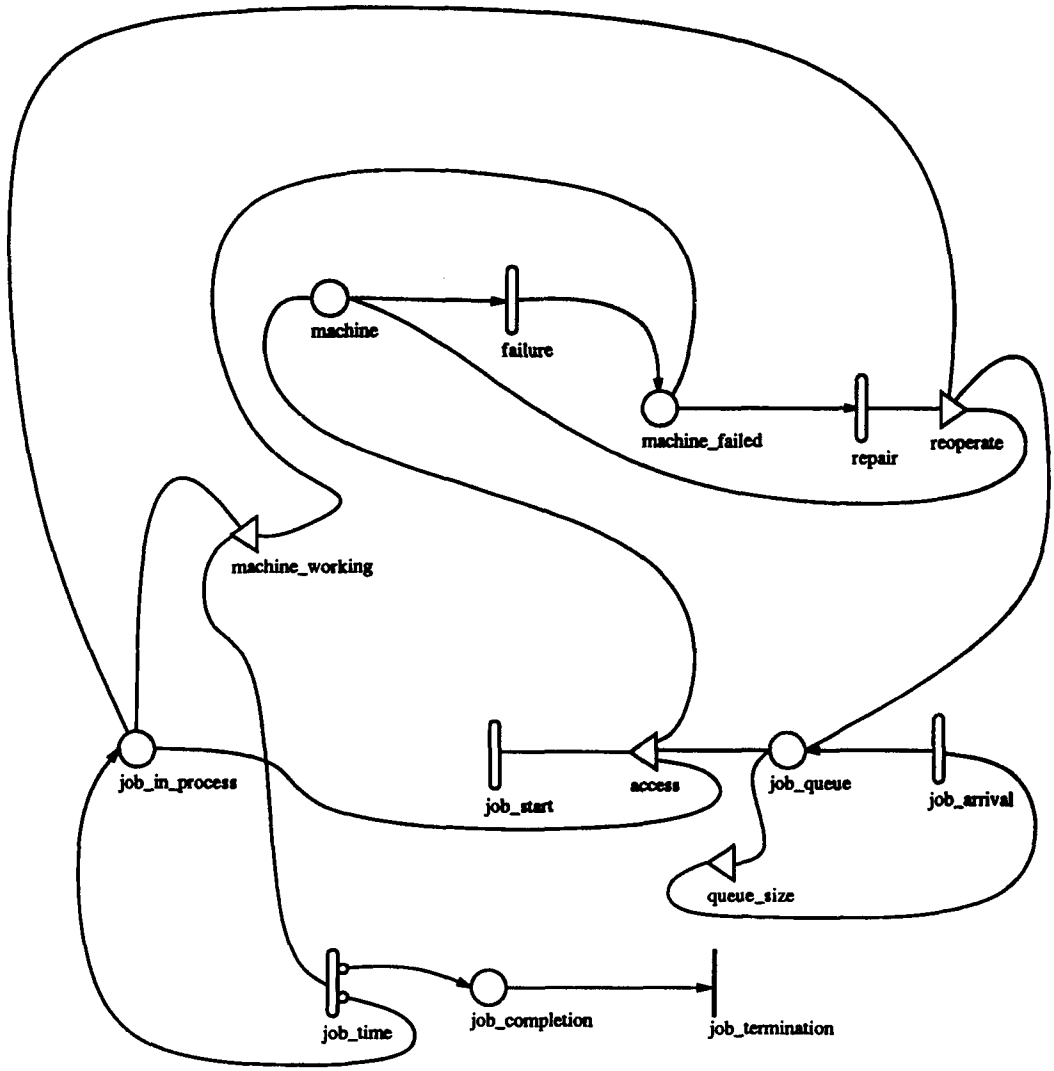


Figure 2.2: SAN Model of a Simple Manufacturing System

Gate	Enabling Predicate	Function
access	MARK(machine) > 0 && MARK(job_queue) > 0 && MARK(job_in_process) == 0	MARK(job_queue) -= 1; MARK(job_in_process) += 1;
machine_working	MARK(machine_failed) == 0 && MARK(job_in_process) > 0	MARK(job_in_process) -= 1;
queue_size	MARK(job_queue) <= 20	Identity

Table 2.1: Input Gates Used in SAN Model of the Simple Manufacturing System

Gate	Function
reoperate	if(MARK(job_in_process) != 0) MARK(job_queue) += 1; MARK(machine) += 1;

Table 2.2: Output Gates Used in SAN Model of the Simple Manufacturing System

Activity	Rate	Probability	
		case 1	case 2
failure	exponential(.00001)	1	-
job_arrival	exponential(varied)	1	-
job_start	deterministic(1)	1	-
job_termination	Instantaneous	1	-
job_time	deterministic(100)	.999	.001
repair	exponential(.01)	1	-

Table 2.3: Activity Parameters for SAN Model of the Simple Manufacturing System

aborted. Job processed at the machine are successfully processed with a certain probability. This probability is represented by the cases of the timed activity *job_time*. Placement of a token in place *job_completion* indicates successful processing of a job. Instantaneous activity *job_termination* then removes a token from the place *job_completion*.

Failures and repairs of the machine are now considered, and are represented by the top half of the SAN of figure 2.2. In this figure, a token in place *machine* represents the machine in working condition, and timed activity *failure* shows the failure of the machine. The place *machine_failed* represents a failed machine. If the machine has failed then it must be repaired. Timed activity *repair* represents repairing process of the machine. After completion of this activity, function of the output gate *reoperate* is executed which puts the job back to start if the machine failed, while the job was in operation. It also puts a token back into place *machine* putting the machine in working state, again.

To illustrate the use of this SAN, we solve it to obtain machine utilization and job queue length, given the specific model parameters of tables 2.1, 2.2, 2.3. By varying the arrival rate of jobs (in our model changing the activity time distribution function associated with timed activity *job_arrival*) one can obtain different values for the performance variables. Figures 2.3 and 2.4 show machine utilization and job queue length plots, respectively.

In this chapter, a simple manufacturing system model was discussed using SANs. However, understanding of the SAN models of manufacturing systems becomes difficult as the system's structural complexity increases. In this situation, a simple high level representation can make understanding of manufacturing flow easier. At the same time, by developing conversion algorithms for the high level representation, SAN models can be generated from them to evaluate the performance of the system. One such high level representation is discussed in the next chapter.

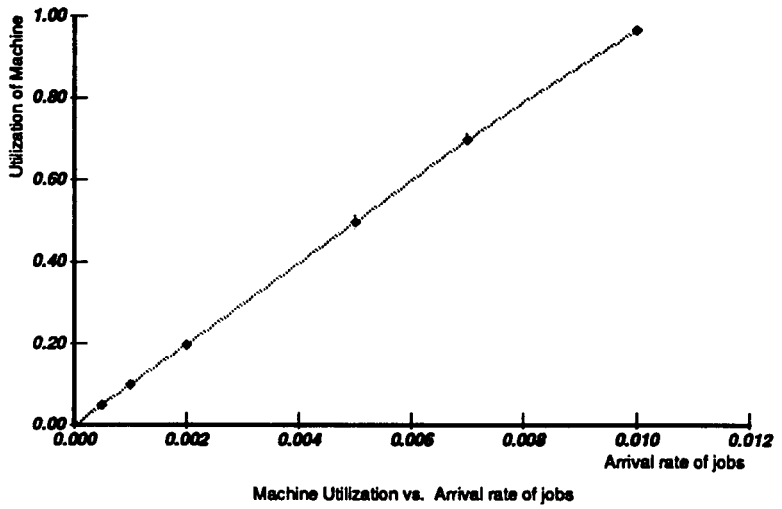


Figure 2.3: Machine Utilization vs. Arrival Rate of Jobs

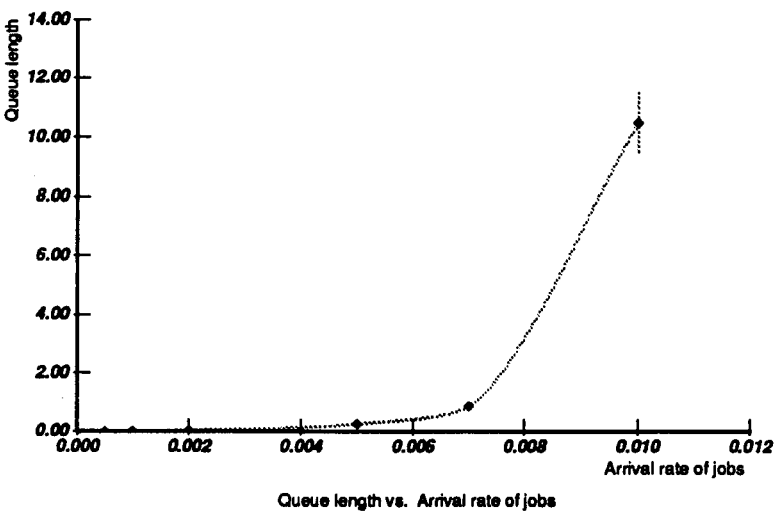


Figure 2.4: Job Queue Length vs. Arrival Rate of Jobs

CHAPTER 3

Manufacturing Flow Representation

3.1 Terms and Definitions Used in Manufacturing

Before modeling any manufacturing system, the modeler must understand the manufacturing flow in the system. The purpose of this chapter is to represent the manufacturing flow in the system in such a way that it simplifies its modeling. Definitions of some of the terms that will be used quite frequently in discussion of the manufacturing flow in the manufacturing system are given below:

1. A *Machine* is an entity used in a manufacturing system to perform task.
2. A *Step* is a task performed at any machine in a manufacturing system.
3. An *Operation* is performed at a chain of one or more machines.
4. A *Product* is a produced after performing a chain of one or more operations on a raw material.
5. A *database manager* is used to keep information about machine breakdowns, shortages, machine activities, routing of products, products completed per period etc.

To manage the complexity of realistic manufacturing systems, a hierarchical modeling approach has been used by many researchers [28, 37, 48]. In this approach, each layer performs well-defined functions which are distinct, with respect to one another. Each layer can communicate with adjacent layer or layers even though their functionalities are different. This approach thus aids in

understanding the complexity of manufacturing systems by separately defining the functionality of each layer and the interactions between layers. In the remainder of this work, we consider the hierarchy of three layers shown in figure 3.1. The detailed functions of each layer are as follows:

- **The Manufacturing flow layer:** This layer is the topmost layer in the manufacturing system; the raw products are introduced in this layer. At this layer, resource allocation, routing of products, and resource scheduling is performed. The manufacturing flow at this layer is viewed at two levels, the *operation level* and the *machine level.* At the operation level, the operations performed on products in the system are described. In contrast, the machine level gives details of how each operation is performed in the system, by receiving service at one or more machines. After performing one or more operations on a raw product, a finished product is produced. Products can be produced by different routes, at both the operation and machine level.

- **The Control layer:** The control layer performs and monitors tasks at machines. A step performed at any machine is controlled at this layer. Control of the system can be of different types: operational control, loading control, transportation control etc. Furthermore, this layer keeps track of tasks performed at machines and reports the status back to the database manager.

- **The Communication layer:** Machine controllers, database manager and machines communicate with each other at this layer. The component models at this layer can be network nodes, bridges, gateways, database managers, among others.

The major advantages of this hierarchical structure are:

1. Each layer has a capability of communicating with other layers.

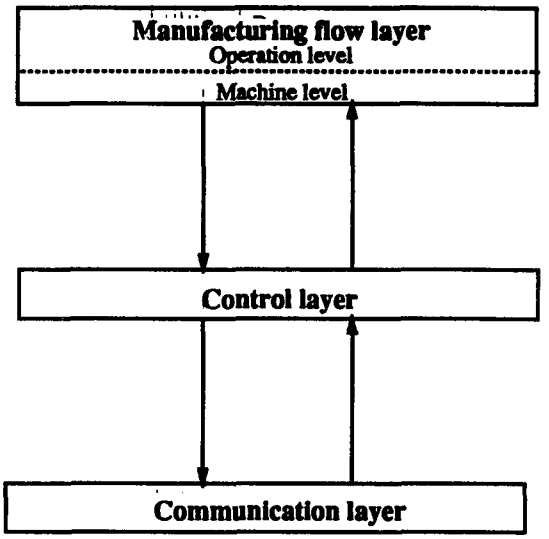


Figure 3.1: Hierarchical Representation of a Manufacturing System

2. The internal functions performed at each layer are not concerned with the internal functions performed at other layers. So, the internal structural changes in one layer doesn't affect the internal functionality of other layers.

3. The models for each layer can be developed separately and then they can be joined together to form a composite model of the system.

3.2 Manufacturing Flow Networks (MFNs)

The representation of behavior at the manufacturing flow layer can be very complex, and difficult to describe directly as a stochastic activity network. To aid in this description, we have developed a new model type which makes it easy to describe flow in a manufacturing system. As will be seen in the following, these models can be converted in a straightforward manner to SANs, which then can be solved using known techniques. The models are known as 'manufacturing flow networks (MFNs)', and have the following components. (A formal definition of a MFN will be given in chapter 5.) MFNs are made up of blocks, of three types, and nodes, of four types.

A (ordinary) *block* is used to represent the delay associated with processing of a product at an operation or machine. An *input block* is used to represent an arrival of a product to an operation or machine and an *output block* represents the completion of the processing of a product at an operation or machine. A block has an incoming arc and an outgoing arc. An input block has an outgoing arc and no incoming arcs. An output block has an incoming arc and no outgoing arcs.

Four different types of nodes are used: *select, merge, fork* and *join.* Directional arcs are used to represent incoming and outgoing flows from a block or a node. A select node is used to show selection of a particular route by a product and has a single incoming arc and multiple outgoing arcs. For a given incoming arc, a select node selects one of the outgoing arcs. A merge node has more than one incoming arcs and a single outgoing arc. It is used for selection of one inflow out of multiple inflows. A fork node represents the branching of an incoming flow into multiple outgoing flows. Finally, a join node allows us to show convergence of parallel inflows into an outflow, and it has multiple incoming arcs and an outgoing arc. Each of these components is shown in figure 3.2.

3.3 Rules for Representation of Manufacturing System Using MFN

The rules for interconnecting these components together to represent specific manufacturing flows are defined below.

1. An incoming arc of any block or output block or node can be
   (a) The outgoing arc of another block or an input block.
   (b) One of the outgoing arcs of a select node or a fork node.
   (c) The outgoing arc of a merge node or a join node.
2. An outgoing arc of any block or input block or node can be
   (a) The incoming arc of other block or an output block.

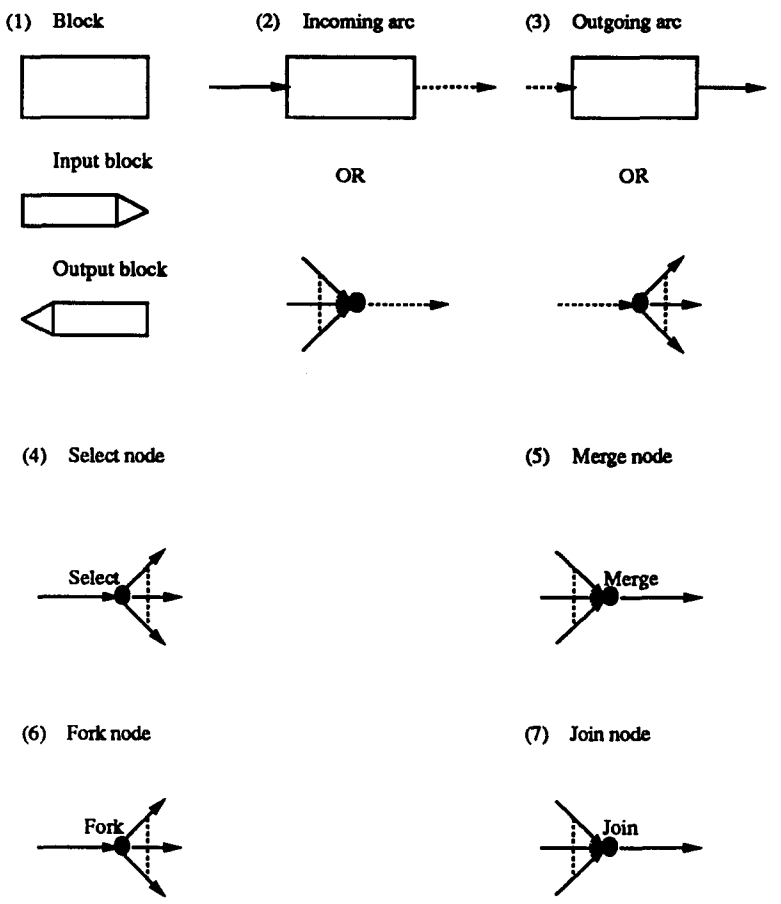


Figure 3.2: Components Used in Manufacturing Flow Representation

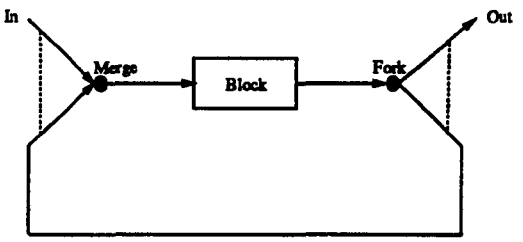


Figure 3.3: Feedback Representation in MFN Using Fork and Merge Nodes

(b) One of the incoming arcs of a merge node or a join node.

(c) The incoming arc of a select node or a fork node.

3. Each block in a MFN can be replaced by another MFN until we reach the lowest layer of manufacturing flow representation.

4. Feedback to any block can be represented by using fork and merge nodes as shown in figure 3.3.

A more formal definition of these rules can be found in chapter 5.

3.4 MFN Representation of a Simple Manufacturing System

A simple manufacturing system, which produces 2 products, is used as an example throughout the remainder of this thesis. Each product in the manufacturing system is produced by performing several operations. There are four different kinds of operations performed, and four different types of machines are used to perform the operations. Furthermore, it is assumed that the manufacturing system is producing two products: ProductA and ProductB.

Each finished product is an outcome of a sequence of operations. A product can be produced by using different routes, with each route having a unique sequence of operations. Furthermore, the

Product	Operations performed for different Routes	
	Route 1	Route 2
Product A	operation1, operation3 operation4.	operation2, operation3 operation4.
Product B	operation1, operation3 and operation4.	

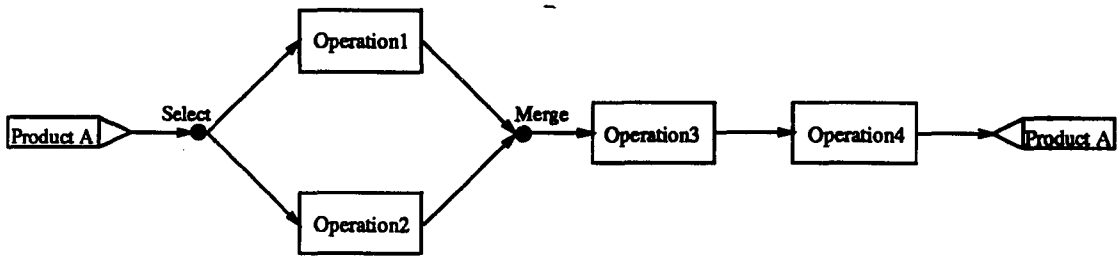
Table 3.1: Operations Performed for Products

manufacturing system has four different types of operations: operation1, operation2, operation3 and operation4. This information is shown in table 3.1. The MFNs for ProductA and ProductB at operation level are shown in figure 3.4.

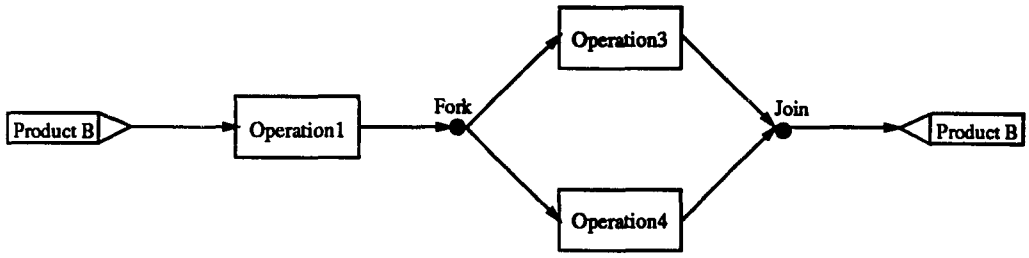
Each operation is done in one or more steps, and each step can be performed at different machines. In a given manufacturing system, the machines are considered at the lowest level of manufacturing flow. The four different types of machines are: machine1, machine2, machine3 and machine4. Table 3.2 shows operations and machines which perform the operational steps. Figures 3.5 shows the MFN for each operation. Here each dotted box represented as a block in the MFN representation of the system at operation level.

3.5 Conversion of MFN into SAN Models

In the last section, a new graphical representation of the manufacturing system, MFN was introduced. MFNs make it easy to represent flows, without having to explicitly consider interaction between production of different products at the same machines and operations. This is because in a MFN, multiple representations of a block are possible. Furthermore, different MFNs can have same blocks. Hence, in order to show contention and solve the models, we need to convert MFNs



Manufacturing flow of ProductA



Manufacturing flow of ProductB

Figure 3.4: MFNs of the Manufacturing System at the Operation Level

Operation	Machines used for different Routes	
	Route 1	Route 2
operation 1	machine1, machine4.	machine2, machine4.
operation 2	machine1, machine3, machine4.	
operation 3	machine4, machine1 and machine2, machine3.	
operation 4	machine3, machine4.	

Table 3.2: Operations Performed at Several Machines in Steps

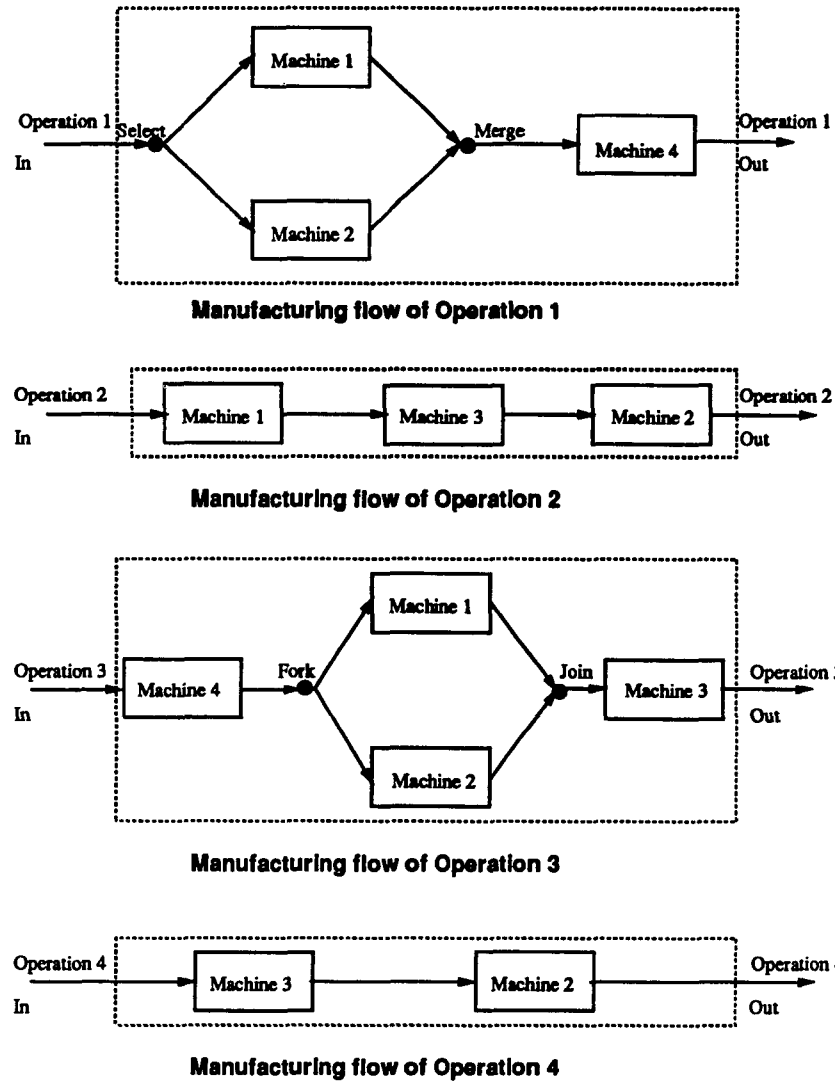


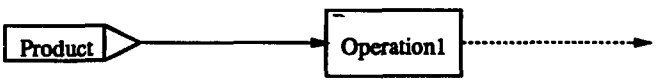
Figure 3.5: MFNs of the Manufacturing System at the Machine Level

into SANs. In the remaining chapter, the conversion of MFN into SAN models at different levels is discussed.

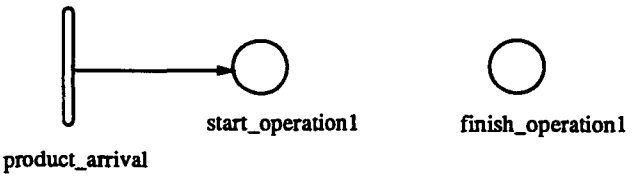
3.5.1 Conversion of MFN into SAN Models for the Product Flow at the Operation Level in Manufacturing System

First, the conversion of MFN into stochastic activity networks of product flow at the operation level is discussed. Without loss of generality, we can break the MFN up into three stages, the first stage, one or more intermediate stages and the last stage, and consider the conversion for each stage separately. These individual SANs can then be put together to form a complete SAN model of the manufacturing flow layer. In the starting stage, the product inflow enters either a block or a node.

A SAN model of product inflow to a block is as shown in figure 3.6. Here, the timed activity *product_arrival* represents arrival of a request to produce a product. Place *start_operation1* represents the starting point of operation1, while the place *finish_operation1* represents the end of operation1. Figure 3.7 shows MFN and SAN models for the case where product inflow is to a select node. Timed activity *product_arrival* represents arrival of the product. Place *place_product* represents the place of arrived products and from here they are routed. Places *start_operation1*, ..., *start_operationN* and *finish_operation1*, ..., *finish_operationN* represent the starting and the ending points of operation1, ..., operationN, respectively. Input gate *Operation_alloc* is used for operation scheduling. By adding additional places to create local memory, and defining an appropriate gate function, any scheduling policy can be realized. For example, the place *Track_op* can be used to keep track of which operation was most recently started for an arrived product request, to obtain a round-robin scheduling algorithm. On completion of the instantaneous activity *product_str*, scheduling is done. Alternatively, a probabilistic selection can be made, as shown in figure 3.8. The difference here is the place *place_product*, the input gate *Operation_alloc* and



(a) MFN model



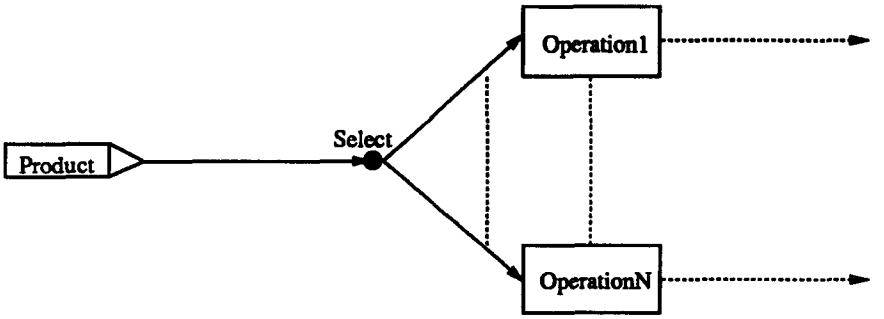
(b) SAN model

Figure 3.6: MFN and SAN Models of Product Inflow to a Block

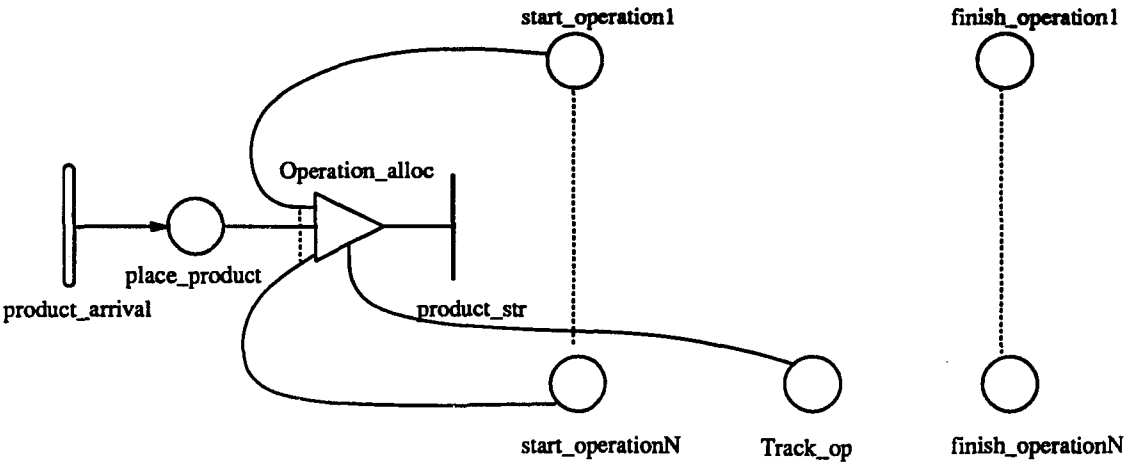
the instantaneous activity *product_str* are replaced by cases of the timed activity *product_arrival*, where each case has fixed probability assigned to it.

Next, product inflow to a fork node is considered. MFN and SAN models for this case are shown in figure 3.9. Here, also, the timed activity *product_arrival* represents arrival of the request to produce a product. Places *start_operation1*, ..., *start_operationN* and *finish_operation1*, ..., *finish_operationN* represent the starting and the ending points of operation1, ..., operationN, respectively.

We now consider SAN models of product outflow from a MFN. Outflow can come from either a block or a node. Figure 3.10 shows SAN model of a MFN where product outflow is coming from a block. In this figure, place *finish_operationI* represents the completion of the preceding operation, and instantaneous activity *start_opJ* represents starting of the operationJ. Place *start_operationJ* shows the starting point of the operationJ and the place *finish_operationJ* represents the ending point of the operationJ. Instantaneous activity *product_out* represents outflow of the product.

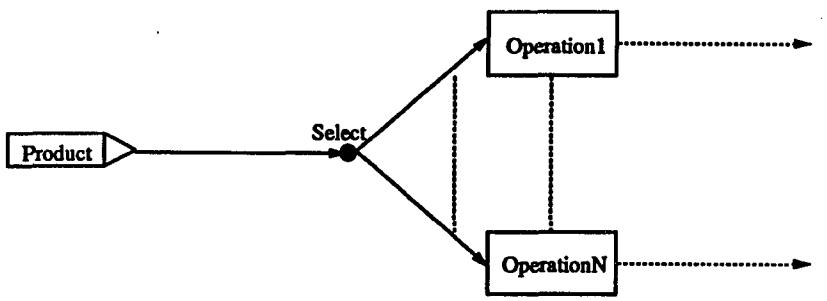


(a) MFN model

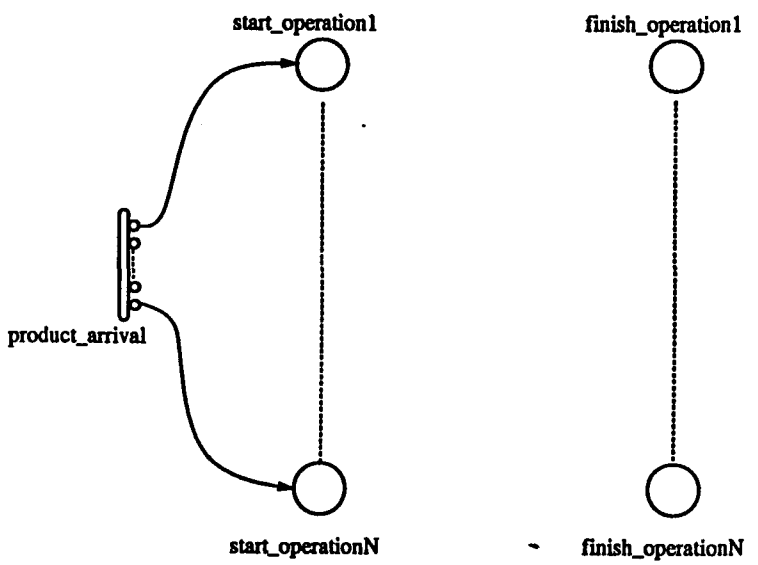


(b) SAN model

Figure 3.7: MFN and SAN Models of Product Inflow to a Select Node

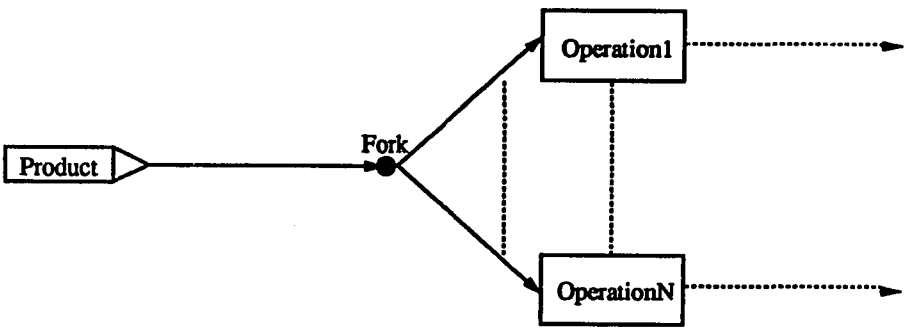


(a) MFN model

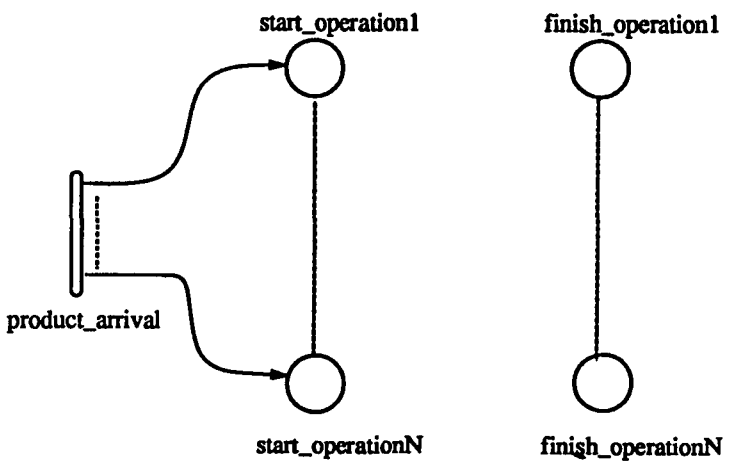


(b) SAN model

Figure 3.8: MFN and SAN Models of Product Inflow to a Select Node with Probabilistic Selection



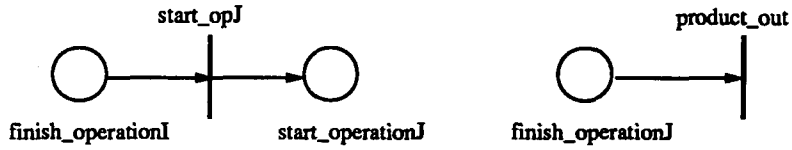
(a) MFN model



(b) SAN model

Figure 3.9: MFN and SAN Models of Product Inflow to a Fork Node

(a) MFN model

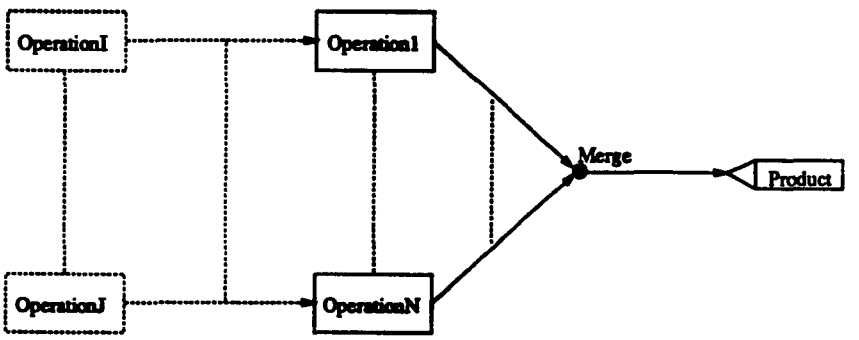


(b) SAN model

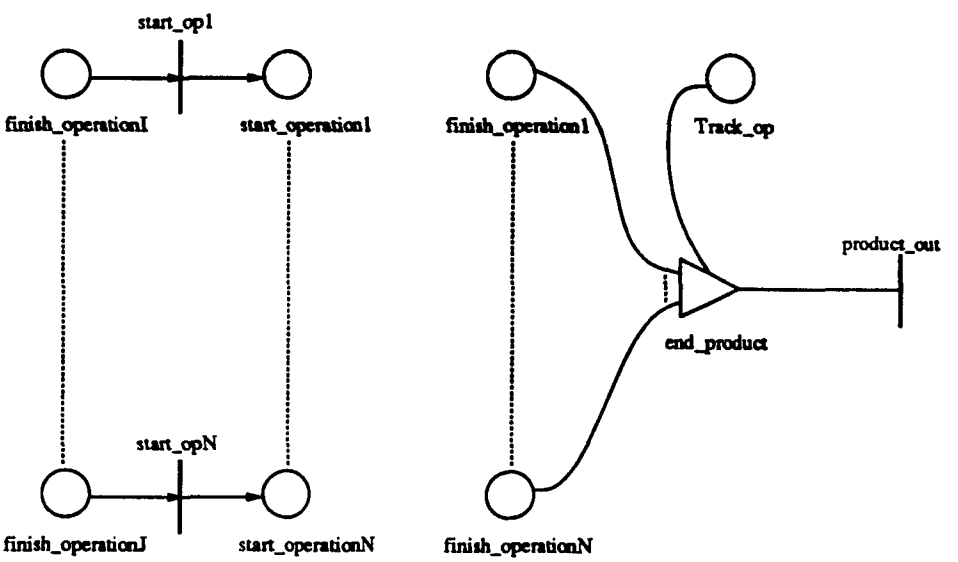
Figure 3.10: MFN and SAN Models of Product Outflow From a Block

Figure 3.11 shows a SAN model for a MFN where product outflow is from a merge node. In this model, places *finish_operationI*, ..., *finish_operationJ* show the completions of the preceding operations: operationI, ...,operationJ. Places *start_operation1*, ..., *start_operationN* show the starting points of the operation1, ..., operationN and the places *finish_operation1*, ..., *finish_operationN* represent the ending points of the operation1, ..., operationN. Input gate *end_product* checks whether an operation has finished. If so, the instantaneous activity *product_out* is enabled and completion of this activity represents outflow of the product from the MFN.

Finally, figure 3.12 shows a SAN model for a MFN where product outflow is from a join node. In this figure, places *finish_operationI*, ..., *finish_operationJ* represent the ending points of the preceding operations. Instantaneous activities *start_opI*, ..., *start_opJ* represent completions of the preceding operations. Places *start_operation1*, ..., *start_operationN* and *finish_operation1*, ..., *finish_operationN* represent the starting and the ending points of the operation1, ..., operationN. Instantaneous activity *product_out* represents the outflow of product.

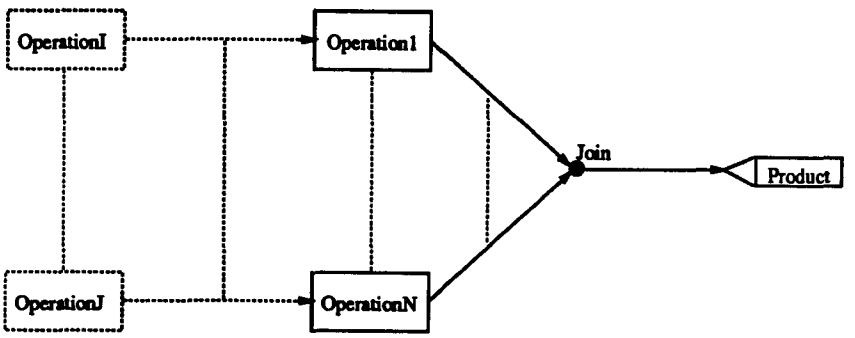


(a) MFN model

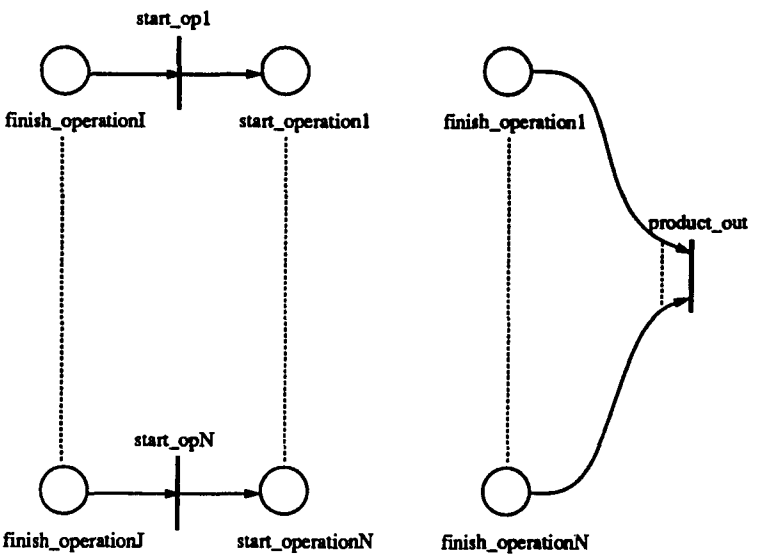


(b) SAN model

Figure 3.11: MFN and SAN Models of Product Outflow From a Merge Node

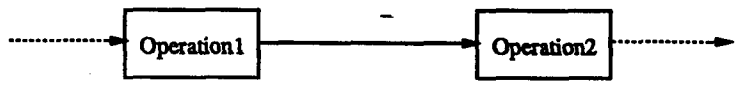


(a) MFN model

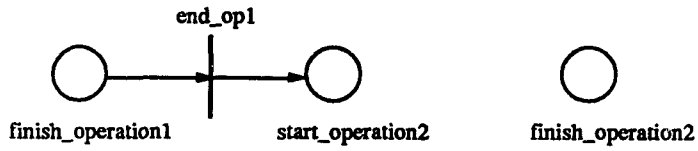


(b) SAN model

Figure 3.12: MFN and SAN Models of Product Outflow From a Join Node



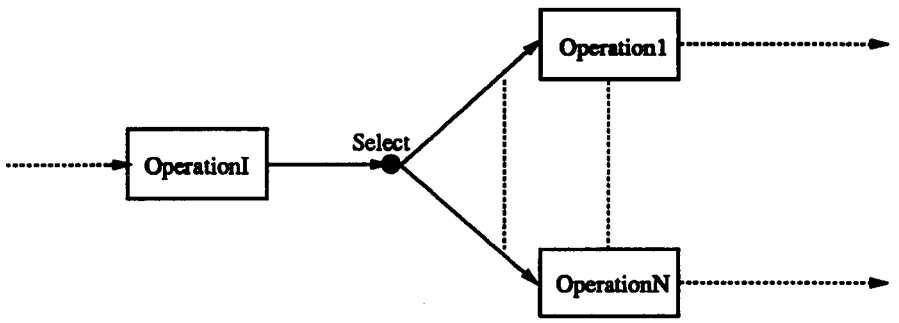
(a) MFN model



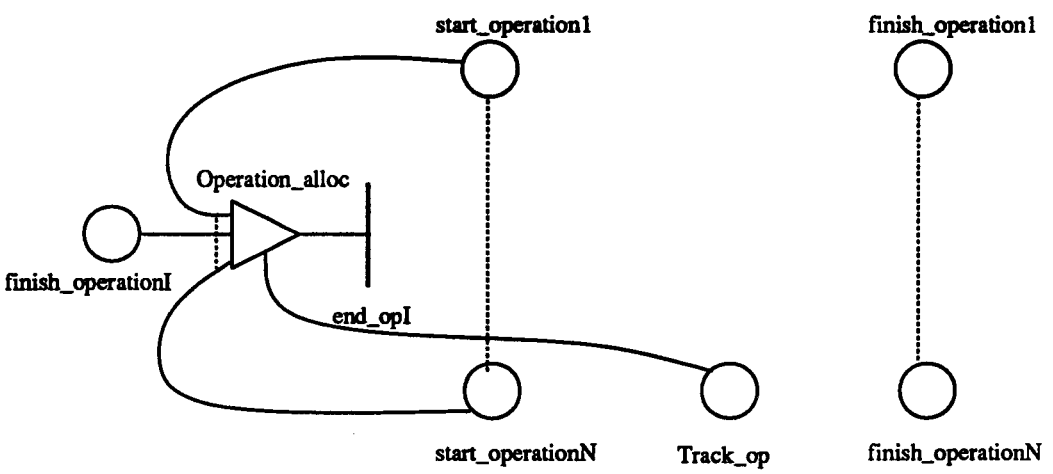
(b) SAN model

Figure 3.13: MFN and SAN Models of the Intermediate Stage of Product Flow between 2 Blocks

Conversion of intermediate stages of MFN models into SANs is straightforward, and does not need to be discussed in detail. Instead, refer to figures 3.13, 3.14, 3.15, 3.16, 3.17, 3.18, for each of these cases.

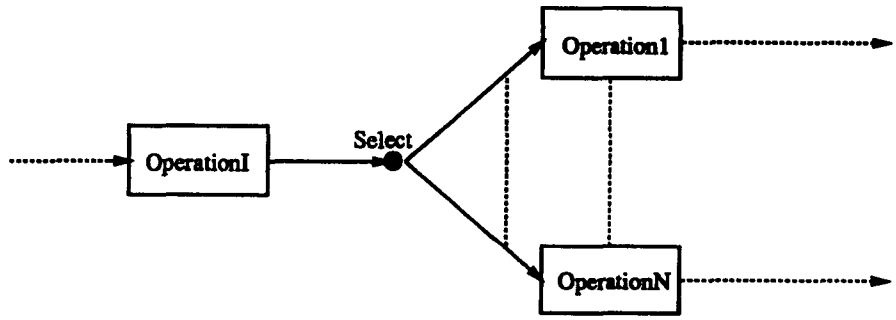


(a) MFN model

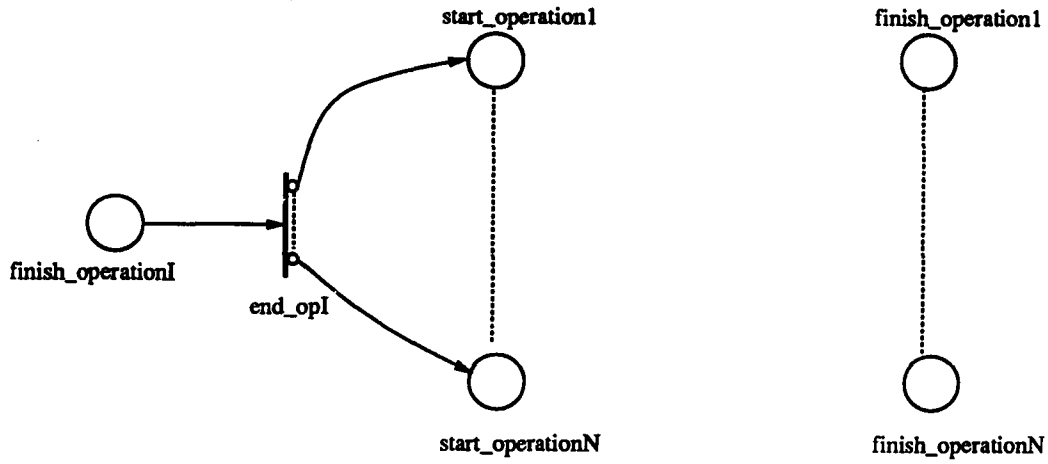


(b) SAN model

Figure 3.14: MFN and SAN Models of the Intermediate Stage of Product Flow between a Select Node and Blocks

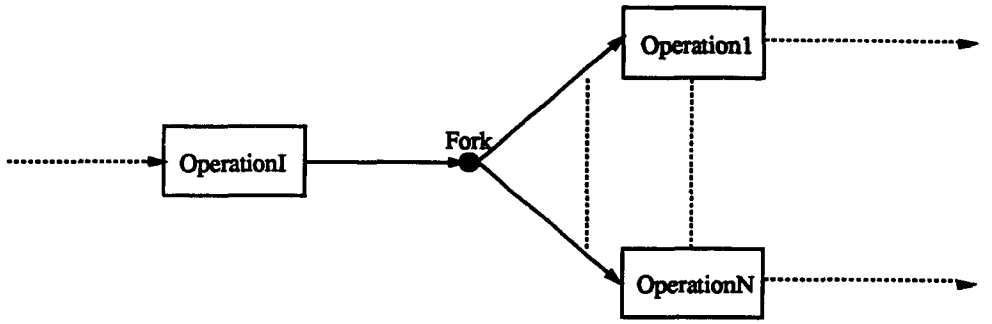


(a) MFN model

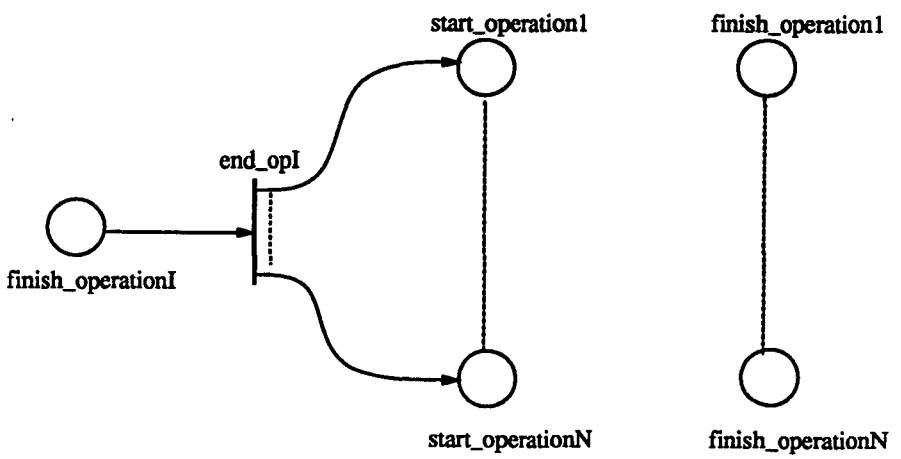


(b) SAN model

Figure 3.15: MFN and SAN Models of the Intermediate Stage of Product Flow between a Select Node and Blocks with Probabilistic Selection

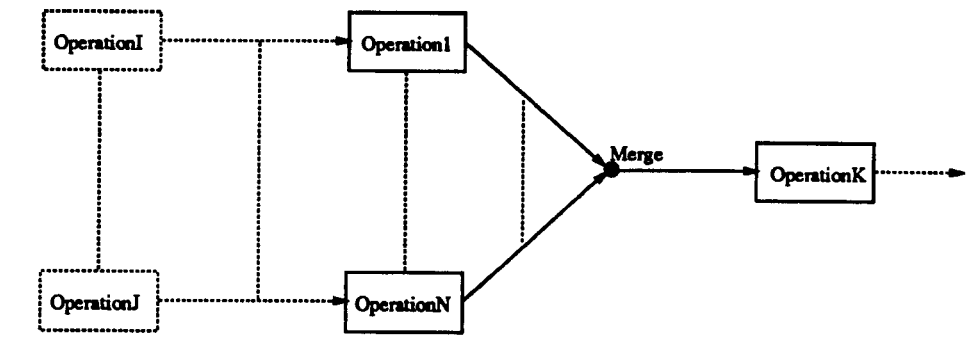


(a) MFN model

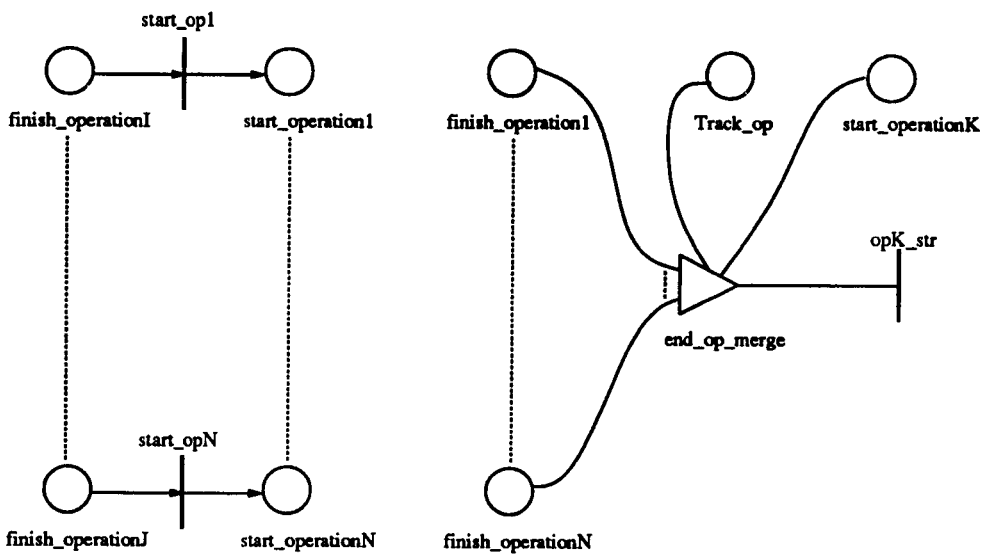


(b) SAN model

Figure 3.16: MFN and SAN Models of the Intermediate Stage of Product Flow between a Fork Node and Blocks



(a) MFN model



(b) SAN model

Figure 3.17: MFN and SAN Models of the Intermediate Stage of Product Flow between Blocks and a Merge Node

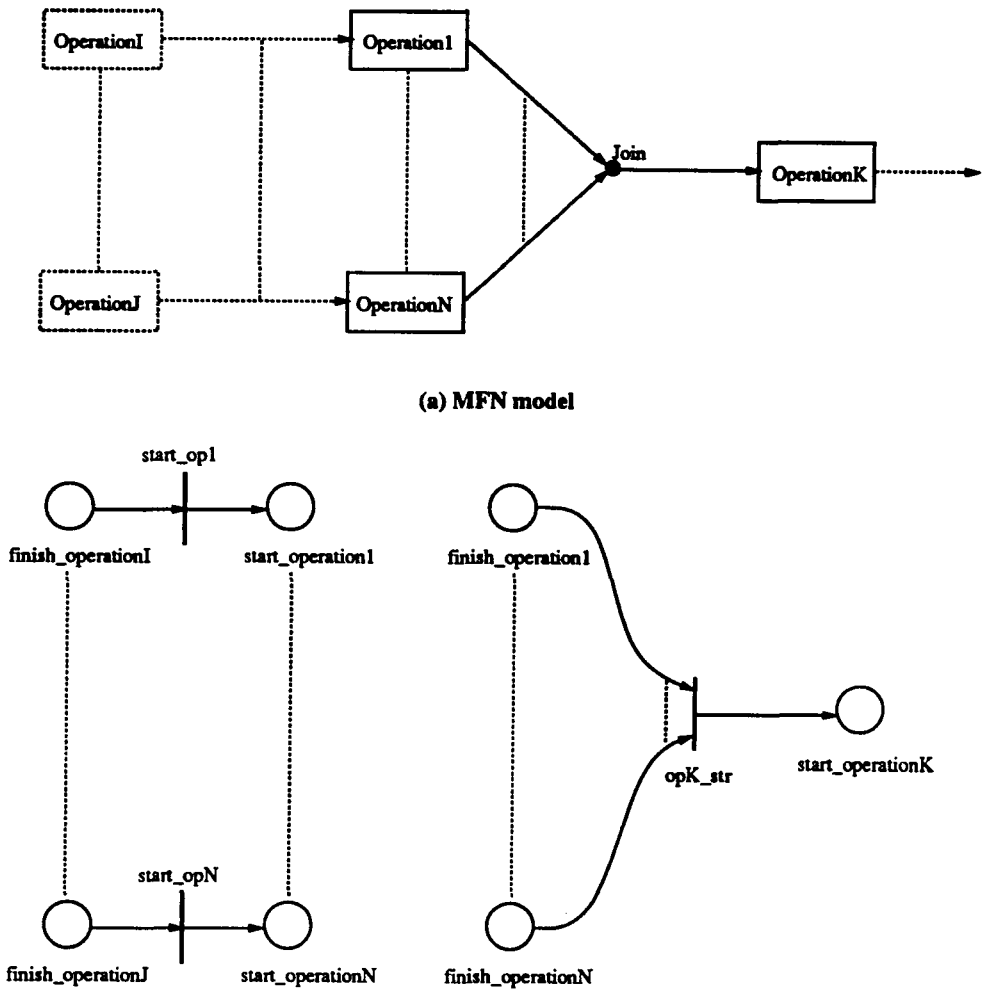


Figure 3.18: MFN and SAN Models of the Intermediate Stage of Product Flow between Blocks and a Join Node

3.5.2 Conversion of MFN into SAN Models for Product Flow at the Machine Level in Manufacturing System

In this section, the conversion of MFN into SAN models for product flow at the machine level is considered. In remainder of this section, the flow of product at the machine level is referred to as operation flow, since we are now representing the flow of product through an operation. As with product flow, operation flow can be shown in three different stages: the first, the intermediate and the last. The first stage of operation flow can be either to a block or a select or fork node. Figure 3.19 represents operation inflow to a block. In this model, timed activity *operation_arrival* is shown dotted because operation flow arrival occurs form another operation completion, or an arrival at the product level. Here the place *Machine1* represents a machine of type machine1, and place *start_operation* represents the start of the operation. Instantaneous activity *Mach_str* is used to indicate the start of the task on machine1. Places *start_Machine1_op* and *finish_Machine1_op* represent the starting and ending point of the task performed on Machine1 for given operation. If we have more than one machines of type machine1 (e.g. K, in the figure) then the places *start_Machine1_op_I* and *finish_Machine1_op_I* represent the starting and the ending points of the task performed on the machine of type machine1 in I^{th} position. *Machine1_op_I*, represents whether the machine of type machine1 is busy in the I^{th} position. In this SAN, the input gate *Machine1_alloc* is used to assign an available machine of type machine1 to the product and start processing at the machine.

Figure 3.20 represents operation inflow into a select node and its MFN and SAN model representation. In this SAN, the timed activity *operation_arrival* is shown dotted because operation flow arrival occurs from another operation completion, or an arrival at the product level. machine1, ..., machineN are the N different types of machines available for performing this operation. Place *start_operation* holds requests to start the operation. Instantaneous activity *Mach_str* is used to allocate a machine to perform the operation. When this activity completes, input gate *Machine_alloc*

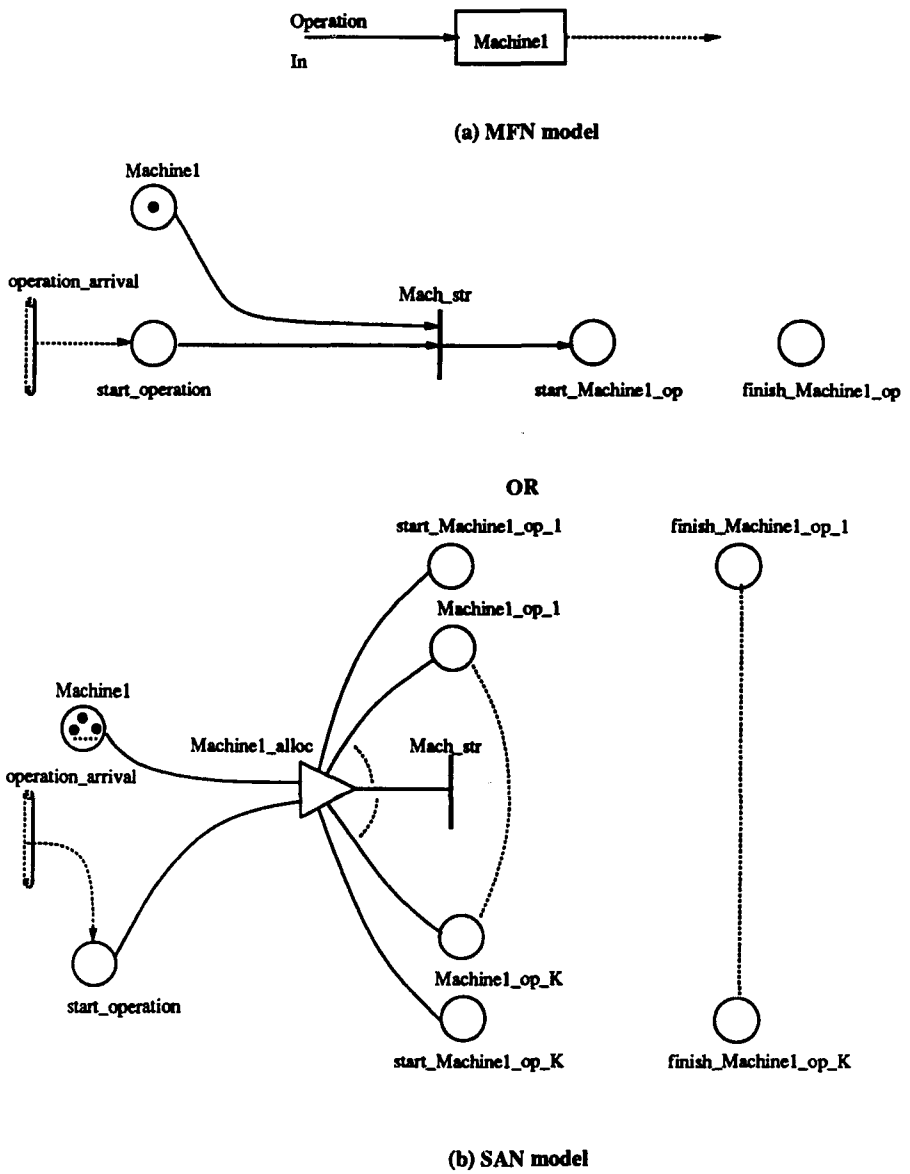


Figure 3.19: MFN and SAN Models of Operation Inflow to a Block

selects a machine to be used from the set of available machines. Places *start_Machine1_op*, ..., *start_MachineN_op* and *finish_Machine1_op*, ..., *finish_MachineN_op* represent the starting and ending points of the tasks performed on machines of types machine1, ..., machineN. The place *Track_Machine* is used for implementing different scheduling policies. For example, the place can be used to keep track of which type of machine was last used for operation, and can be used to implement round-robin scheduling policy.

If we have multiple machines of a single type, the model looks somewhat different. For illustration purposes, we consider in the lower SAN of figure 3.20, which shows the scheduling of multiple machines of type 1. Here the places *start_Machine1_op_I* and *finish_Machine1_op_I* represent the starting and ending points of the task performed on the machine of type machine1 in the I^{th} position. *Machine1_op_I* represents whether the machine of type machine1 is busy in the I^{th} position. Probabilistic scheduling of machines is also possible, but not illustrated since it is relatively uncommon in manufacturing systems. However, such models would be very similar to models used for probabilistic selection of operations in the previous discussion.

Figure 3.21 illustrates the conversion of a MFN which represents operation inflow into a fork node to the equivalent SAN model. Here the timed activity *operation_arrival* is shown dotted because operation flow arrival occurs from another operation completion, or an arrival at the product level. machine1, ..., machineN are N different types of machines available for performing the operation. A token in place *start_operation* represents a request to start the operation. Instantaneous activity *Mach_str* is used to represent operation inflow splitting to machines of N different types. The places *Machine1_op_str*, ..., *MachineN_op_str* represent divergence of operation inflow. Instantaneous activities *Ma1_str*, ..., *MaN_str* are used for starting tasks at the associated machines, by placing tokens in places *start_Machine1_op*, ..., *start_MachineN_op* and *finish_Machine1_op*, ..., *finish_MachineN_op* respectively, when the appropriate machine is available.

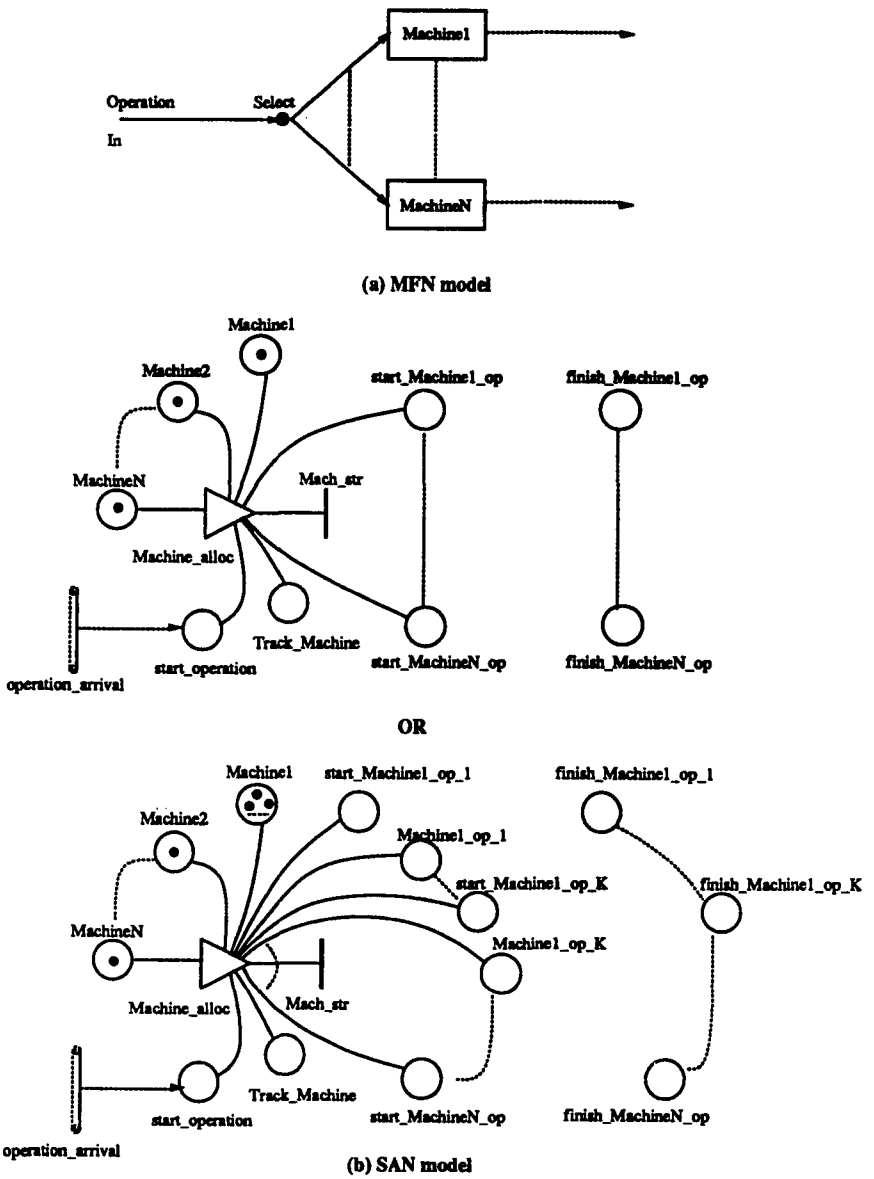


Figure 3.20: MFN and SAN Models of Operation Inflow to a Select Node

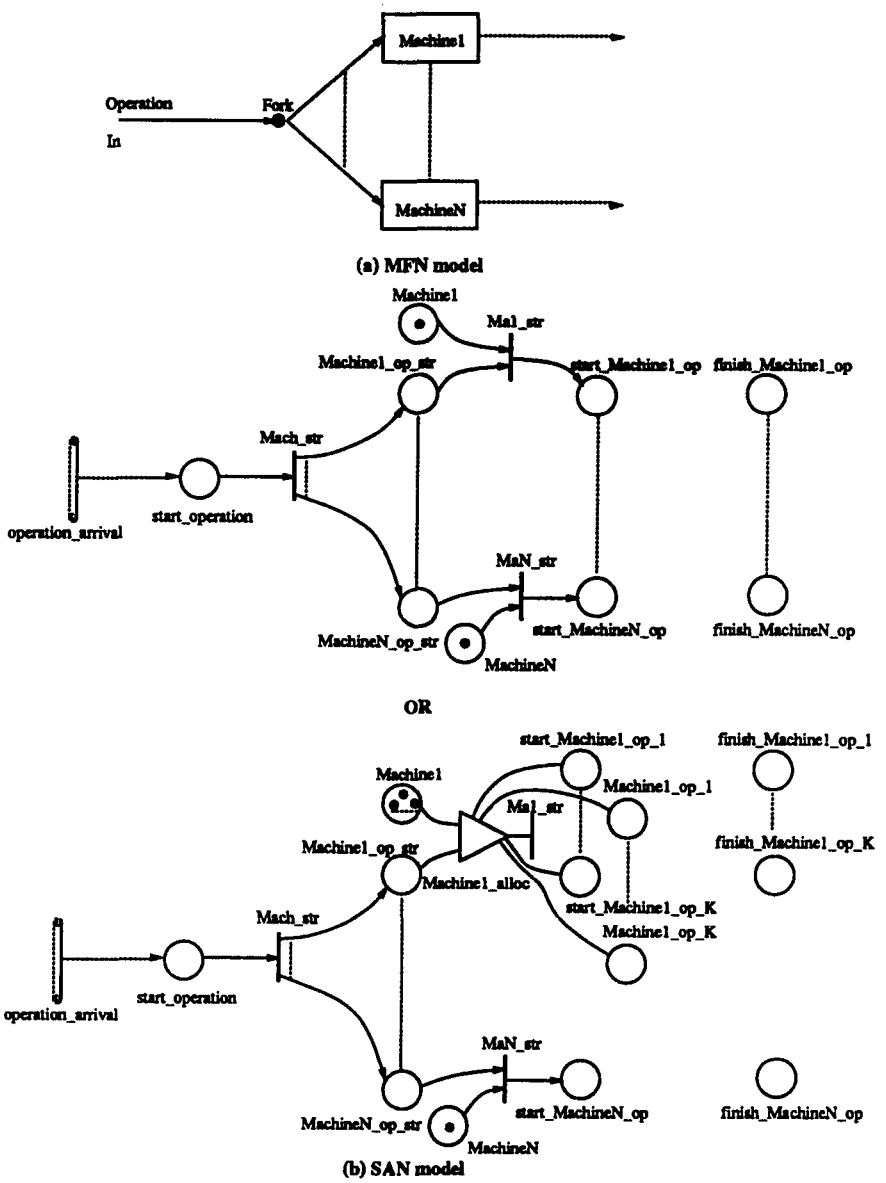


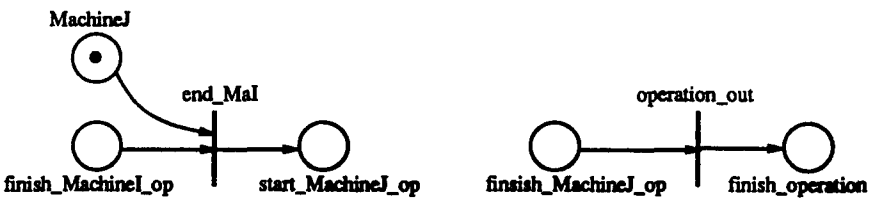
Figure 3.21: MFN and SAN Models of Operation Inflow to a Fork Node

As with the previous SAN conversion discussion, the resulting SAN is somewhat more complicated if multiple machines of each type are considered. The lower SAN in figure 3.21 illustrates this for multiple machines of type 1. Here the places *start_Machine1_op_I* and *finish_Machine1_op_I* represent the starting and the ending points of the task performed on machine1 in the I^{th} position, and *Machine1_op_I* represents whether the machine of type machine1 is used in the I^{th} position. Furthermore, the input gate *Machine1_alloc* is used to assign available machine of type machine1 and it starts task at the available position out of a total of K positions.

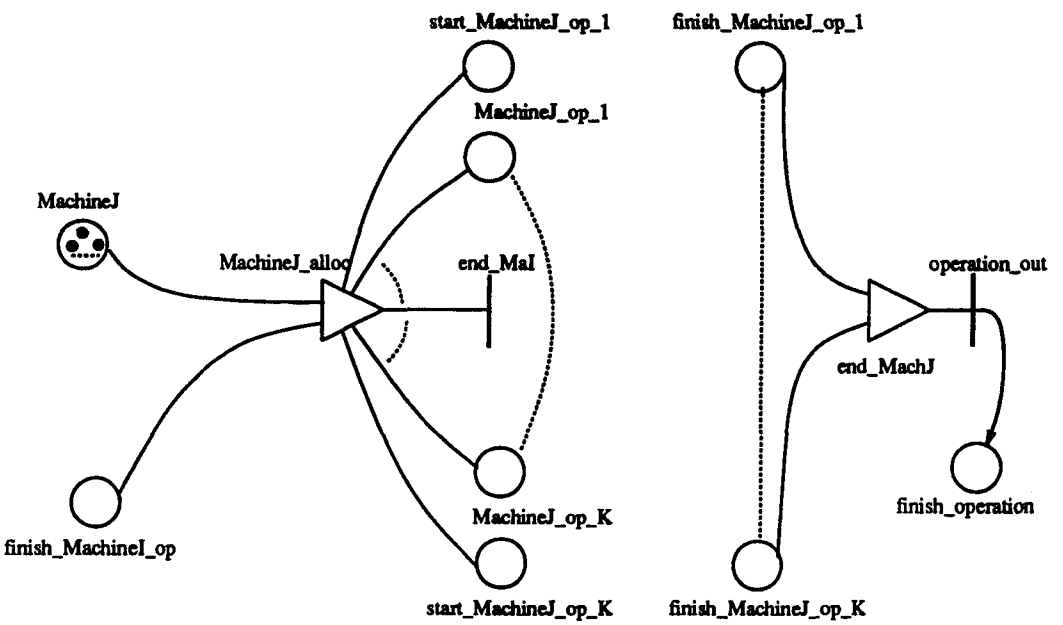
The conversion of MFN models of operation of operation outflow to their corresponding SAN models are now considered. SAN models for the last stage of operation, that is, operation outflow are shown. Operation outflow can be either from a block or from a node. Figure 3.22 shows the corresponding SAN model for a MFN where operation outflow is from a block. In this SAN, a token in place *finish_MachineI_op* represents the completion of a task on the preceding machine. Completion of instantaneous activity *end_MaI* represents the completion of a task on a machine of type machineI and the starting of the task on machineJ. Places *start_MachineJ_op* and *finish_MachineJ_op* represent the starting and ending points of the task performed on machineJ. Place *finish_operation* represents the ending point of the operation.

The extension to multiple machines of a single type is shown in figure 3.22. In this figure, places *start_MachineJ_op_I* and *finish_MachineJ_op_I* represent the starting and ending points of the task performed on the machine of type machineJ in the I^{th} position, and *MachineJ_op_I* represents whether the machine of type machineJ is used in the I^{th} position. Here the input gate *MachineJ_alloc* is used to assign available machine of type machineJ and it starts task at the available position out of a total of K positions. Instantaneous activity *operation_out* represents the outflow of operation.

(a) MFN model



OR



(b) SAN model

Figure 3.22: MFN and SAN Models of Operation Outflow From a Block

Figure 3.23 illustrates conversion of a MFN representing operation outflow from a merge node into the corresponding SAN model representation. Here, places *finish_MachineI_op*, ..., *finish_MachineJ_op* represent the ending points of the task performed on machines of types machineI, ..., machineJ. Places *start_Machine1_op*, ..., *start_MachineN_op* and *finish_Machine1_op*, ..., *finish_MachineN_op* represent the starting and the ending points of the tasks performed on the machines of types machine1, ..., machineN respectively. Input gate *end_opr* is checks if any of the machines has finished task for the given operation. A scheduling policy can be implemented by using input gate function. For example, the place *Track_Mach* can be used to keep track of which type of machine was last chosen for operation outflow, and can be used to implement round-robin scheduling policy. If any machine has finished its task then the instantaneous activity *operation_out* is enabled and completion of this activity is outflow of operation. Place *finish_operation* represents the end of operation.

As with the previous SAN conversion discussion, the resulting SAN is somewhat more complicated if multiple machines of each type are considered. The lower SAN in figure 3.23 illustrates this for multiple machines of type 1. Here places *start_Machine1_op_I* and *finish_Machine1_op_I* represent the starting and ending points of task performed on the machine of type machine1 in the I^{th} position, and *Machine1_op_I* represents whether the machine of type machine1 is used in the I^{th} position. Furthermore, the input gate *Machine1_alloc* is used to assign available machine of type of machine1 and it starts task at the available position out of total K positions.

Finally, figure 3.24 illustrates the conversion of a MFN which represents operation outflow from a join node to the equivalent SAN model. Places *finish_MachineI_op*, ..., *finish_MachineJ_op* represent the ending points of tasks performed at the machines of types machineI, ..., machineJ for the given operation. Places *start_Machine1_op*, ..., *start_MachineN_op* and *finish_Machine1_op*,..., *finish_MachineN_op* represent the starting and ending points of the tasks performed on the machines of types machine1, ..., machineN respectively. As with the previous SAN conversion

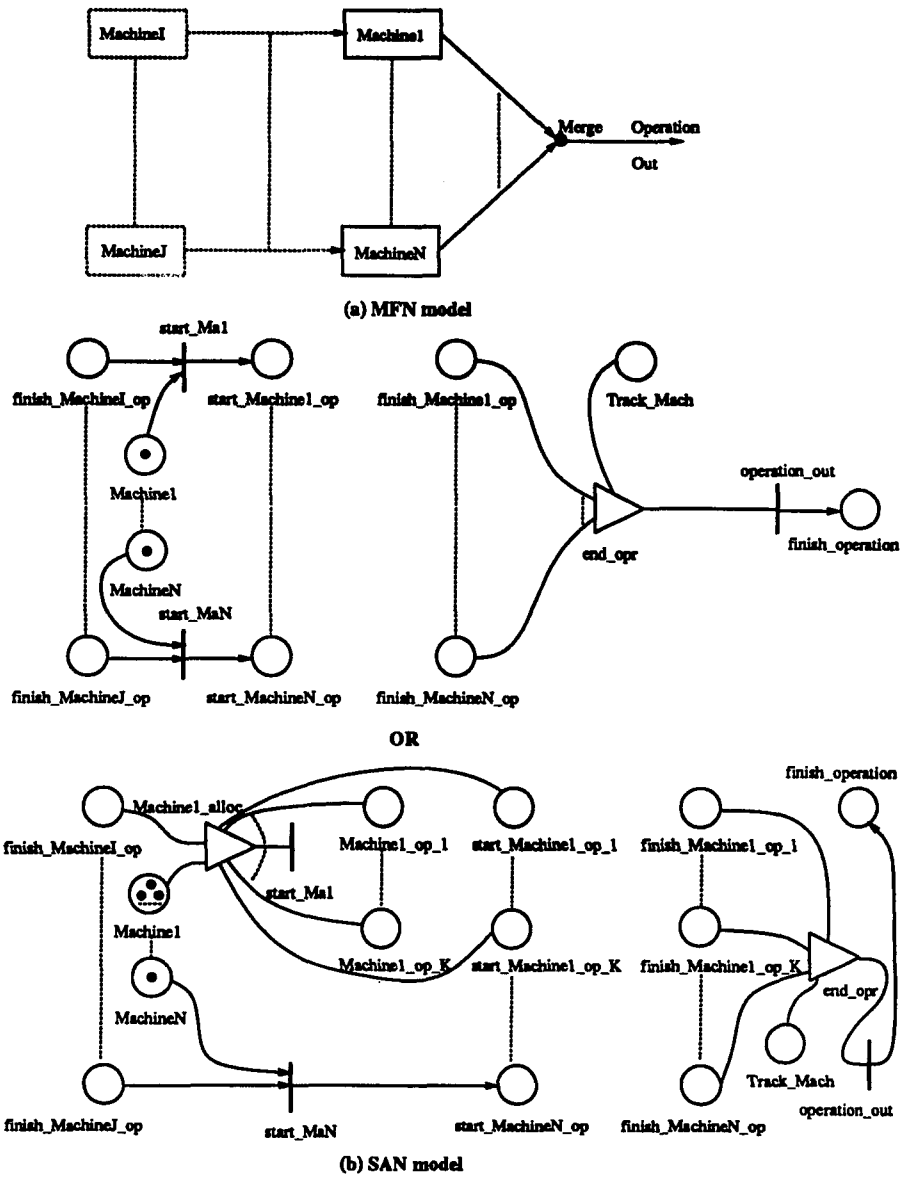


Figure 3.23: MFN and SAN Models of Operation Outflow From a Merge Node

discussion, the resulting SAN is somewhat more complicated if multiple machines of each type are considered. The lower SAN in figure 3.24 illustrates this for multiple machines of type 1. Here the places *start_Machine1_op_I* and *finish_Machine1_op_I* represent the starting and the ending points of the task performed on the machine of type machine1 in the I^{th} position, and *Machine1_op_I* represents whether the machine of type machine1 is used in the I^{th} position. Furthermore, the input gate *Machine1_alloc* is used to assign available machine of type of machine1 and it starts task at any of the available position out of total K positions. Input gate *end_opr* checks that the task is completed on a machine of each type and if so then it enables the activity *operation_out* which represents outflow of operation. Place *finish_operation* represents the ending point of the operation.

Conversion of intermediate stages of MFN models into SANs is straightforward, and does not need to be discussed in detail. Instead, refer to figures 3.25, 3.26, 3.27, 3.28, 3.29, for each of theses cases.

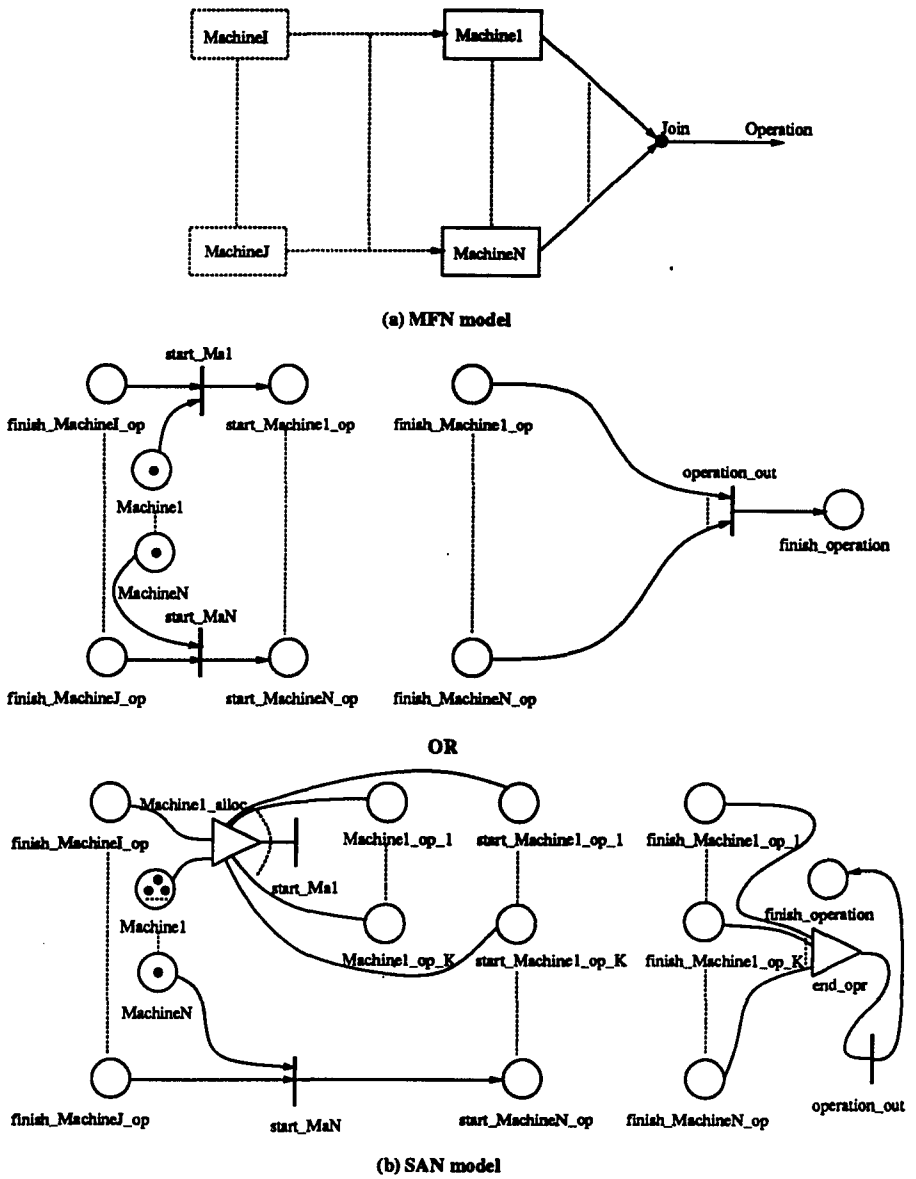
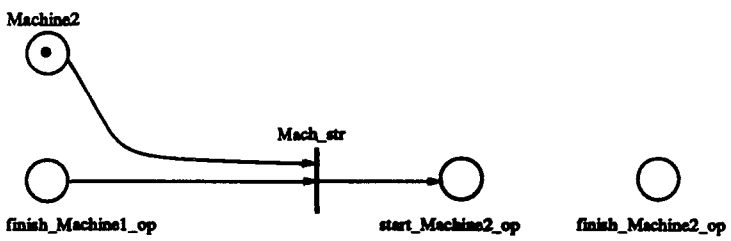
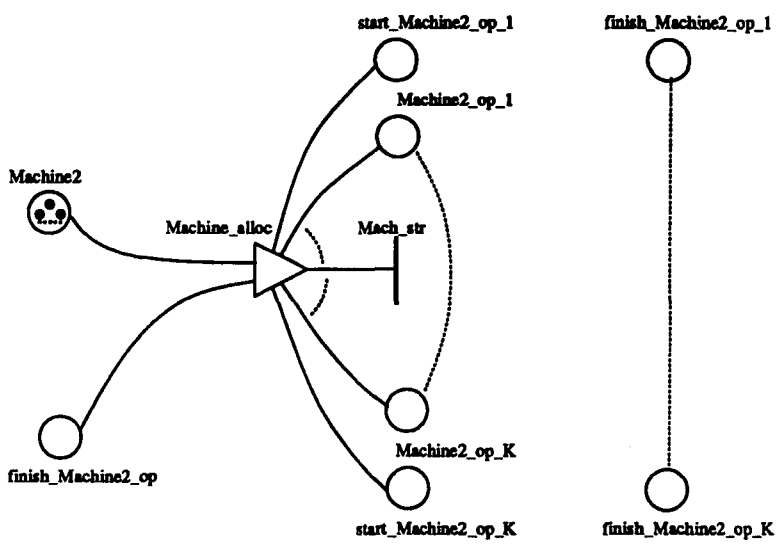


Figure 3.24: MFN and SAN Models of Operation Outflow From a Join Node

(a) MFN model



OR



(b) SAN model

Figure 3.25: MFN and SAN Models of the Intermediate Stage of Operation Flow Between 2 Blocks

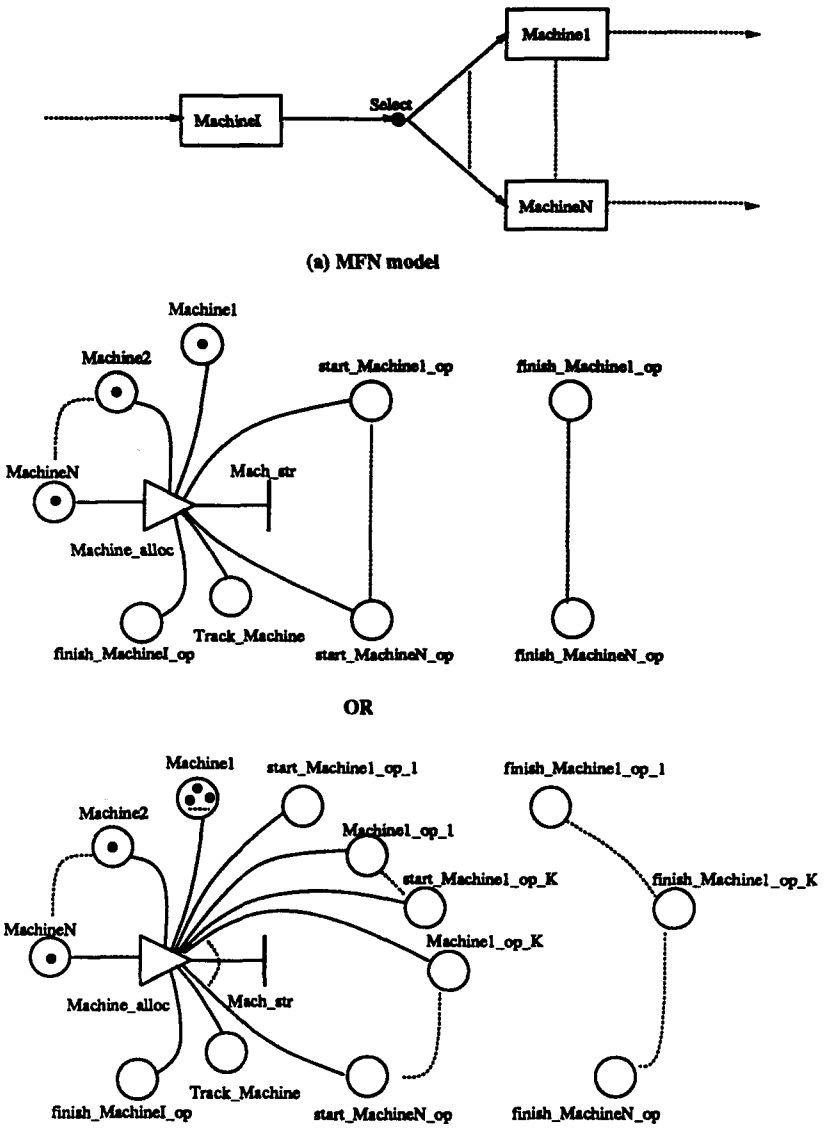


Figure 3.26: MFN and SAN Models of the Intermediate Stage of Operation Flow Between a Select Node and Blocks

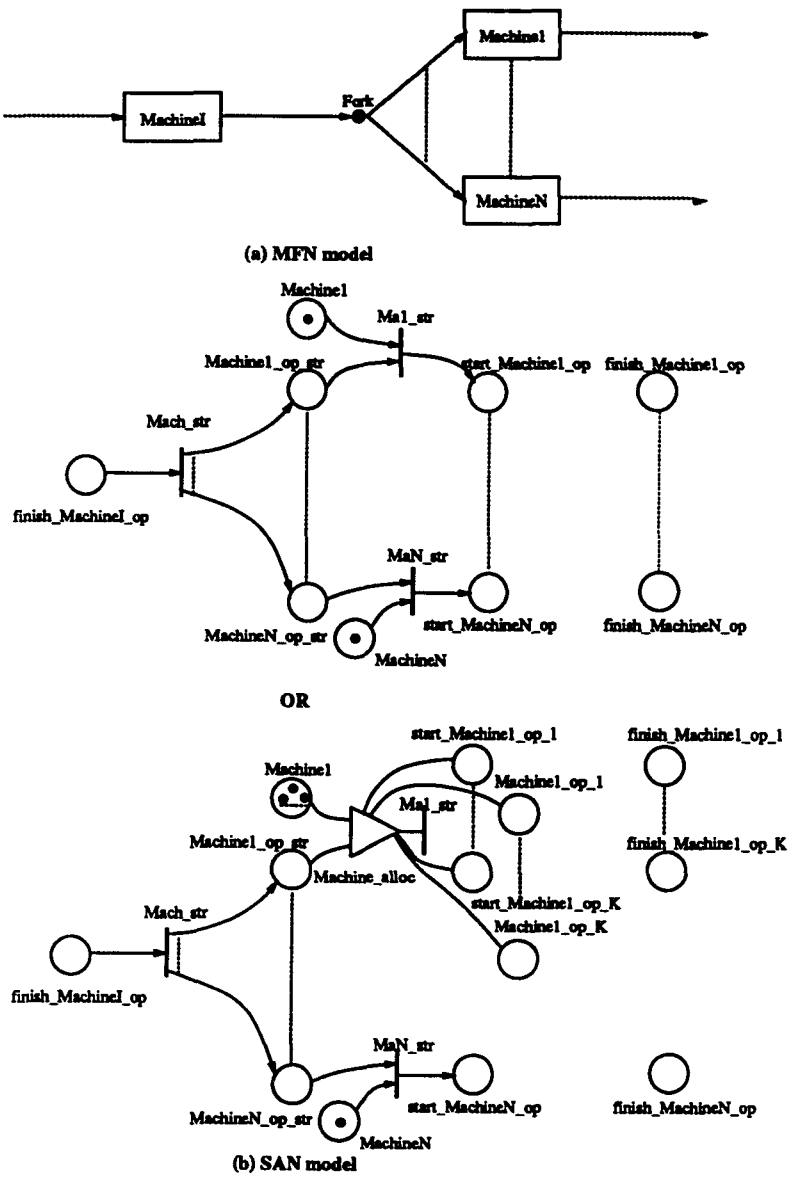


Figure 3.27: MFN and SAN Models of the Intermediate Stage of Operation Flow Between a Fork Node and Blocks

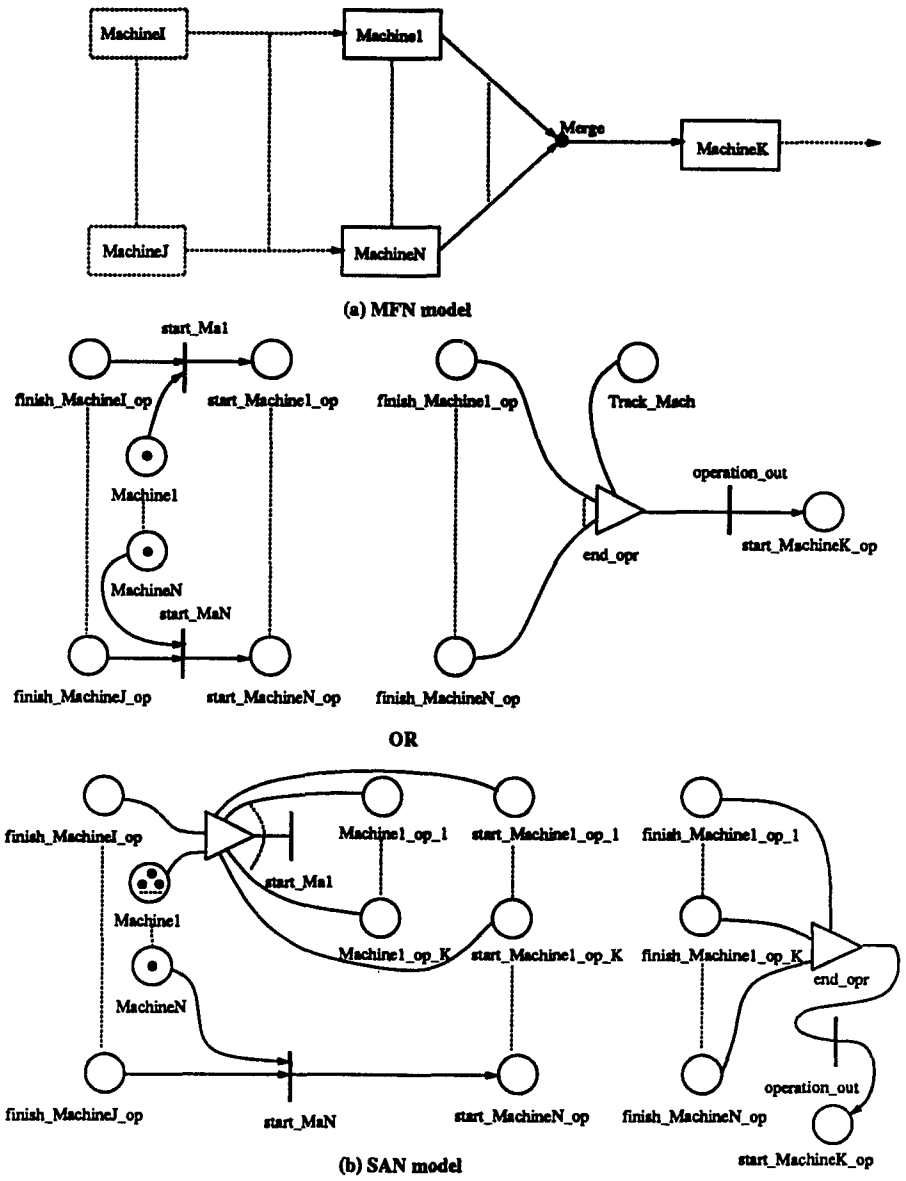


Figure 3.28: MFN and SAN Models of the Intermediate Stage of Operation Flow Between Blocks and a Merge Node

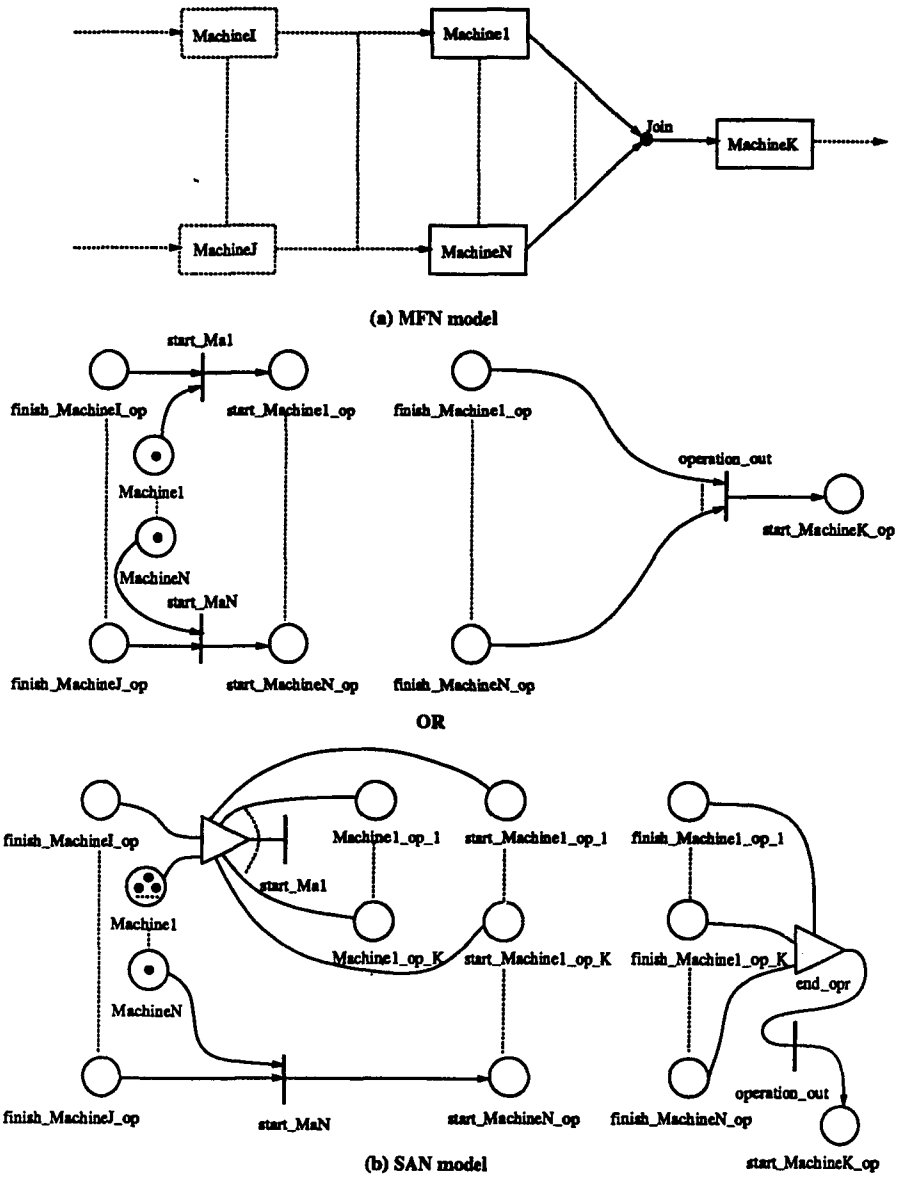


Figure 3.29: MFN and SAN Models of the Intermediate Stage of Operation Flow Between Blocks and a Join Node

CHAPTER 4

Control and Communication Layers of Manufacturing System

In the previous chapter, a new graphical representation, the manufacturing flow network was introduced to facilitate the understanding of flow in manufacturing systems. Conversion of MFNs into SANs was done to solve the MFN models. At the machine level, each block representing a machine can go through one or more control actions. In manufacturing systems, these control activities are very closely tied with production activities. Control of the production of a product is either done manually or by machines. In modern manufacturing systems, automated control is more common than manual control. In such systems, the controllers can be robots, machine tools, material handling systems, and automatic guided vehicles, among others.

When machines are automated, considerable intelligence is required to perform tasks and a significant amount of data transfer, between the machines and/or controlling nodes, is often necessary. For this data transfer to occur some type of communication medium is required. These days, the use of computer networks in manufacturing environments as a communication medium is gaining popularity. This chapter is devoted to studying the modeling of the control and communication layers of manufacturing systems. Models will be developed that can interact with the manufacturing flow layer models, providing a complete three layer hierarchy.

4.1 Control Layer

The control layer of the manufacturing system is discussed first. In general, some of the main objectives of control layer of the manufacturing system are as follows:

1. To support the material handling at the equipment or machines of manufacturing systems.
2. To coordinate the actions or tasks performed on machines.
3. To monitor tasks performed on machines.
4. To assist the system with error messages in case of a machine failure or an operational failure.
5. To provide status of tasks performed on machines.

In order to model the control layer, models representing the sequence of tasks a machine performs are used. There are some significant issues related to modeling the control layer which need to be considered:

1. Mutual exclusion should be provided for shared resources. This can be done by having common places representing these resources in different SAN models, both at the manufacturing flow layer and the control layer.
2. The failures of operations and of machines should be represented. This can be done by having more than one case associated with a timed activity in the SAN model of the machine which represents the action or task performed at the machine. Failure of the machine itself can also be represented by a timed activity.
3. Control layer models can have common places with models of both the manufacturing flow layer and communication layer. Thus, the control layer provides a bridge between manufacturing flow layer and communication layer. This means that the control layer should be able

to handle tasks introduced by the manufacturing flow layer as well as by the communication layer.

4. The model should be able to handle the control actions associated with a task. These can be shown by having output gates on the output side of each timed activity which represents a task performed at machine whose output functions perform the control actions. A model can have starting of a task dependent on successful completion of the previous task and correct response from the communication medium.

4.2 Models of Control Layer

SAN models of control layer are now discussed. A simple model is shown in figure 4.1. In this model, the place *start_Machine_op* represents a request to start an operation on a machine. Timed activity *step* represents time to perform the task at the machine. After completion of a task, a message is sent through the network and the machine waits for the response from main controller or main database. Here the place *message* represents the message request which may be put on the network while the place *response* represents the response which may be received from the network. Finally, place *end_step* represents the completion of a task by a machine. After receiving a response from the network, the instantaneous activity *end* then frees the machine. The number of tokens in place *Machine* represents the number of machines of a certain type and the placement of a token in place *finish_Machine_op* represents the end of an operation performed on the machine.

Table 4.1: Activity Time Distributions of the Simplified Machine Model

Activity	Distribution
end	inst
step	deter(varied)

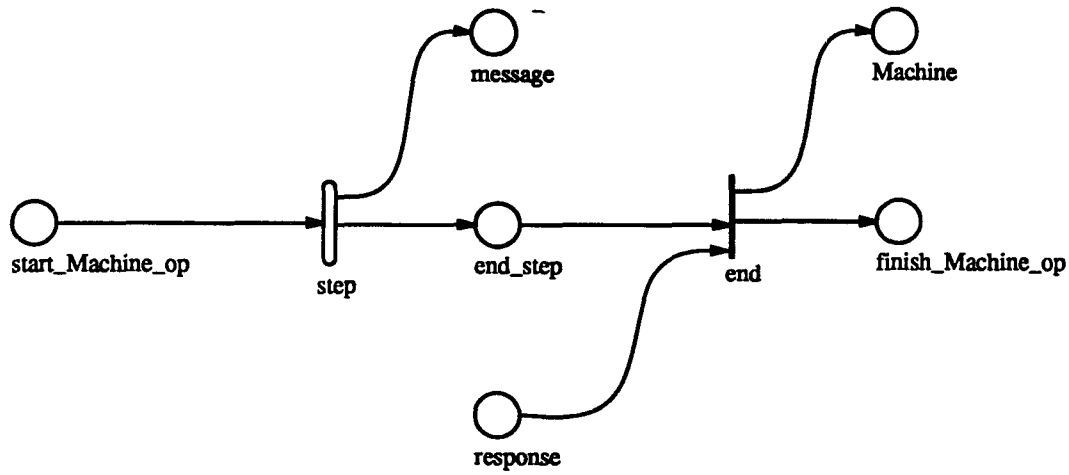


Figure 4.1: A Simplified Machine Model

A more complicated model of machine is given in figure 4.2. This model is similar to the one just discussed. But now instead of performing a single task, the machine performs several tasks to complete its service. Here timed activities *step1*, *step2* and *step3* represent different tasks. There are three cases associated with each of the timed activity. Case 1 represents a complete operation failure, and thus puts a token into place *Machine* in order to free the machine and to abort the operation. Case 2 represents a successful completion of step on the machine. On completion of this case, machine advances to perform next step. Case 3 represents the failure of step and the situation where, the machine must start again from the first step, and hence puts token back in place *start_Machine_op*. In this model, machine failure and repair are not considered, but could be easily implemented by using a seperate SAN model.

Table 4.2: Activity Time Distributions of the Complex Machine Model

Activity	Distribution
end	inst
step1	deter(varied)
step2	deter(varied)
step3	deter(varied)

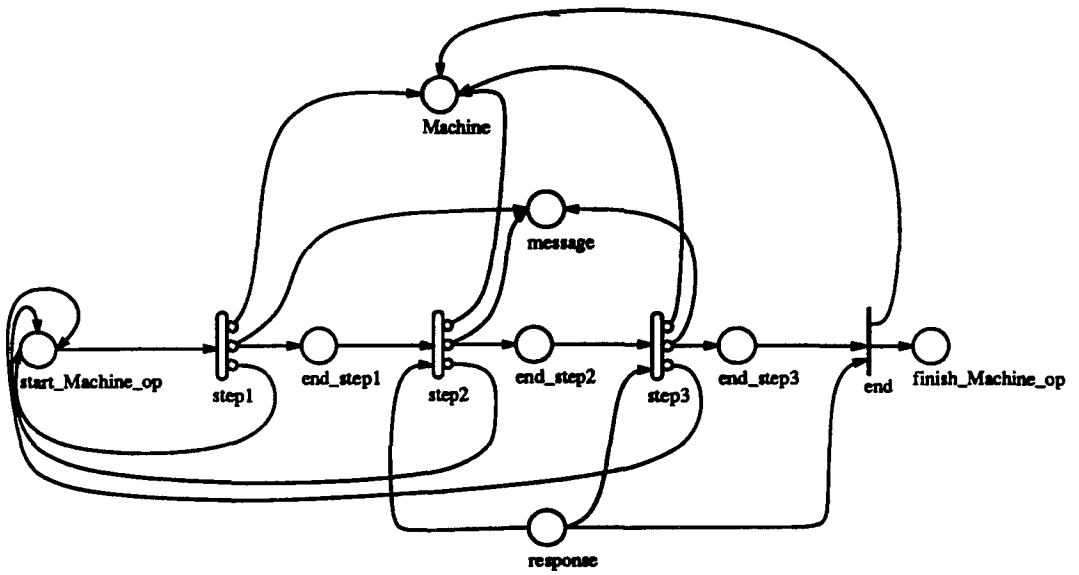


Figure 4.2: A Complex Machine Model

In order to make it easy to model machine failures and repairs, we make one reasonable assumption. In particular, in manufacturing environment a machine does not typically fail frequently relative to part production times. In this case, it is reasonable to assume that if a machine fails during an operation, its failure is deferred until the completion of the operation. Given this assumption, a model of machine failure is shown in figure 4.3. In this model, the place *Mach_working* represents machines those are in a working condition. Here the failure rate of a machine is λ and repair rate is μ. Timed activity *failure* represents the failure of machine and is exponentially distributed. Its rate is dependent on marking of the place *Mach_working*. Place *Mach_failed* represents the number of machines that have failed but not put in for repair. The marking of the place *Machine* represents the number of machines which are available for performing tasks. If the places *Mach_failed* and *Machine* has one or more tokens in them, the instantaneous activity *Fail* is enabled. Upon completion, the instantaneous activity *Fail* puts a token in the place *Mach_fail_in_repair* and removes a token each from the places *Machine* and *Mach_failed*. The marking of place *Mach_fail_in_repair* represents the number of machines put into repair after their

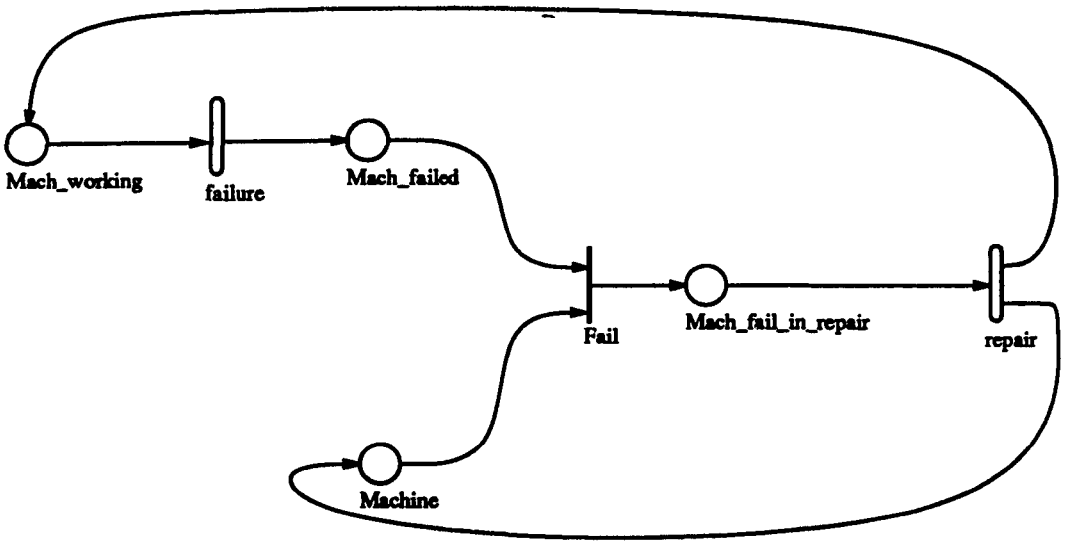


Figure 4.3: A Failure Model of Machine

failure. This model ensures the assumption which was made above because when a machine starts performing tasks a token is taken out from the place *Machine* which is not put back until the completion of tasks. So, while performing tasks that particular machine cannot fail. Timed activity *repair* represents the repairing process of machines and it is exponentially distributed with its rate dependent on the marking of place *Mach_fail_in_repair*. Upon completion, timed activity *repair* puts a token each into places *Mach_working* and *Machine*, thus making a machine again available for performing tasks.

Table 4.3: Case Probabilities for Activities of the Complex Machine Model

Activity	Case	Probability
step3	1	.01
	2	.98
	3	.01
step2	1	.01
	2	.98
	3	.01
step1	1	.01
	2	.98
	3	.01

Table 4.4: Activity Time Distributions of the Failure Model of Machine

Activity	Distribution
Fail	inst
failure	expon(λ * MARK(Machine))
repair	expon(μ * MARK(Mach_fail_in_repair))

4.3 Communication Layer

In many manufacturing systems, a significant amount of data transfer is involved. For this data transfer to occur, computer networks are typically used. The two most commonly used protocols in a manufacturing environment are CSMA/CD and Token bus. Many SAN models of media access protocols exist, and can be directly used at this layer. To illustrate this we discuss a SAN model of a CSMA/CD network. More details regarding a similar model can be found in [34]. A token bus network model is not discussed here but is described in [38].

Figure 4.4 shows CSMA/CD SAN model of a single station. In this model, place *message* represents the queue of message to be transmitted over the channel. After a packet enters this queue, it can begin transmission. Input gate *access* enables activity *Idledly* if the channel is idle or there is an unpropagated message on the channel. Whether a given station will begin transmission or not depends on whether or not the station has just been involved in a collision, and is governed by the standard exponential binary backoff algorithm. The binary backoff algorithm is modeled, in part, by three places (*message*, *go*, and *wait*). In addition, two timed activities are required (*slotdly* and *Idledly*). The timed activity *Idledly* performs the function of delaying until the channel becomes idle. The cases associated with *Idledly* determine whether the station will attempt to capture the bus (case 1) or will defer for a slot time (case 2), which is modeled by activity *slotdly*. The probabilities of these cases are determined by the number of previous collisions that the station has encountered. In particular, probability that a station will attempt to capture the bus is one if there are no tokens in the place *count*. Otherwise, the probability is reduced by powers of two as specified by the binary backoff algorithm. (See table 4.6 , for the exact case probabilities used.)

Given that a transmission is attempted, its time must be modeled. The two-way transmission time for a packet is modeled by the timed activity *send*. A token in place *pack* represents a packet in transmission. Upon completion of the activity *send*, a token is placed in *stop*, indicating

the completion of the transmission for that packet. Termination of a successful transmission is accomplished by the output gate denoted as *remove*. Output gate *remove* has the function of removing the token representing the packet from the place *channel* and resets the count of transmission attempts (*count*) to zero. Completion of the instantaneous activity *stoptrig* puts a token in *response*.

The status of the channel, as seen by the station, is determined by the marking of place *channel*. If there are no tokens in *channel*, then the station is idle. A single token represents unpropagated message on channel. Propagated message is represented by having marking of *channel* as 2. Three tokens in place *channel* represent a collision. Output gate *unprop* puts a token in *channel* when channel is idle and station starts transmitting a message and thus it puts unpropagated message on channel. If *channel* has a token in it (unpropagated message) and the station starts transmitting then *unprop* makes marking of *channel* 3 and which shows that collision has occurred.

When collision is detected, input gate *telljam* enables the instantaneous activity *clear* which removes a token from place *pack* and puts a token each in places *count* and *message*. Thus activity *clear* aborts the transmission and puts a message back into message queue for retransmission. Place *count* is used to keep track of the number of collisions that have occurred in message transmission. Input gate *errorclr* is enabled when the number of collisions reaches some threshold. In this case timed activity *error* will complete, and a message is removed from message queue and marking of place *count* is reinitialized to 0.

The physical layer characteristics of the network can be modeled as shown in figure 4.5. Here the timed activity *propdly* represents the propagation time for a message to propagate on the channel. Here the channel as seen by a station is not distinguished from that seen by other station. This can be considered as described by Prodromides and Sanders [34]. Propagation delay taken here is the time required for a message to propagate between the two farthest stations. This gives the worst case analysis because the number of collisions will now be more than that when the channel

as seen by a station is represented differently. On completion of the activity *propdly*, input gate *pda* changes the marking of *channel* from 1 to 2, thus making a message propagated. When a collision occurs, the input gate *Jamclr* enables the timed activity *jamsend1* which represents the time required to clear a jammed signal from the channel.

We have now completed the development of complete SAN models of all three layers: the manufacturing flow layer, the control layer and the network layer of manufacturing system. In succeeding chapters, we will investigate the analysis and solution of such models.

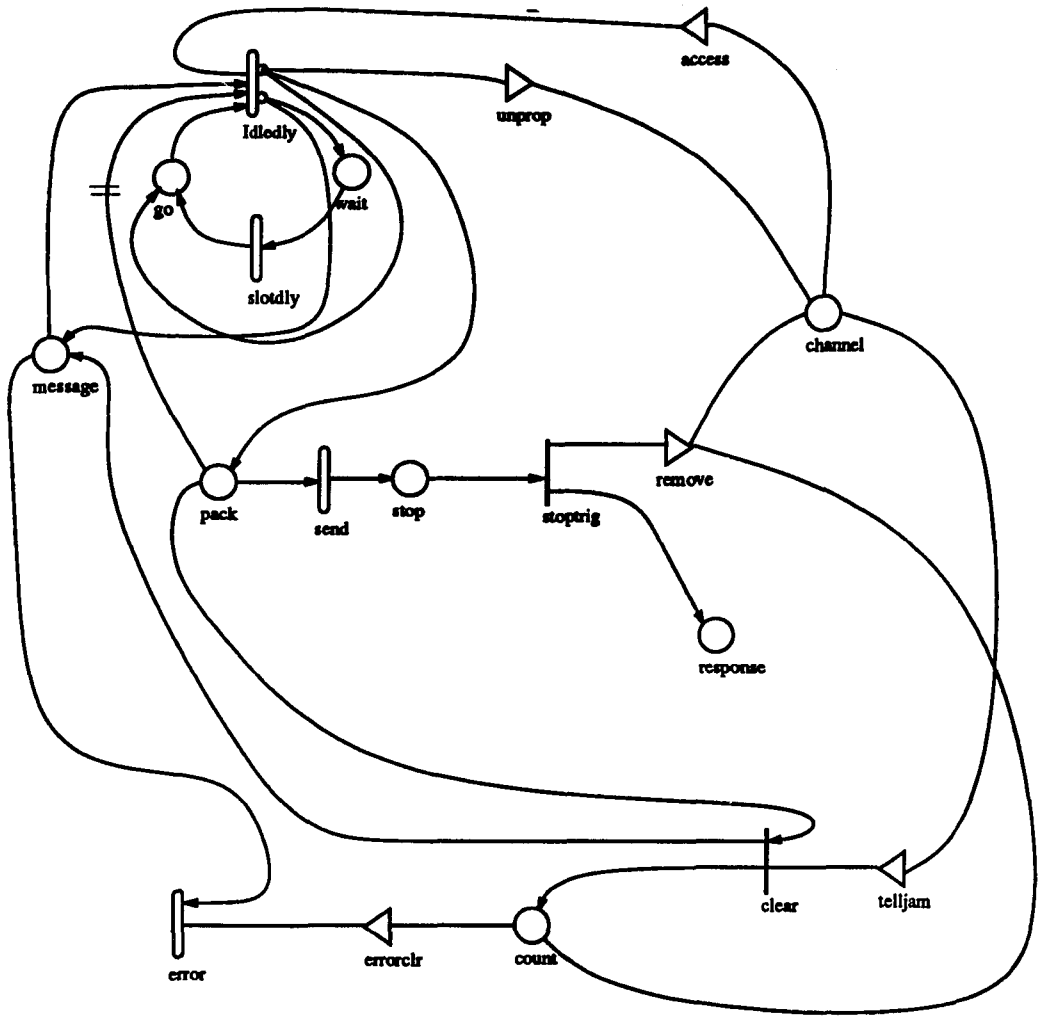


Figure 4.4: CSMA/CD SAN Model of a Single Station

Table 4.5: Activity Time Distributions of the CSMA/CD Model

Activity	Distribution
stoptrig	inst
clear	inst
Idledly	deter(.0000001)
error	deter(.000000001)
send	uniform(.000068,.000576)
slotdly	deter(.000032)

Table 4.6: Case Probabilities for Activities of the CSMA/CD Model

Case	Probability
	Idledly
1	$exp2(-(double)(MARK(count)))$
2	$1 - exp2(-(double)(MARK(count)))$

Table 4.7: Input Gate Predicates and Functions of the CSMA/CD Model

Gate	Enabling Predicate	Function
access	$MARK(channel) ==$ $0 \mid\mid MARK(channel) == 1$	*identity*
telljam	$MARK(channel) == 3$	*identity*
errorclr	$MARK(count) > 15$	$MARK(count) = 0;$

Table 4.8: Output Gate Functions of the CSMA/CD Model

Gate	Function
unprop	$if(MARK(channel) == 0)$ $MARK(channel) = 1;$ $else\ if(MARK(channel) == 1)$ $MARK(channel) = 3;$
remove	$MARK(channel) = 0;$ $MARK(count) = 0;$

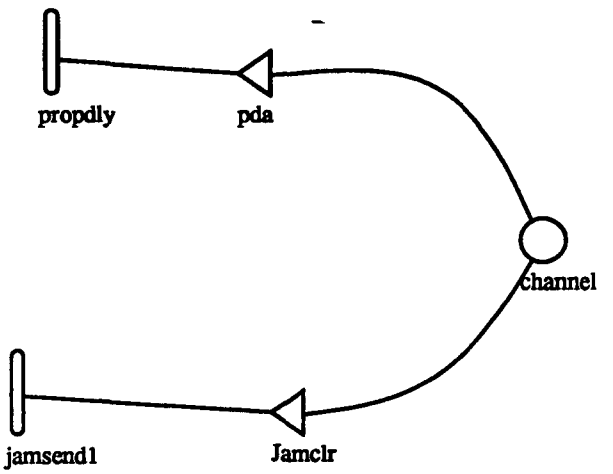


Figure 4.5: SAN Model of Physical Layer of Computer Network

Table 4.9: Activity Time Distributions of the Physical Layer Model

Activity	Distribution
jamsend1	deter(.0000001)
propdly	deter(.00000005)

Table 4.10: Input Gate Predicates and Functions of the Physical Layer Model

Gate	Enabling Predicate	Function
pda	$MARK(channel) == 1$	$MARK(channel) = 2;$
Jamclr	$MARK(channel) == 3$	$MARK(channel) = 0;$

CHAPTER 5

Basic Definitions, Conditions and Analysis of MFNs

In chapter 3 the manufacturing flow networks were qualitatively defined. Several examples of MFNs were considered and their conversion into SAN models was discussed in order to understand the conversion of MFNs to SANs. We now investigate a formal definition of a MFN, and a method by which MFNs can be decomposed at the operation level, for certain variables and parameter values.

Before investigating the method, a formal definition of a MFN is given and connection rules for different elements of MFNs are discussed. Later in the chapter some mathematical proofs are provided which help to decompose MFNs at the operation level. The decomposition of MFNs helps in reducing the complexity of a model (as perceived by the model at model specification time) as well as simulation time required for performance/dependability evaluation.

5.1 Definition of MFN

Formally, a *manufacturing flow network* (MFN) is a tuple

$$MFN = (B, S, M, F, J, I, O, i_B, i_S, i_M, i_F, i_J, i_O, c_B, P_S),$$

where B is a set of *blocks*, S is a set of *select* nodes, M is a set of *merge* nodes, F is a set of *fork* nodes and J is a set of *join* nodes, I is a set of *input blocks* and O is a set of *output blocks*. Furthermore, i_B, i_S, i_M, i_F, i_J, i_O are the element interconnection functions on sets $B \times N$, S, M, F, J, and O respectively, c_B is a copy function assigned to each element of B, and P_S is a

probability function assigned to each element of the output set of each element of S. B, S, M, F, J, I, O all are disjoint and each node and input block represented in a MFN is unique. To aid in describing the connection rules that follow, a *general set G* of a MFN is defined as:

$$G = B \times N \bigcup S \bigcup M \bigcup F \bigcup J \bigcup I.$$

and a *common set C* of a MFN is defined as

$$C = S \bigcup F$$

5.1.1 Element Interconnection Functions

A precise definition of each of the functions that define the interconnection between MFN elements is now given.

1. **Block interconnection**

   A block can occur one or more times in a MFN. To incorporate this feature, the copy function, c_B, which maps an element of B to a positive number, is defined as

   $$c_B : B \rightarrow N$$

   where $c_B(b)$ is the number of copies of b.

   The input function, i_B, which maps a copy of element of B to the input element connected to it, is defined as

   $$i_B : B \times N \rightarrow G$$

   where, $i_B(b, i)$ is the *input element* of i^{th} $(1 \leq i \leq c_B(b))$ copy of b.

   The restrictions on this function are as follows

   (a) If the input elements of the copies of different blocks are the same then they are either a fork node or a select node of the MFN. Mathematically,

$$\text{if } i_B(b_x, i_x) = i_B(b_y, i_y) = \cdots = i_B(b_z, i_z) = c \text{ then } c \in C.$$

Where, $b_x, b_y, \cdots, b_z \in B$, $b_x \neq b_y \neq \cdots \neq b_z$, and $1 \leq i_x \leq c_B(b_x), 1 \leq i_y \leq c_B(b_y), \cdots, 1 \leq i_z \leq c_B(b_z)$.

2. **Select node interconnection**

The input function, i_S, which maps an element of S to the input element connected to it, is defined as

$$i_S : S \rightarrow G$$

where, $i_S(s)$ is the *input element* of s. The *output set of* s, $s \in S$, O_s, is defined as

$$O_s = \{\ g \mid g \in G \text{ and } s \text{ is an input element of } g\ \}.$$

The probability function P_S which is an assignment of probability to each element of O_s for each $s \in S$ is,

$$P_S : \{\ (s,a) \mid s \in S \wedge a \in O_s\ \} \rightarrow [0,1]$$

Here $P_S(s,a)$ is the probability associated with output a of s, where $s \in S$ and $a \in O_s$.

The restrictions on this function are that

(a) A select node cannot be input element of itself. Mathematically,

$$i_S(s) \neq s,\ s \in S.$$

(b) If the input elements of the different select nodes are the same then they are either a fork node or a select node of the MFN. Mathematically,

$$\text{if } i_S(s_x) = i_S(s_y) = \cdots = i_S(s_z) = c, \text{ then } c \in C.$$

Where $s_x, s_y, \cdots, s_z \in S$ and $s_x \neq s_y \neq \cdots \neq s_z$

3. **Merge node interconnection**

The input function, i_M, which maps an element of M to the set of elements which are connected to it, is defined as,

$$i_M : M \rightarrow \wp(G)$$

where, $i_M(m)$ is the *input set of* m, and $\wp$(G) is the power set of G.

The restrictions on this function are that:

(a) A merge node cannot be the element of the input set of itself; i.e.

$$m \notin i_M(m)$$

(b) C is a superset of the intersection of input sets of the different merge nodes. Mathematically,

$$i_M(m_x) \bigcap i_M(m_y) \bigcap \cdots \bigcap i_M(m_z) \subseteq C.$$

Where, $m_x, m_y, \cdots, m_z \in M$ and $m_x \neq m_y \neq \cdots \neq m_z$

4. **Fork node interconnection**

The input function, i_F, which maps an element of F to the input element connected to it, is defined as,

$$i_F : F \rightarrow G.$$

where, $i_F(f)$ is the *input element* of f.

The restrictions on this function are that:

(a) A fork node cannot be the input element of itself. Mathematically,

$$i_F(f) \neq f, f \in F.$$

(b) If the input elements of the different fork nodes are the same then they are either a fork node or a select node of a given MFN. Mathematically,

$$\text{if } i_F(f_x) = i_F(f_y) = \cdots = i_F(f_z) = c \text{ then } c \in C.$$

Where $f_x, f_y, \cdots, f_z \in F$ and $f_x \neq f_y \neq \cdots \neq f_z$

5. **Join node interconnection**

The input function, i_J, which maps the element of J to the set of elements which are connected to it, is defined as,

$$i_J : J \rightarrow \wp(G).$$

Where, $i_J(j)$ is the input set of j, and $\wp$(G) is the power set of G.

The restrictions on this function are that:

(a) A join node cannot be element of the input set of itself.

$$j \notin i_J(j)$$

(b) C is a superset of intersection of the input sets of the different join nodes. Mathematically,

$$i_J(j_x) \bigcap i_J(j_y) \bigcap \cdots \bigcap i_J(j_z) \subseteq C.$$

Where, $j_x, j_y, \cdots, j_z \in J$ and $j_x \neq j_y \neq \cdots \neq j_z$

6. **Output block interconnection**

The input function, i_O, which maps an element of O to the input element connected to it, is defined as,

$$i_O : O \rightarrow G.$$

Here, $i_O(o)$ is the *input element* of o.

The restrictions on this function are that:

(a) If the input elements of the different output blocks are the same then they are either a fork node or a select node of a given MFN. Mathematically,

$$\text{if } i_O(o_x) = i_O(o_y) = \cdots = i_O(o_z) = c \text{ then } c \in C.$$

Where $o_x, o_y, \cdots, o_z \in O \;\&\; o_x \neq o_y \neq \cdots \neq o_z$

5.2 Conditions for Connection of Elements in MFN

In the previous section, the MFN and its members were defined. However, using these definitions, further restrictions are necessary on the connection of MFN elements, in order to obtain MFNs that make physical sense. For example, mathematically, the following can be said:

$$i_B(b,i) = i_S(s) = m. \text{ Where, } b \in B,\ s \in S\ ,\ 1 \leq i \leq c_B(b) \text{ and } m \in M.$$

Above statement is mathematically correct per our previous mathematical definitions of block and select nodes. But, it is definitely wrong because now we have two outgoing arcs on the output side of a merge node m, one on the input side of block b and the other one on the input side of select node s. This violates the description of a merge node. To avoid such situations some more constraints on our definitions are needed. Following are the necessary conditions for interconnecting manufacturing flow network elements.

5.2.1 Block Conditions

1. If the input element of a copy of a block and the input element of a select node of a MFN are the same, then the input element is either a fork node or a select node of the MFN. Mathematically,

if $i_B(b,i) = i_S(s) = c$, then $c \in C$, where $b \in B$, $1 \leq i \leq c_B(b)$ and $s \in S$.

2. If the input element of a copy of a block and the input element of a fork node of a MFN are the same, then the input element is either a fork node or a select node of the MFN. Formally,

if $i_B(b,i) = i_F(f) = c$ then $c \in C$, where $b \in B$, $1 \leq i \leq c_B(b)$ and $f \in F$.

3. If the input element of a copy of a block is the element of the input set of a merge node of a MFN, then it is either a select or fork node of the MFN. Formally,

if $i_B(b,i) \in i_M(m)$ then $i_B(b,i) \in C$, where $b \in B$, $1 \leq i \leq c_B(b)$ and $m \in M$.

4. If the input element of a copy of a block is the element of the input set of a join node of a MFN, then it is either a select node or a fork node of the MFN. Formally,

if $i_B(b,i) \in i_J(j)$ then $i_B(b,i) \in C$, where $b \in B$, $1 \leq i \leq c_B(b)$ and $j \in J$.

5. If the input element of a copy of a block and the input element of an output block of a MFN are the same, then the input element is either a select node or a fork node of the MFN. Formally,

if $i_B(b,i) = i_O(o) = c$ then $c \in C$, where $b \in B$, $1 \leq i \leq c_B(b)$ and $o \in O$.

5.2.2 Select Node Conditions

1. If the input element of a select node and the input element of a fork node of a MFN are the same, then the input element is either a fork node or a select node of the MFN. Formally,

if $i_S(s) = i_F(f) = c$ then $c \in C$, where $s \in S$, $f \in F$.

2. If the input element of a select node and the input element of an output block of a MFN are the same, then the input element is either a fork node or a select node of the MFN. Formally,

$$\text{if } i_S(s) = i_O(o) = c \text{ then } c \in C, \text{ where } s \in S, o \in O.$$

3. If the input element of a select node is the element of input set of a merge node of a MFN, then it is either a fork node or a select node of the MFN. Formally,

$$\text{if } i_S(s) \in i_M(m) \text{ then } i_S(s) \in C, \text{ where } s \in S, m \in M.$$

4. If the input element of a select node is the element of input set of a join node of a MFN, then it is either a fork node or a select node of the MFN. Formally,

$$\text{if } i_S(s) \in i_J(j) \text{ then } i_S(s) \in C, \text{ where } s \in S, j \in J.$$

5.2.3 Fork Node Conditions

1. If the input element of a fork node and the input element of an output block of a MFN are the same, then the input element is either a fork node or a select node of the MFN. Formally,

$$\text{if } i_F(f) = i_O(o) = c \text{ then } c \in C, \text{ where } f \in F, o \in O.$$

2. If the input element of a fork node is the element of input set of a merge node of a MFN, then it is either a fork node or a select node of the MFN. Formally,

$$\text{if } i_F(f) \in i_M(m) \text{ then } i_F(f) \in C, \text{ where } f \in F, m \in M.$$

3. If the input element of a fork node is the element of input set of a join node of a MFN, then it is either a fork node or a select node of the MFN. Formally,

$$\text{if } i_F(f) \in i_J(j) \text{ then } i_F(f) \in C, \text{ where } f \in F, j \in J.$$

5.2.4 Output Block Conditions

1. If the input element of an output block is the element of input set of a merge node of a MFN, then it is either a select node or a fork node of the MFN. Formally,

$$\text{if } i_O(o) \in i_M(m) \text{ then } i_O(o) \in C, \text{ where } o \in O,\ m \in M.$$

2. If the input element of an output block is the element of input set of a join node of a MFN, then it is either a select node or a fork node of the MFN. Formally,

$$\text{if } i_O(o) \in i_J(j) \text{ then } i_O(o) \in C, \text{ where } o \in O,\ j \in J.$$

5.2.5 Merge Node and Join Node Condition

1. The intersection of the input set of a merge node and the input set of a join node of a given MFN must be a subset of C. Formally,

$$i_M(m) \bigcap i_J(j) \subseteq C, \text{ where } m \in M,\ j \in J.$$

These rules together with the formal definition presented earlier, form a complete framework for specifying MFNs. The rate of flow to each MFN element can now be calculated, using the results of the next section.

5.3 Calculation of the Average Arrival Rates in SAN Model Representation of MFN Members

5.3.1 Input Rate Function

Calculation of average arrival rate of product to each element of MFN is important from a performance point of view. Furthermore, under specific conditions, it aids in reducing the complexity of models by allowing a decomposition of MFNs at the operation level which improves

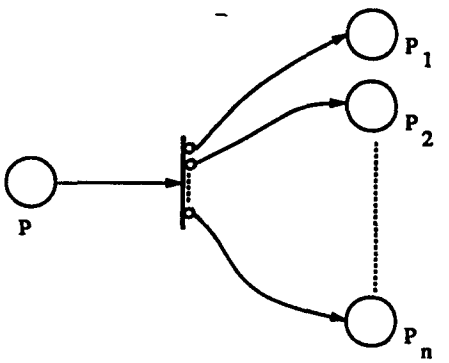


Figure 5.1: SAN Model of a Select Node

the efficiency of solution of models. Formally, the input rate function r, which maps each element of all seven sets of a MFN to a positive real number, is defined as,

$$r : G \bigcup O \rightarrow R^{+}$$

where, $r(x)$ is the input rate of x, $x \in G \bigcup O$. The methods to calculate the average arrival rate of tokens to places of the SAN model representation of different type of simple MFN models are discussed. Furthermore, under specific conditions, the use of these methods in decomposing MFNs at operation level is discussed.

5.3.2 Block

By simple conservation of flow, it can be said that, in steady state, average outgoing rate of product from a block will be the same as the average arrival rate of product arriving in that block. This result holds true because product is neither created nor destroyed in the block.

5.3.3 Select Node

The SAN model representation of a select node is shown in figure 5.1. Let, random variables N_t and N_t^i represent the number of tokens that have arrived in the place P during time $[0, t]$ and the number of tokens arrived in the place P_i during time $[0, t]$, respectively. Furthermore, let λ be

the average arrival rate of tokens at the place P and λ_i be average arrival rate of tokens at place P_i. Now, by the execution rules of a SAN,

$$N_t = N_t^1 + N_t^2 + \cdots + N_t^n \tag{5.1}$$

$$E[N_t] = E[N_t^1] + E[N_t^2] + \cdots + E[N_t^n] \tag{5.2}$$

$$E[N_t] = \lambda \cdot t \tag{5.3}$$

$$E[N_t^i] = \lambda_i \cdot t \tag{5.4}$$

And, by substituting equations 5.3 and 5.4 in 5.2,

$$\lambda \cdot t = \lambda_1 \cdot t + \lambda_2 \cdot t + \cdots + \lambda_n \cdot t, and \tag{5.5}$$

$$\lambda = \lambda_1 + \lambda_2 + \cdots + \lambda_n \tag{5.6}$$

Furthermore, for a probabilistic select where i^{th} case is chosen with probability p_i we have

$$\lambda_i = \frac{p_i}{p_j} \cdot \lambda_j \tag{5.7}$$

Substituting equation 5.7 in 5.6,

$$\lambda_i \cdot \left[\frac{p_1}{p_i} + \frac{p_2}{p_i} + \cdots + \frac{p_n}{p_i}\right] = \lambda \tag{5.8}$$

$$\lambda_i \cdot \left[\frac{p_1 + p_2 + \cdots + p_n}{p_i}\right] = \lambda \tag{5.9}$$

and,

$$p_1 + p_2 + \cdots + p_n = 1 \tag{5.10}$$

Therefore,

$$\frac{\lambda_i}{p_i} = \lambda \tag{5.11}$$

$$\lambda_i = p_i \cdot \lambda \tag{5.12}$$

Therefore, for a probabilistic select, the average arrival rate of tokens at the place P_i is the product of probability of that case and the average arrival rate of tokens at the place P. In general,

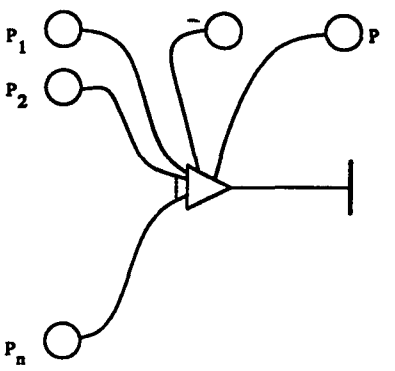


Figure 5.2: SAN Model of a Merge Node

average outgoing rate of product from each outgoing arc is the average incoming rate of product at select node times probability assigned to that outgoing arc.

5.3.4 Merge Node

The SAN model representation of a merge node is shown in figure 5.2. With respect to this figure, let random variables N_t and N_t^i represent the number of tokens which arrive to the place P during time $[0,t]$ and the number of tokens which arrive in the place P_i during time $[0,t]$. Now, let λ be the average arrival rate of tokens at the place P and λ_i be the average arrival rate of tokens at the place P_i. Now, because of the definition of the input gate of a merge node, it can be stated that

$$N_t = N_t^1 + N_t^2 + \cdots + N_t^n \tag{5.13}$$

$$E[N_t] = E[N_t^1] + E[N_t^2] + \cdots + E[N_t^n] \tag{5.14}$$

But,

$$E[N_t] = \lambda \cdot t \tag{5.15}$$

$$E[N_t^i] = \lambda_i \cdot t \tag{5.16}$$

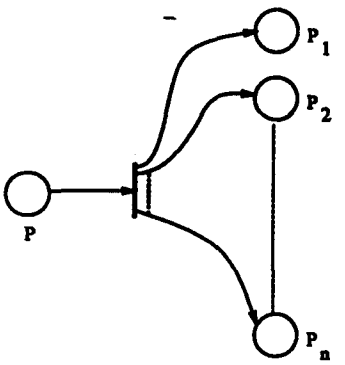


Figure 5.3: SAN Model of a Fork Node

Substituting equations 5.15 and 5.16 into 5.14 we obtain

$$\lambda \cdot t = \lambda_1 \cdot t + \lambda_2 \cdot t + \cdots + \lambda_n \cdot t \tag{5.17}$$

$$\lambda = \lambda_1 + \lambda_2 + \cdots + \lambda_n \tag{5.18}$$

Hence, from this result it can be said that the average arrival rate of tokens at place P is the sum of average arrival rates of tokens at places $P_1, P_2, \cdots, P_n$. In general, the average outgoing rate of product from a merge is sum of average incoming rate of product for each incoming arc of a merge node.

5.3.5 Fork Node

The SAN model representation of a fork node is shown in figure 5.3. Let, random variables N_t and N_t^i represent the number of tokens which arrive in place P during time $[0, t]$ and the number of tokens which arrive in place P_i during time $[0, t]$. Now, let λ be the average arrival rate of tokens at place P and let λ_i be the average arrival rate of tokens at the place P_i. Now, by the execution rules of a SAN, it can be stated that

$$N_t^1 = N_t^2 = \cdots = N_t^n = N_t \tag{5.19}$$

$$E[N_t^1] = E[N_t^2] = \cdots = E[N_t^n] = E[N_t] \tag{5.20}$$

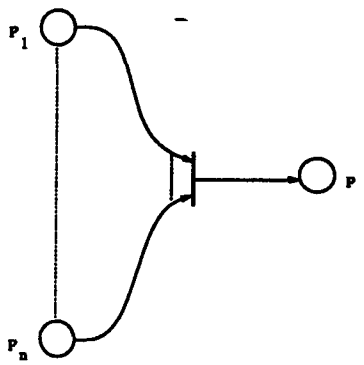


Figure 5.4: SAN Model of a Join Node

Furthermore,

$$E[N_t] = \lambda \cdot t \tag{5.21}$$

$$E[N_t^i] = \lambda_i \cdot t \tag{5.22}$$

Substituting equations 5.21 and 5.22 into 5.20 we get

$$\lambda_1 \cdot t = \lambda_2 \cdot t = \cdots = \lambda_n \cdot t = \lambda \cdot t \tag{5.23}$$

Therefore, the following results from equation 5.23 are obtained.

$$\lambda_1 = \lambda_2 = \cdots = \lambda_n = \lambda \tag{5.24}$$

Thus, the average arrival rate of tokens at the places, $P_1, P_2, \cdots, P_n$ is same as the average arrival rate of tokens at the place P. In general, the average outgoing rate of product from each outgoing arc of fork node is equal to the average arrival rate of product at fork node.

5.3.6 Join Node

Finally, consider the SAN model representation of a join node as shown in figure 5.4. Let the random variables N_t and N_t^i represent the number of tokens which arrive in the place P during time $[0, t]$ and the number of tokens which arrive in the place P_i during $[0, t]$, respectively. Now,

let λ be the average arrival rate of tokens at the place P and λ_i be the average arrival rate of tokens at the place P_i. Now, by the execution rules of a SAN, it can be stated that

$$N_t = min(N_t^1, N_t^2, \cdots, N_t^n) \tag{5.25}$$

$$E[N_t] = \lambda \cdot t \tag{5.26}$$

$$E[N_t^i] = \lambda_i \cdot t \tag{5.27}$$

It can be proved that,

$$\lambda = \lambda_{min} = min(\lambda_1, \lambda_2, \cdots, \lambda_n) \tag{5.28}$$

by induction. Specifically, for $n = 1$,

$$N_t = min(N_t^1) = N_t^1 \tag{5.29}$$

$$E[N_t] = E[N_t^1] = \lambda_1 \cdot t \tag{5.30}$$

$$\lambda = \lambda_1 = min(\lambda_1) \tag{5.31}$$

Therefore, equation 5.28 is correct for $n = 1$. Now assume that result 5.28 is true for $n - 1$, i.e.,

$$N_t = min(N_t^1, N_t^2, \cdots, N_t^{n-1}) = N_t^i \tag{5.32}$$

where

$$\lambda = \lambda_i = min(\lambda_1, \lambda_2, \cdots, \lambda_{n-1}) \tag{5.33}$$

$$E[N_t^i] = \lambda_i \cdot t \tag{5.34}$$

for $i \leq n - 1$.

Now, the case for n is considered. Here,

$$N_t = min(N_t^1, N_t^2, \cdots, N_t^{n-1}, N_t^n) \tag{5.35}$$

There are two possible cases,

- *Case 1:* Let,

$$N_t^n \geq N_t^i$$

therefore,

$$E[N_t^n] \geq E[N_t^i] \tag{5.36}$$

$$\lambda_n \cdot t \geq \lambda_i \cdot t \tag{5.37}$$

which gives the result,

$$\lambda_n \geq \lambda_i \tag{5.38}$$

From 5.33 and 5.38 it can be said that,

$$\lambda_i = min(\lambda_1, \lambda_2, \cdots, \lambda_n) \tag{5.39}$$

Now,

$$N_t = N_t^i \tag{5.40}$$

$$E[N_t] = E[N_t^i] \tag{5.41}$$

$$\lambda \cdot t = \lambda_i \cdot t \tag{5.42}$$

$$\lambda = \lambda_i = min(\lambda_1, \lambda_2, \cdots, \lambda_n) \tag{5.43}$$

Hence, for this case, result as given in eqn. 5.28 is true.

- *Case 2:* Now, the second possibility about N_t^n is,

$$N_t^n < N_t^i$$

therefore,

$$E[N_t^n] < E[N_t^i] \tag{5.44}$$

$$\lambda_n \cdot t < \lambda_i \cdot t \tag{5.45}$$

this gives,

$$\lambda_n < \lambda_i \tag{5.46}$$

Thus from 5.33 and 5.46,

$$\lambda_n = min(\lambda_1, \lambda_2, \cdots, \lambda_{n-1}, \lambda_n) \tag{5.47}$$

Now,

$$N_t = min(N_t^1, N_t^2, \cdots, N_t^n) = N_t^n \tag{5.48}$$

therefore,

$$E[N_t] = E[N_t^n] \tag{5.49}$$

$$\lambda \cdot t = \lambda_n \cdot t \tag{5.50}$$

$$\lambda = \lambda_n = min(\lambda_1, \lambda_2, \cdots, \lambda_{n-1}, \lambda_n) \tag{5.51}$$

Hence, it is proved that result 5.28 is true for n if it is true for $n-1$. Therefore, our claim is true by induction. Thus, the average arrival rate of tokens at the place P is the minimum of average arrival rate of tokens at the places $P_1, P_2, \cdots, P_n$. In general, average outgoing rate of product from a join node is equal to the minimum of average incoming rate of product at each incoming arc of a join node.

5.4 MFN Decomposition

Under specific conditions, a decomposition technique, applied to MFNs using the rate calculations discussed in the previous section, aids in reducing the complexity involved in modeling the manufacturing system represented by MFNs and the simulation time required to evaluate performance variables. Here, it is assumed that internal structure and parameters of copies of a block in a MFN are identical. Furthermore, internal structure and parameters of common blocks

in different MFNs are identical. In this section, the decomposition of MFNs at operation level is discussed. In order to do this, the equivalent arrival rates for different operations of MFNs must be calculated. This can be done by using the results of the previous section. In particular,

1. For each copy of a block,

$$r(b,i) = p \cdot r(x)$$

where, $x \in G$, $b \in B$, $x = i_B(b,i)$. Here if $x = s$, $s \in S$ then $p = P_S(s,b,i)$ else $p = 1$.

2. For each input block,

$$r(i) = \lambda_i$$

where, $\lambda_i \in R^+$

3. For each output block,

$$r(o) = p \cdot r(x)$$

where, $x \in G$, $o \in O$, $x = i_O(o)$. Here if $x = s$, $s \in S$ then $p = P_S(s,o)$ else $p = 1$.

4. For each select node,

$$r(s) = p \cdot r(x)$$

where, $x \in G$, $s \in S$, $x = i_S(s)$. Here if $x = q$, $q \in S$ then $p = P_S(q,s)$ else $p = 1$.

5. For each merge node,

$$r(m) = \sum_{i=1}^{|i_M(m)|} p_i \cdot r(x_i)$$

where, $X = i_M(m)$, $x_i \in G$, $m \in M$ Here if $x_i = s$, $s \in S$ then $p_i = P_S(s,m)$ else $p_i = 1$.

6. For each fork node,

$$r(f) = p \cdot r(x)$$

where, $x \in G$, $f \in F$, $x = i_F(f)$. Here if $x = s$, $s \in S$ then $p = P_S(s, f)$ else $p = 1$.

7. For each join node,

$$r(j) = p_a \cdot r(x_a)$$

where, $X = i_J(j)$, $x_a \in G$, $j \in J$. Here $p_a \cdot r(x_a) = min(p_1 \cdot r(x_1), p_2 \cdot r(x_2), \cdots, p_n \cdot r(x_n))$, where, $n = \mid X \mid$, $x_i \in G$, $j \in J$ and $1 \leq i \leq n$. If $x_i = s$, $s \in S$ then $p_i = P_S(s, j)$ else $p_i = 1$.

8. Now we can assign an input rate function to each element of set B as

$$r : B \rightarrow R^+.$$

So for a block the input arrival rate of product can be calculated as,

$$r(b) = \sum_{i=1}^{c_B(b)} r(b, i).$$

Thus for any given MFN we can write n linear equations of n variables where each variable is of type $r(x)$ as described above, where $n = \mid G \mid + \mid O \mid$. A solution for all $r(x)$ from these n linear equations can be obtained. After getting a solution for all these $r(x)$, the total arrival rate of product to each block in a MFN can be calculated at operation level. Furthermore, when many different MFNs shares blocks at the operation level, the total arrival rate of product to each block which is present in one or more MFNs out of given m MFNs can be calculated. An algorithm for calculating rates for decomposed model is described:

5.4.1 Algorithm for Decomposition of m MFNs at the Operation Level

Construct superblock set(A) as

$$A = \bigcup_{i=1}^{m} B^i$$

(where B^i is block set of i^{th} MFN)

For $j = 1$ to $| A |$

$\Lambda(a_j) = 0$

($\Lambda(a_j)$ is the total input rate for element a_j.)

For $k = 1$ to $| A |$

For $l = 1$ to m

if $(a_k \in B^l)$ then $\Lambda(a_k) = \Lambda(a_k) + r^l(a_k)$.

($r^l(a_k)$ is Input rate calculated for element a_k of l^{th} MFN.)

5.4.2 Calculation of Rates of Operations for Example MFNs

The application of the decomposition technique and algorithm as discussed above to calculate input rate of operations can be illustrated by decomposing example MFNs of figure 3.4. Here, the two input blocks represent product arrivals of productA and productB. Figure 3.4 has two MFNs, one for productA and the other for productB. Equations for both MFNs at operation level can be written as follows:

$$B^1 = \{Operation1, Operation2, Operation3, Operation4\}.$$

$$B^2 = \{Operation1, Operation3, Operation4\}.$$

Now let,

p_1 = Probability of performing Operation1 on productA in the first stage, and

p_2 = Probability of performing Operation2 on productA in the first stage,

where $p_1 + p_2 = 1$.

$$r^1(ProductA) = \lambda_a.$$

$$r^2(ProductB) = \lambda_b.$$

Furthermore, the input rate for each block of both MFNs can be calculated separately. For the first MFN,

$$r^1(Operation1, 1) = p_1.r^1(ProductA) = p_1.\lambda_a.$$

$$r^1(Operation2, 1) = p_2.r^1(ProductA) = p_2.\lambda_a.$$

$$r^1(Operation3, 1) = r^1(Operation1, 1) + r^1(Operation2, 1) = p_1.\lambda_a + p_2.\lambda_a = \lambda_a.$$

$$r^1(Operation4, 1) = r^1(Operation3, 1) = \lambda_a.$$

Since we have only one copy of each block in the first MFN at operation level we can write input rate for its each block as:

$$r^1(Operation1) = p_1.\lambda_a.$$

$$r^1(Operation2) = p_2.\lambda_a.$$

$$r^1(Operation3) = \lambda_a.$$

$$r^1(Operation4) = \lambda_a.$$

Now for the second MFN, the rate equations for each copy of a block are written as

$$r^2(Operation1, 1) = r^2(ProductB) = \lambda_b.$$

$$r^2(Operation3, 1) = r^2(Operation1, 1) = \lambda_b.$$

$$r^2(Operation4, 1) = r^2(Operation1, 1) = \lambda_b.$$

Similarly, we can write input rate for each block of the second MFN as

$$r^2(Operation1) = \lambda_b.$$

$$r^2(Operation3) = \lambda_b.$$

$$r^2(Operation4) = \lambda_b.$$

Here, the set A is

$$A = \{Operation1, Operation2, Operation3, Operation4\}.$$

Now, total input rate for each element of A can be calculated as following:

$$\Lambda(Operation1) = r^1(Operation1) + r^2(Operation1) = p_1.\lambda_a + \lambda_b.$$

$$\Lambda(Operation2) = r^1(Operation2) = p_2.\lambda_a.$$

$$\Lambda(Operation3) = r^1(Operation3) + r^2(Operation3) = \lambda_a + \lambda_b.$$

$$\Lambda(Operation4) = r^1(Operation4) + r^2(Operation4) = \lambda_a + \lambda_b.$$

In this manner, the total input rate of each operation can be calculated by decomposing MFNs at operation level. Thus, analysis of any MFN can be performed in a similar fashion. We now evaluate (solve) both the decomposed and original models of this manufacturing system, in order to investigate the usefulness of the decomposition technique.

5.5 SAN Models for Non-decomposed and Decomposed MFN Models

Using conversion techniques as discussed in Chapter 3, the SAN models for the non-decomposed MFN models as well as decomposed MFN models were developed. The resulting SANs for the non-decomposed MFN models as represented by figures 3.4 and 3.5 are shown in Appendix A. Similarly, SAN models of the manufacturing flow layer for the decomposed MFN models are given in Appendix B. For the control and network layer models, SAN models as shown in figures 4.2, 4.4 and 4.5 are used.

The Following assumptions are made in the models:

1. Product arrivals are exponentially distributed.

2. In the control layer model of the machine, operation failure is not taken into account (i.e. probabilities of case1 of step1, step2 and step3 are 0). This assumption is made so that the

decomposition technique can be applied to the models. This ensures that in steady-state average inflow into a block is same as average outflow from the block.

3. The arrival rates of both products are same.

4. Machine failure model is not included, to reduce the complexity in modeling the whole manufacturing system.

Only the second assumption is needed to apply the decomposition technique to the models. The remaining assumptions were made to facilitate modeling and reducing complexity.

5.6 Performance Variables Specification

For this example, we study the three performance variables listed below:

- the utilization of each machine type, which is defined as the ratio of the number of machines of a particular type to the total number of machines of that type in the system.
- the availability of each machine type, which is defined as the probability that at least one machine of that type available for use.
- the average queue length of an operation, which is the average number of products waiting for some machine to perform the operation.

These performance variables can be measured by defining a reward structure for the constructed SAN models for each variable. In SAN models, impulse rewards are assigned to activity completions and rate rewards are assigned to particular numbers of tokens in places. For example, if there are n number of machines of type machine then, using the complex machine model as represented in figure 4.2, the utilization and availability of machine type are defined in terms of the following reward structure:

The utilization is defined as

$$C_{\mathcal{U}}(a) = 0, \forall \textit{ activities } a$$

$$\mathcal{R}_{\mathcal{U}}(v) = (n-i)/n, \text{ where } v = \{(\text{Machine,i})\}.$$

Here $C_{\mathcal{U}}(a)$ is the impulse reward associated with a, and $\mathcal{R}_{\mathcal{U}}(v)$ is a rate reward defined to be ratio of the machines that are being used in the system (i.e. the total number of machines used in the system - the marking of place *Machine*) to the total number of machines in the system (i.e. the initial marking of place *Machine*). Then, if V_t is the instant-of-time variable [39] at time t and $V_{t\to\infty}$ as the limit of this variable, $E[V_{t\to\infty}]$ is the steady-state utilization of the machine type.

Similarly, the availability of the machine can be defined in terms of a reward structure,

$$C_{\mathcal{A}}(a) = 0, \forall \textit{ activities } a$$

$$\mathcal{R}_{\mathcal{A}}(v) = \begin{cases} 0 & \text{if } v = \{(\text{Machine,0})\} \\ 1 & \text{otherwise.} \end{cases}$$

Here $C_{\mathcal{A}}(a)$ is the impulse reward associated with a, and $\mathcal{R}_{\mathcal{A}}(v)$ is a rate reward of zero, when the marking of place *Machine* is 0, and one, otherwise. Then, if V_t is the instant-of-time variable at time t and $V_{t\to\infty}$ as the limit of this variable, $E[V_{t\to\infty}]$ is the steady-state availability of this type of machine.

5.7 Simulation Results of Performance Variables

In the example study, we study the following specific variables: the utilizations of machines 1-4, the availabilities of machines 1-4, the queue lengths of operations 1-4, utilization of the communication channel, and the probability the channel is idle. Each of these performance variables was estimated using steady-state simulation. The results of these performance variables are as shown in appendix C. These results are shown in graphical form in figures. 5.5 to 5.18.

The decomposition of MFN model does not provide information about the distribution of the operation arrival but only the means, and hence, we must choose (rather arbitrarily) the

distribution of arrivals to operation in the decomposed model. For the sake of comparing, all the performance variables are measured for three different distributions of the operation arrival: exponential, uniform and normal with small variance. Looking at the results and error tables, as well as the curves, the following points can be concluded:

- The decomposition technique gives accurate values of utilization and availability of machines. The curves of the utilization and availability of machines for non-decomposed and decomposed models seem to be quite similar. This can be justified, quite easily, since every product arriving in the system carries some fixed amount of work for particular machines. So, the average utilization and availability of machines depend on average arrival rate of products at machines. The decomposition technique keeps the average arrival rate of products at operations and machines to be same as that for non-decomposed models and decomposed models. So, the values of these performance variables are the same for both non-decomposed and decomposed models.

- However, for the decomposed models, the values of average queue length of operations varies from values obtained for the non-decomposed models. This is because these performance variables are very sensitive to arrival distribution. At the operation level, a block may receive a product arrival from a block, an input block or a node. In a MFN there is a delay associated with each block and each node follows certain rules for inflows and outflows of products from it. The distribution of product arrival at a particular block is determined by all these factors. This knowledge is not captured by the average rates computed using the decomposition technique. Furthermore, since blocks may be common blocks to many MFNs, the distribution of product arrival at a block may be even more difficult. So, by using the decomposition technique without the knowledge of an accurate product arrival distribution at operation level, it is difficult to get correct values of performance variables which depend on the distribution of product arrival at a block.

- Out of all three distributions used for operation arrival, exponential distribution seems to be result in the values of queue length that a closest to the values obtained from the non-decomposed models. This is despite the fact that at very low or high loads, the values vary much more than those at medium loads. Still, these values are not correct.

So, it is concluded that the decomposition technique works well when the performance variables of interest are not dependent on the distribution of product arrival. Under heavy loads the decomposition technique would work well even for the performance variables which are dependent on the distribution of product arrival. This is because now the queues in the system are always full and thus outflow of a product from any element of MFN depends only on the service time distribution at that particular element. Using decomposition, the complexity of the composite model reduces and some speedup is obtained while running the simulation. For the example manufacturing system, a speedup factor of approximately 1.6 in simulation is obtained by using the decomposition technique.

In this chapter, the MFN and its connection rules were defined. A decomposition technique, which can be used for certain variables, was developed to reduce complexity of models and simulation time required to evaluate the performance of manufacturing system. The usefulness of MFN and SAN models for modeling a simple, but realistic manufacturing system is illustrated in the next chapter.

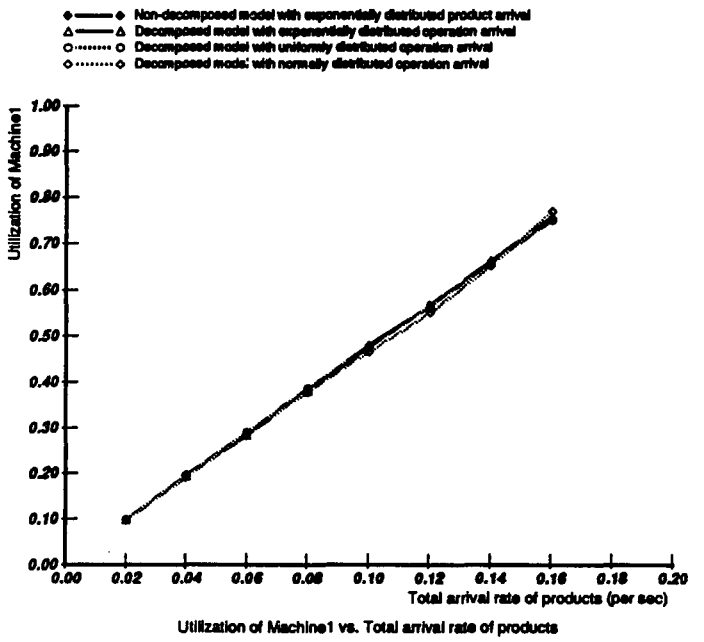


Figure 5.5: Utilization of Machines of Type 1

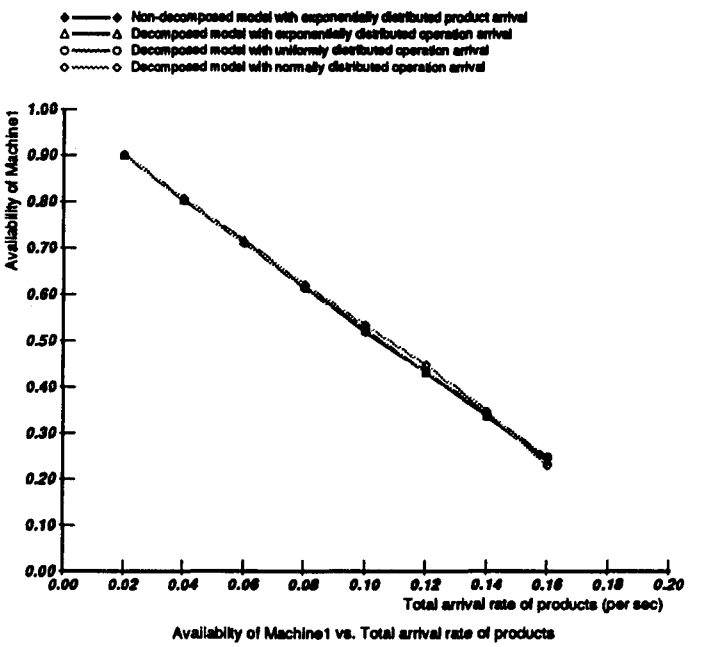


Figure 5.6: Availability of Machines of Type 1

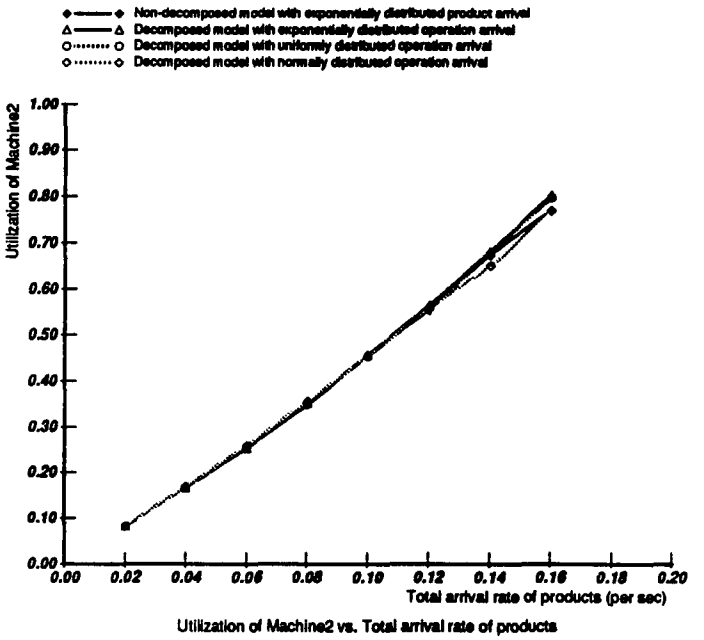


Figure 5.7: Utilization of Machines of Type 2

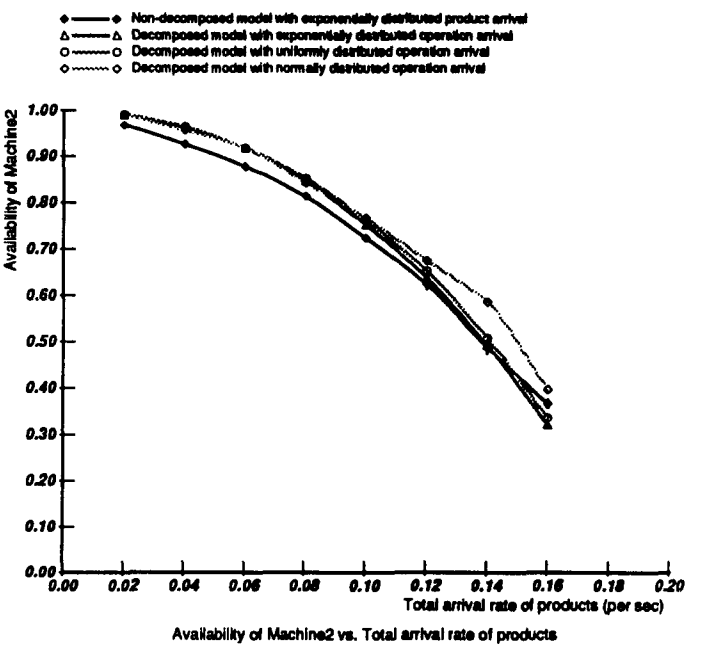


Figure 5.8: Availability of Machines of Type 2

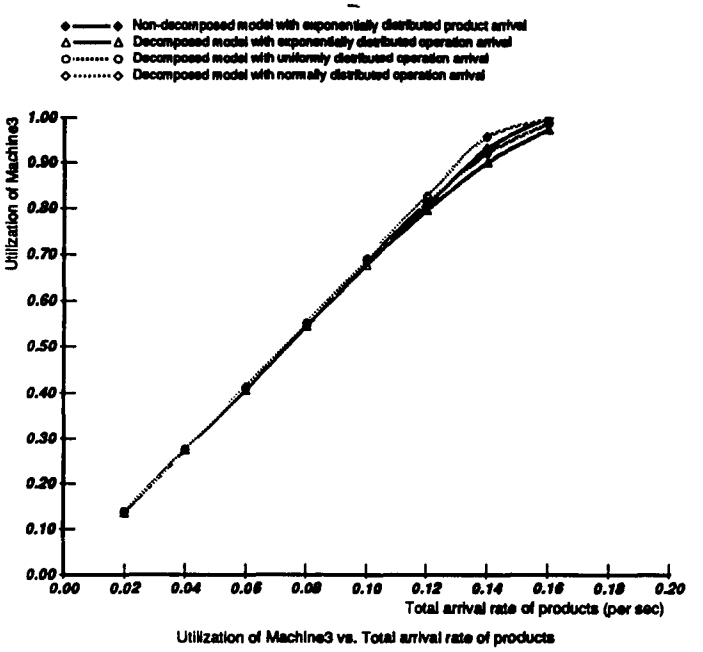


Figure 5.9: Utilization of Machines of Type 3

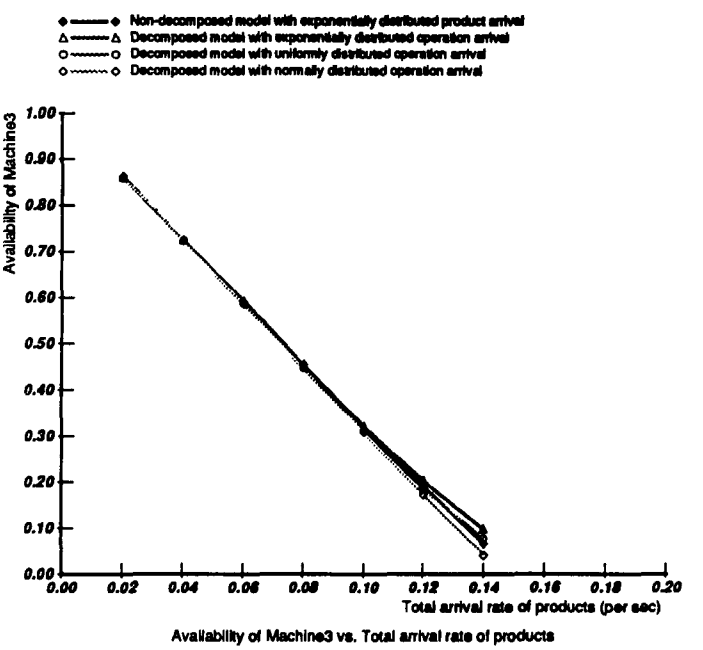


Figure 5.10: Availability of Machines of Type 3

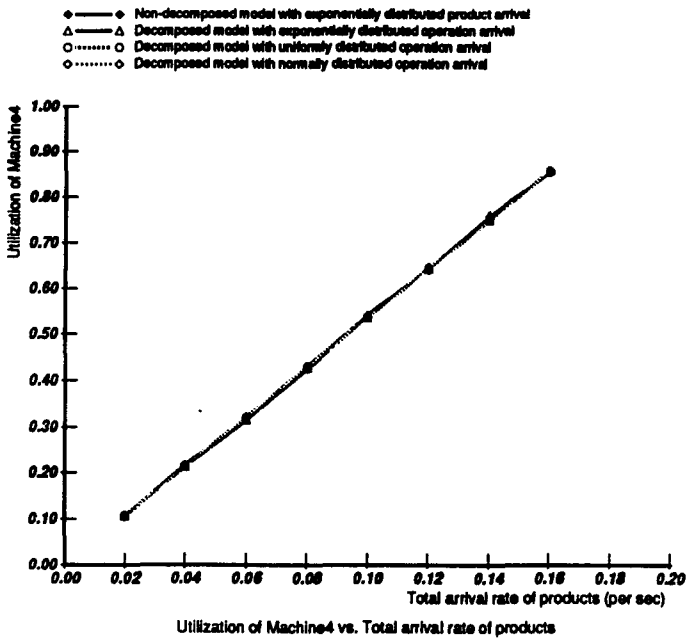


Figure 5.11: Utilization of Machines of Type 4

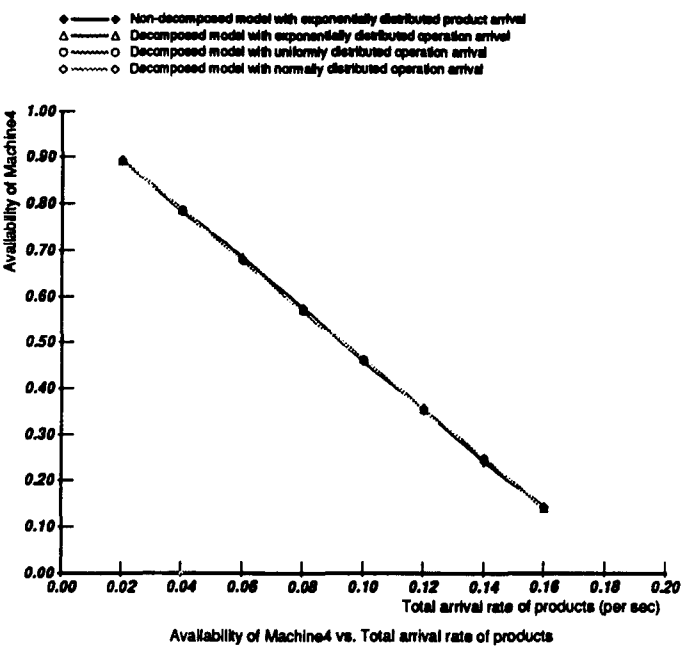


Figure 5.12: Availability of Machines of Type 4

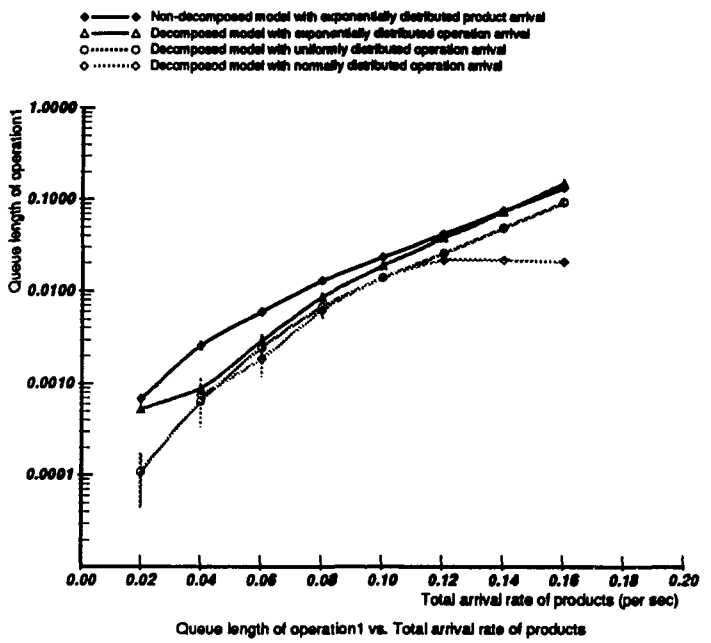


Figure 5.13: Queue Length of Operation1

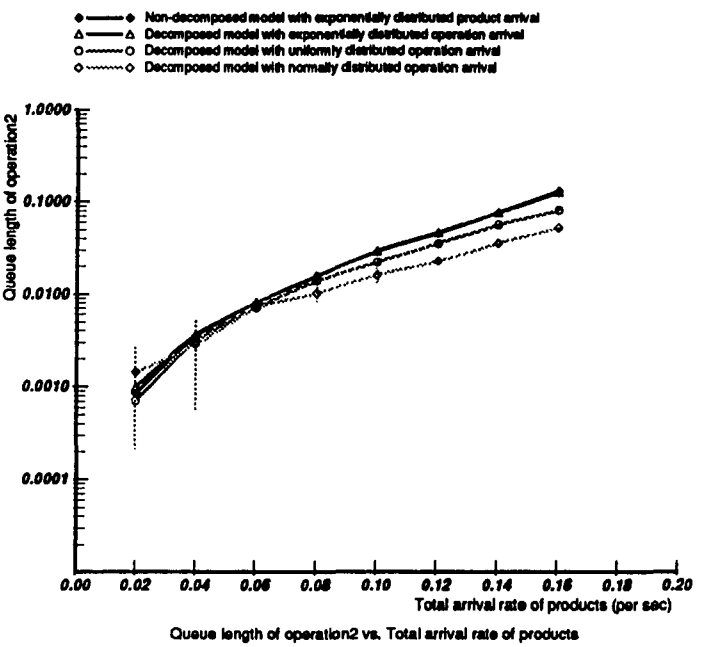


Figure 5.14: Queue Length of Operation2

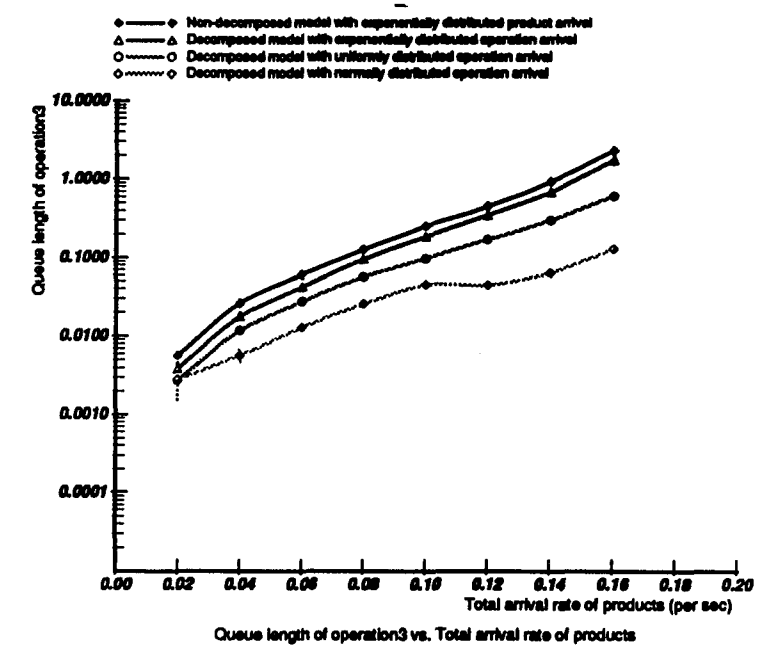


Figure 5.15: Queue Length of Operation3

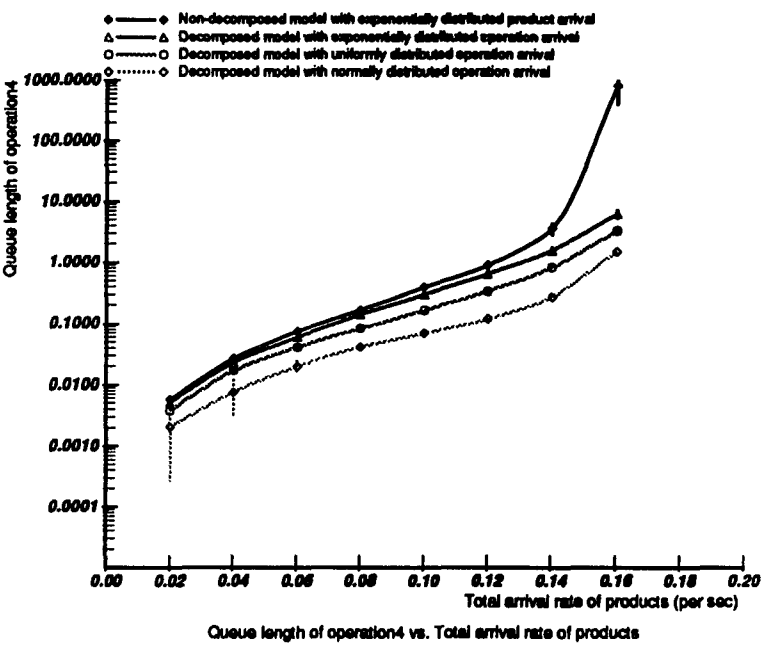


Figure 5.16: Queue Length of Operation4

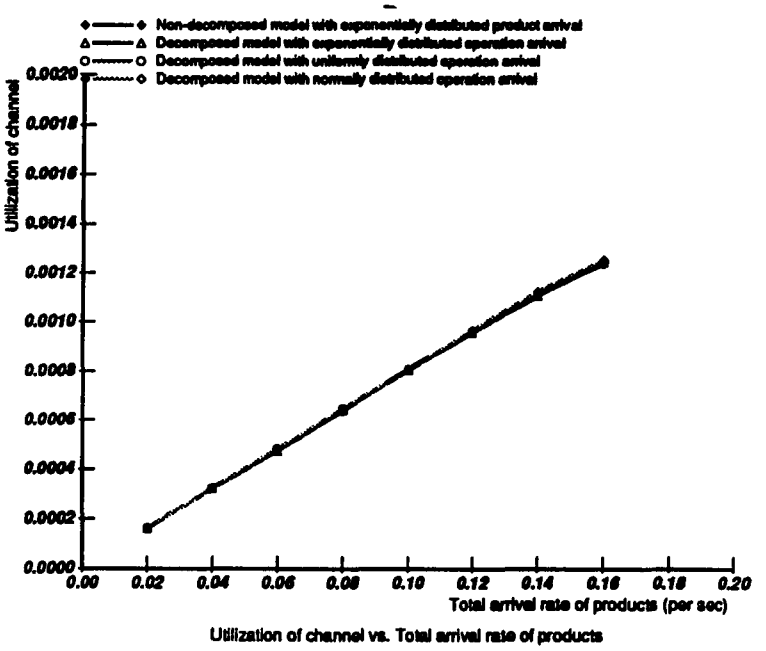


Figure 5.17: Utilization of Channel

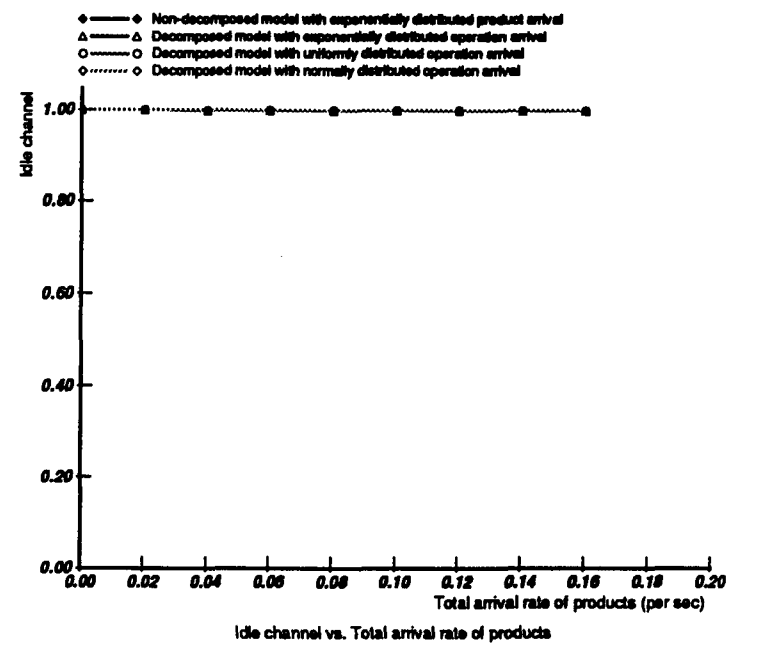


Figure 5.18: Idle Channel

CHAPTER 6

An Example System

6.1 Introduction

The previous chapters described a generic manufacturing system and discussed how to build models for each of its levels when it was viewed as hierarchical system of three different layers: the manufacturing flow layer, the control layer and the communication layer. An informal definition of a manufacturing flow network was given in chapter 3, in order to simplify the representation of manufacturing flow. The formal and more mathematical definition of MFN was derived in chapter 5. Conditions and rules for connecting elements of the MFN were discussed. Finally the technique for decomposing a MFN at a higher level was developed.

This chapter illustrates an example use of MFN and SAN models in performance evaluation of manufacturing systems, by considering a small manufacturing system example. In this system, two different parts of a product are milled and drilled separately, and then joined together to form a finished product. Milling is a basic machining process in which a surface is generated by the progressive formulation and the removal of chips of material from the workpiece. A milling machine is a rotating cutter to which workpiece is fed. The tool used in milling is known as a milling cutter. Likewise, drilling is a process used for boring holes in hard materials, usually by abrasion or repeated blows. An implement, with cutting edges or a pointed end, is used for drilling.

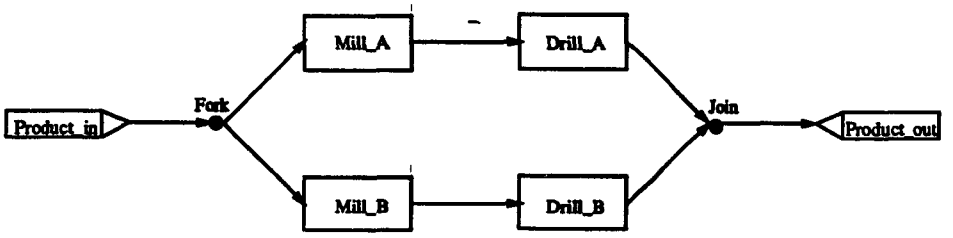


Figure 6.1: MFN of the Example Manufacturing System at the Operation Level

6.2 Description of Example Manufacturing System

Suppose we have a factory whose MFN representation at operation and machine levels are as shown in figures 6.1 and 6.2. A brief description of the functionality of this manufacturing system is discussed in succeeding part of this section. In figure 6.1, input block *Product_in* represents the arrival of raw material from which a product will be produced. The raw material gets split into two different parts. Divergence of the raw material is represented by fork node after *Product_in*. Then, both parts go through milling and drilling operations. Blocks *Mill_A* and *Mill_B* show the milling process performed on part A and part B respectively. Blocks *Drill_A* and *Drill_B* represent the drilling process done on part A and part B after they have gone through milling process. After completion of drilling, both parts fit together to form the final product. Output block *Product_out* represents finished product. *join* node before *Product_out* represents fitting of the two parts together to form finished product.

The details of these operations, at the machine level, are given in figure 6.2. Several types of machines are used, and are listed below:

1. *Carrier machines:* These machines are used for

   (a) Carrying parts to milling area where milling process can be performed,

   (b) Carrying parts to drilling area when milling process is performed, and

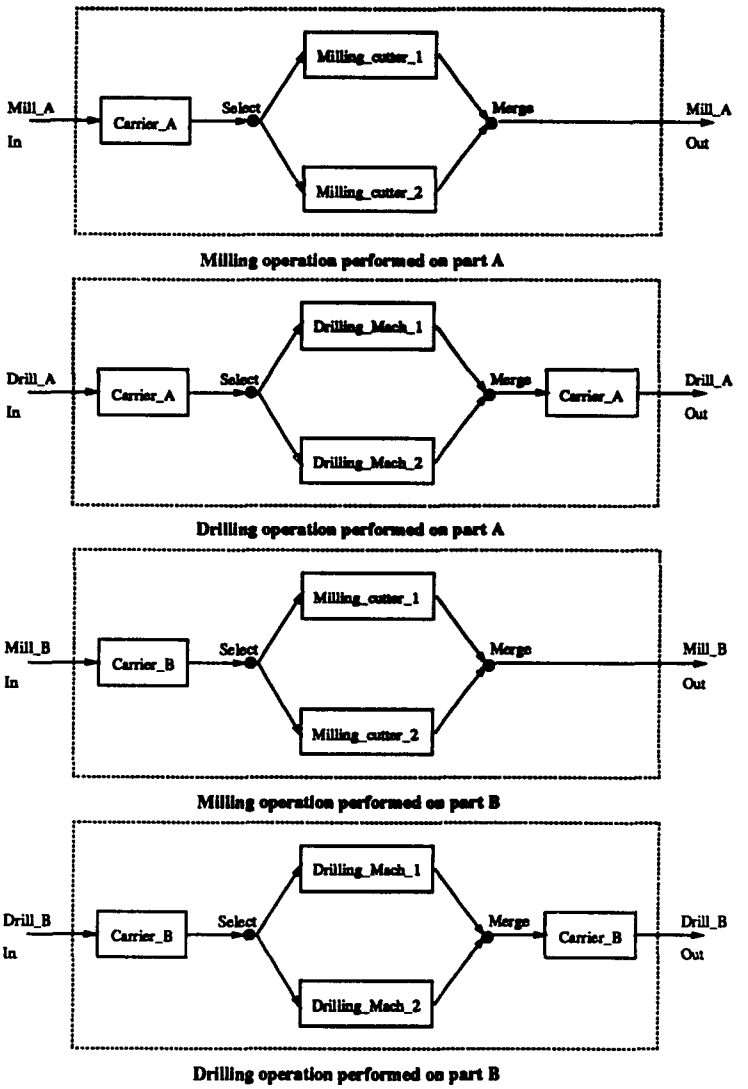


Figure 6.2: MFNs of the Example Manufacturing System at the Machine Level

(c) Carrying parts to fitting area where they can be joined together to form finished products.

Blocks *Carrier_A* and *Carrier_B* in the MFNs of figure 6.2 represent the carrier machines used in the system to perform above mentioned tasks on part A and part B of the product respectively.

2. *Milling machines:* Two milling machines are used in the system to perform milling process on the parts. Selection strategies for them can be defined at *select* nodes as shown in MFN representation of Milling operations of part A and part B in figure 6.2. Blocks *Milling_cutter_1* and *Milling_cutter_2* represent the two milling machines in the system.

3. *Drilling machines:* Similar to milling machines, two drilling machines are used to perform drilling operation on parts. *select* nodes in the MFN representations of drilling of part A and part B have scheduling policies associated with them for selection of a drilling machine. Blocks *Drilling_Mach_1* and *Drilling_Mach_2* represent the two drilling machines used in the system.

All four of these MFNs at machine level are simply represented by blocks *Mill_A*, *Drill_A*, *Mill_B* and *Drill_B* in MFN representation of the system at operation level.

6.3 Model Assumptions and System Parameters

Before evaluating the performance of the system the model assumptions and input parameters of the system are discussed.

1. **Model assumptions**

   Following are the assumptions which are made in the example models:

   (a) Product arrivals are exponentially distributed.

   (b) Operation failure is not taken into account but machine failures are accounted for. Machine failures and repairs are exponentially distributed.

   (c) Processing times for tasks performed at machines are uniformly distributed.

   (d) Drilling machines fail more often than milling machines.

2. **System parameters**

   The input parameters to this system can be listed as below:

Parameter	Value
Arrival rate of products	0.05 per hour to 0.5 per hour
Processing time required for milling	0.5 to 1 hour
Processing time required for drilling	1 to 1.5 hour
Failure rate of milling machine	0.001 per hour
Failure rate of drilling machine	0.005 per hour
Repair rate of milling machine	1 per hour
Repair rate of drilling machine	1 per hour
Time required for carrier machines to carry the material to milling area	10-15 minutes
Time required for carrier machines to carry the material from milling area to drilling area	4-6 minutes
Time required for carrier machines to carry the material from drilling area	10-15 minutes

6.4 Performance Variables

In this study we consider the utilization of each machine, the availability of each machine, and average time to produce the product. These performance variables can be determined by defining a reward structure for the constructed SAN models as discussed in section 5.6. Using conversion algorithms of chapter 3, SAN models for the manufacturing flow layer of this system were constructed. The simple machine model and its failure model as discussed in section 4.2 were used at the control layer. Finally, the CSMA/CD network model as shown in figure 4.4 was used at the network layer. The specific SAN models used can be found in Appendix D.

6.5 Simulation Results of Performance Variables

The system was then evaluated for the desired performance variables using steady-state simulation with a confidence level of 0.95. The plots for these performance variables are given in figures. 6.3 to 6.10. Looking at the results as well as curves the following points can be concluded:

1. As the load increases, machine utilization increases while machine availability decreases. This relation is almost linear for stable loads. This is a quite obvious result.

2. In this particular manufacturing system, time to perform tasks at the machine is quite large compared to the time required to transmit or receive a message on the computer network. Hence, machine utilization is the bottleneck, rather than the computer network.

3. As the load increases, the average time to produce a product increases, but the response time doesn't linearly vary with the variation in loads. This result is valid because as the load increases, queue length of products at machines increases and so does the waiting times of products at machines.

4. For the chosen parameter values, the bottleneck is the drilling machines, which are highly overutilized, compared to the carrier and milling machines.

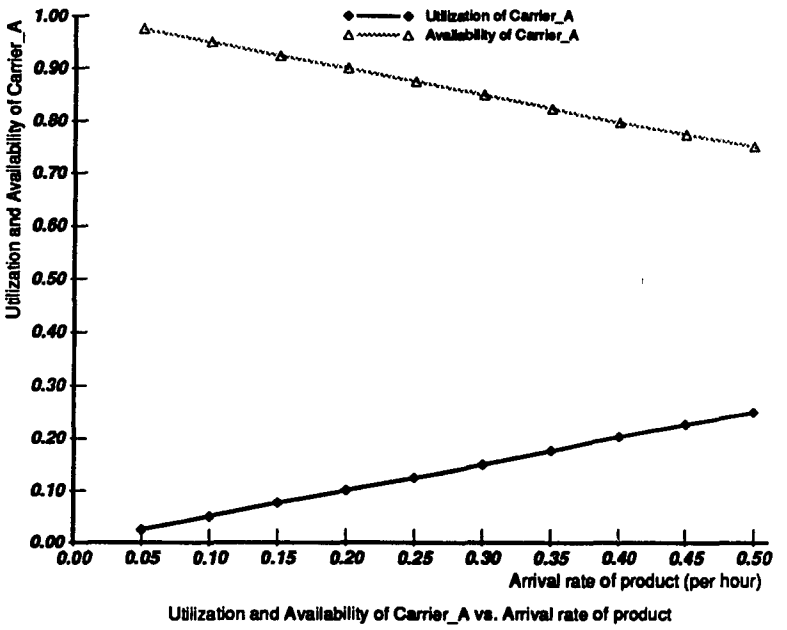


Figure 6.3: Utilization and Availability of Carrier_A

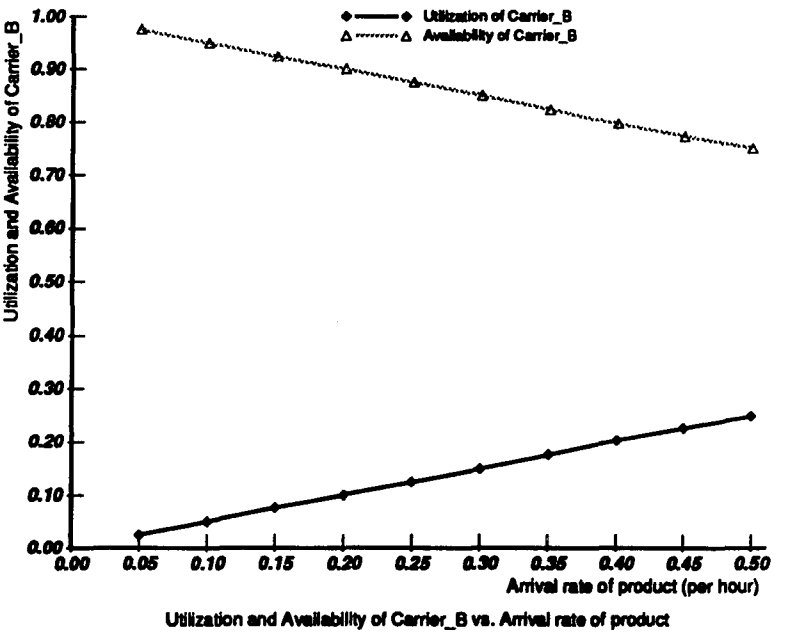


Figure 6.4: Utilization and Availability of Carrier_B

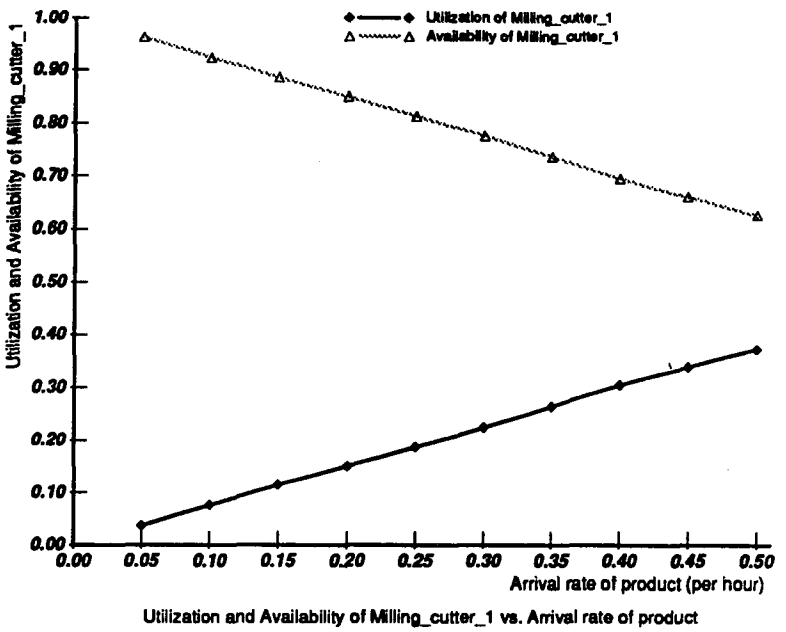


Figure 6.5: Utilization and Availability of Milling_cutter_1

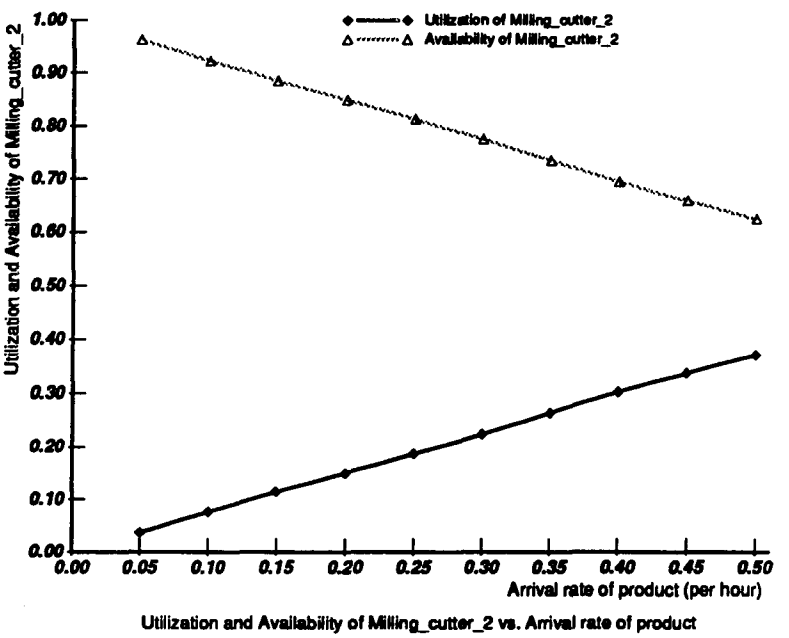


Figure 6.6: Utilization and Availability of Milling_cutter_2

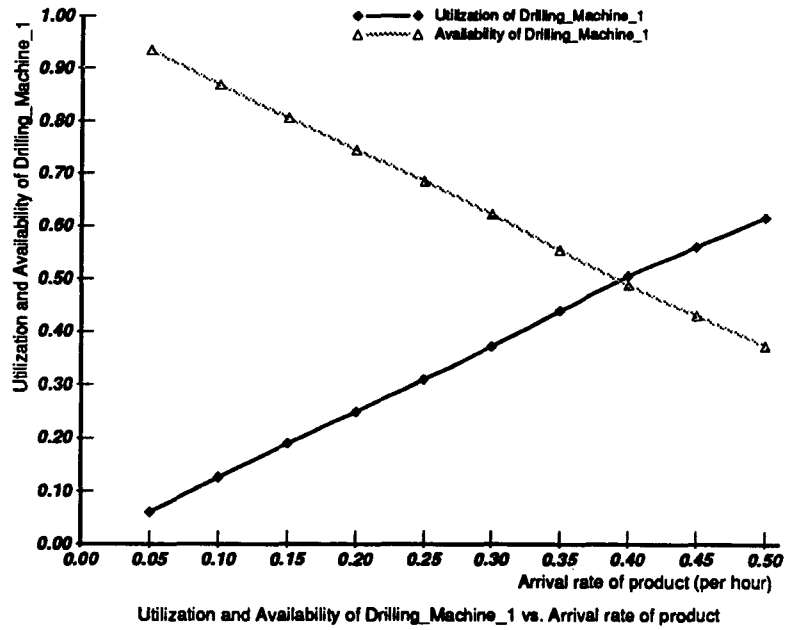


Figure 6.7: Utilization and Availability of Drilling_Mach_1

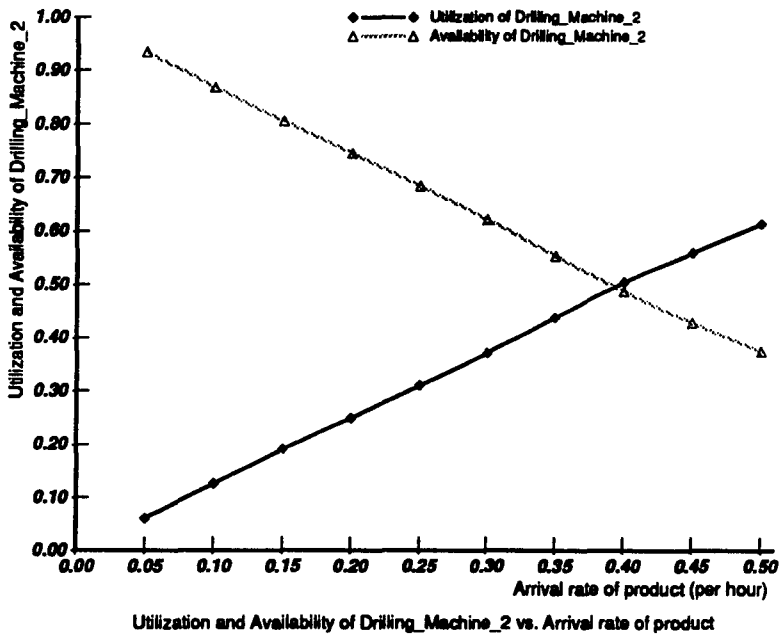


Figure 6.8: Utilization and Availability of Drilling_Mach_2

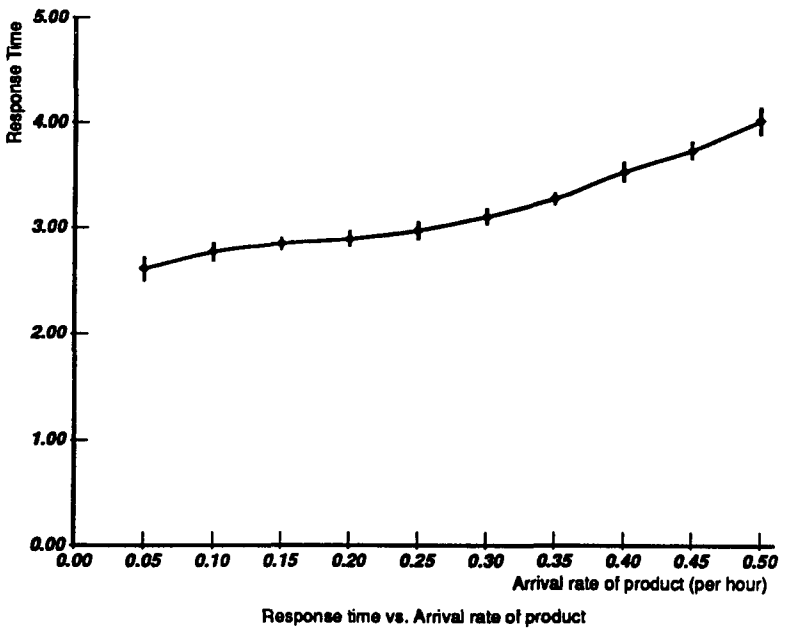


Figure 6.9: Response Time of Product

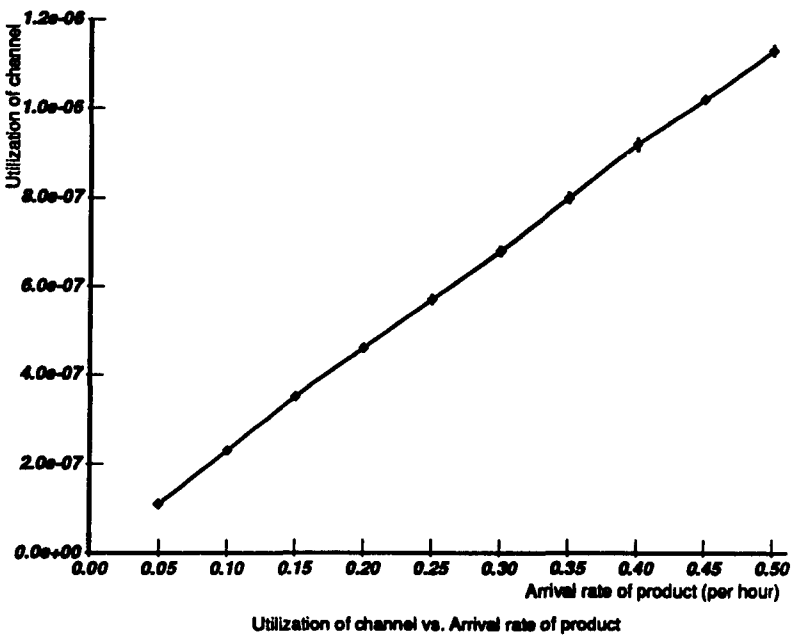


Figure 6.10: Utilization of Channel

CHAPTER 7

Conclusions and Further Research

The main objectives of this research were:

1. Build hierarchical models of manufacturing systems that realistically represent flow, control and information transfer aspects of operation in a single model.
2. Develop a high-level representation of manufacturing flow and develop conversions of this representation to SANs.
3. Investigate a decomposition technique for models to increase the efficiency of their solution and make their representation easier.

To accomplish these objectives, the following was done:

- Developed a new graphical representation, the manufacturing flow network for representing manufacturing flow within manufacturing systems.
- Discussed the conversion of MFN into SAN models.
- Developed the hierarchical models of manufacturing systems for representing the manufacturing flow, control and communication activities in the systems.
- Developed a decomposition technique on models to increase the efficiency of their solution, and studied its accuracy using an example.
- Finally, conducted a performance evaluation of a realistic, but small example manufacturing system.

The first two and the fourth accomplishments were challenging. Much abstract thinking and new mathematical proofs made those things possible. Set theory helped in defining MFN and other mathematics related to MFN. The last two objectives were the application of the theory which was developed. They gave insight into how well the proved theory about the MFN works. The decomposition technique helps us in conducting a performance evaluation of a manufacturing system where product flow is very complicated with many products produced in the system.

7.1 Areas of Further Research

Modeling, in general, is a field which is playing an important role in performance evaluation. The manufacturing flow network formalism described in this thesis is in its initial stage. A new software tool could be developed which would allow to represent manufacturing system at this level, in order to hide details about SAN models. It can be imagined as taking the MFN models as input with specified system parameters and producing values and curves of the desired performance variables as output.

Another area of interest would be to try to determine distribution of operation arrival for specific conditions when a MFN with exponential product arrival is decomposed at product level into operations. This would require a significant amount of mathematical work and thorough understanding of queueing networks, stochastic activity networks, and mathematical theory behind them.

APPENDIX A

SAN Models of Example MFNs of Figures 3.5 and 3.6

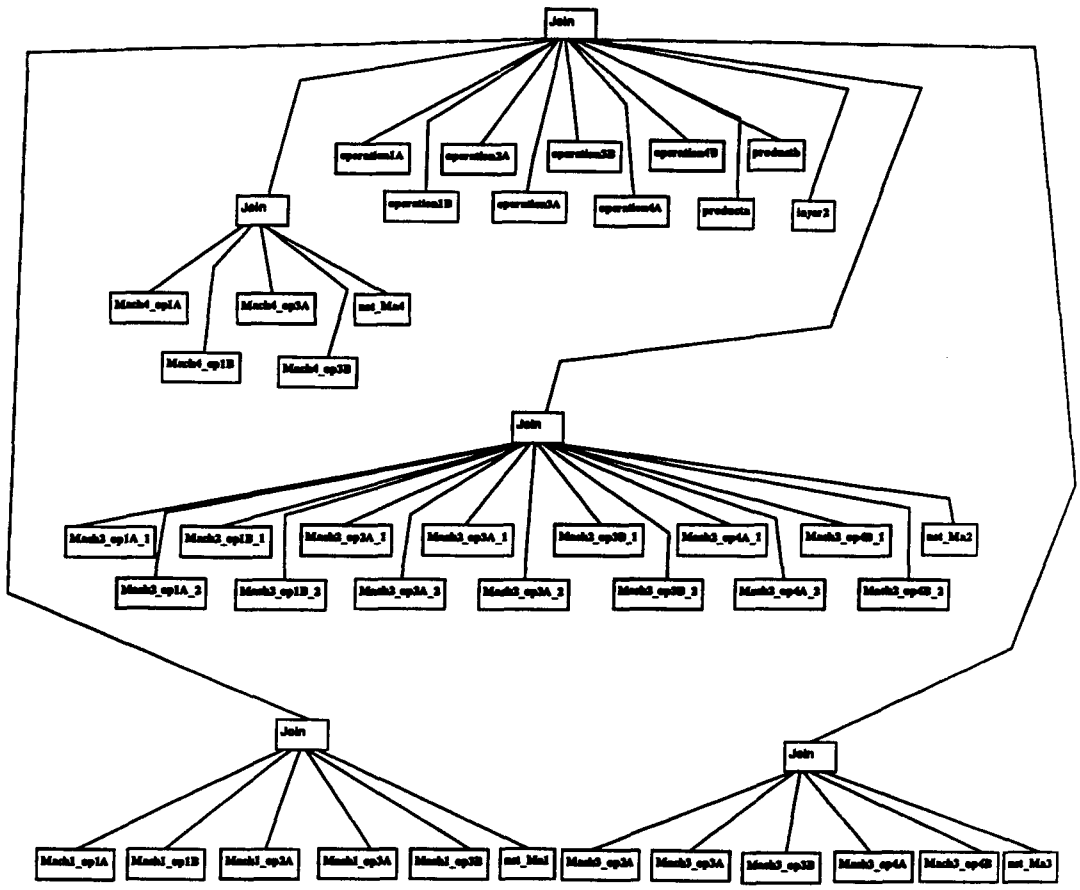


Table A.1: Composite Model of Non-decomposed MFNs Shown in Figures 3.5 and 3.6

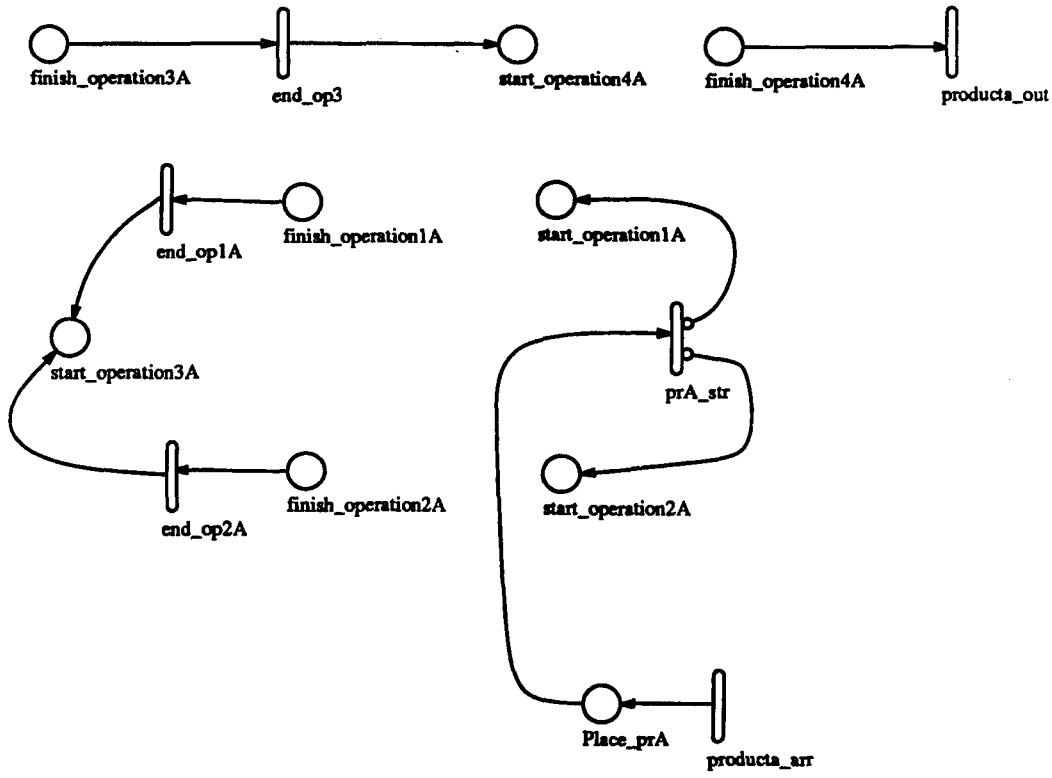


Table A.2: SAN Model Showing Manufacturing Flow of ProductA

Table A.3: Activity Time Distributions

Activity	Distribution
end_op1A	USAN_expon(10000000)
end_op2A	USAN_expon(10000000)
end_op3	USAN_expon(10000000)
prA_str	USAN_expon(10000000)
producta_arr	USAN_expon(varied)
producta_out	USAN_expon(10000000)

Table A.4: Case Probabilities for Activities

Activity	Case	Probability
prA_str	1	0.5
	2	0.5

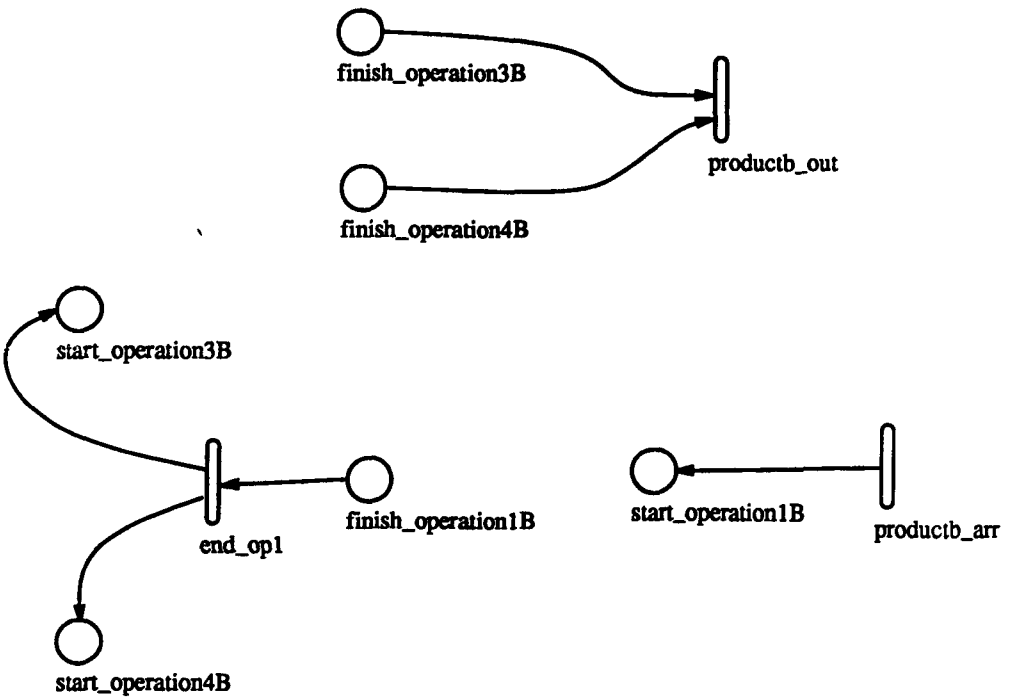


Table A.5: SAN Model of Manufacturing Flow of ProductB

Table A.6: Activity Time Distributions

Activity	Distribution
end_op1	USAN_exponential(10000000)
productb_arr	USAN_exponential(varied)
productb_out	USAN_exponential(10000000)

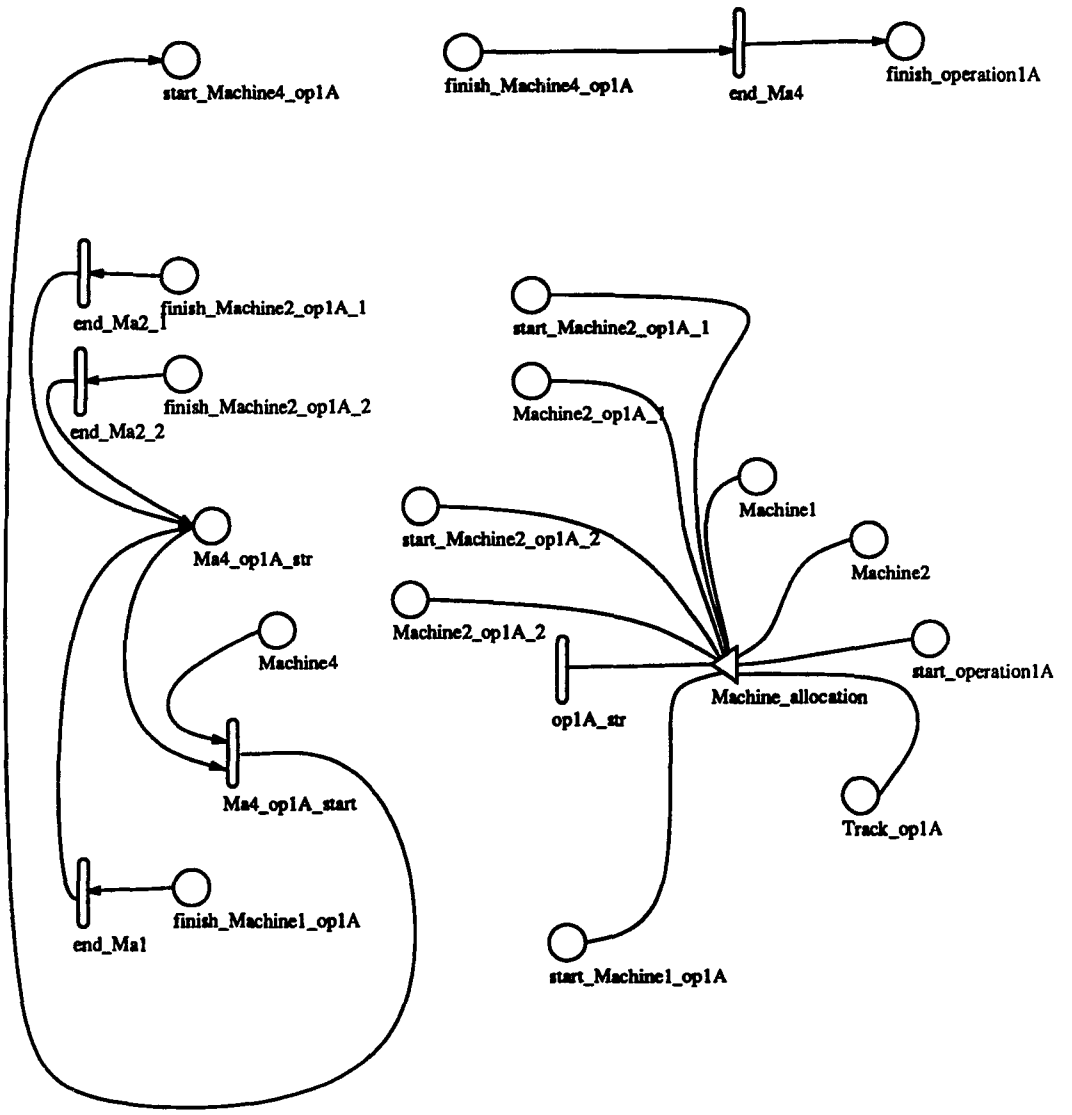


Table A.7: SAN Model Showing Manufacturing Flow of Operation1 Performed on ProductA

Table A.8: Activity Time Distributions

Activity	Distribution
Ma4_op1A_start	USAN_expon(10000000)
end_Ma1	USAN_expon(10000000)
end_Ma2_1	USAN_expon(10000000)
end_Ma2_2	USAN_expon(10000000)
end_Ma4	USAN_expon(10000000)
op1A_str	USAN_expon(10000000)

Table A.9: Input Gate Predicates and Functions

Gate	Enabling Predicate	Function
Machine_allocation	*MARK(start_operation1A) >* *0 && (MARK(Machine1) >* *0 || MARK(Machine2) > 0)*	*MARK(start_operation1A)– = 1;* *switch(MARK(Track_op1A)){* *case (1) : if(MARK(Machine2) > 0){* *MARK(Machine2)– = 1;* *MARK(Track_op1A) = 2;* *if(MARK(Machine2_op1A_1) == 0){* *MARK(Machine2_op1A_1) = 1;* *MARK(start_Machine2_op1A_1) = 1;* *}else{* *MARK(Machine2_op1A_2) = 1;* *MARK(start_Machine2_op1A_2) = 1;* *}* *}else{* *MARK(Machine1)– = 1;* *MARK(Track_op1A) = 1;* *MARK(start_Machine1_op1A) = 1;* *}* *break;* *case (2) : if(MARK(Machine1) > 0){* *MARK(Machine1)– = 1;* *MARK(Track_op1A) = 1;* *MARK(start_Machine1_op1A) = 1;* *}else{* *if(MARK(Machine2_op1A_1) == 0){* *MARK(Machine2_op1A_1) = 1;* *MARK(start_Machine2_op1A_1) = 1;* *}else{* *MARK(Machine2_op1A_1) = 1;* *MARK(start_Machine2_op1A_2) = 1;* *}* *MARK(Track_op1A) = 2;* *MARK(Machine2)– = 1;* *}* *break;* *}*

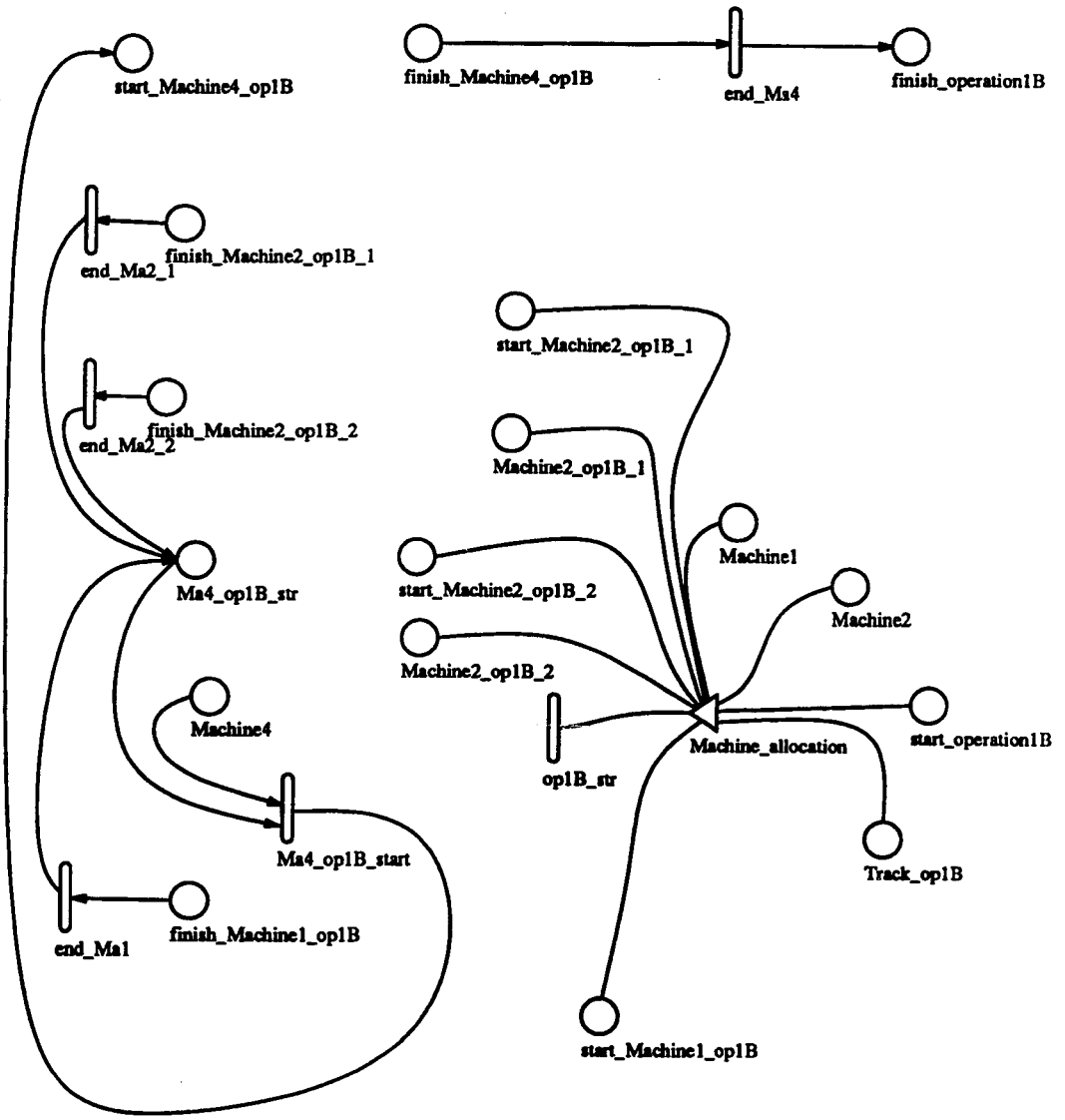


Table A.10: SAN Model Showing Manufacturing Flow of Operation1 Performed on ProductB

Table A.11: Activity Time Distributions

Activity	Distribution
Ma4_op1B_start	USAN_expon(10000000)
end_Ma1	USAN_expon(10000000)
end_Ma2_1	USAN_expon(10000000)
end_Ma2_2	USAN_expon(10000000)
end_Ma4	USAN_expon(10000000)
op1B_str	USAN_expon(10000000)

Table A.12: Input Gate Predicates and Functions

Gate	Enabling Predicate	Function
Machine_allocation	*MARK(start_operation1B)* > 0 && (*MARK(Machine1)* > 0 || *MARK(Machine2)* > 0)	*MARK(start_operation1B)*– = 1; switch(*MARK(Track_op1B)*){ case (1) : if(*MARK(Machine2)* > 0){ *MARK(Machine2)*– = 1; *MARK(Track_op1B)* = 2; if(*MARK(Machine2_op1B_1)* == 0){ *MARK(Machine2_op1B_1)* = 1; *MARK(start_Machine2_op1B_1)* = 1; }else{ *MARK(Machine2_op1B_2)* = 1; *MARK(start_Machine2_op1B_2)* = 1; } }else{ *MARK(Machine1)*– = 1; *MARK(Track_op1B)* = 1; *MARK(start_Machine1_op1B)* = 1; } break; case (2) : if(*MARK(Machine1)* > 0){ *MARK(Machine1)*– = 1; *MARK(Track_op1B)* = 1; *MARK(start_Machine1_op1B)* = 1; }else{ if(*MARK(Machine2_op1B_1)* == 0){ *MARK(Machine2_op1B_1)* = 1; *MARK(start_Machine2_op1B_1)* = 1; }else{ *MARK(Machine2_op1B_1)* = 1; *MARK(start_Machine2_op1B_2)* = 1; } *MARK(Track_op1B)* = 2; *MARK(Machine2)*– = 1; } break; }

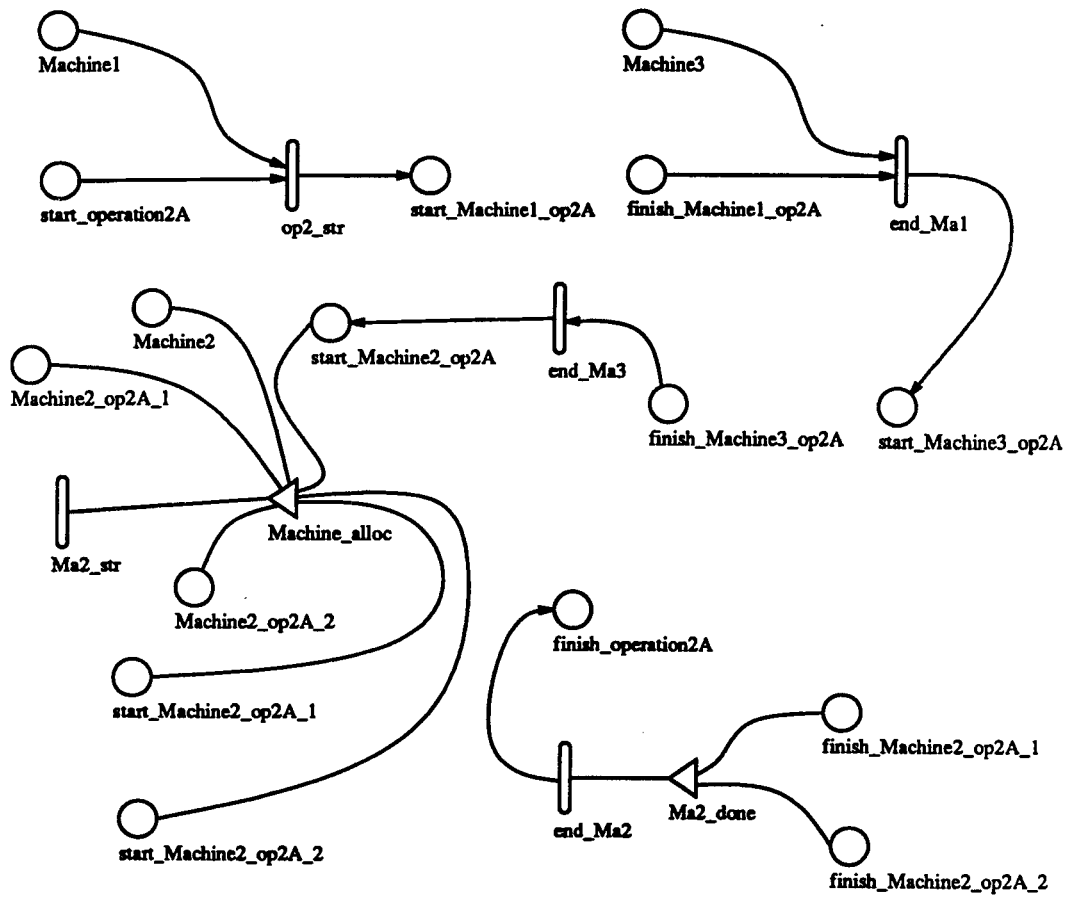


Table A.13: SAN Model Showing Manufacturing Flow of Operation2 Performed on ProductA

Table A.14: Activity Time Distributions

Activity	Distribution
Ma2_str	USAN_expon(10000000)
end_Ma1	USAN_expon(10000000)
end_Ma2	USAN_expon(10000000)
end_Ma3	USAN_expon(10000000)
op2_str	USAN_expon(10000000)

Table A.15: Input Gate Predicates and Functions

Gate	Enabling Predicate	Function
Machine_alloc	MARK(start_Machine2_op2A) > 0 && MARK(Machine2) > 0	if(MARK(Machine2_op2A_1) == 0){ MARK(Machine2_op2A_1) = 1; MARK(start_Machine2_op2A_1) = 1; }else { MARK(Machine2_op2A_2) = 1; MARK(start_Machine2_op2A_2) = 1; } MARK(Machine2)- = 1; MARK(start_Machine2_op2A)- = 1;
Ma2_done	MARK(finish_Machine2_op2A_1) > 0 || MARK(finish_Machine2_op2A_2) > 0	if(MARK(finish_Machine2_op2A_1) > 0) MARK(finish_Machine2_op2A_1)- = 1; else MARK(finish_Machine2_op2A_2)- = 1;

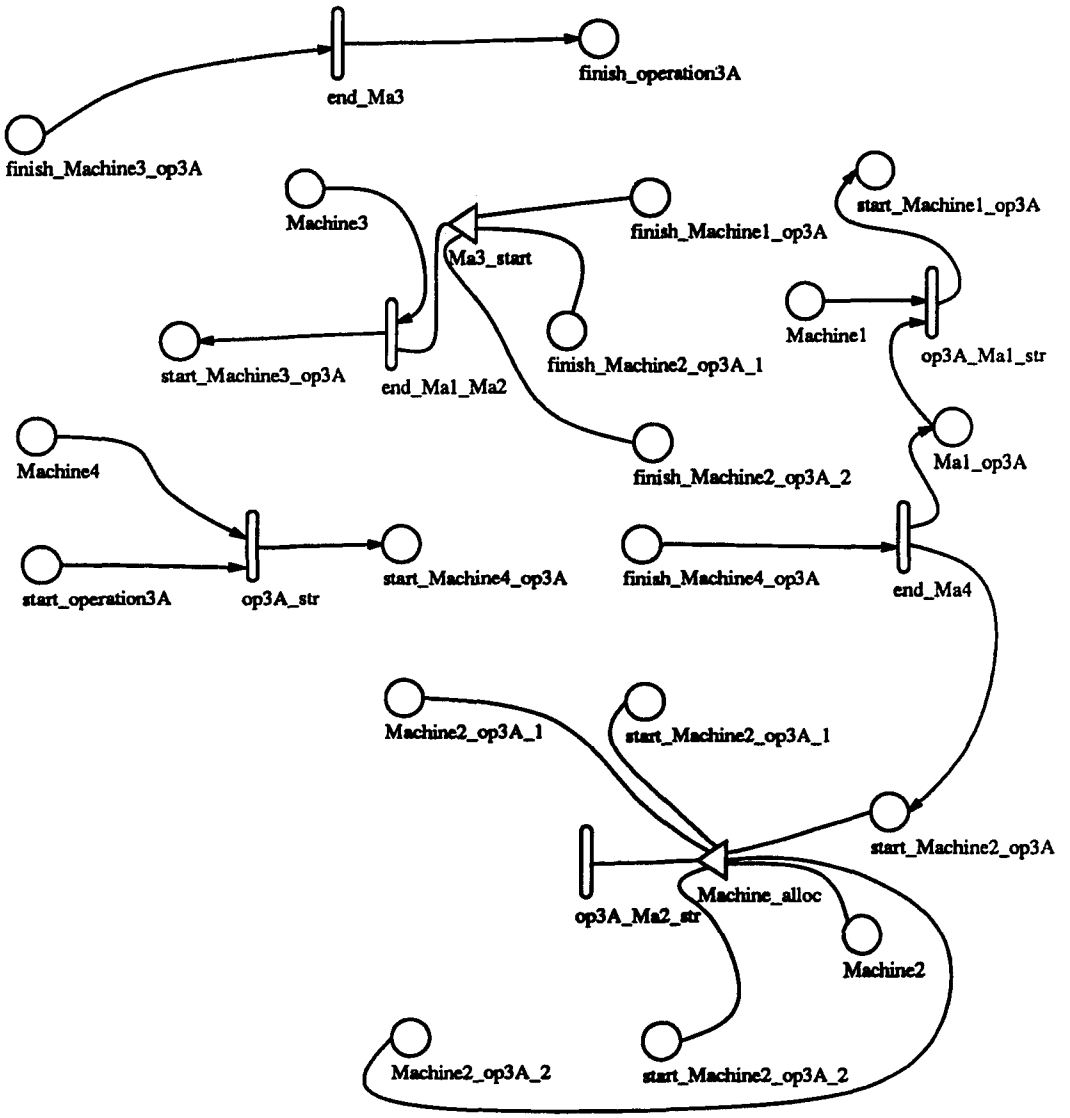


Table A.16: SAN Model Showing Manufacturing Flow of Operation3 Performed on ProductA

Table A.17: Activity Time Distributions

Activity	Distribution
end_Ma1_Ma2	USAN_exponential(10000000)
end_Ma3	USAN_exponential(10000000)
end_Ma4	USAN_exponential(10000000)
op3A_Ma1_str	USAN_exponential(10000000)
op3A_Ma2_str	USAN_exponential(10000000)
op3A_str	USAN_exponential(10000000)

Table A.18: Input Gate Predicates and Functions

Gate	Enabling Predicate	Function
Machine_alloc	*MARK(Machine2) >* *0 && MARK(start_Machine2_op3A) > 0*	*if(MARK(Machine2_op3A_1) == 0){* *MARK(Machine2_op3A_1) = 1;* *MARK(start_Machine2_op3A_1) = 1;* *}else{* *MARK(Machine2_op3A_2) = 1;* *MARK(start_Machine2_op3A_2) = 1;* *}* *MARK(start_Machine2_op3A)- = 1;* *MARK(Machine2)- = 1;*
Ma3_start	*MARK(finish_Machine1_op3A) >* *0 && (MARK(finish_Machine2_op3A_1) >* *0 || MARK(finish_Machine2_op3A_2) > 0)*	*MARK(finish_Machine1_op3A)- = 1;* *if(MARK(finish_Machine2_op3A_1) >* *0) MARK(finish_Machine2_op3A_1)- = 1;* *if(MARK(finish_Machine2_op3A_2) >* *0) MARK(finish_Machine2_op3A_2)- = 1;*

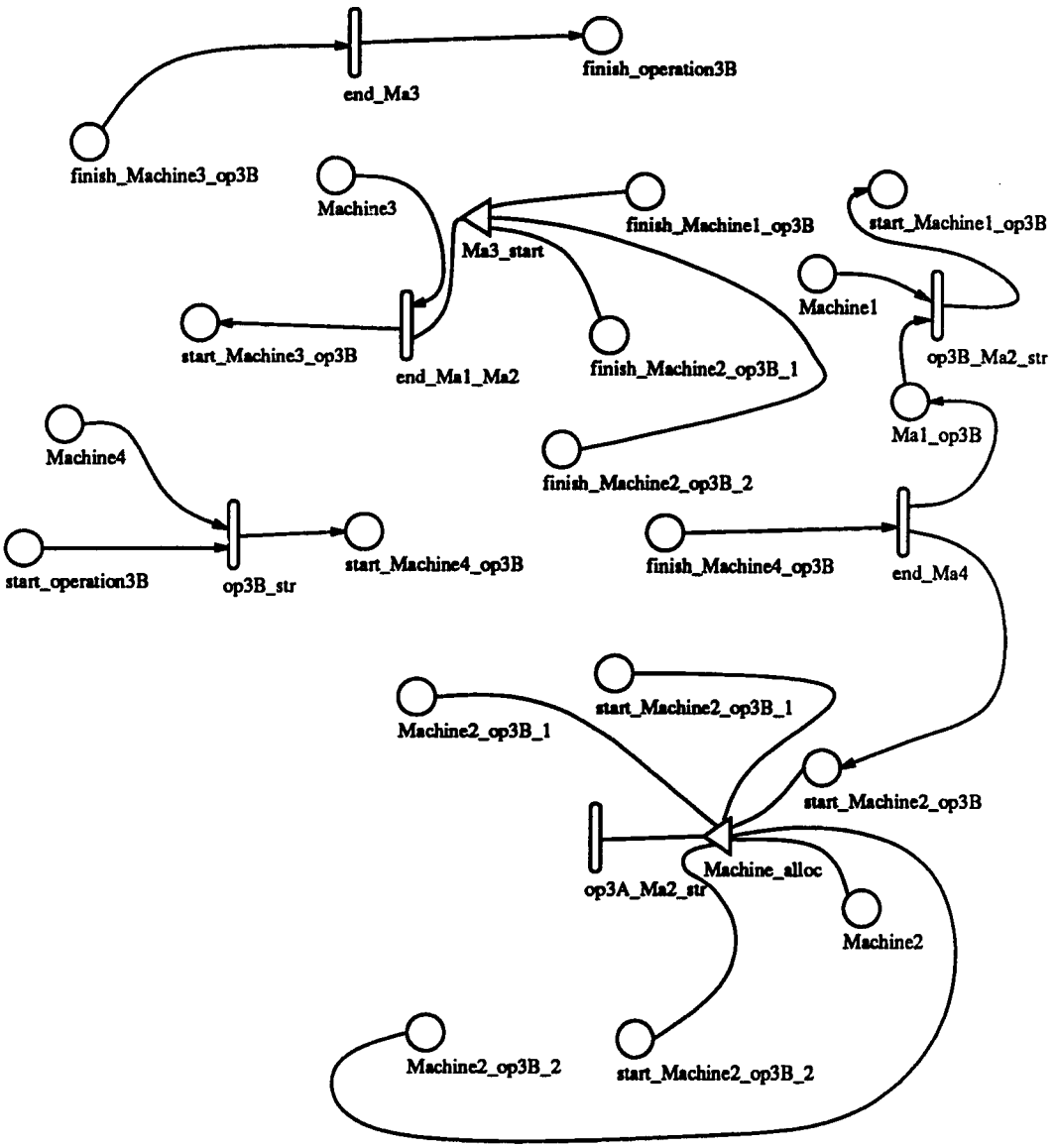


Table A.19: SAN Model Showing Manufacturing Flow of Operation3 Performed on ProductB

Table A.20: Activity Time Distributions

Activity	Distribution
end_Ma1_Ma2	USAN_expon(10000000)
end_Ma3	USAN_expon(10000000)
end_Ma4	USAN_expon(10000000)
op3A_Ma2_str	USAN_expon(10000000)
op3B_Ma2_str	USAN_expon(10000000)
op3B_str	USAN_expon(10000000)

Table A.21: Input Gate Predicates and Functions

Gate	Enabling Predicate	Function
Machine_alloc	*MARK(Machine2) >* *0 && MARK(start_Machine2_op3B) > 0*	*if(MARK(Machine2_op3B_1) == 0){* *MARK(Machine2_op3B_1) = 1;* *MARK(start_Machine2_op3B_1) = 1;* *}else{* *MARK(Machine2_op3B_2) = 1;* *MARK(start_Machine2_op3B_2) = 1;* *}* *MARK(start_Machine2_op3B)- = 1;* *MARK(Machine2)- = 1;*
Ma3_start	*MARK(finish_Machine1_op3B) >* *0 && (MARK(finish_Machine2_op3B_1) >* *0 || MARK(finish_Machine2_op3B_2) > 0)*	*MARK(finish_Machine1_op3B)- = 1;* *if(MARK(finish_Machine2_op3B_1) >* *0) MARK(finish_Machine2_op3B_1)- = 1;* *if(MARK(finish_Machine2_op3B_2) >* *0) MARK(finish_Machine2_op3B_2)- = 1;*

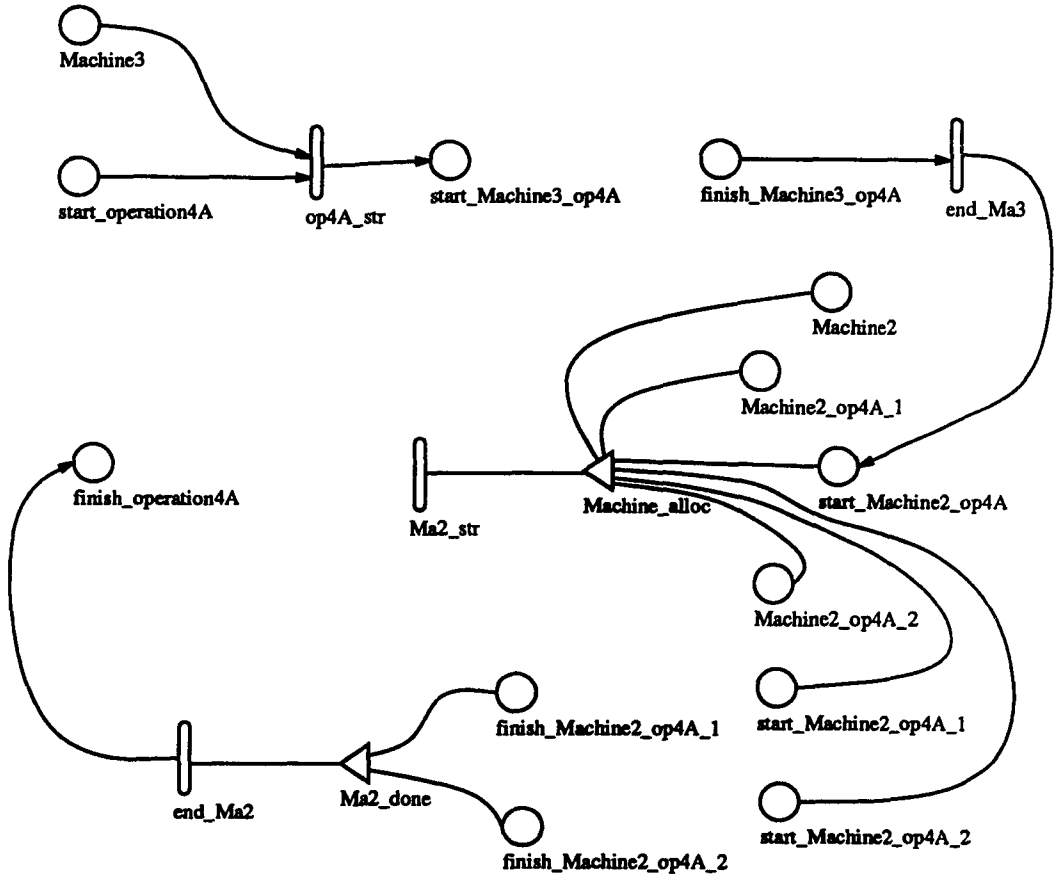


Table A.22: SAN Model Showing Manufacturing Flow of Operation4 Performed on ProductA

Table A.23: Activity Time Distributions

Activity	Distribution
Ma2_str	USAN_expon(10000000)
end_Ma2	USAN_expon(10000000)
end_Ma3	USAN_expon(10000000)
op4A_str	USAN_expon(10000000)

Table A.24: Input Gate Predicates and Functions

Gate	Enabling Predicate	Function
Machine_alloc	*MARK(start_Machine2_op4A) > 0 && MARK(Machine2) > 0*	*if(MARK(Machine2_op4A_1) == 0){* *MARK(Machine2_op4A_1) = 1;* *MARK(start_Machine2_op4A_1) = 1;* *}else{* *MARK(Machine2_op4A_2) = 1;* *MARK(start_Machine2_op4A_2) = 1;* *}* *MARK(Machine2)- = 1;* *MARK(start_Machine2_op4A)- = 1;*
Ma2_done	*MARK(finish_Machine2_op4A_1) > 0 || MARK(finish_Machine2_op4A_2) > 0*	*if(MARK(finish_Machine2_op4A_1) > 0) MARK(finish_Machine2_op4A_1)- = 1;* *else MARK(finish_Machine2_op4A_2)- = 1;*

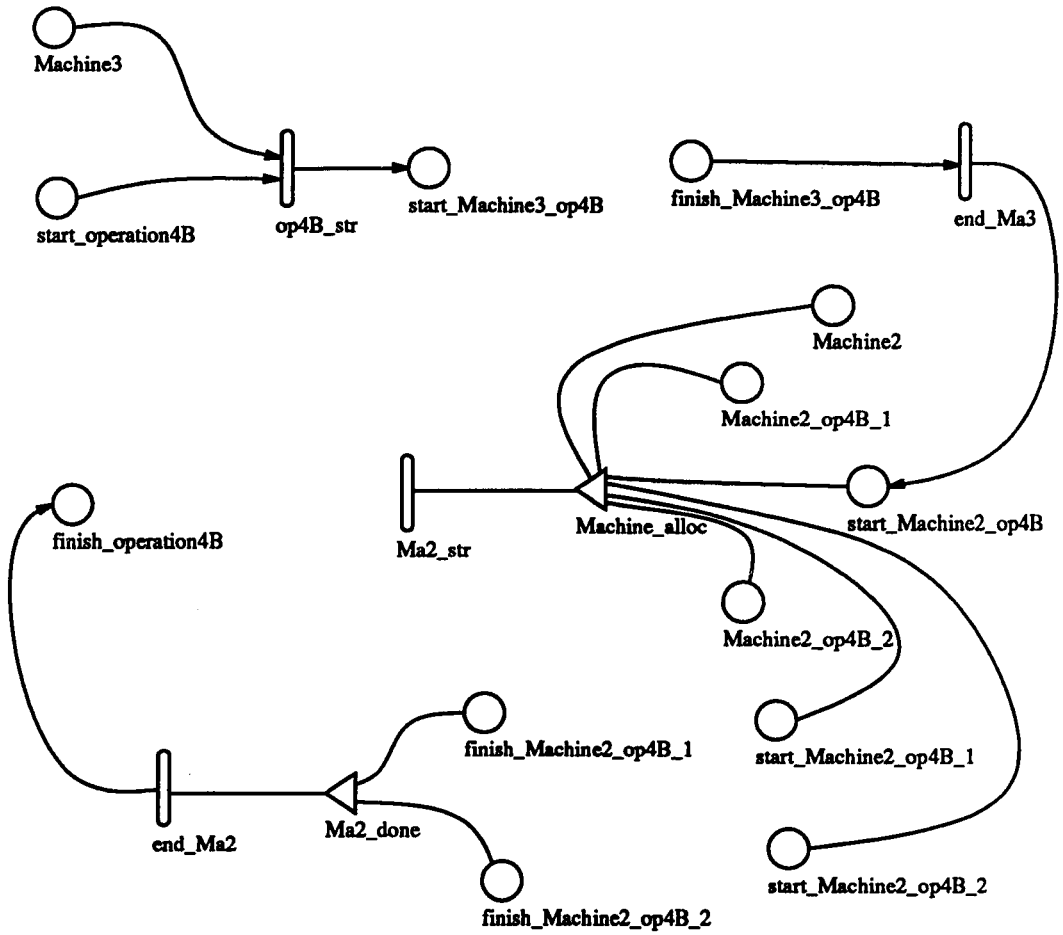


Table A.25: SAN Model Showing Manufacturing Flow of Operation4 Performed on ProductB

Table A.26: Activity Time Distributions

Activity	Distribution
Ma2_str	USAN_expon(10000000)
end_Ma2	USAN_expon(10000000)
end_Ma3	USAN_expon(10000000)
op4B_str	USAN_expon(10000000)

Table A.27: Input Gate Predicates and Functions

Gate	Enabling Predicate	Function
Machine_alloc	MARK(start_Machine2_op4B) > 0 && MARK(Machine2) > 0	if(MARK(Machine2_op4B_1) == 0){ MARK(Machine2_op4B_1) = 1; MARK(start_Machine2_op4B_1) = 1; }else{ MARK(Machine2_op4B_2) = 1; MARK(start_Machine2_op4B_2) = 1; } MARK(Machine2)-- = 1; MARK(start_Machine2_op4B)-- = 1;
Me2_done	MARK(finish_Machine2_op4B_1) > 0 || MARK(finish_Machine2_op4B_2) > 0	if(MARK(finish_Machine2_op4B_1) > 0) MARK(finish_Machine2_op4B_1)-- = 1; else MARK(finish_Machine2_op4B_2)-- = 1;

APPENDIX B

SAN Models of Decomposed Example MFNs

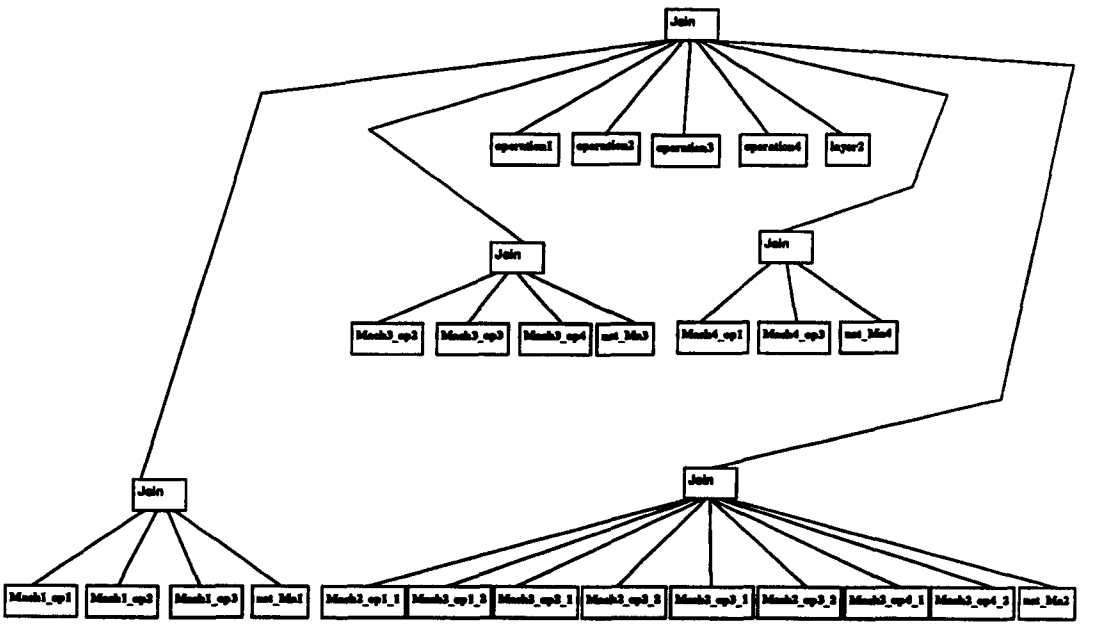


Table B.1: Composite Model of Decomposed MFNs

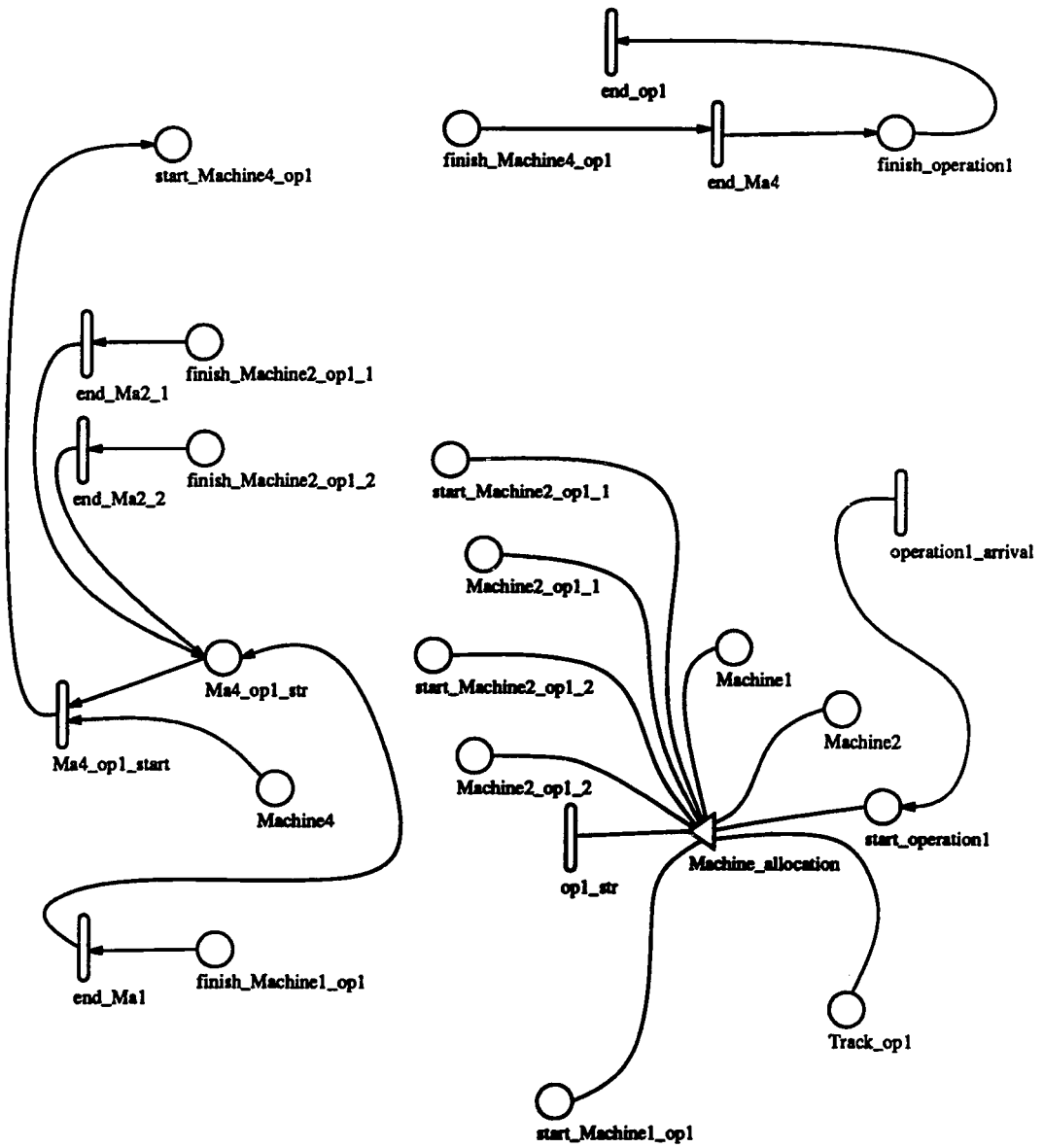


Table B.2: SAN Model Showing Manufacturing Flow of Operation1

Table B.3: Activity Time Distributions

Activity	Distribution
Mc1_op1_start	USAN_expon(10000000)
end_Mc1	USAN_expon(10000000)
end_Mc2_1	USAN_expon(10000000)
end_Mc2_2	USAN_expon(10000000)
end_Mc4	USAN_expon(10000000)
end_op1	USAN_deter(.00000000001)
op1_str	USAN_expon(10000000)
operation1_arrival	USAN_expon(varied)

Table B.4: Input Gate Predicates and Functions

Gate	Enabling Predicate	Function
Machine_allocation	MARK(start_operation1) > 0 && (MARK(Machine1) > 0 || MARK(Machine2) > 0)	MARK(start_operation1)-- = 1; switch(MARK(Track_op1)){ case (1) : if(MARK(Machine2) > 0){ MARK(Machine2)-- = 1; MARK(Track_op1) = 2; if(MARK(Machine2_op1_1) == 0){ MARK(Machine2_op1_1) = 1; MARK(start_Machine2_op1_1) = 1; }else{ MARK(Machine2_op1_2) = 1; MARK(start_Machine2_op1_2) = 1; } }else{ MARK(Machine1)-- = 1; MARK(Track_op1) = 1; MARK(start_Machine1_op1) = 1; } break; case (2) : if(MARK(Machine1) > 0){ MARK(Machine1)-- = 1; MARK(Track_op1) = 1; MARK(start_Machine1_op1) = 1; }else{ if(MARK(Machine2_op1_1) == 0){ MARK(Machine2_op1_1) = 1; MARK(start_Machine2_op1_1) = 1; }else{ MARK(Machine2_op1_1) = 1; MARK(start_Machine2_op1_2) = 1; } MARK(Track_op1) = 2; MARK(Machine2)-- = 1; } break; }

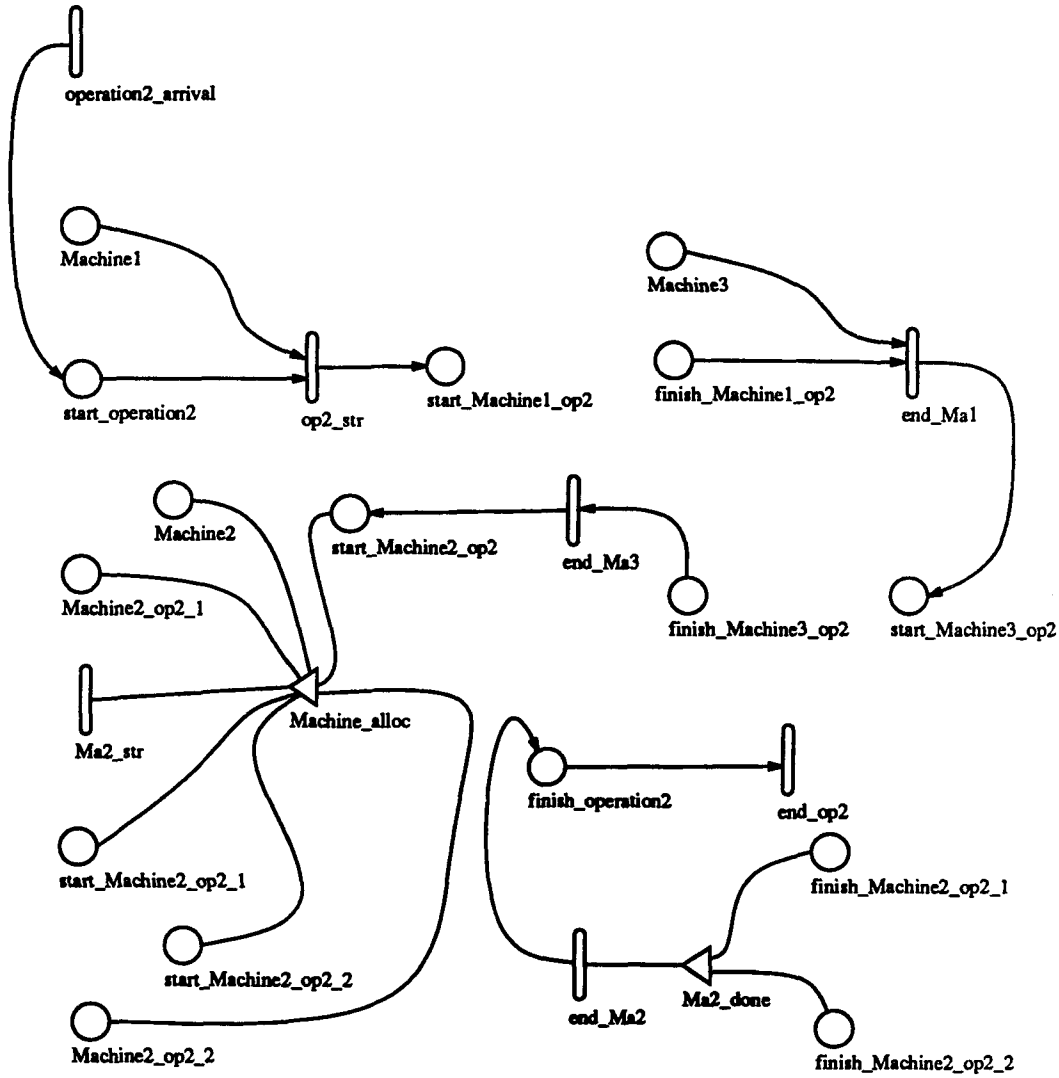


Table B.5: SAN Model Showing Manufacturing Flow of Operation2

Table B.6: Activity Time Distributions

Activity	Distribution
Ma2_str	USAN_expon(10000000)
end_Ma1	USAN_expon(10000000)
end_Ma2	USAN_expon(10000000)
end_Ma3	USAN_expon(10000000)
end_op2	USAN_expon(10000000)
op2_str	USAN_expon(10000000)
operation2_arrival	USAN_expon(varied)

Table B.7: Input Gate Predicates and Functions

Gate	Enabling Predicate	Function
Machine_alloc	*MARK(start_Machine2_op2)* > 0 && *MARK(Machine2)* > 0	if(*MARK(Machine2_op2_1)* == 0){ *MARK(Machine2_op2_1)* = 1; *MARK(start_Machine2_op2_1)* = 1; }else { *MARK(Machine2_op2_2)* = 1; *MARK(start_Machine2_op2_2)* = 1; } *MARK(Machine2)*– = 1; *MARK(start_Machine2_op2)*– = 1;
M2_done	*MARK(finish_Machine2_op2_1)* > 0 || *MARK(finish_Machine2_op2_2)* > 0	if(*MARK(finish_Machine2_op2_1)* > 0) *MARK(finish_Machine2_op2_1)*– = 1; else *MARK(finish_Machine2_op2_2)*– = 1;

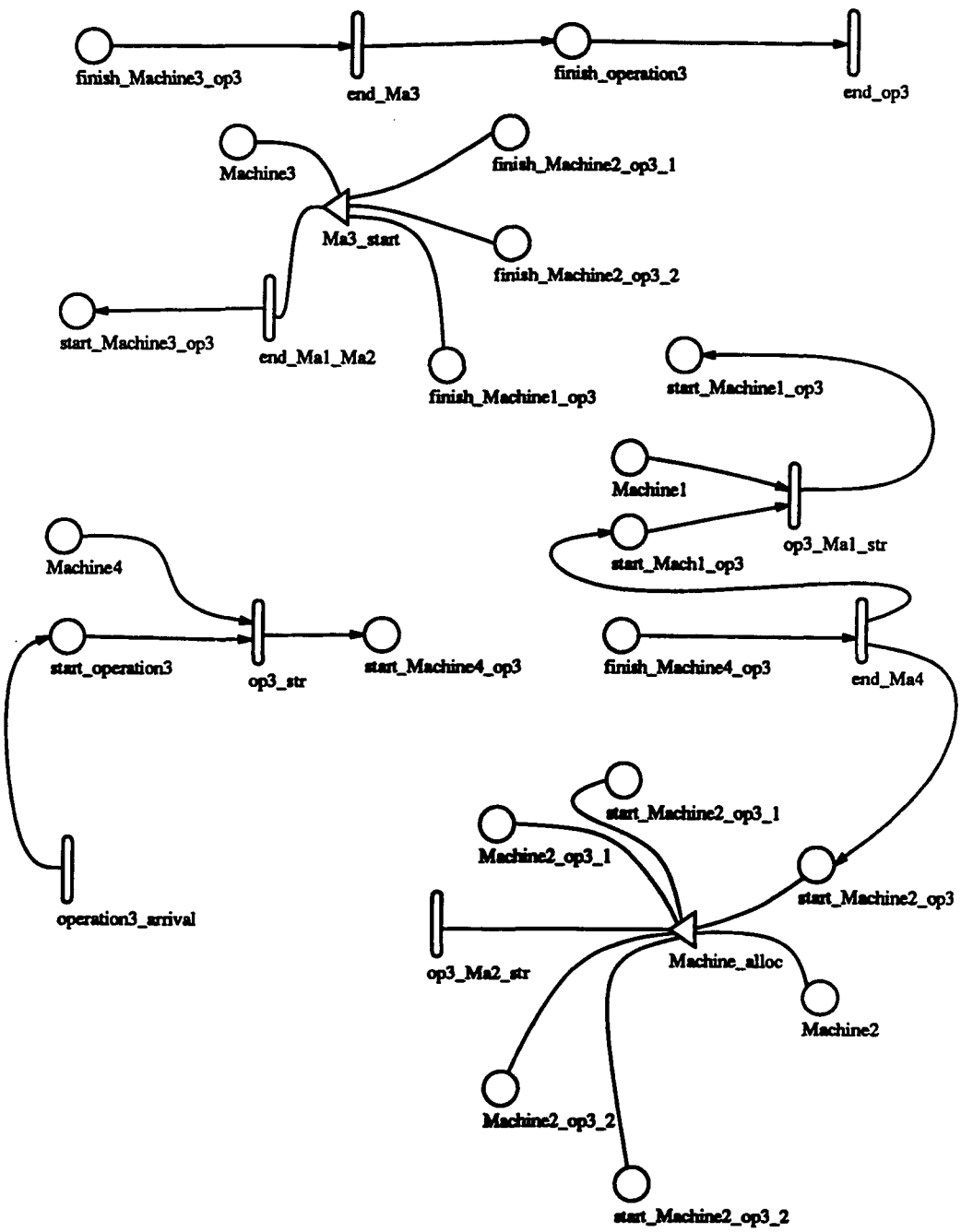


Table B.8: SAN Model Showing Manufacturing Flow of Operation3

Table B.9: Activity Time Distributions

Activity	Distribution
end_Ma1_Ma2	USAN_expon(10000000)
end_Ma3	USAN_expon(10000000)
end_Ma4	USAN_expon(10000000)
end_op3	USAN_expon(10000000)
op3_Ma1_str	USAN_expon(10000000)
op3_Ma2_str	USAN_expon(10000000)
op3_str	USAN_expon(10000000)
operation3_arrival	USAN_expon(varied)

Table B.10: Input Gate Predicates and Functions

Gate	Enabling Predicate	Function
Machine_alloc	*MARK(start_Machine2_op3) >* *0 && MARK(Machine2) > 0*	*if (MARK(Machine2_op3_1) == 0){* *MARK(start_Machine2_op3_1) = 1;* *MARK(Machine2_op3_1) = 1;* *}else{* *MARK(start_Machine2_op3_2) = 1;* *MARK(Machine2_op3_2) = 1;* *}* *MARK(start_Machine2_op3)- = 1;* *MARK(Machine2)- = 1;*
Ma3_start	*MARK(finish_Machine1_op3) >* *0 && (MARK(finish_Machine2_op3_1) >* *0 || MARK(finish_Machine2_op3_2) >* *0) && MARK(Machine3) > 0*	*MARK(finish_Machine1_op3)- = 1;* *if(MARK(finish_Machine2_op3_1) >* *0) MARK(finish_Machine2_op3_1)- = 1;* *else MARK(finish_Machine2_op3_2)- = 1;* *MARK(Machine3)- = 1;*

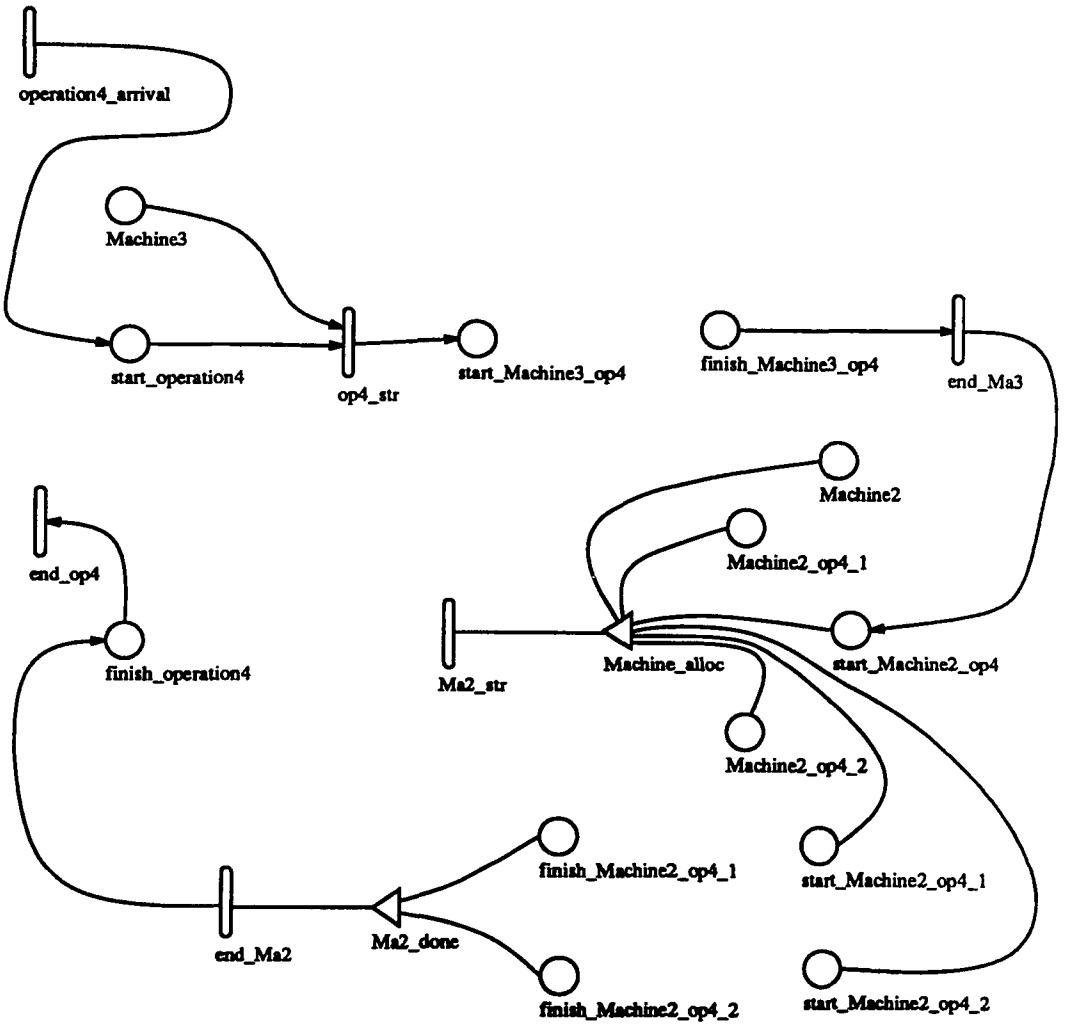


Table B.11: SAN Model Showing Manufacturing Flow of Operation4

Table B.12: Activity Time Distributions

Activity	Distribution
Ma2_str	USAN_expon(10000000)
end_Ma2	USAN_expon(10000000)
end_Ma3	USAN_expon(10000000)
end_op4	USAN_expon(10000000)
op4_str	USAN_expon(10000000)
operation4_arrival	USAN_expon(varied)

Table B.13: Input Gate Predicates and Functions

Gate	Enabling Predicate	Function
Machine_alloc	MARK(start_Machine2_op4) > 0 && MARK(Machine2) > 0	if(MARK(Machine2_op4_1) == 0){ MARK(Machine2_op4_1) = 1; MARK(start_Machine2_op4_1) = 1; }else{ MARK(Machine2_op4_2) = 1; MARK(start_Machine2_op4_2) = 1; } MARK(Machine2)- = 1; MARK(start_Machine2_op4)- = 1;
Mc2_done	MARK(finish_Machine2_op4_1) > 0 || MARK(finish_Machine2_op4_2) > 0	if(MARK(finish_Machine2_op4_1) > 0) MARK(finish_Machine2_op4_1)- = 1; else MARK(finish_Machine2_op4_2)- = 1;

APPENDIX C

Steady-state Simulation Results, Error Tables and Speedup Table of Performance Variables

Total arrival rate of products (per sec)	Non-decomposed model with exponentially distributed arrivals of products	Decomposed model with exponentially distributed arrivals of operations	Decomposed model with uniformly distributed arrivals of operations	Decomposed model with normally distributed arrivals of operations
.02	9.808177e-02+/- 2.152937e-03	9.92382e-02+/- 2.223388e-03	9.926593e-02+/- 1.359101e-03	9.764747e-02+/- 1.294968e-03
.04	1.965118e-01+/- 4.303845e-03	1.948579e-01+/- 2.803763e-03	1.957336e-01+/- 1.271501e-03	1.930745e-01+/- 1.7618e-03
.06	2.884873e-01+/- 3.673877e-03	2.841658e-01+/- 3.264624e-03	2.891447e-01+/- 3.06073e-03	2.862092e-01+/- 1.364276e-03
.08	3.812978e-01+/- 5.631117e-03	3.813996e-01+/- 7.10973e-03	3.858894e-01+/- 1.606313e-03	3.788396e-01+/- 1.881479e-03
.10	4.806044e-01+/- 5.072811e-03	4.768239e-01+/- 3.490046e-03	4.768879e-01+/- 3.661381e-03	4.667771e-01+/- 1.166367e-03
.12	5.658307e-01+/- 9.712428e-03	5.665549e-01+/- 3.916784e-03	5.65478e-01+/- 2.486481e-03	5.51514e-01+/- 1.806544e-03
.14	6.628686e-01+/- 7.736421e-03	6.62096e-01+/- 5.697873e-03	6.581267e-01+/- 3.823422e-03	6.526204e-01+/- 1.821353e-03
.16	7.551154e-01+/- 7.258527e-03	7.588809e-01+/- 7.61988e-03	7.512109e-01+/- 4.383433e-03	7.687435e-01+/- 1.897843e-03

Table C.1: Utilization of Machines of Type 1

Total arrival rate of products (per sec)	Decomposed model with exponentially distributed arrivals of operations		Decomposed model with uniformly distributed arrivals of operations		Decomposed model with normally distributed arrivals of operations	
	minimum	maximum	minimum	maximum	minimum	maximum
.02	0%	5.76%	0%	4.89%	0%	2.71%
.04	0%	4.36%	0%	3.16%	0%	4.73%
.06	0%	3.85%	0%	2.6%	0%	7.32%
.08	0%	3.42%	0%	3.15%	0%	2.57%
.10	0%	2.54%	0%	2.56%	0%	2%
.12	0%	2.58%	0%	2.18%	0%	4.14%
.14	0%	2.12%	0%	2.43%	0%	2.95%
.16	0%	2.49%	0%	2.04%	0.59%	3.05%

Table C.2: Percentage Error in Value of Utilization of Machines of Type 1 Measured Using Decomposition Technique

Total arrival rate of products (per sec)	Non-decomposed model with exponentially distributed arrivals of products	Decomposed model with exponentially distributed arrivals of operations	Decomposed model with uniformly distributed arrivals of operations	Decomposed model with normally distributed arrivals of operations
.02	9.019182e-01+/- 2.152937e-03	9.007618e-01+/- 2.223388e-03	9.007341e-01+/- 1.359101e-03	9.023525e-01+/- 1.294968e-03
.04	8.034882e-01+/- 4.303845e-03	8.051421e-01+/- 2.803763e-03	8.042664e-01+/- 1.271501e-03	8.069255e-01+/- 1.7618e-03
.06	7.115127e-01+/- 3.673877e-03	7.158342e-01+/- 3.264624e-03	7.108553e-01+/- 3.06073e-03	7.137908e-01+/- 1.364276e-03
.08	6.187022e-01+/- 5.631117e-03	6.186004e-01+/- 7.10973e-03	6.141106e-01+/- 1.606313e-03	6.211604e-01+/- 1.881479e-03
.10	5.193956e-01+/- 5.072811e-03	5.231761e-01+/- 3.490046e-03	5.231121e-01+/- 3.661381e-03	5.332229e-01+/- 1.166367e-03
.12	4.341693e-01+/- 9.712428e-03	4.334451e-01+/- 3.916784e-03	4.34522e-01+/- 2.486481e-03	4.48486e-01+/- 1.806544e-03
.14	3.371314e-01+/- 7.736421e-03	3.37904e-01+/- 5.697873e-03	3.418733e-01+/- 3.823422e-03	3.473796e-01+/- 1.821353e-03
.16	2.448846e-01+/- 7.258527e-03	2.411191e-01+/- 7.61988e-03	2.487891e-01+/- 4.383433e-03	2.312565e-01+/- 1.897843e-03

Table C.3: Availability of Machines of Type 1

Total arrival rate of products (per sec)	Decomposed model with exponentially distributed arrivals of operations		Decomposed model with uniformly distributed arrivals of operations		Decomposed model with normally distributed arrivals of operations	
	minimum	maximum	minimum	maximum	minimum	maximum
.02	0%	0.61%	0%	0.52%	0%	0.43%
.04	0%	1.1%	0%	0.8%	0%	1.19%
.06	0%	1.59%	0%	1.03%	0%	1.04%
.08	0%	2.06%	0%	1.89%	0%	1.62%
.10	0%	2.4%	0%	2.42%	0%	3.9%
.12	0%	3.23%	0%	2.96%	0%	6.09%
.14	0%	4.31%	0%	4.95%	0%	6.01%
.16	0%	7.39%	0%	6.54%	1.88%	9.04%

Table C.4: Percentage Error in Value of Availability of Machines of Type 1 Measured Using Decomposition Technique

Total arrival rate of products (per sec)	Non-decomposed model with exponentially distributed arrivals of products	Decomposed model with exponentially distributed arrivals of operations	Decomposed model with uniformly distributed arrivals of operations	Decomposed model with normally distributed arrivals of operations
.02	8.196886e-02+/- 1.957587e-03	8.22306e-02+/- 1.667332e-03	8.305119e-02+/- 1.101286e-03	8.265516e-02+/- 1.605381e-03
.04	1.690064e-01+/- 4.419861e-03	1.675122e-01+/- 1.700156e-03	1.675902e-01+/- 1.767087e-03	1.702555e-01+/- 1.584931e-03
.06	2.554859e-01+/- 3.672075e-03	2.543089e-01+/- 2.683081e-03	2.572587e-01+/- 1.771392e-03	2.598648e-01+/- 1.269632e-03
.08	3.483836e-01+/- 5.065824e-03	3.528925e-01+/- 4.905366e-03	3.531589e-01+/- 1.504774e-03	3.564718e-01+/- 1.462843e-03
.10	4.565542e-01+/- 6.306141e-03	4.573213e-01+/- 3.807148e-03	4.54273e-01+/- 1.645476e-03	4.549158e-01+/- 8.450727e-04
.12	5.569232e-01+/- 1.126272e-02	5.64777e-01+/- 4.681916e-03	5.590439e-01+/- 3.189298e-03	5.602704e-01+/- 2.037859e-03
.14	6.758141e-01+/- 1.100434e-02	6.815046e-01+/- 5.13299e-03	6.767167e-01+/- 6.193967e-03	6.509702e-01+/- 1.722106e-03
.16	7.688835e-01+/- 6.24634e-03	8.030032e-01+/- 6.000737e-03	7.981961e-01+/- 3.829504e-03	7.708548e-01+/- 1.275915e-03

Table C.5: Utilization of Machines of Type 2

Total arrival rate of products (per sec)	Decomposed model with exponentially distributed arrivals of operations		Decomposed model with uniformly distributed arrivals of operations		Decomposed model with normally distributed arrivals of operations	
	minimum	maximum	minimum	maximum	minimum	maximum
.02	0%	4.32%	0%	5.18%	0%	5.31%
.04	0%	4.39%	0%	4.38%	0%	4.41%
.06	0%	2.91%	0%	2.87%	0%	3.7%
.08	0%	4.22%	0%	3.3%	0%	4.26%
.10	0%	2.42%	0%	2.21%	0%	1.9%
.12	0%	4.36%	0%	3.04%	0%	3.06%
.14	0%	3.28%	0%	2.72%	0%	5.47%
.16	0%	6.08%	0%	5.16%	0%	1.24%

Table C.6: Percentage Error in Value of Utilization of Machines of Type 2 Measured Using Decomposition Technique

Total arrival rate of products (per sec)	Non-decomposed model with exponentially distributed arrivals of products	Decomposed model with exponentially distributed arrivals of operations	Decomposed model with uniformly distributed arrivals of operations	Decomposed model with normally distributed arrivals of operations
.02	9.683175e-01+/- 1.310196e-03	9.909065e-01+/- 7.1640888e-04	9.908484e-01+/- 6.409713e-04	9.917647e-01+/- 7.105006e-03
.04	9.27264e-01+/- 2.874806e-03	9.638368e-01+/- 1.647948e-03	9.649983e-01+/- 1.485927e-03	9.564768e-01+/- 8.461031e-03
.06	8.787866e-01+/- 3.057527e-03	9.186609e-01+/- 2.151113e-03	9.18746e-01+/- 1.807039e-03	9.196704e-01+/- 7.112691e-03
.08	8.143158e-01+/- 3.950338e-03	8.470392e-01+/- 4.221973e-03	8.53333e-01+/- 2.688387e-03	8.436519e-01+/- 8.016714e-03
.10	7.248551e-01+/- 6.81084e-03	7.537917e-01+/- 4.520928e-03	7.635479e-01+/- 2.613826e-03	7.67528e-01+/- 3.82722e-03
.12	6.244867e-01+/- 1.217575e-02	6.388047e-01+/- 5.525704e-03	6.528528e-01+/- 4.563568e-03	6.75186e-01+/- 4.446896e-03
.14	4.890758e-01+/- 1.471025e-02	4.939806e-01+/- 7.159117e-03	5.092503e-01+/- 8.430095e-03	5.862297e-01+/- 3.245811e-03
.16	3.665101e-01+/- 9.506144e-03	3.217937e-01+/- 8.567122e-03	3.366628e-01+/- 6.263785e-03	3.978345e-01+/- 2.3595e-03

Table C.7: Availability of Machines of Type 2

Total arrival rate of products (per sec)	Decomposed model with exponentially distributed arrivals of operations		Decomposed model with uniformly distributed arrivals of operations		Decomposed model with normally distributed arrivals of operations	
	minimum	maximum	minimum	maximum	minimum	maximum
.02	2.19%	2.48%	0%	2.53%	1.55%	3.29%
.04	3.45%	4.45%	3.59%	4.59%	1.92%	4.39%
.06	3.93%	5.15%	4.39%	5.12%	3.48%	5.83%
.08	3%	5.05%	3.96%	5.63%	2.12%	5.1%
.10	2.41%	5.61%	0%	6.7%	0%	7.42%
.12	0%	5.22%	1.83%	7.37%	5.35%	10.99%
.14	0%	5.64%	0%	9.13%	15.74%	24.25%
.16	7.46%	16.7%	3.94%	12.13%	5.17%	12.1%

Table C.8: Percentage Error in Value of Availability of Machines of Type 2 Measured Using Decomposition Technique

Total arrival rate of products (per sec)	Non-decomposed model with exponentially distributed arrivals of products	Decomposed model with exponentially distributed arrivals of operations	Decomposed model with uniformly distributed arrivals of operations	Decomposed model with normally distributed arrivals of operations
.02	1.367472e-01+/- 3.24225e-03	1.376114e-01+/- 2.952897e-04	1.395786e-01+/- 1.633016e-03	1.380896e-01+/- 5.16464e-04
.04	2.766265e-01+/- 6.690664e-03	2.750658e-01+/- 3.247548e-03	2.763903e-01+/- 2.356159e-03	2.757673e-01+/- 5.544736e-04
.06	4.099171e-01+/- 5.181849e-03	4.067759e-01+/- 4.503272e-03	4.128092e-01+/- 3.248733e-03	4.147399e-01+/- 7.682478e-04
.08	5.440091e-01+/- 7.476723e-03	5.439668e-01+/- 8.655073e-03	5.51853e-01+/- 2.164474e-03	5.519105e-01+/- 9.719618e-04
.10	6.884733e-01+/- 8.402788e-03	6.773478e-01+/- 4.421172e-03	6.866836e-01+/- 4.118614e-03	6.907772e-01+/- 8.795636e-04
.12	8.056954e-01+/- 1.274154e-02	7.947967e-01+/- 6.375202e-03	8.116491e-01+/- 3.434512e-03	8.268831e-01+/- 1.088417e-03
.14	9.31493e-01+/- 9.825232e-03	9.005496e-01+/- 6.029879e-03	9.210233e-01+/- 5.129478e-03	9.574662e-01+/- 1.454131e-03
.16	9.999366e-01+/- 1.461289e-03	9.72997e-01+/- 3.456197e-03	9.863654e-01+/- 1.322864e-03	9.977308e-01+/- 2.6092286e-04

Table C.9: Utilization of Machines of Type 3

Total arrival rate of products (per sec)	Decomposed model with exponentially distributed arrivals of operations		Decomposed model with uniformly distributed arrivals of operations		Decomposed model with normally distributed arrivals of operations	
	minimum	maximum	minimum	maximum	minimum	maximum
.02	0%	5.89%	0%	5.77%	0%	3.82%
.04	0%	4.06%	0%	3.28%	0%	2.86%
.06	0%	3.19%	0%	2.8%	0%	2.66%
.08	0%	2.93%	0%	3.26%	0%	3.05%
.10	0%	3.44%	0%	2.05%	0%	1.7%
.12	0%	3.67%	0%	2.79%	0%	4.42%
.14	1.64%	4.97%	0%	2.7%	1.56%	4.04%
.16	2.33%	3.05%	1.21%	1.5%	0.18%	0.25%

Table C.10: Percentage Error in Value of Utilization of Machines of Type 3 Measured Using Decomposition Technique

Total arrival rate of products (per sec)	Non-decomposed model with exponentially distributed arrivals of products	Decomposed model with exponentially distributed arrivals of operations	Decomposed model with uniformly distributed arrivals of operations	Decomposed model with normally distributed arrivals of operations
.02	8.632528e-01+/- 3.24225e-03	8.623886e-01+/- 2.952897e-04	8.604214e-01+/- 1.633016e-03	8.619104e-01+/- 5.16464e-04
.04	7.233735e-01+/- 6.690664e-03	7.249342e-01+/- 3.247548e-03	7.236097e-01+/- 2.356159e-03	7.242327e-01+/- i5.544736e-04
.06	5.900829e-01+/- 5.181849e-03	5.932241e-01+/- 4.503272e-03	5.871908e-01+/- 3.248733e-03	5.852601e-01+/- 7.682478e-04
.08	4.559909e-01+/- 7.476723e-03	4.560332e-01+/- 8.655073e-03	4.48147e-01+/- 2.164474e-03	4.480895e-01+/- 9.719618e-04
.10	3.115267e-01+/- 8.402788e-03	3.226522e-01+/- 4.421172e-03	3.133164e-01+/- 4.118614e-03	3.092228e-01+/- 8.795636e-04
.12	1.943046e-01+/- 1.274154e-02	2.052033e-01+/- 6.375202e-03	1.883509e-01+/- 3.434512e-03	1.731169e-01+/- 1.088417e-03
.14	6.8507e-02+/- 9.825232e-03	9.945041e-02+/- 6.029879e-03	7.89767e-02+/- 5.129478e-03	4.253383e-02+/- 1.454131e-03

Table C.11: Availability of Machines of Type 3

Total arrival rate of products (per sec)	Decomposed model with exponentially distributed arrivals of operations		Decomposed model with uniformly distributed arrivals of operations		Decomposed model with normally distributed arrivals of operations	
	minimum	maximum	minimum	maximum	minimum	maximum
.02	0%	0.82%	0%	0.89%	0%	0.59%
.04	0%	1.6%	0%	1.29%	0%	1.13%
.06	0%	2.19%	0%	1.9%	0%	1.81%
.08	0%	3.61%	0%	3.77%	0%	3.53%
.10	0%	7.9%	0%	4.72%	0%	3.62%
.12	0%	16.53%	0%	10.69%	7.36%	16.9%
.14	19.26%	79.75%	0%	43.3%	25.04%	47.56%

Table C.12: Percentage Error in Value of Availability of Machines of Type 3 Measured Using Decomposition Technique

Total arrival rate of products (per sec)	Non-decomposed model with exponentially distributed arrivals of products	Decomposed model with exponentially distributed arrivals of operations	Decomposed model with uniformly distributed arrivals of operations	Decomposed model with normally distributed arrivals of operations
.02	1.061942e-01+/- 2.86161e-03	1.072415e-01+/- 2.817272e-03	1.079328e-01+/- 1.633016e-03	1.074353e-01+/- 3.239694e-04
.04	2.164073e-01+/- 4.851095e-03	2.141777e-01+/- 3.439492e-03	2.137043e-01+/- 1.388893e-03	2.146676e-01+/- 5.264795e-04
.06	3.186587e-01+/- 3.147259e-03	3.152612e-01+/- 4.577756e-03	3.204037e-01+/- 2.462428e-03	3.224287e-01+/- 6.781819e-04
.08	4.247693e-01+/- 4.542595e-03	4.281878e-01+/- 5.980505e-03	4.318172e-01+/- 2.873069e-03	4.295735e-01+/- 7.820105e-04
.10	5.414692e-01+/- 5.923374e-03	5.371955e-01+/- 4.173174e-03	5.388595e-01+/- 3.307686e-03	5.370468e-01+/- 5.051945e-04
.12	6.444757e-01+/- 1.091842e-02	6.443979e-01+/- 4.276246e-03	6.450318e-01+/- 3.979006e-03	6.441110e-01+/- 9.124785e-04
.14	7.573755e-01+/- 1.157328e-02	7.510211e-01+/- 9.298837e-03	7.52426e-01+/- 3.486166e-03	7.51289e-01+/- 1.241958e-03
.16	8.55483e-01+/- 8.1292e-03	8.58678e-01+/- 8.747211e-03	8.574823e-01+/- 4.146215e-03	8.591258e-01+/- 1.367932e-03

Table C.13: Utilization of Machines of Type 4

Total arrival rate of products (per sec)	Decomposed model with exponentially distributed arrivals of operations		Decomposed model with uniformly distributed arrivals of operations		Decomposed model with normally distributed arrivals of operations	
	minimum	maximum	minimum	maximum	minimum	maximum
.02	0%	6.51%	0%	6.03%	0%	4.28%
.04	0%	4.75%	0%	4.04%	0%	3.22%
.06	0%	3.46%	0%	2.33%	0%	7.6%
.08	0%	3.32%	0%	3.44%	0%	2.41%
.10	0%	2.63%	0%	2.16%	0%	1.98%
.12	0%	2.33%	0%	2.44%	0%	1.86%
.14	0%	3.54%	0%	2.6%	0%	2.46%
.16	0%	2.34%	0%	1.68%	0%	1.55%

Table C.14: Percentage Error in Value of Utilization of Machines of Type 4 Measured Using Decomposition Technique

Total arrival rate of products (per sec)	Non-decomposed model with exponentially distributed arrivals of products	Decomposed model with exponentially distributed arrivals of operations	Decomposed model with uniformly distributed arrivals of operations	Decomposed model with normally distributed arrivals of operations
.02	8.938058e-01+/- 2.86161e-03	8.927585e-01+/- 2.817272e-03	8.920672e-01+/- 1.633016e-03	8.925647e-01+/- 3.239694e-04
.04	7.835927e-01+/- 4.851095e-03	7.858223e-01+/- 3.439492e-03	7.862957e-01+/- 1.388893e-03	7.853324e-01+/- 5.264795e-04
.06	6.813413e-01+/- 3.147259e-03	6.847388e-01+/- 4.577756e-03	6.795963e-01+/- 2.462428e-03	6.775713e-01+/- 6.781819e-04
.08	5.752307e-01+/- 4.542595e-03	5.718122e-01+/- 5.980505e-03	5.681828e-01+/- 2.873069e-03	5.704265e-01+/- 7.820105e-04
.10	4.585308e-01+/- 5.923374e-03	4.628045e-01+/- 4.173174e-03	4.611405e-01+/- 3.307686e-03	4.629532e-01+/- 5.051945e-04
.12	3.555243e-01+/- 1.091842e-02	3.556021e-01+/- 4.276246e-03	3.549682e-01+/- 3.979006e-03	3.55889e-01+/- 9.124785e-04
.14	2.426245e-01+/- 1.157328e-02	2.489789e-01+/- 9.298837e-03	2.47574e-01+/- 3.486166e-03	2.48711e-01+/- 1.241958e-03
.16	1.44517e-01+/- 8.1292e-03	1.41322e-01+/- 8.747211e-03	1.425177e-01+/- 4.146215e-03	1.408742e-01+/- 1.367932e-03

Table C.15: Availability of Machines of Type 4

Total arrival rate of products (per sec)	Decomposed model with exponentially distributed arrivals of operations		Decomposed model with uniformly distributed arrivals of operations		Decomposed model with normally distributed arrivals of operations	
	minimum	maximum	minimum	maximum	minimum	maximum
.02	0%	0.75%	0%	0.7%	0%	0.42%
.04	0%	1.36%	0%	1.15%	0%	0.91%
.06	0%	1.64%	0%	1.07%	0%	0.91%
.08	0%	2.4%	0%	2.49%	0%	1.75%
.10	0%	3.17%	0%	2.61%	0%	2.39%
.12	0%	4.43%	0%	4.22%	0%	3.54%
.14	0%	11.78%	0%	8.66%	0%	8.18%
.16	0%	12.97%	0%	9.35%	0%	8.61%

Table C.16: Percentage Error in Value of Availability of Machines of Type 4 Measured Using Decomposition Technique

Total arrival rate of products (per sec)	Non-decomposed model with exponentially distributed arrivals of products	Decomposed model with exponentially distributed arrivals of operations	Decomposed model with uniformly distributed arrivals of operations	Decomposed model with normally distributed arrivals of operations
.02	6.759578e-04+/- 6.902108e-05	5.286855e-04+/- 3.550738e-05	1.090631e-04+/- 6.44198e-05	8.660157e-05+/- 1.911469e-04
.04	2.606069e-03+/- 3.044058e-04	8.923783e-04+/- 1.766579e-04	6.41105e-04+/- 1.872743e-04	7.427703e-04+/- 4.044203e-04
.06	6.006587e-03+/- 4.990099e-04	2.889584e-03+/- 4.838848e-04	2.451533e-03+/- 3.226244e-04	1.864097e-03+/- 6.637845e-04
.08	1.284096e-02+/- 8.785892e-04	8.60202e-03+/- 8.117789e-04	6.769794e-03+/- 4.658168e-04	6.184152e-03+/- 1.094143e-03
.10	2.308737e-02+/- 1.653748e-03	1.866912e-02+/- 8.633092e-04	1.38186e-02+/- 5.874689e-04	1.396612e-02+/- 1.038364e-03
.12	4.195111e-02+/- 3.821087e-03	3.770752e-02+/- 8.566229e-04	2.558086e-02+/- 1.221152e-03	2.132103e-02+/- 7.9946e-04
.14	7.547225e-02+/- 5.436549e-03	7.361583e-02+/- 3.433067e-03	4.903338e-02+/- 2.551177e-03	2.141331e-02+/- 1.204542e-03
.16	1.33399e-01+/- 8.398269e-03	1.517865e-01+/- 9.811275e-03	9.283005e-02+/- 4.443156e-03	2.048172e-02+/- 5.27183e-04

Table C.17: Queue Length of Operation1

Total arrival rate of products (per sec)	Decomposed model with exponentially distributed arrivals of operations		Decomposed model with uniformly distributed arrivals of operations		Decomposed model with normally distributed arrivals of operations	
	minimum	maximum	minimum	maximum	minimum	maximum
.02	85.44%	97.67%	71.42%	94%	54.2%	85.9%
.04	53.4%	75.5%	64.01%	84.41%	50.16%	88.38%
.06	38.75%	63.02%	49.63%	67.28%	54.1%	81.55%
.08	21.3%	43.22%	39.51%	54.05%	39.16%	63.9%
.10	8.87%	28.03%	33.79%	46.52%	30%	47.75%
.12	0%	19.49%	32.85%	46.78%	41.99%	55.17%
.14	0%	13.26%	26.35%	42.55%	67.71%	75.02%
.16	0.13%	29.28%	22.18%	37.67%	83.21%	85.91%

Table C.18: Percentage Error in Value of Queue Length of Operation1 Measured Using Decomposition Technique

Total arrival rate of products (per sec)	Non-decomposed model with exponentially distributed arrivals of products	Decomposed model with exponentially distributed arrivals of operations	Decomposed model with uniformly distributed arrivals of operations	Decomposed model with normally distributed arrivals of operations
.02	8.537609e-04+/- 1.671354e-04	9.908371e-04+/- 1.648676e-04	7.109918e-04+/- 2.989102e-04	1.435877e-03+/- 1.221257e-03
.04	3.48952e-03+/- 4.604093e-04	3.655665e-03+/- 3.716562e-04	3.107892e-03+/- 4.242699e-04	2.85494e-03+/- 2.308211e-03
.06	8.118793e-03+/- 9.076387e-04	8.154962e-03+/- 8.143166e-04	7.193379e-03+/- 8.031593e-04	7.080622e-03+/- 2.134728e-04
.08	1.600548e-02+/- 1.612393e-03	1.577204e-02+/- 1.153021e-03	1.401464e-02+/- 7.646607e-04	1.035445e-02+/- 1.916214e-03
.10	2.944937e-02+/- 2.531761e-03	2.997331e-02+/- 1.435292e-03	2.256058e-02+/- 6.95139e-04	1.644445e-02+/- 3.025072e-03
.12	4.666951e-02+/- 4.275467e-03	4.771369e-02+/- 1.969005e-03	3.601886e-02+/- 1.800398e-03	2.299973e-02+/- 1.573753e-03
.14	7.838466e-02+/- 5.762012e-03	7.775326e-02+/- 4.862834e-03	5.748842e-02+/- 2.505021e-03	3.646668e-02+/- 3.596425e-03
.16	1.342843e-01+/- 8.084761e-03	1.293872e-01+/- 7.2989e-03	8.249688e-02+/- 3.078605e-03	5.387576e-02+/- 2.867889e-03

Table C.19: Queue Length of Operation2

Total arrival rate of products (per sec)	Decomposed model with exponentially distributed arrivals of operations		Decomposed model with uniformly distributed arrivals of operations		Decomposed model with normally distributed arrivals of operations	
	minimum	maximum	minimum	maximum	minimum	maximum
.02	0%	68.3%	0%	59.64%	61.3%	96.87%
.04	0%	32.95%	0%	32.06%	0%	86.16%
.06	0%	24.3%	0%	29.2%	0%	45.21%
.08	0%	17.02%	0%	24.79%	14.75%	52.1%
.10	0%	16.68%	15.34%	31.63%	27.67%	58.04%
.12	0%	17.2%	10.79%	32.83%	42.04%	57.94%
.14	0%	13.38%	17.39%	34.66%	44.83%	60.94%
.16	0%	14.25%	32.19%	44.72%	55.04%	64.17%

Table C.20: Percentage Error in Value of Queue Length of Operation2 Measured Using Decomposition Technique

Total arrival rate of products (per sec)	Non-decomposed model with exponentially distributed arrivals of products	Decomposed model with exponentially distributed arrivals of operations	Decomposed model with uniformly distributed arrivals of operations	Decomposed model with normally distributed arrivals of operations
.02	5.627186e-03+/- 5.681412e-04	3.835694e-03+/- 7.389593e-04	2.713677e-03+/- 4.907632e-04	2.796977e-03+/- 1.41216e-03
.04	2.609497e-02+/- 2.733028e-03	1.760926e-02+/- 2.034406e-03	1.170607e-02+/- 8.837441e-04	5.621174e-03+/- 1.070539e-03
.06	5.971255e-02+/- 6.533285e-03	4.125593e-02+/- 2.352215e-03	2.710488e-02+/- 2.294174e-03	1.272713e-02+/- 1.103842e-03
.08	1.258381e-01+/- 6.801231e-03	9.467826e-02+/- 6.557294e-03	5.608511e-02+/- 2.175927e-03	2.532955e-02+/- 9.032443e-04
.10	2.509896e-01+/- 1.232948e-02	1.851923e-01+/- 6.232381e-03	9.741645e-02+/- 4.894468e-03	4.42501e-02+/- 5.472658e-04
.12	4.438216e-01+/- 4.463992e-02	3.461747e-01+/- 1.176231e-02	1.693966e-01+/- 8.247889e-03	4.383678e-02+/- 1.202479e-03
.14	9.179294e-01+/- 8.797202e-02	6.774385e-01+/- 5.774225e-02	2.974033e-01+/- 1.470394e-02	6.351804e-02+/- 2.000337e-03
.16	2.302683e+00+/- 2.602122e-01	1.746121e+00+/- 2.556265e-01	6.031545e-01+/- 4.393879e-02	1.275217e-01+/- 1.454746e-03

Table C.21: Queue Length of Operation3

Total arrival rate of products (per sec)	Decomposed model with exponentially distributed arrivals of operations		Decomposed model with uniformly distributed arrivals of operations		Decomposed model with normally distributed arrivals of operations	
	minimum	maximum	minimum	maximum	minimum	maximum
.02	9.57%	50.01%	36.66%	64.12%	16.82%	78.12%
.04	15.92%	45.97%	46.11%	62.46%	71.36%	84.21%
.06	18%	41.27%	44.72%	62.55%	74%	82.45%
.08	14.95%	33.56%	51.06%	59.36%	71.13%	87.7%
.10	19.79%	32.04%	57.13%	64.86%	81.2%	83.4%
.12	10.33%	31.54%	55.5%	67.01%	88.72%	90.49%
.14	11.42%	38.39%	62.39%	71.9%	92.11%	93.88%
.16	1.99%	41.84%	68.32%	78.18%	93.69%	95.08%

Table C.22: Percentage Error in Value of Queue Length of Operation3 Measured Using Decomposition Technique

Total arrival rate of products (per sec)	Non-decomposed model with exponentially distributed arrivals of products	Decomposed model with exponentially distributed arrivals of operations	Decomposed model with uniformly distributed arrivals of operations	Decomposed model with normally distributed arrivals of operations
.02	5.699539e-03+/- 5.92946e-04	5.159755e-03+/- 6.60546e-04	3.807947e-03+/- 5.292273e-04	2.041751e-03+/- 1.774233e-03
.04	2.678912e-02+/- 2.109199e-03	2.315068e-02+/- 1.473025e-03	1.69673e-02+/- 1.062726e-03	7.580246e-03+/- 4.404418e-03
.06	7.49422e-02+/- 6.605816e-03	6.050565e-02+/- 2.712386e-03	4.155505e-02+/- 1.561964e-03	2.008602e-02+/- 4.790184e-03
.08	1.678452e-01+/- 9.738273e-03	1.420243e-01+/- 8.39847e-03	8.363467e-02+/- 3.253229e-03	4.155282e-02+/- 5.05363e-03
.10	3.933125e-01+/- 4.272747e-02	3.003509e-01+/- 1.054544e-02	1.676584e-01+/- 6.204515e-03	7.000045e-02+/- 5.038184e-03
.12	8.918224e-01+/- 1.529076e-01	6.465494e-01+/- 3.15204e-02	3.389968e-01+/- 1.994743e-02	1.199524e-01+/- 5.054748e-03
.14	3.661352+00+/- 8.193978e-01	1.598298+00+/- 1.618203e-01	8.329809e-01+/- 5.295617e-02	2.750521e-01+/- 9.574177e-03
.16	8.076048e+02+/- 4.092372e+02	6.356254e+00+/- 1.166937e+00	3.333653e+00+/- 3.091133e-01	1.530711e+00+/- 1.367357e-01

Table C.23: Queue Length of Operation4

Total arrival rate of products (per sec)	Decomposed model with exponentially distributed arrivals of operations		Decomposed model with uniformly distributed arrivals of operations		Decomposed model with normally distributed arrivals of operations	
	minimum	maximum	minimum	maximum	minimum	maximum
.02	0%	28.5%	15.07%	47.89%	25.27%	94.76%
.04	0.22%	24.98%	26.94%	44.96%	51.44%	89.01%
.06	7.5%	29.12%	36.9%	50.96%	64%	81.24%
.08	4.86%	24.75%	45.04%	54.74%	70.52%	79.45%
.10	11.32%	33.54%	50.41%	62.97%	78.6%	85.1%
.12	16.76%	41.13%	51.42%	69.46%	83.08%	89%
.14	38.07%	67.94%	68.83%	82.59%	89.98%	94.08%

Table C.24: Percentage Error in Value of Queue Length of Operation4 Measured Using Decomposition Technique

Total arrival rate of products (per sec)	Non-decomposed model with exponentially distributed arrivals of products	Decomposed model with exponentially distributed arrivals of operations	Decomposed model with uniformly distributed arrivals of operations	Decomposed model with normally distributed arrivals of operations
.02	1.596791e-04+/- 3.491918e-06	1.610039e-04+/- 3.442271e-06	1.629349e-04+/- 1.966269e-06	1.609644e-04+/- 3.890699e-07
.04	3.230681e-04+/- 7.599088e-06	3.206872e-04+/- 3.355232e-06	3.219731e-04+/- 2.618944e-06	3.216308e-04+/- 6.421853e-07
.06	4.781724e-04+/- 5.738098e-06	4.738207e-04+/- 5.094483e-06	4.814337e-04+/- 3.55506e-06	4.828936e-04+/- 9.12189e-07
.08	6.368171e-04+/- 8.792327e-06	6.396057e-04+/- 9.184191e-06	6.45916e-04+/- 2.82099e-06	6.443687e-04+/- 1.1412e-06
.10	8.090319e-04+/- 8.67888e-06	8.021234e-04+/- 5.30552e-06	8.052343e-04+/- 4.472734e-06	8.048222e-04+/- 9.50165e-07
.12	9.559845e-04+/- 1.532905e-05	9.554464e-04+/- 6.520246e-06	9.612084e-04+/- 3.916903e-06	9.661247e-04+/- 1.316184e-06
.14	1.116724e-03+/- 1.391427e-05	1.107097e-03+/- 7.459479e-06	1.114143e-03+/- 6.178698e-06	1.124292e-03+/- 1.413469e-06
.16	1.236679e-03+/- 6.541506e-06	1.2444507e-03+/- 7.252022e-06	1.247701e-03+/- 3.985616e-06	1.254417e-03+/- 1.645246e-06

Table C.25: Utilization of Channel

Total arrival rate of products (per sec)	Decomposed model with exponentially distributed arrivals of operations		Decomposed model with uniformly distributed arrivals of operations		Decomposed model with normally distributed arrivals of operations	
	minimum	maximum	minimum	maximum	minimum	maximum
.02	0%	5.29%	0%	5.56%	0%	3.3%
.04	0%	4.03%	0%	3.42%	0%	2.93%
.06	0%	3.14%	0%	2.61%	0%	2.41%
.08	0%	3.31%	0%	3.3%	0%	2.78%
.10	0%	2.55%	0%	2.07%	0%	1.69%
.12	0%	2.3%	0%	2.6%	0%	4.18%
.14	0%	2.74%	0%	2%	0%	2.29%
.16	0%	1.76%	0%	1.75%	0.768%	2.11%

Table C.26: Percentage Error in Value of Utilization of Channel Measured Using Decomposition Technique

Total arrival rate of products (per sec)	Non-decomposed model with exponentially distributed arrivals of products	Decomposed model with exponentially distributed arrivals of operations	Decomposed model with uniformly distributed arrivals of operations	Decomposed model with normally distributed arrivals of operations
.02	9.998403e-01+/- 3.495252e-06	9.99839e-01+/- 3.4423245e-06	9.99837e-01+/- 1.96656e-06	9.99839e-01+/- 3.892684e-07
.04	9.996769e-01+/- 7.600391e-06	9.996793e-01+/- 3.355815e-06	9.99678e-01+/- 2.619243e-06	9.996783e-01+/- 6.421409e-07
.06	9.995217e-01+/- 5.738901e-06	9.995261e-01+/- 5.095304e-06	9.995185e-01+/- 3.555659e-06	9.99517e-01+/- 9.122324e-07
.08	9.993631e-01+/- 8.798653e-06	9.993603e-01+/- 9.185629e-06	9.99354e-01+/- 2.821273e-06	9.993555e-01+/- 1.141272e-06
.10	9.991908e-01+/- 8.680346e-06	9.991977e-01+/- 5.306386e-06	9.991946e-01+/- 4.473541e-06	9.99195e-01+/- 9.50201e-07
.12	9.990439e-01+/- 1.533151e-05	9.990444e-01+/- 6.521283e-06	9.990386e-01+/- 3.917495e-06	9.990337e-01+/- 1.316316e-06
.14	9.988831e-01+/- 1.391612e-05	9.988927e-01+/- 7.46074e-06	9.988857e-01+/- 6.179593e-06	9.988755e-01+/- 1.413586e-06
.16	9.987631e-01+/- 6.542274e-06	9.987553e-01+/- 7.252886e-06	9.987521e-01+/- 3.985919e-06	9.987454e-01+/- 1.645358e-06

Table C.27: Idle Channel

Total arrival rate of products (per sec)	Decomposed model with exponentially distributed arrivals of operations		Decomposed model with uniformly distributed arrivals of operations		Decomposed model with normally distributed arrivals of operations	
	minimum	maximum	minimum	maximum	minimum	maximum
.02	0%	8.24e-04%	0%	8.76e-04%	0%	1.8e-04%
.04	0%	1.34e-03%	0%	1.13e-03%	0%	6.51e-02%
.06	0%	1.52e-03%	0%	1.25e-03%	0%	1.14e-02%
.08	0%	2.08e-03%	0%	2.07e-03%	0%	1.78e-03%
.10	0%	2.09e-03%	0%	1.7e-03%	0%	1.38e-03%
.12	0%	2.23e-03%	0%	2.46e-03%	0%	2.69e-03%
.14	0%	1.61e-03%	0%	2.27e-03%	0%	2.3e-03%
.16	0%	2.16e-03%	0%	2.16e-03%	9.52e-04%	2.76e-03%

Table C.28: Percentage Error in Value of Idle Channel Measured Using Decomposition Technique

Total arrival rate of products (per sec) arrivals of products	Simulation time for Non-decomposed model with exponentially distributed arrivals of operations (in seconds)	Simulation time Decomposed model with exponentially distributed arrivals of operations (in seconds)	Speed-up factor
.02	14685	9010	1.63
.04	30382	18228	1.67
.06	44856	27347	1.64
.08	59789	37025	1.61
.10	75394	46855	1.61
.12	88713	55841	1.59
.14	102357	66612	1.54
.16	110976	71166	1.56

Table C.29: Speed up Obtained in Simulation Time Using Decomposition Technique

APPENDIX D

SAN Models of Example Manufacturing System as Shown in Figures 6.1 and 6.2

Table D.1: Composite Model of the Manufacturing System Represented in Figures 6.1 and 6.2

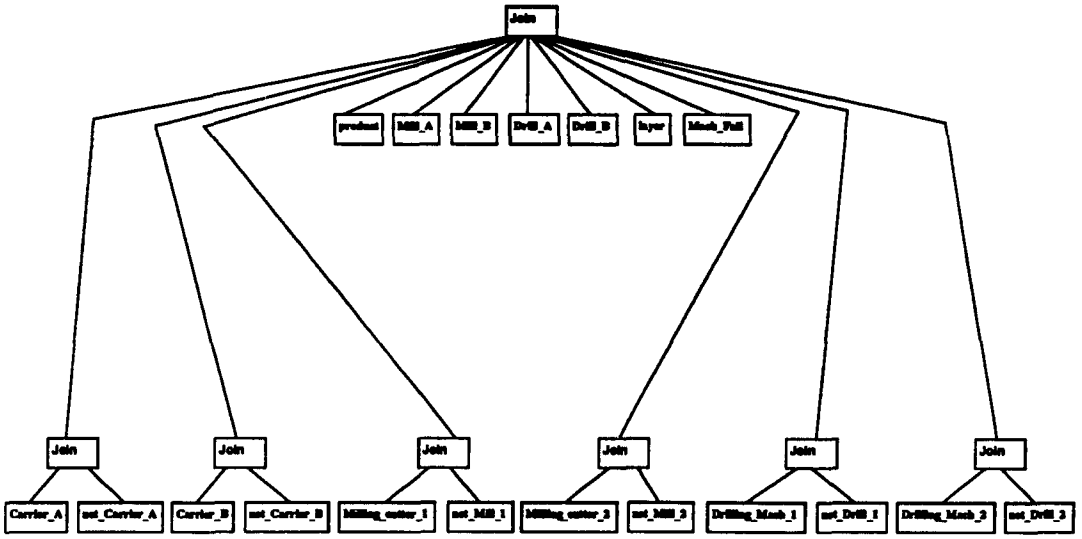


Table D.2: SAN Model of Manufacturing Flow of Product

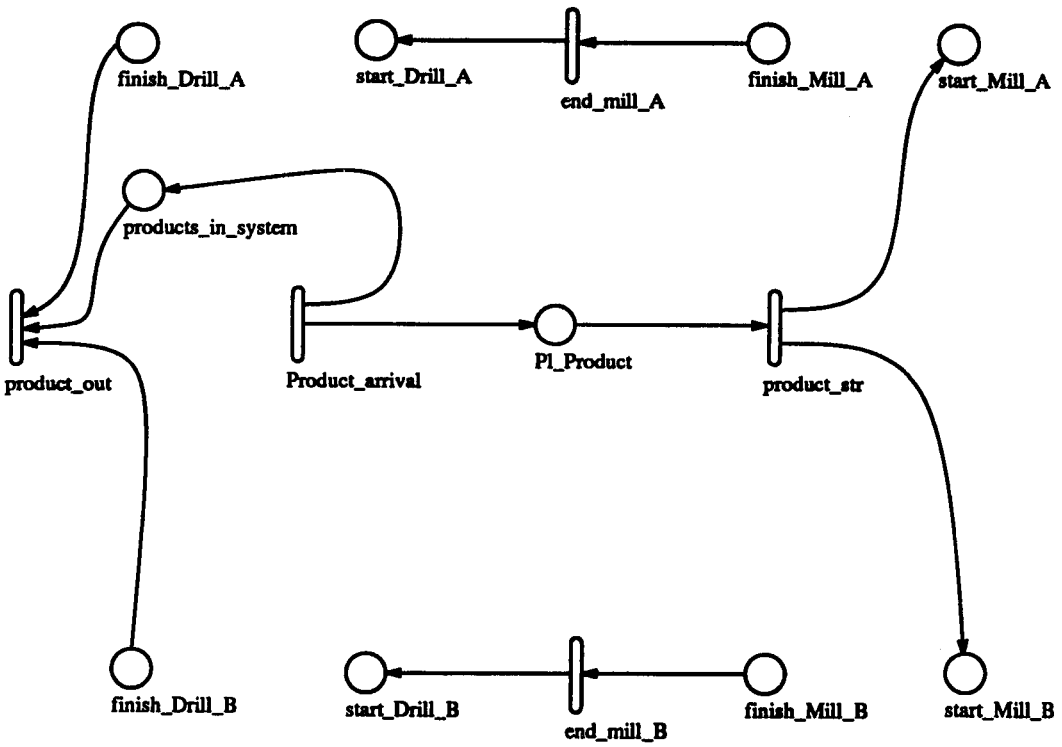


Table D.3: Activity Time Distributions

Activity	Distribution
Product_arrival	USAN_expon(.0000138889)
end_mill_A	USAN_expon(10000000)
end_mill_B	USAN_expon(10000000)
product_out	USAN_expon(10000000)
product_str	USAN_expon(10000000)

Table D.4: SAN Model of Milling Operation Performed on Part A

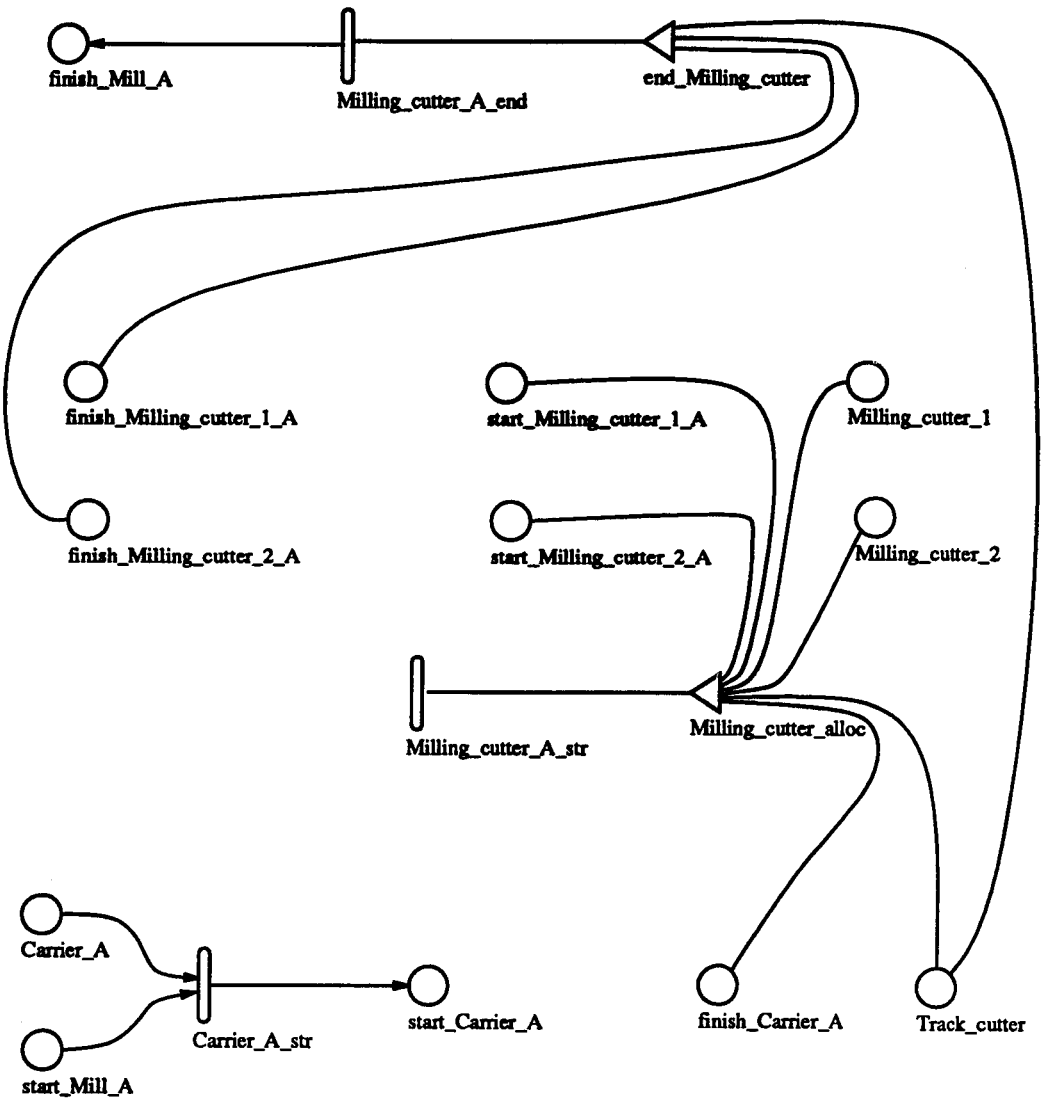


Table D.5: Activity Time Distributions

Activity	Distribution
Carrier_A_str	USAN_expon(10000000)
Milling_cutter_A_end	USAN_expon(10000000)
Milling_cutter_A_str	USAN_expon(10000000)

Table D.6: Input Gate Predicates and Functions

Gate	Enabling Predicate	Function
end_Milling_cutter	MARK(finish_Milling_cutter_1_A) > 0 || MARK(finish_Milling_cutter_2_A) > 0	switch(MARK(Track_cutter)){ case (1) : if (MARK(finish_Milling_cutter_2_A) > 0) MARK(finish_Milling_cutter_2_A) − −; else MARK(finish_Milling_cutter_1_A) − −; break; case (2) : if (MARK(finish_Milling_cutter_1_A) > 0) MARK(finish_Milling_cutter_1_A) − −; else MARK(finish_Milling_cutter_2_A) − −; break; }
Milling_cutter_alloc	MARK(finish_Carrier_A) > 0 && (MARK(Milling_cutter_1) > 0 || MARK(Milling_cutter_2) > 0)	switch(MARK(Track_cutter)){ case (1) : if (MARK(Milling_cutter_2) > 0){ MARK(Milling_cutter_2) − −; MARK(start_Milling_cutter_2_A) + +; MARK(Track_cutter) = 2; }else { MARK(Milling_cutter_1) − −; MARK(start_Milling_cutter_1_A) + +; MARK(Track_cutter) = 1; } break; case (2) : if (MARK(Milling_cutter_1) > 0){ MARK(Milling_cutter_1) − −; MARK(start_Milling_cutter_1_A) + +; MARK(Track_cutter) = 1; }else { MARK(Milling_cutter_2) − −; MARK(start_Milling_cutter_2_A) + +; MARK(Track_cutter) = 2; } break; } MARK(finish_Carrier_A) − −;

Table D.7: SAN Model of Milling Operation Performed on Part B

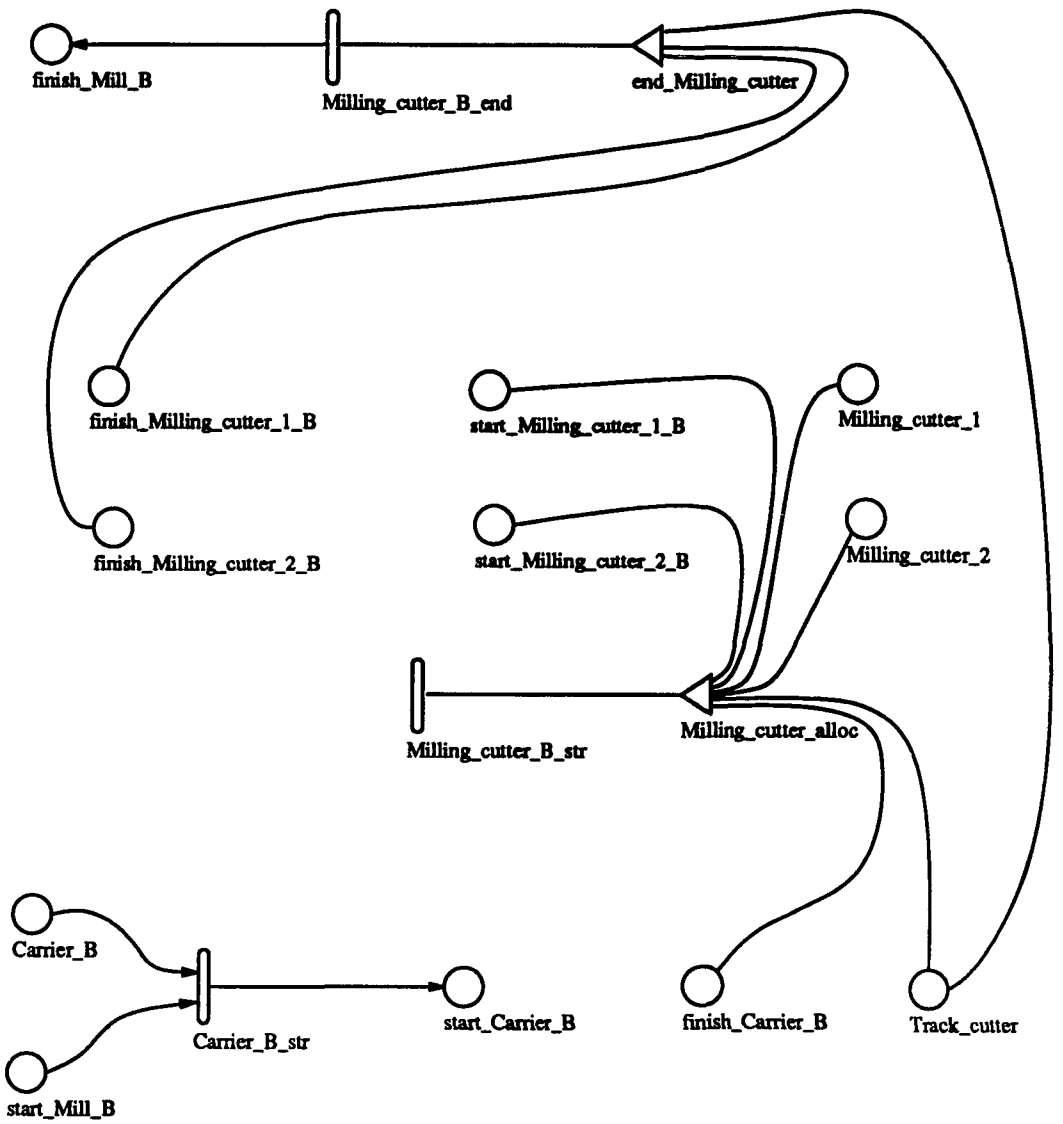


Table D.8: Activity Time Distributions

Activity	Distribution
Milling_Cutter_B_str	Inst
Milling_Cutter_B_end	Inst
Carrier_B_str	USAN_expon(10000000)
Milling_cutter_B_end	USAN_expon(10000000)
Milling_cutter_B_str	USAN_expon(10000000)

Table D.9: Input Gate Predicates and Functions

Gate	Enabling Predicate	Function
end_Milling_cutter	MARK(finish_Milling_cutter_1_B) > 0 || MARK(finish_Milling_cutter_2_B) > 0	switch(MARK(Track_cutter)){ case (1) : if (MARK(finish_Milling_cutter_2_B) > 0) MARK(finish_Milling_cutter_2_B) − −; else MARK(finish_Milling_cutter_1_B) − −; break; case (2) : if (MARK(finish_Milling_cutter_1_B) > 0) MARK(finish_Milling_cutter_1_B) − −; else MARK(finish_Milling_cutter_2_B) − −; break; }
Milling_cutter_alloc	MARK(finish_Carrier_B) > 0 && (MARK(Milling_cutter_1) > 0 || MARK(Milling_cutter_2) > 0)	switch(MARK(Track_cutter)){ case (1) : if (MARK(Milling_cutter_2) > 0){ MARK(Milling_cutter_2) − −; MARK(start_Milling_cutter_2_B) + +; MARK(Track_cutter) = 2; }else { MARK(Milling_cutter_1) − −; MARK(start_Milling_cutter_1_B) + +; MARK(Track_cutter) = 1; } break; case (2) : if (MARK(Milling_cutter_1) > 0){ MARK(Milling_cutter_1) − −; MARK(start_Milling_cutter_1_B) + +; MARK(Track_cutter) = 1; }else { MARK(Milling_cutter_2) − −; MARK(start_Milling_cutter_2_B) + +; MARK(Track_cutter) = 2; } break; } MARK(finish_Carrier_B) − −;

Table D.10: SAN Model of Drilling Operation Performed on Part A

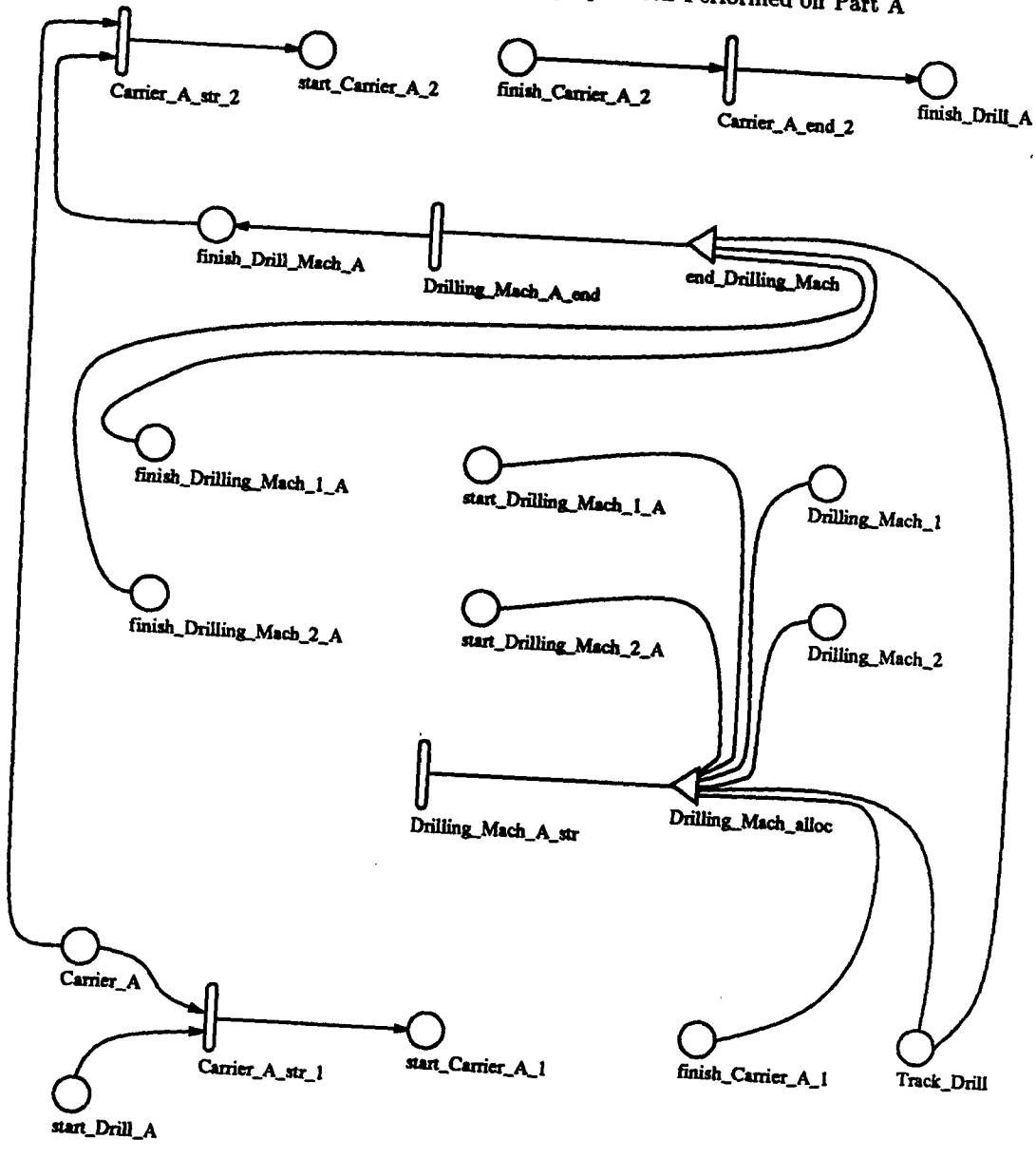


Table D.11: Activity Time Distributions

Activity	Distribution
Carrier_A_snd_3	USAN_expon(10000000)
Carrier_A_str_1	USAN_expon(10000000)
Carrier_A_str_3	USAN_expon(10000000)
Drilling_Mach_A_snd	USAN_expon(10000000)
Drilling_Mach_A_str	USAN_expon(10000000)

Table D.12: Input Gate Predicates and Functions

Gate	Enabling Predicate	Function
Drilling_Mach_alloc	MARK(finish_Carrier_A_1) > 0 && (MARK(Drilling_Mach_1) > 0 || MARK(Drilling_Mach_2) > 0)	switch(MARK(Track_Drill)){ case (1) : if (MARK(Drilling_Mach_2) > 0){ MARK(Drilling_Mach_2) − −; MARK(start_Drilling_Mach_2_A) + +; MARK(Track_Drill) = 2; }else { MARK(Drilling_Mach_1) − −; MARK(start_Drilling_Mach_1_A) + +; MARK(Track_Drill) = 1; } break; case (2) : if (MARK(Drilling_Mach_1) > 0){ MARK(Drilling_Mach_1) − −; MARK(start_Drilling_Mach_1_A) + +; MARK(Track_Drill) = 1; }else { MARK(Drilling_Mach_2) − −; MARK(start_Drilling_Mach_2_A) + +; MARK(Track_Drill) = 2; } break; } MARK(finish_Carrier_A_1) − −;
end_Drilling_Mach	MARK(finish_Drilling_Mach_1_A) > 0 || MARK(finish_Drilling_Mach_2_A) > 0	switch(MARK(Track_Drill)){ case (1) : if (MARK(finish_Drilling_Mach_2_A) > 0) MARK(finish_Drilling_Mach_2_A) − −; else MARK(finish_Drilling_Mach_1_A) − −; break; case (2) : if (MARK(finish_Drilling_Mach_1_A) > 0) MARK(finish_Drilling_Mach_1_A) − −; else MARK(finish_Drilling_Mach_2_A) − −; break; }

Table D.13: SAN Model of Drilling Operation Performed on Part B

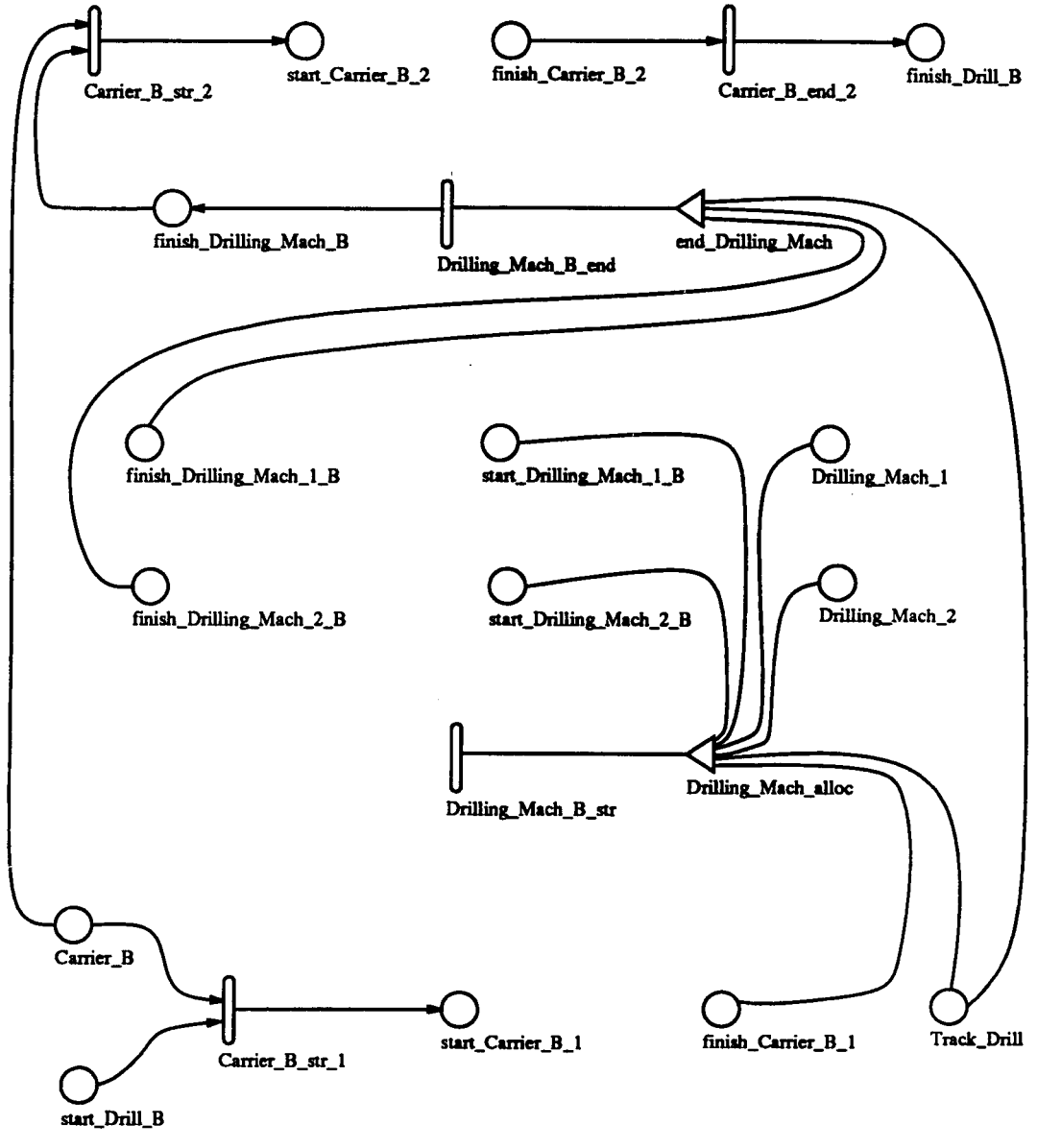


Table D.14: Activity Time Distributions

Activity	Distribution
Carrier_B_snd_2	USAN_expon(10000000)
Carrier_B_str_1	USAN_expon(10000000)
Carrier_B_str_2	USAN_expon(10000000)
Drilling_Mach_B_snd	USAN_expon(10000000)
Drilling_Mach_B_str	USAN_expon(10000000)

Table D.15: Input Gate Predicates and Functions

Gate	Enabling Predicate	Function
Drilling_Mach_alloc	MARK(finish_Carrier_B_1) > 0 && (MARK(Drilling_Mach_1) > 0 || MARK(Drilling_Mach_2) > 0)	switch(MARK(Track_Drill)){ case (1) : if (MARK(Drilling_Mach_2) > 0){ MARK(Drilling_Mach_2) – –; MARK(start_Drilling_Mach_2_B) + +; MARK(Track_Drill) = 2; }else { MARK(Drilling_Mach_1) – –; MARK(start_Drilling_Mach_1_B) + +; MARK(Track_Drill) = 1; } break; case (2) : if (MARK(Drilling_Mach_1) > 0){ MARK(Drilling_Mach_1) – –; MARK(start_Drilling_Mach_1_B) + +; MARK(Track_Drill) = 1; }else { MARK(Drilling_Mach_2) – –; MARK(start_Drilling_Mach_2_B) + +; MARK(Track_Drill) = 2; } break; } MARK(finish_Carrier_B_1) – –;
end_Drilling_Mach	MARK(finish_Drilling_Mach_1_B) > 0 || MARK(finish_Drilling_Mach_2_B) > 0	switch(MARK(Track_Drill)){ case (1) : if (MARK(finish_Drilling_Mach_2_B) > 0) MARK(finish_Drilling_Mach_2_B) – –; else MARK(finish_Drilling_Mach_1_B) – –; break; case (2) : if (MARK(finish_Drilling_Mach_1_B) > 0) MARK(finish_Drilling_Mach_1_B) – –; else MARK(finish_Drilling_Mach_2_B) – –; break; }

Table D.16: SAN Model of Carrier_A

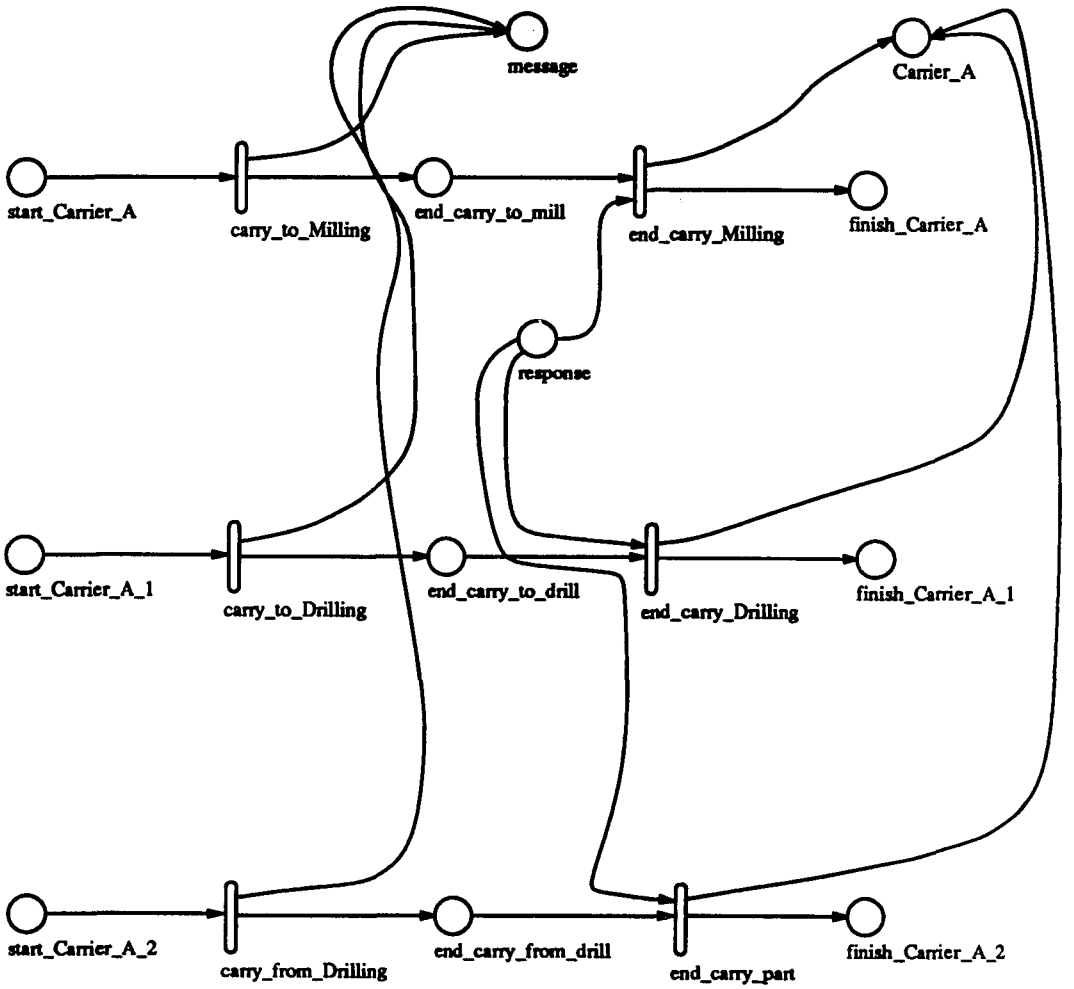


Table D.17: Activity Time Distributions

Activity	Distribution
carry_from_Drilling	USAN_uniform(600,900)
carry_to_Drilling	USAN_uniform(240,360)
carry_to_Milling	USAN_uniform(600,900)
end_carry_Drilling	USAN_expon(10000000)
end_carry_Milling	USAN_expon(10000000)
end_carry_part	USAN_expon(10000000)

Table D.18: SAN Model of Carrier_B

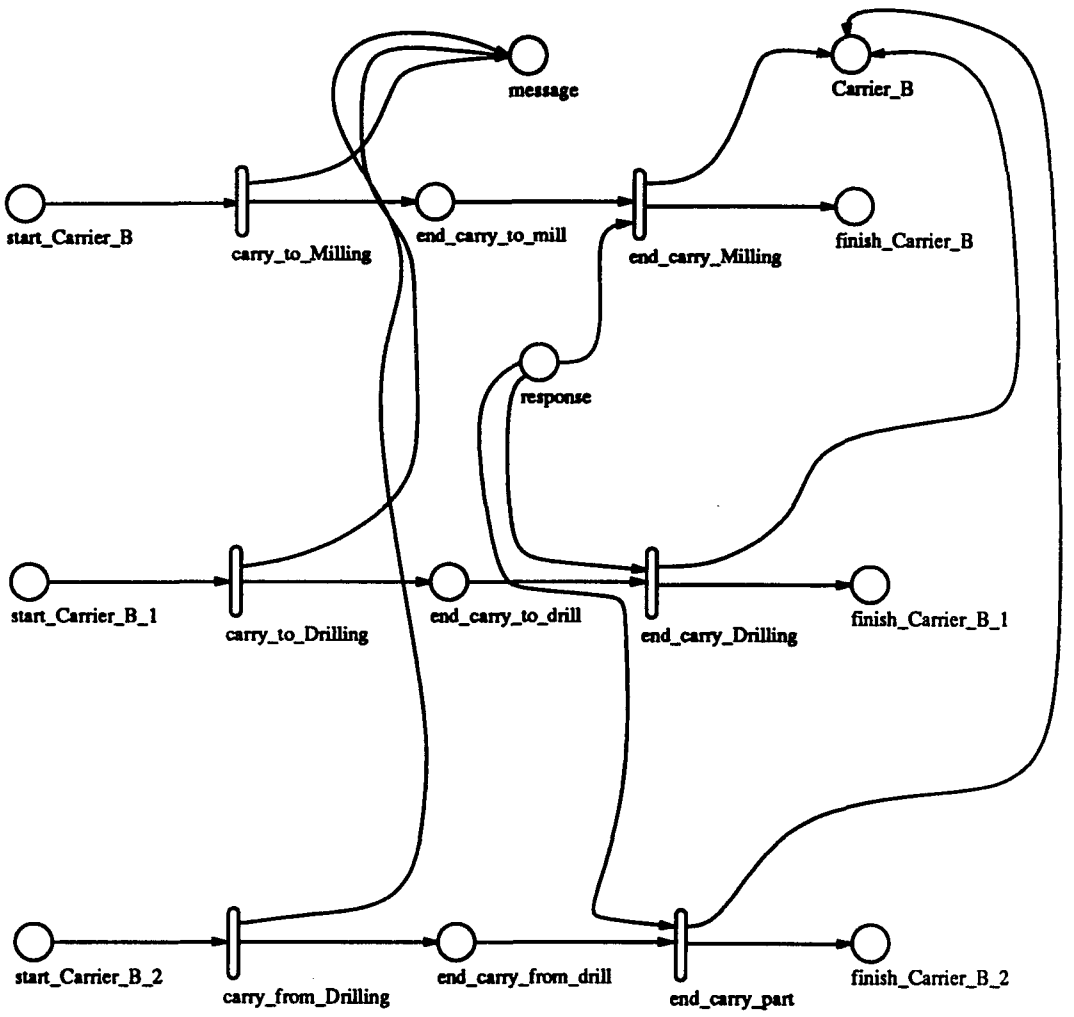


Table D.19: Activity Time Distributions

Activity	Distribution
carry_from_Drilling	USAN_uniform(600,900)
carry_to_Drilling	USAN_uniform(240,360)
carry_to_Milling	USAN_uniform(600,900)
end_carry_Drilling	USAN_expon(10000000)
end_carry_Milling	USAN_expon(10000000)
end_carry_part	USAN_expon(10000000)

Table D.20: SAN Model of Milling_cutter_1

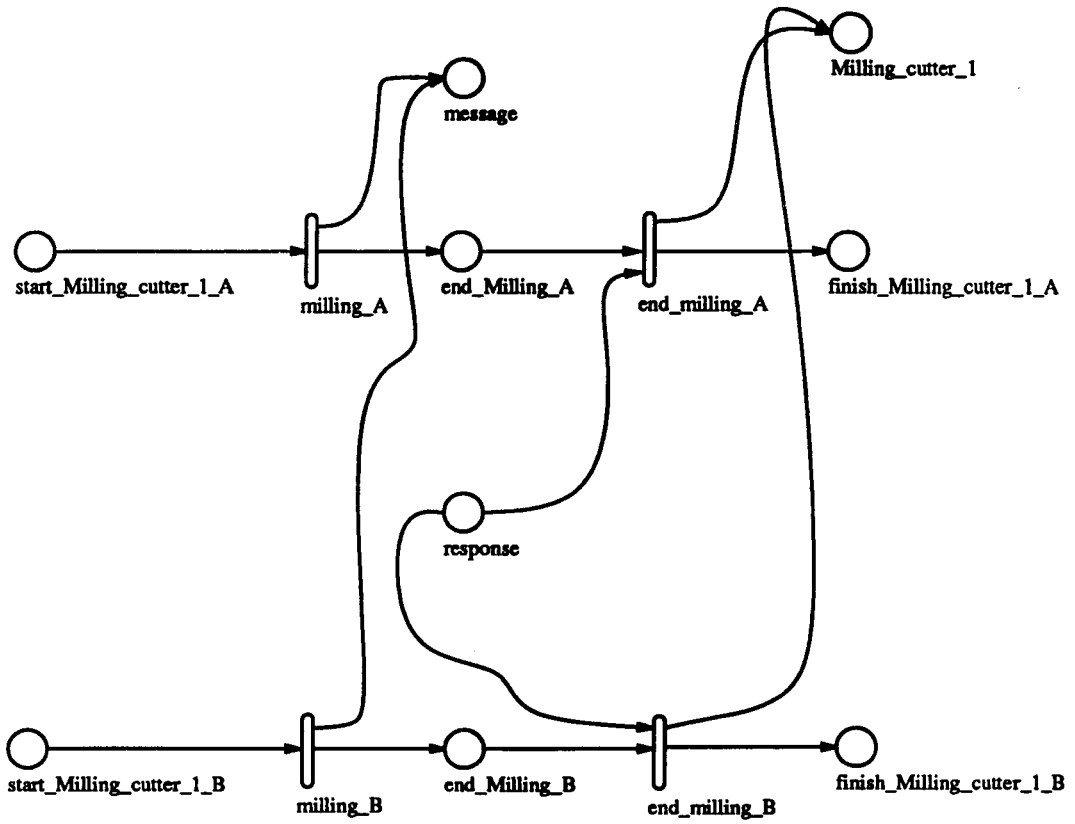


Table D.21: Activity Time Distributions

Activity	Distribution
end_milling_A	USAN_expon(10000000)
end_milling_B	USAN_expon(10000000)
milling_A	USAN_uniform(1800,3600)
milling_B	USAN_uniform(1800,3600)

Table D.22: SAN Model of Milling_cutter_2

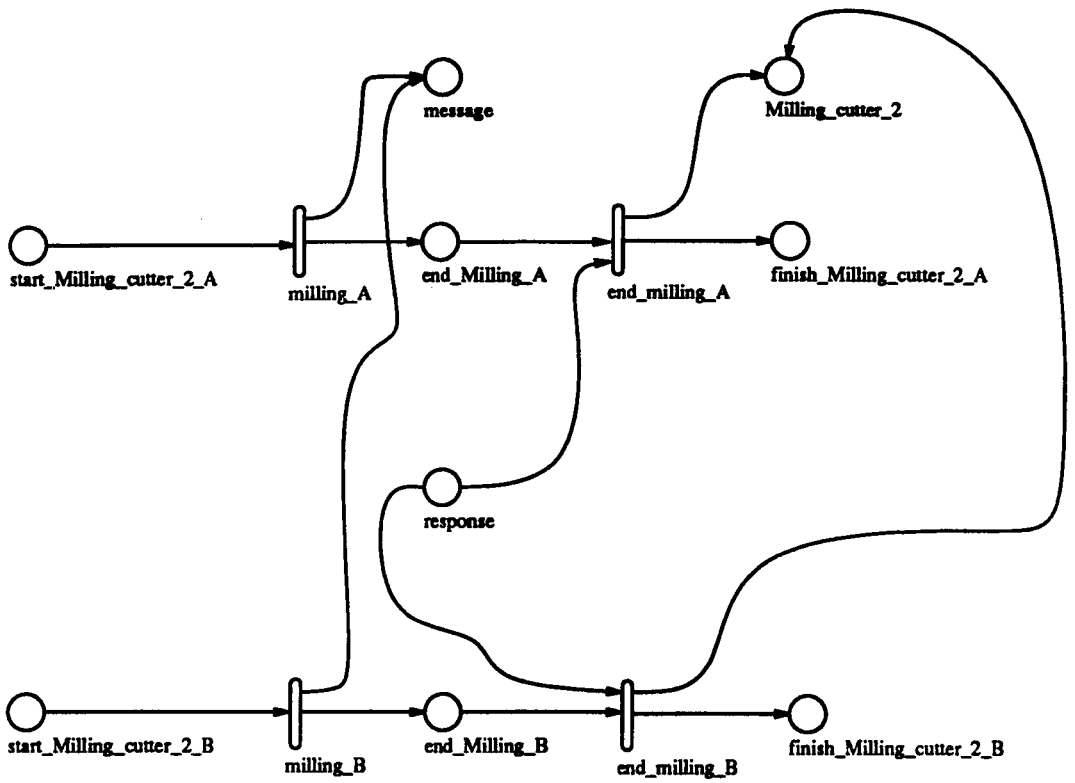


Table D.23: Activity Time Distributions

Activity	Distribution
end_milling_A	USAN_expon(10000000)
end_milling_B	USAN_expon(10000000)
milling_A	USAN_uniform(10000000)
milling_B	USAN_uniform(10000000)

Table D.24: SAN Model of Drilling_Mach_1

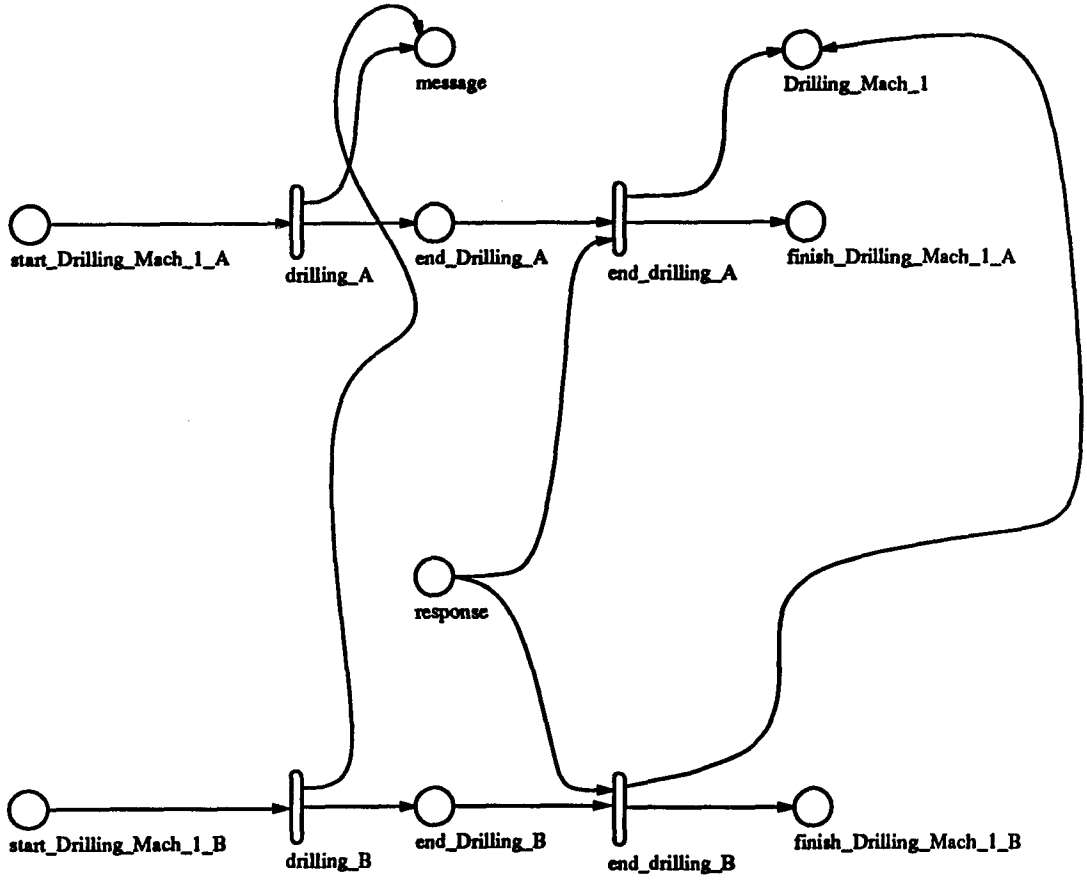


Table D.25: Activity Time Distributions

Activity	Distribution
drilling_A	USAN_uniform(3600,5400)
drilling_B	USAN_uniform(3600,5400)
end_drilling_A	USAN_expon(10000000)
end_drilling_B	USAN_expon(10000000)

Table D.26: SAN Model of Drilling_Mach_2

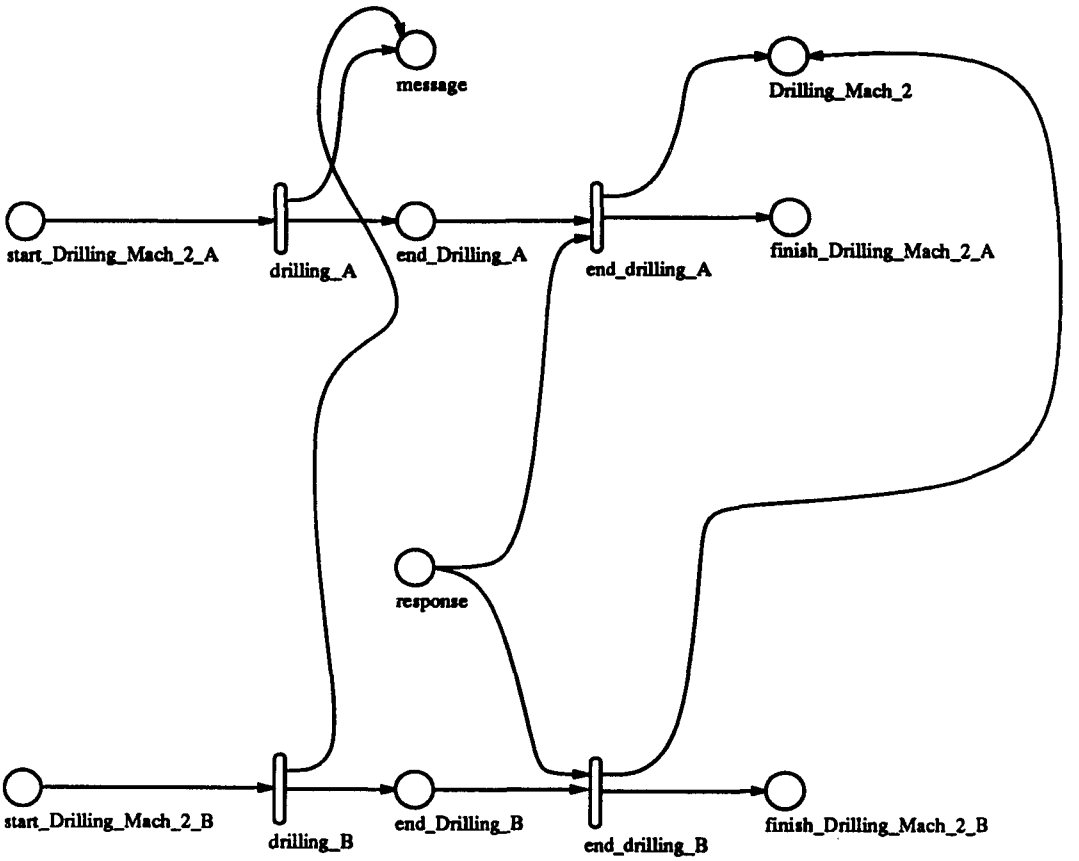


Table D.27: Activity Time Distributions

Activity	Distribution
drilling_A	USAN_uniform(3600,5400)
drilling_B	USAN_uniform(3600,5400)
end_drilling_A	USAN_expon(10000000)
end_drilling_B	USAN_expon(10000000)

Table D.28: SAN Model Representing Failures and Repairs of Machines

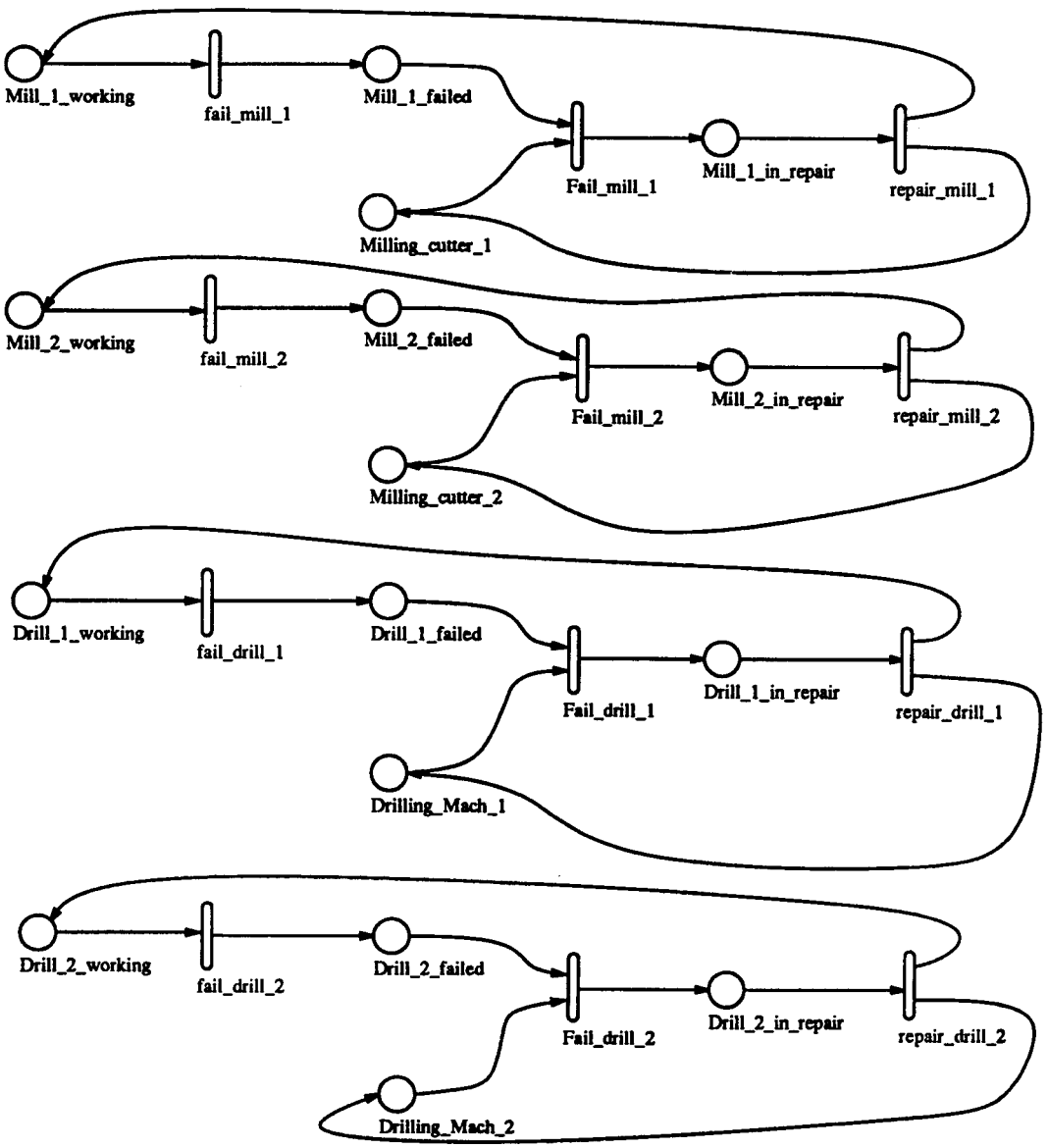


Table D.29: Activity Time Distributions

Activity	Distribution
Fail_drill_1	USAN_expon(10000000)
Fail_drill_2	USAN_expon(10000000)
Fail_mill_1	USAN_expon(10000000)
Fail_mill_2	USAN_expon(10000000)
fail_drill_1	USAN_expon(.0000013889)
fail_drill_2	USAN_expon(.0000013889)
fail_mill_1	USAN_expon(.0000002778)
fail_mill_2	USAN_expon(.0000002778)
repair_drill_1	USAN_expon(.0002777778)
repair_drill_2	USAN_expon(.0002777778)
repair_mill_1	USAN_expon(.0002777778)
repair_mill_2	USAN_expon(.0002777778)

REFERENCES

[1] G. Acaccia, M. Bovone, R. Michelini, R. Molfino and F. Spinosa, "Rule-based Dispatching Govern for Flexible Manufacturing:Example Implementation of a Shop-floor Part-transportation System", *IEEE International Conference on Robotics & Automation*, Vol.1, pp. 558-565, 1987.

[2] R. Al-jaar and A. Desrochers, "Performance Evaluation of Automated Manufacturing Systems Using Generalized Stochastic Petri Nets", *IEEE Transactions on Robotics & Automation*, Vol.6, No.6, pp. 621-639, Dec. 1990.

[3] G. Balbo, S. Bruell and S. Ghanta, "Combining Queueing Network and Generalized Stochastic Petri Net Models for the Analysis of Some Software Blocking Phenomena", *IEEE Transactions on Software Engineering*, vol. SE-12, No.4, pp. 561-576, Apr. 1986.

[4] G. Balbo, G. Chiola, G. Franceschinis and G. Molinar Roet, "Generalized Stochastic Petri Nets for Performance Evaluation of FMS", *IEEE International Conference on Robotics & Automation*, Vol.2, pp. 1013-1018, 1987.

[5] R. Bauman and T. Turano, "Production Based Language Simulation of Petri Nets", *Simulation 47:5*, pp. 191-198, 1986.

[6] C. Beck and B. Krogh, "Models for Simulation and Discrete Control of Manufacturing Systems", *IEEE International Conference on Robotics & Automation*, vol.1, pp. 305-310, 1986.

[7] G. Bruno and M.Morisio, "Petri-Net Based Simulation of Manufacturing Cells", *IEEE International Conference on Robotics & Automation*, Vol.2, pp. 1174-1179, 1987.

[8] G. Bruno, M. Morisio, "The Role of Rule Based Programming for Production Scheduling", *IEEE International Conference on Robotics & Automation*, Vol.1, pp. 545-550, 1987.

[9] A. Camurri and P. Franchi, "An Approach to the Design and Implementation of the Hierarchical Control System of FMS, Combining Structured Knowledge Representation Formalisms and High-Level Petri Nets", *IEEE International conference on Robotics & Automation*, pp. 520-525, 1990.

[10] J. Cavaille and D. Dubois, "Heuristic Methods Based on Mean-value Analysis for Flexible Manufacturing Systems Performance Evaluation", *Proceedings of the 21st IEEE Conference on Decision and Control*, pp. 1061-1065, 1982.

[11] C. Chen, C. Lee and C. McGillem, "Task Assignment and Load Balancing of Autonomous Vehicles in a Flexible Manufacturing System", *IEEE International Conference on Robotics & Automation*, Vol.2, pp. 1033-1039, 1987.

[12] W. Chow, E. Macnair and C. Sauer, "Analysis of Manufacturing Systems by Research Queueing Package", *IBM Journal of Research Developement*, Vol.29, No.4, Jul. 1985.

[13] E. Cinlar, *Introduction to Stochastic Processes*, Englewood Cliffs: Prentice-Hall, 1975.

[14] D. Corbeel, J. Gentina and C. Vercauter, "Application of an Extension of Petri Nets to Modelization of Control and Production Processes", *Advances in Petri Nets*, pp. 162-180, 1985.

[15] Y. Dallery, "On Modelling Flexible Manufacturing Systems Using Closed Queueing Networks", *North-Holland , Large Scale Systems 11(1986)*, pp. 109-119, 1986.

[16] Y. Dotan and D. Ben-Arieh, "Modeling Flexible Manufacturing Systems: The Concurrent Logic Programming Approach", *IEEE International Conference on Robotics & Automation*, Vol. 7 , No. 1, pp. 135-148, Feb. 1991.

[17] D. Dubois, "A Mathematical Model of a Flexible Manufacturing System with a Limited In-process Inventory", *European Journal of Operational Research 14*, pp. 66-78, 1982.

[18] D. Dubois, K. Stecke, "Using Petri Nets to Represent Production Processes", *IEEE Transaction on Software Engineering*, pp. 1062-1067, 1983.

[19] R. Freire, *A Technique for Simulating Composed SAN-based Reward Models*, Master's thesis, Univ. of Arizona, 1990.

[20] J. Gentina and D. Corbeel, "Coloured Adaptive Structured Petri-net : A Tool for the Automatic Synthesis of Hierarchical Control of Flexible Manufacturing Systems (F.M.S)", *IEEE International Conference on Robotics & Automation*, Vol.2 , pp. 1166-1173, 1987.

[21] N. Greenwood, *Implementing Flexible Manufacturing Systems*, Published by John Wiley & Sons, New York, 1988.

[22] M. Hasegawa, M. Takata, T. Temmyo and H. Matsuka, "Modeling of Exception Handling in Manufacturing Cell Control and Its Application to PLC programming", *Proc. of IEEE International Conference on Robotics &Automation*, pp. 514-519, 1990.

[23] R. Hildebrant, "Scheduling Flexible Machining Systems Using Mean Value Analysis", *Proc. of the IEEE Conference on Decision and Control*, pp. 701-706, 1980.

[24] J. Holland, *Flexible Manufacturing Systems*, Published by Society of Manufacturing Engineers, Dearborn, Michigan, 1984.

[25] R. Hurley, P. Coffman, J. Dixon and J. Walacavage, "The Use of Physical Model Simulation to Emulate an AGV Material Handling System", *IEEE International Conference on Robotics & Automation*, Vol.2, pp. 1040-1045, 1987.

[26] M.Jafari, "Petri Net Based Shop Floor Controller and Recovery Analysis", *IEEE International Conference on Robotics & Automation*, pp. 532-537, 1990.

[27] M. Kamath and N. Vishwanadham, "Applications of Petri Net Based Models in the Modeling and Analysis of Flexible Manufacturing Systems", *IEEE International Conference on Robotics & Automation*, vol.1, pp. 312-317, 1986.

[28] E. Kasturia, F. Dicesare and A. Desrochers, "Real Time Control of Multilevel Manufacturing Systems Using Colored Petri Nets", *IEEE International Conference on Robotics & Automation*, Vol.2, pp. 1114-1119, 1988.

[29] L. Kleinrock, *Queueing Systems, Volume I:Theory*, New York: John Wiley, 1975.

[30] J. Martinez, P.Muro and M.Silva, "Modeling, Validation and Software Implementation of Production Systems Using High Level Petri Nets", *IEEE International Conference on Robotics & Automation*, Vol.2, pp. 1180-1185, 1987.

[31] V. Milacic, V. Majstorovic and I. Race, "Building Expert System for Diagnosis and Maintenace in FMS", *IEEE International Conference on Robotics & Automation*, Vol.2, pp. 1126-1129, 1988.

[32] J. L. Peterson, *Petri Net Theory and the Modeling of Systems*, Englewood Cliffs: Prentice-Hall, 1981.

[33] C. Petri, *Kommunikation mit Automaten*, Ph.D. dissertation, University of Bonn, Bonn, West Germany, 1962.

[34] K. H. Prodromides and W. H. Sanders, "Performability Evaluation of CSMA/CD and CSMA/DCR Protocols Under Transient Fault Conditions", *Tenth Symposium on Reliable Distributed Systems*, 1991.

[35] M. Rai, *Design and Implementation of a Reduced Base Model Construction Technique for Stochastic Activity Networks*, Master's thesis, Univ. of Arizona, 1990.

[36] R. Ravichandran and A. Chakravarty, "Decision Support in Flexible Manufacturing Systems Using Timed Petri Nets", *Journal of Manufacturing System*, vol.5, No.2, pp. 89-101, 1986.

[37] A. Sahraoui, H.Atabakhche, M. Courvoisier, R. Valette, "Joining Petri Nets and Knowledge Based Systems for Monitoring Purposes", *IEEE Transaction on Software Engineering*, pp. 1160-1165, 1987.

[38] W. H. Sanders, *Construction and Solution of Performability Models Based on Stochastic Activity Networks*, PhD thesis, Univ of Michigan, 1988.

[39] W. H. Sanders and J. F. Meyer, "Reduced Base Model Construction Methods for Stochastic Activity Networks", *IEEE Journal of Selected Areas of communication*, pp. 25-36, Jan. 1991.

[40] J. Couvillion, R. Freire, R. Johnson, W. D. Obal II, M. A. Qureshi, M. Rai, W. H. Sanders, and J. E. Tvedt, "Performability Modeling with *UltraSAN*", *IEEE software, Special issue on software for performance analysis*, Sept. 1991.

[41] J. G. Shanthikumar and J. A. Buzacott, "Open Queueing Network Models of Dynamic Job Shops", *International Journal of Production Research*, Vol. 19, No. 3, pp. 255-266, 1981.

[42] J. Sifakis, "Use of Petri Nets for Performance Evaluation, Measuring, Modeling and Evaluating Computer Systems", *North-Holland Publishing Company*, 1977.

[43] J. J. Solberg and S. Y. Nof, "Analysis of Flow Control in Alternative Manufacturing Configurations", *Journal of Dynamic Systems, Measurement, and Control*, Vol. 102, pp. 141-147, Sept. 1980.

[44] P. Stotts, R. Newcomb and Z. Cai, "Modelling the Logical Structure of Flexible Manufacturing Systems with Petri-Nets", *Butterworth & Co(Publishers) Ltd.*, Vol.12, No.4, Aug. 1989.

[45] R. Suri and R. R. Hildebrant, "Modelling Flexible Manufacturing Systems Using Mean-Value Analysis", *Journal of Manufacturing Systems*, Vol. 3, No. 1, pp. 27-38, 1984.

[46] A. Villa, "A Hierarchical Knowledge-based/Analytical Approach to Fault-tolerant Control in Flexible Manufacturing", *IEEE International Conference on Robotics & Automation*, Vol.2, pp. 1120-1125, 1988.

[47] Prof. H. J. Warnecke and R. Steinhilper, *Flexible Manufacturing Systems*, IFS(Publications) Ltd, UK, 1985.

[48] H. Wedekind and G. Zoerntlein, "Conceptual Basis for Database applications in Flexible Manufacturing Systems (FMS)", *IEEE International Conference on Robotics & Automation*, Vol.1, pp. 551-557, 1987.

[49] M. Zhou, F. DiCesare and A. Desrochers, "A Top-down Approach to Systematic Synthesis of Petri Net Models for Manufacturing Systems", *IEEE International conference on Robotics & Automation*, Vol.1, pp. 534-539, 1989.

[50] M. Zhou and F.DiCesare, "A Petri Net Design Method for Automated Manufacturing Systems with Shared Resources", *Proc. of IEEE Robotics and Automation conferences*, pp. 526-531, 1990.