Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be "used for any purpose other than private study, scholarship, or research." If a, user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of "fair use" that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation

Printing note: If you do not wish to print this page, then select "Pages from: first page # to: last page #" on the print dialog screen

ABSTRACT

Modeling, Evaluation, and Control of a Flexible Manufacturing Cell Using Petri Nets

by
Glenn Arthur Thorniley

This thesis describes the usefulness of Petri nets in modeling, evaluating, and controlling a Flexible Manufacturing Cell (FMC). The basics of Petri net theory are explained and a specific FMC is examined. First, the FMC is modeled. The purpose of modeling is to facilitate the evaluation and provide a framework on which the control methodologies can be applied. The objective of the evaluation is to determine how the FMC would benefit most by replacing or adding equipment. Several ideas on control are combined to form a useful framework for the designing of the control net. With this framework the control net is developed directly from the Petri net used in the modeling and evaluation phases. Through the use of special symbols incorporated into the control net, the basic input and output requirements of the system can be derived from the graphical control net.

MODELING, EVALUATION, AND CONTROL OF A FLEXIBLE MANUFACTURING CELL USING PETRI NETS

by
Glenn Arthur Thorniley

A Thesis
Submitted to the Faculty of New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Master of Science
Department of Manufacturing Engineering
October, 1992

APPROVAL PAGE
Modeling, Evaluation, and Control of a Flexible Manufacturing Cell Using Petri Nets

by
Glenn Arthur Thorniley

8/4/92

Dr. MengChu Zhou, Thesis Adviser
Assistant Professor
Department of Electrical and Computer Engineering
Manufacturing Engineering Program, NJIT

8. 4. 92

Dr. Raj Sodhi, Committee Member
Director of Manufacturing Engineering Program
Department of Mechanical and Industrial Engineering, NJIT

Aug. 4. 1992

Dr. Xiuli Chao, Committee Member
Assistant Professor
Department of Mechanical and Industrial Engineering, NJIT

BIOGRAPHICAL SKETCH

Author: Glenn Arthur Thorniley

Degree: Master of Science in Manufacturing Engineering

Date: October, 1992

Date of Birth:

Place of Birth:

Undergraduate and Graduate Education:

- Master of Science in Manufacturing Engineering, New Jersey Institute of Technology, Newark, New Jersey, 1992.
- Bachelor of Science in Mechanical Engineering, The Pennsylvania State University, University Park, Pennsylvania, 1986.

Major: Manufacturing Engineering

This thesis is dedicated to
my mother,
Grace J. Thorniley

ACKNOWLEDGMENT

The author wishes to thank his thesis adviser, Dr. MengChu Zhou, for his outstanding support throughout this research. Dr. Zhou's willingness to answer questions and his ability to explain the subject matter are exceptional.

In addition, thanks go to Dr. Raj Sodhi and Dr. Xiuli Chao for their assistance in serving as committee members for this thesis.

A special thank you goes to the author's wife, Denise, whose patience and support were crucial to the completion of this thesis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 INTRODUCTION TO PETRI NETS

1.1 History

The fundamental idea on which current Petri net (PN) theory has been built was originated in 1962 by Carl Adam Petri at the University of Darmstadt, Germany. Petri's work was soon recognized by A.W. Holt who piloted the Information System Theory Project of Applied Data Research, Inc., in the United States. During the 1970's, MIT had conducted research through their Computational Structure Group. The Europeans became more involved in the early 1980's by arranging workshops and publishing conference proceedings. By the later part of the 1980's, Petri net research was being conducted on a broader spectrum including Japan and Australia in addition to most European countries and the United States [17]. Research is currently being conducted at many universities including, New Jersey Institute of Technology, University of Illinois, Rensselaer Polytechnic Institute, Carnegie-Mellon University, and Duke University.

1.2 Why Use Petri Nets?

Petri Nets are a simple, powerful, logical tool which enables the engineer to model, evaluate, and control complex dynamic discrete event systems. Petri net methodology utilizes both a mathematical and a graphical model. A graphical representation of a Petri net can be constructed in a building block type manner where each newly revised *net* is developed by further detailing a previous version. After the graphical model has been completed, a mathematical model can be written systematically. These models are

used to evaluate the properties and characteristics of the system under investigation. The graphical model can even be animated with computer software to reflect real-time activities or simulated activity. Although this paper addresses the use of PN's in flexible manufacturing, further development of Petri net theory would be beneficial in other areas as well. Petri nets can be utilized to represent a variety of dynamic systems including manufacturing, computer network, communication, and traffic systems. Petri nets are also adept at modeling *Just-in-Time* manufacturing systems [21]. They are particularly useful in modeling flexible manufacturing systems due to their ability to model: asynchronous operations, concurrence, deadlock, conflicting events, and event driven systems [12]. In addition, Petri nets have the ability to model all states of a given system with a single model using different initial markings or conditions, whereas *Finite State Machines* need often be changed drastically [12]. Unlike many other modeling techniques, PN's can be used from design through evaluation to control. This makes it possible to compile the net into control code or data for implementation and execution on the shop floor [26]. As the different entities of manufacturing systems become more and more integrated, Petri nets could become the universal language of manufacturing systems. The remainder of this paper pertains to the application of Petri nets in Flexible Manufacturing.

2 BASIC PETRI NET THEORY

2.1 The Graphical Model

The graphical model is constructed with *places* (circles), *transitions* (short lines or boxes), *arcs* (arrows) and *tokens* (dots). These symbols are illustrated in Figure 1. Although these four symbols seem elementary, they can be combined to model an infinite variety of dynamic behavior.

Figure 1 - Petri Net Symbols

Places and transitions are connected alternately with arcs to form a directed graph or net. Tokens flow from place to place via transitions. Tokens may only follow paths designated by the arcs and only in the specified direction. Places can represent operations, buffers, or resources. Tokens, when occupying a place, signify a true condition. For example, a place can represent the availability of a machine (resource). If that place contains a token then the machine (resource) is available. Conversely, the absence of a token in that resource place signifies that the particular resource is not available. Transitions generally designate the beginning or the completion of an event. For example, a transition could represent the beginning of a machining process. Arcs connect places to transitions and vice versa. These arcs designate the logical paths in which the tokens can follow. An arc from a place to a transition is said to be an *input*

arc for that transition. An arc from a transition to a place is called an *output arc*. In addition, arcs are assigned weights. *Arc weights* are assumed to be one unless they are labeled otherwise. If an input arc has a weight of two, then two tokens are required before the corresponding transition is *enabled*. The set of input places or *pre-set* for a transition is designated as ${}^{\bullet}t_j$ and a set of output places or *post-set* is designated as $t_j^{\bullet}$. A transition is said to be enabled if all of its input places contain the required number of tokens. An enabled transition may *fire*. This firing removes tokens (number determined by the input arc weight) from the input places and generates tokens (number determined by the output arc weight) to the output places. A detailed explanation of this *transition firing rule* can be found in [17]. For a continuous process, such as a machining operation, a *time delay* (τ) is associated with each transition. The time delay is an amount of time that elapses from the instant a transition is enabled until the instant it fires. For example, if an input place to a certain transition represents a machining operation, then the transition designates the completion of the machining cycle and the corresponding time delay is the machining or processing time. Due to the stochastic nature of most real-life processes, the process times are usually random and exponentially distributed. If the time delay is random and exponentially distributed, the resulting *firing rate*, λ, is equal to the inverse of the expected time delay, $E(\tau)$.

$$\lambda = {}^{1}/E(\tau) \qquad \text{where } \tau \text{ is exponentially distributed.} \qquad (1)$$

Finally, the *initial marking*, m_0, is the initial set of tokens including their quantities and locations at system start-up.

A Petri net model of a Flexible Manufacturing Cell (FMC) is illustrated in Figure 2A. This FMC contains a horizontal machining center. One unit of raw material is machined to produce two finished parts. Once the machining operation has been completed and the parts are unloaded, the machine becomes available. The completion of the process triggers the release of more raw material. In Figure 2A, place p_1 along with its initial marking of one token signifies that there is one unit of raw material which is waiting to be machined.

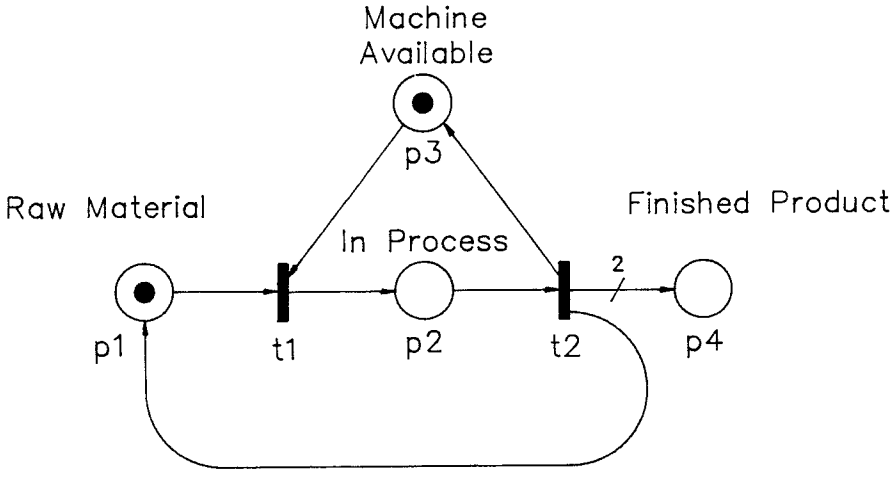

Figure 2A - Petri Net of a Machining Operation

The availability of the machine is monitored through p_3. When p_3 contains a token, the machine is available. Transition t_1 represents the start of the machining process. The machining process can begin if and only if the machine and the raw material are both available. Since p_1 and p_3 have tokens, t_1 is enabled. Firing t_1 results in the net shown in Figure 2B. Here tokens are removed from the input places of t_1, namely p_1 and p_3, and a token is generated in the output place of t_1, namely p_2. At this point, t_2 is

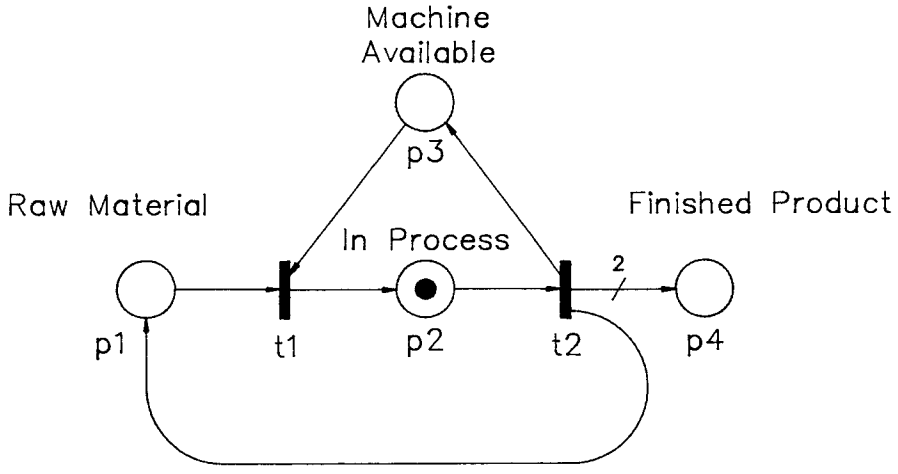

Figure 2B - After t_1 Fires

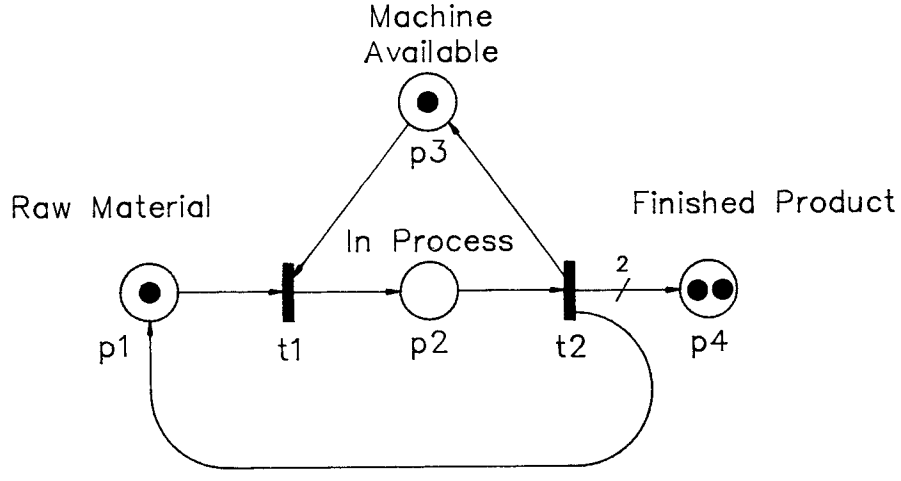

Figure 2C - After t_2 Fires

enabled. The time delay, τ, of t_2 corresponds to the machining or process time. After this delay time, t_2 fires resulting in the net shown in Figure 2C. Notice that p_4 now has two tokens due to the output arc weight of t_2. This symbolizes that two finished products are machined from one unit of raw material. Also notice that p_3 again has a

token meaning that the machine is available again. Finally the firing of t_2 also releases more raw material, represented in p_1. References [17] and [12] offer a detailed description of Petri net theory.

2.2 The Mathematical Model

Input Matrix I

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 0 \end{bmatrix}$$

Output Matrix O

$$O = \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ 0 & 2 \end{bmatrix}$$

Incidence Matrix C

$$C = O - I$$

$$C = \begin{bmatrix} -1 & 1 \\ 1 & -1 \\ -1 & 1 \\ 0 & 2 \end{bmatrix}$$

Figure 3 - Input, Output, and Incidence Matrices

Once the graphical model has been constructed, the mathematical model can be developed. This model consists of several matrices. The *input matrix*, I, derived from the Petri net of Figure 2A can be seen in Figure 3. The input matrix is an (s x n) matrix where s is the number of places and n is the number of transitions. In other

words, the row number corresponds to the place number and the column number corresponds to the transition number. The numbers in the input matrix correspond to the arc weight of the input arc from a place (row) to a transition (column). For instance, the upper left member of the input matrix of Figure 3 designates that there is an input arc from p_1 to t_1 with an arc weight of one. Similarly, the *output matrix*, O, which is also an (s x n) matrix, is constructed from the output arcs and is also shown in Figure 3. The *Incidence Matrix*, C, describes the dynamic characteristics of the system and is equal to the difference between the output and input matrices, O - I. This matrix represents the change from input to output. The incidence matrix for the Petri net in Figure 2 is also illustrated in Figure 3. The "2" in the fourth row, second column of this matrix signifies that two tokens are produced in p_4 upon firing t_2. Negative numbers in the incidence matrix signify the consumption of tokens. This is shown by the -1 in row 2, column 2, where a token of p_2 (row 2) is consumed upon firing t_2 (column 2). The state or *marking* of a Petri net denotes the amount of tokens occupying each place and is captured in a one dimensional matrix of size *s*. The *initial marking*, m_0, of the PN of Figure 2A can be seen in Equation (2).

$$m_0 = [1\ 0\ 1\ 0] \qquad (2)$$

The above matrices are used to fully define a Petri net, Z, as shown in Equation (3) where P and T are the sets of places and transitions in the net, respectively. A Petri net is represented as a five-tuple:

$$Z = (P,T,I,O,m_0) \qquad (3)$$

2.3 Behavioral Properties

Properties of a PN which depend on the initial marking are called *behavioral properties*. From a manufacturing standpoint behavioral properties depend on the conditions at system start-up. The most useful behavioral properties for manufacturing applications are reachability, boundedness, liveness, reversibility, and persistence [17].

Reachability - The reachability set $R(m_0)$ of a Petri net, is defined as the set of all markings (states) which are obtainable from the initial marking, m_0, through some firing sequence $L(m_0)$ [17].

Boundedness - A PN is said to be k-bounded if for any reachable marking, m, none of the places contain more than k tokens. A 1-bounded PN is called *safe* [17]. No place in a safe net will ever contain more than one token.

Liveness - There are five degrees of liveness, L0 - L4. They are defined with respect to a single transition, t, as follows [17]:

- L0 "dead" if t can never be fired in any sequence, $L(m_0)$.
- L1 "potentially firable" if t can be fired at least once in some firing sequence $L(m_0)$.
- L2 given any positive integer n, t can be fired at least n times in some firing sequence $L(m_0)$.
- L3 if t appears infinitely often in some firing sequence, $L(m_0)$.
- L4 "live" if t is at least L1-live for every marking, m, in $R(m_0)$.

The entire PN is said to be "live" if all of the transitions in the net are L4-live. The liveness of a PN determines how prone the net is to reaching a deadlocked state. A live PN is one which cannot be deadlocked.

Reversibility - A reversible PN is one that can always return to a *home state* via some firing sequence in $L(m_n)$. The home state is usually, but not necessarily, the initial state [17].

Persistence - A PN is considered to be persistent if the firing of any enabled transition does not disable another previously enabled transition [17].

According to the above definitions the following observations can be made of Figure 2. The markings shown in Figures 2B and 2C are in the reachability set, $R(m_{2A})$, where m_{2A} is the initial marking shown in Figure 2A.

$$R(m_{2A}) = \{m_{2B}, m_{2C}, \ldots\} \qquad (4)$$

The firing sequence to reach the marking of Figure 2C (m_{2C}) from the marking of Figure 2A, (m_{2A}) is:

$$L(m_{2A}) = \{t_1, t_2\} \qquad (5)$$

This net is unbounded (not safe) since p_4 will continually accumulate tokens. It is live since both transitions are L4-live. It is not reversible since there is no firing sequence in which m_{2A} can be reached from m_{2C}. This PN is persistent because t_1 and t_2 cannot be enabled simultaneously; therefore, no conflicts will arise. A bounded, live, reversible net is desirable in the FMS context. A bounded net guarantees that system will not produce product uncontrollably. A reversible system captures the cyclic character of a FMS. Lastly, liveness insures the system will not deadlock [26].

2.4 Structural Properties

Properties which capture characteristics of a Petri net and are independent of the initial marking (state) are called *structural properties*.

Structural Liveness - If there exists a live initial marking, a Petri net is considered to be structurally live [17].

Controllability - If every marking is in the reachability set of any other feasible marking then a Petri net is completely controllable [17].

Structural Boundedness - A Petri net is said to be structurally bounded if it is bounded for any finite initial marking m_0 [17].

Conservativeness - A Petri net is said to be (partially) conservative if there exists a positive integer y(p) for every (some) place, p, such that the weighted sum of tokens, $m^T y = m_0^T y$ = a constant for every $m \epsilon R(m_0)$ and for any fixed marking m_0 [17].

Repetitiveness - A Petri net is said to be (partially) repetitive if there exists a marking m_0 and a firing sequence $L(m_0)$ such that every (some) transition fires infinitely often in $L(m_0)$ [17].

Consistency - A Petri net is said to be (partially) consistent if there exists a marking m_0 and a firing sequence $L(m_0)$ returning to m_0 such that every (some) transition appears at least once in $L(m_0)$ [17].

Some other important characteristics of the PN are described by *p-invariants* and *t-invariants*. A p-invariant is a subset of places which always share a certain number

of tokens regardless of the firing order, given the initial marking. A t-invariant indicates a firing sequence through which a Petri net will return to its current marking or state.

2.5 Classical Manufacturing Measures

In addition to behavioral and structural properties, typical quantitative concerns such as throughput, utilization, down time, work-in-process, and lead time can be determined from the Petri net model. The effects of changes in resources, process times, and buffer sizes on these performance measures can be determined easily by making the appropriate changes in the model and analyzing the results. This can be done with the aid of computer software such as SPNP, GRAMAN, SIMAN, and SLAM.

3 MODELING A FLEXIBLE MANUFACTURING CELL

3.1 Problem Statement

As previously mentioned, Petri nets can be used to model and evaluate Flexible Manufacturing Systems. In this chapter a specific Flexible Manufacturing Cell (FMC) will be modeled. The purpose of modeling is to facilitate the evaluation and provide a framework on which the control methodologies can be applied. The objective of the evaluation (Chapter 4) is to determine how the FMC will benefit most as a result of upgrading some of the existing equipment. The capital reserved for this improvement is only enough to replace or add two major pieces of equipment. Candidates for upgrading include three CNC machines and an inspection station. An additional robot could be added to assume half of the load of an existing robot. In Chapter 5 process control via Petri nets will be investigated and applied to operate the cell effectively and efficiently.

3.2 Understanding the FMC

Before a system can be modeled, the builder should have a thorough understanding of the systems dynamic behavior, a drawing of the cell layout, and a collection of time delays for any events which do not occur instantaneously. These time delays include process times, loading and unloading times, and inspection times. The drawing of the cell layout (Figure 4), shows the location of the equipment in the FMC. Machine C is a CNC machining center and machines A and B are CNC lathes. Parts #1, #2, and #3 must all be machined and inspected within the cell. Raw material #1 must be

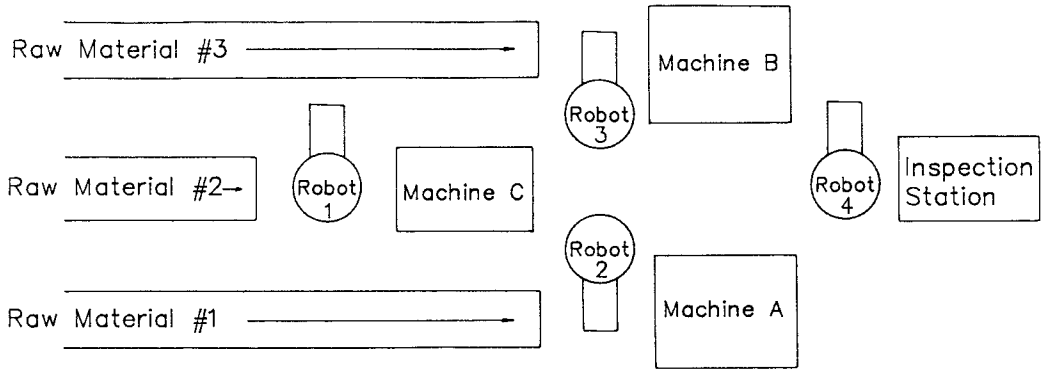

Figure 4 - Flexible Manufacturing Cell Layout

processed on machine A and inspected. Raw material #2 must be processed by machine C, followed by machine A or machine B and then inspected. Raw material #3 must be processed on machine B and inspected. The robot assignments are as follows: Robot #1 loads machine C. Robot #2 transfers part #2 from machine C to machines A or B. Robot #3, if it were added, would transfer part #2 from machine C to machine B, thus relieving robot #2 to transfer from machine C to machine A, exclusively. The routings for each part along with the current and upgraded process times are contained in Table 1. The current process times are those obtained with the presently used equipment. The upgraded process times are those which would be obtained should the decision be made to upgrade the corresponding machine or operation. The decision of whether or not to upgrade certain equipment will be evaluated in Section 4.5.

Table 1 - Current and Upgraded Process Times

Part #	Process	Current Process Time (min.)	Upgraded Process Time (min.)
1	1- Load w/Robot 2	.500	-
	2- Machine A	7.500	2.500
	3- Unload w/Robot 2	.700	-
	4- Inspection	.750	.075
2	1- Load w/Robot 2	.500	-
	2- Machine C	30.000	10.000
	3- Transfer w/Robot 2	.800	-
	4- Machine A or B	4.500	1.500
	5- Unload w/Robot 3	.700	-
	6- Inspection	.750	.075
3	1- Load w/ Robot 2	.500	-
	2- Machine B	13.200	4.400
	3- Unload w/ Robot 3	.700	-
	4- Inspection	.750	.075

3.3 Building a Graphical Model

Several methods have been developed to aid in the construction of complex nets. The *Bottom-Up Method* or *Modular Approach* designs individual nets for the smaller components first, then combines these components into subsystems and finally, integrates the subsystems into a complete model. For example, a net representing a machine (such as the net in Figure 2A) would be an component. Next, this machine along with other components would be combined into a work cell or subsystem. Finally the subsystems or *subnets* are integrated to form a final net of the entire system. As will be discussed in section 4.4, having subnets can greatly reduce the overwhelming computational burden of evaluating the entire system simultaneously. Analogously the hierarchical structure of a FMS is reflected through subnets. The *Top-*

Down Method, as the name suggests, starts with a simplified net of the overall system and works towards a complete net by adding detail and complexity with each refinement. A combination of these two methods is called the *Hybrid Method.*

The Hybrid Method will be used to construct a Petri net for the FMC of Figure 4. First the Top-Down Method is used. Here the entire cell is viewed as a single place and transition. This pair is built upon by expanding into three parallel sections representing the three parts. The section for each part includes a *raw material* place, and an *in-process* place(s). These first two steps are illustrated in Figure 5A and Figure 5B, respectively.

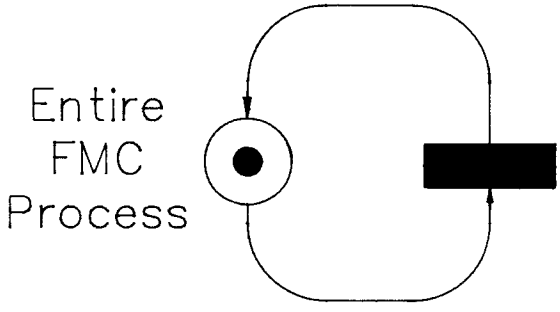

Figure 5A - Hybrid Method, Step 1

Notice that the alternate routing through machines A or B for part #2 is reflected in this structure. In Figure 5C the net is expanded again by adding places representing inspection availability and buffers. In Figure 5D, places for loading and unloading each machine are added along with post-inspection buffers. At this point the skeletal structure of the system is intact. The Bottom-Up Method is now used to add resource places (Figure 5E). These include robot, machine, and inspection station availabilities. This leads to the creation of one and only one place for each machine, robot, and inspection station. These resource places are connected, via arcs, to the functions in

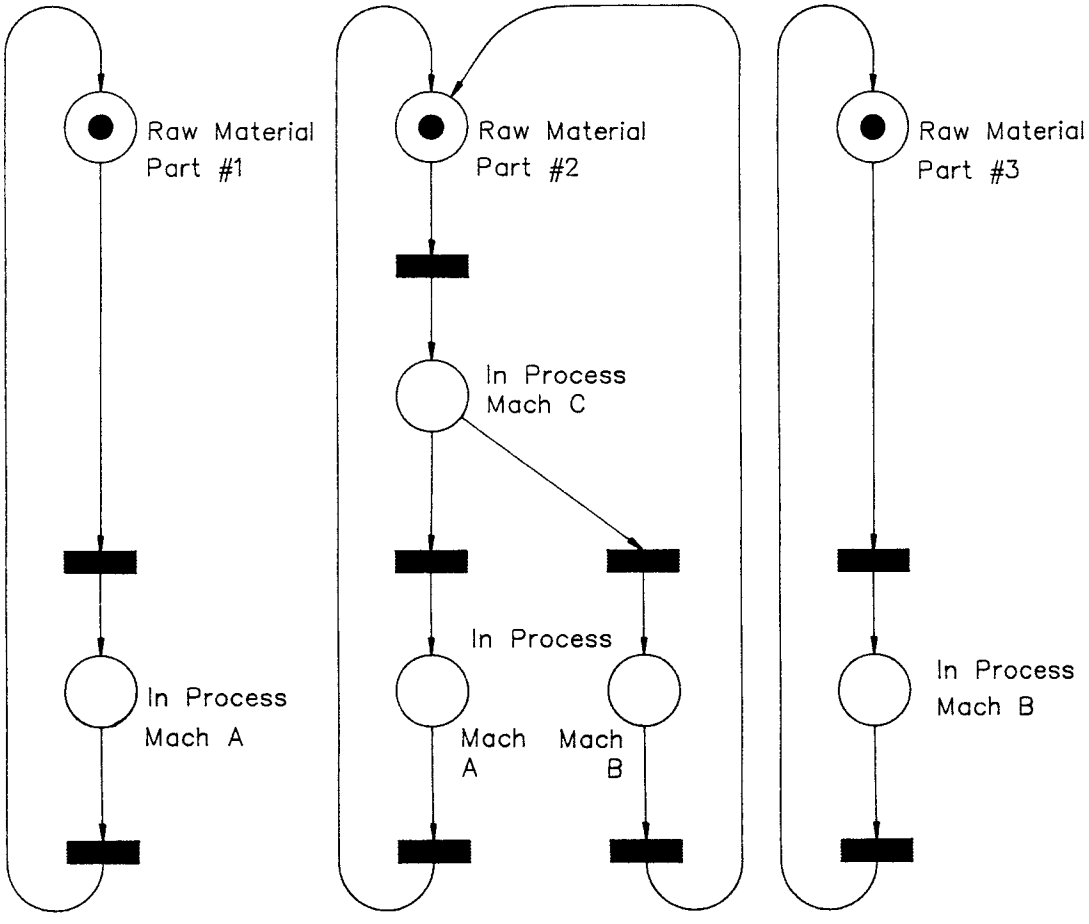

Figure 5B - Hybrid Method, Step 2

which they are required for each part. As seen in Figure 5E, a part cannot be loaded onto a machine unless the required robot and machine are available. In other words, both resource places must contain tokens. After the machine is loaded the robot becomes available again (a token is generated in the corresponding robot resource place). However, the machine is still in use; therefore, it does not become available, i.e. a token is not generated in its resource place, until the process is completed and the machine is unloaded. Similarly, a part cannot be inspected unless the inspection station is available. Notice that there are two arcs exiting each inspection place. These correspond to the acceptance or rejection of the part. If the part is rejected, more raw

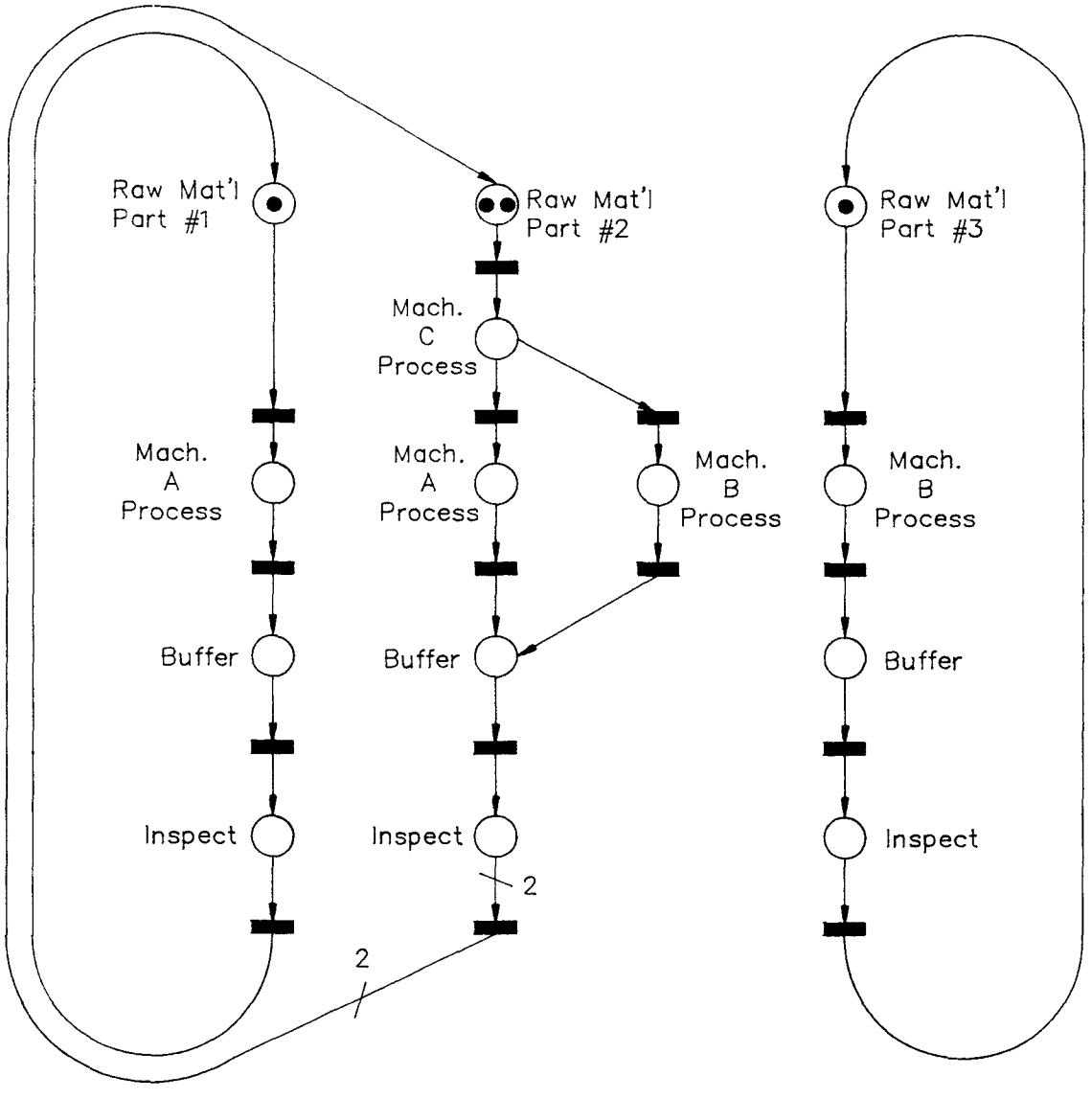

Figure 5C - Hybrid Method, Step 3

material is immediately released to replenish the scrapped part. If a part passes the inspection, a token is generated in the acceptance buffer. After the Petri net is completed, all places and transitions are numbered. The complete Petri net is shown in Figure 5E. Notice there is exactly one place for each resource, i.e. robots, machines, inspection, or raw materials. Other places represent the status of operations such as loading, transferring, processing, or process complete. All transitions represent

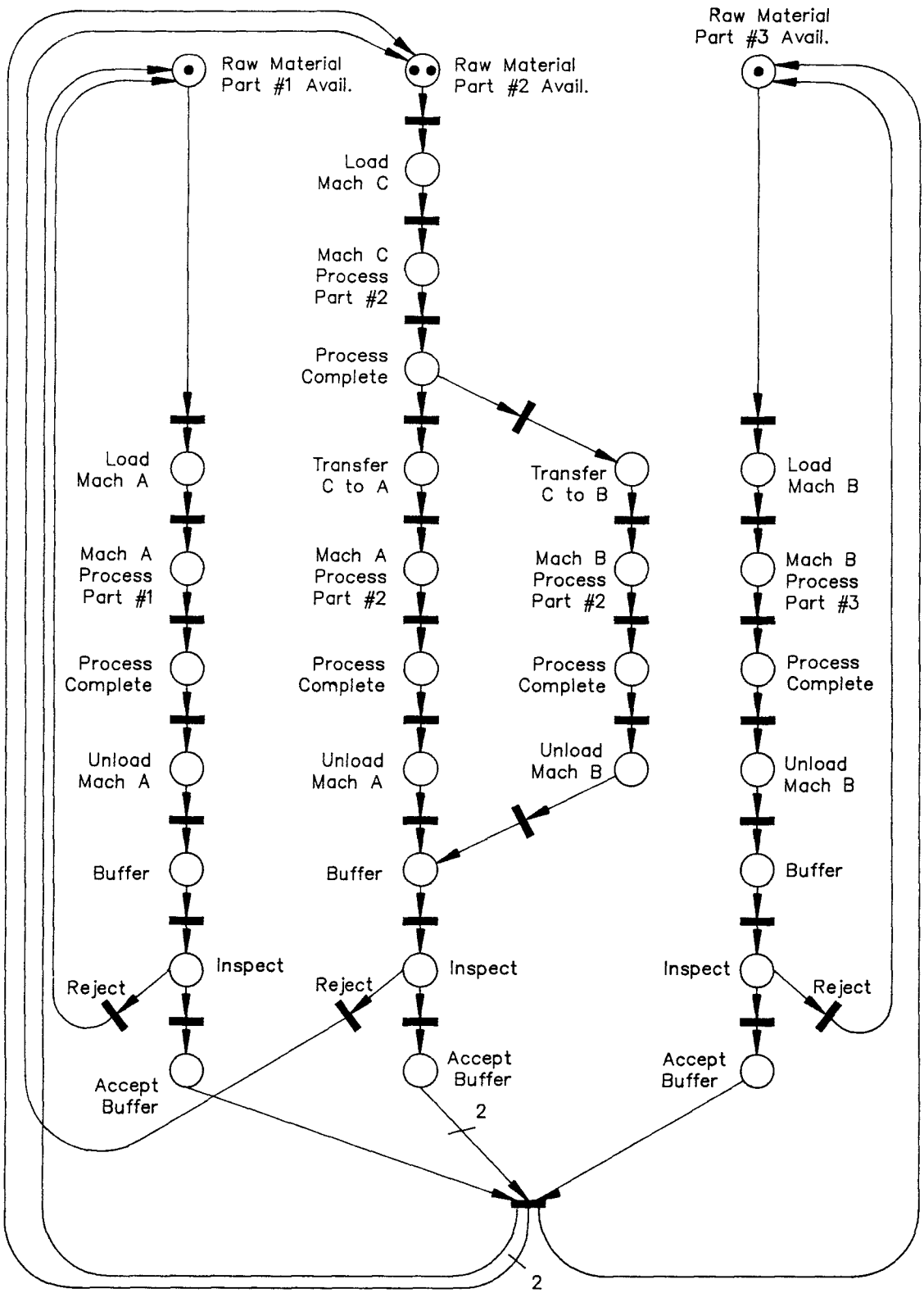

Figure 5D - Hybrid Method, Step 4

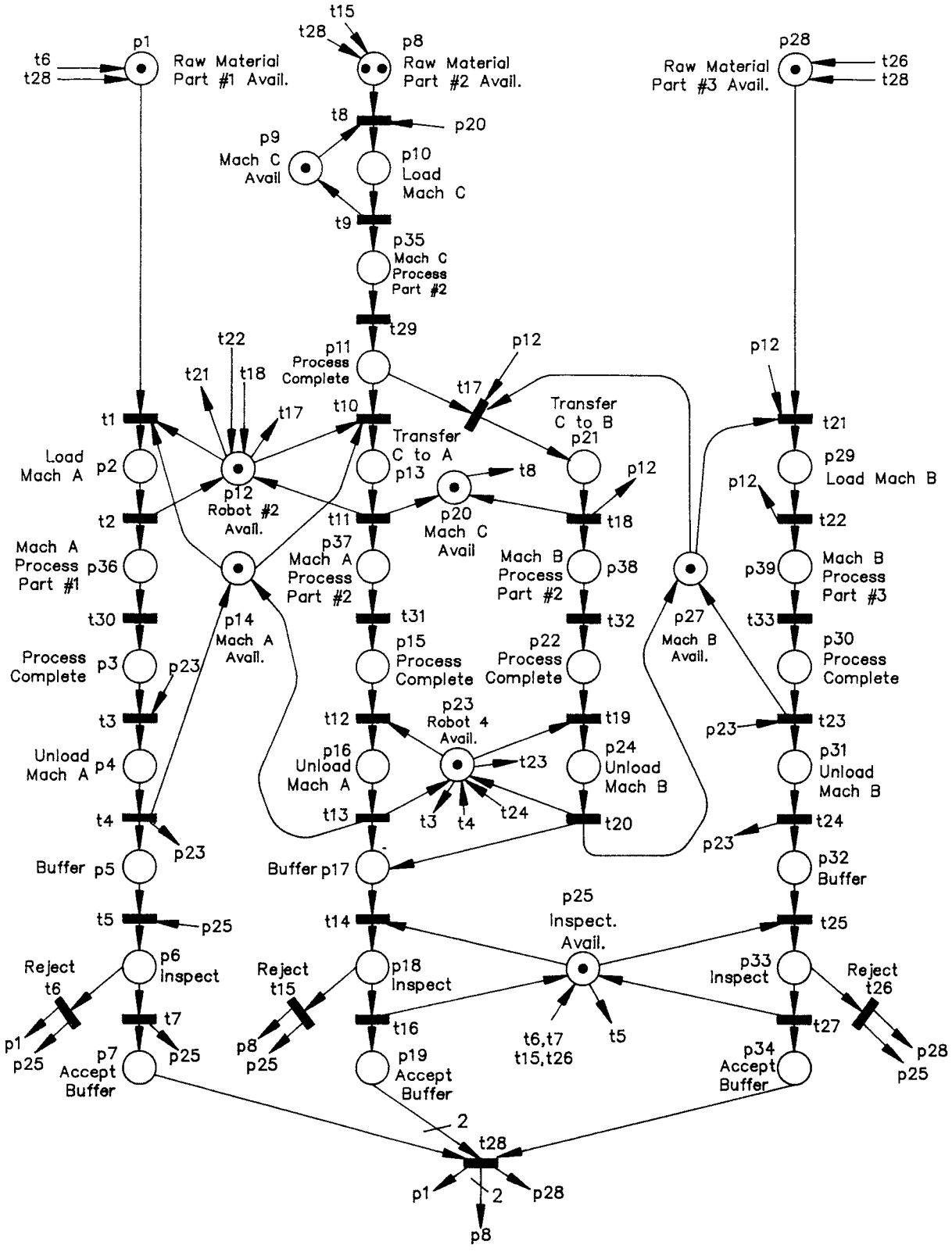

Figure 5E - Hybrid Method, Step 5 (Complete Net Without Robot #3)

either the beginning or completion of an event. The corresponding time delays and firing rates are in Table 2. The designations of the places and transitions are provided in Tables 3 and 4, respectively. Firing rates of transitions immediately following operation places are derived from the average time required to perform that operation. Firing rates for transitions immediately following resource places represent the rate at which the cells status is sampled. This is the time it takes the host computer to detect that a transition has been enabled. The status of this system is evaluated 1000 times each minute, resulting in a time delay of .001 min. The purpose of t_{28} is to synchronize the ratio of parts being manufactured in the cell. In this particular case, (1) part #1 and (2) parts #2 are required for every (1) of part #3. So, the input arcs to t_{28} are weighted 1, 2, and 1 for parts 1, 2, and 3, respectively. In addition, the output arcs, representing the release of more raw materials, are also weighted 1, 2, and 1, respectively.

3.4 Equivalent Inspection Firing Rates

It should be noted that firing rates for the transitions associated with inspection are not calculated directly from taking the inverse of the inspection time delay. In Figure 5E, p_6 represents the inspection of part #1. A token may be consumed from this place by firing either t_6 or t_7 signifying either the rejection or an acceptance of the part, respectively. The fact that the expected inspection time delay is .75 min. regardless of the outcome suggests that t_6 and t_7 should have equal time delays, thus having equal firing rates, which would result in 50% rejection and 50% acceptance. This would not reflect the actual 5% probability of rejection. So, to model the inspection process

accurately, *equivalent* firing rates were calculated for t_6 and t_7. These result in an average time delay between the two transitions of .75 min. and also reflect the 5% defective probability. The inspection transition rates for parts #2 and #3 were calculated in the same manner. Details of these calculations are in Appendix 1.

Table 2 - Time Delays and Firing Rates

Transition	τ_{old}	τ_{new}	λ_{old}	λ_{new}
1	.001	-	1000.000	-
2	.500	-	2.000	-
3	.001	-	1000.000	-
4	.700	-	1.428	-
5	.001	-	1000.000	-
6	76.900	7.690	.013	.130
7	3.950	.395	.253	2.530
8	.001	-	1000.000	-
9	.500	-	2.000	-
10	.001	-	1000.000	-
11	.800	-	1.250	-
12	.001	-	1000.000	-
13	.700	-	1.428	-
14	.001	-	1000.000	-
15	76.900	7.690	.013	.130
16	3.950	.395	.253	2.530
17	.001	-	1000.000	-
18	.800	-	1.250	-
19	.001	-	1000.000	-
20	.700	-	1.428	-
21	.001	-	1000.000	-
22	.500	-	2.000	-
23	.001	-	1000.000	-
24	.700	-	1.428	-
25	.001	-	1000.000	-
26	76.900	7.690	.013	.130
27	3.950	.395	.253	2.530
28	.001	-	1000.000	-
29	30.000	10.000	.033	1.000
30	7.500	2.500	.133	.400
31	4.500	1.500	.222	.667
32	4.500	1.500	.222	.667
33	13.200	4.400	.075	.227

Table 3 - Transition Designations

Trans.	Designation
1	Begin loading part #1 onto mach A with robot 2.
2	Finish loading part #1 onto machine A with robot 2 & begin processing.
3	Begin unload part #1 from mach A with robot 4.
4	Finish unload part #1 from mach A with robot 4.
5	Begin inspection of part #1.
6	Finish inspection of part #1 - Reject.
7	Finish inspection of part #1 - Accept.
8	Begin load part #2 onto machine C with robot 1.
9	Finish loading part #2 onto machine C with robot 1 & begin processing.
10	Begin transfer of part #2 from machine C to Machine A with robot 2.
11	Finish transfer of part #2 from machine C to Machine A & begin process.
12	Begin unload part #2 from mach A with robot 4.
13	Finish unload part #2 from mach A with robot 4.
14	Begin inspection of part #2.
15	Finish inspection of part #2 - Reject.
16	Finish inspection of part #2 - Accept.
17	Begin transfer of part #2 from machine C to machine B with robot 2.
18	Finish transfer of part #2 from machine C to machine B & begin process.
19	Begin unload part #2 from machine B with robot 4.
20	Finish unload part #2 from machine B with robot 4.
21	Begin loading part #3 onto machine B with robot 2.
22	Finish loading part #3 onto machine B with robot 2.
23	Begin unload part #3 from machine B with robot 4.
24	Finish unload part #3 from machine B with robot 4.
25	Begin inspection of part #3.
26	Finish inspection of part #3 - Reject.
27	Finish inspection of part #3 - Accept.
28	Release another set of raw materials.
29	Finish processing part #2 on machine C.
30	Finish processing part #1 on machine A.
31	Finish processing part #2 on machine A.
32	Finish processing part #2 on machine B.
33	Finish processing part #3 on machine B.

Table 4 - Place Designations

Place	Designation
1	Raw material part #1 is available.
2	Robot 2 is loading part #1 onto machine A.
3	Part #1 is done processing on machine A.
4	Robot 4 is unloading part #1 from machine A
5	Buffer for part #1.
6	Part 1 is being inspected.
7	Buffer for accepted part #1.
8	Raw material part 2 is available.
9	Robot 1 is available.
10	Robot 1 is loading part #2 onto machine C.
11	Part #2 is done processing on machine C.
12	Robot 2 is available.
13	Robot 2 transferring part #2 from machine C to A.
14	Machine A is available.
15	Part #2 is done processing on machine A.
16	Robot 4 is unloading part #2 from machine A.
17	Buffer for part #2.
18	Part #2 is being inspected.
19	Buffer for accepted part #2.
20	Machine C is available.
21	Robot 2 transferring part #2 from machine C to B.
22	Part #2 is done processing on machine B.
23	Robot 4 is available.
24	Robot 4 is unloading part #2 from machine B.
25	Inspection resource available.
26	*Reserved for* Robot 3
27	Machine B is available.
28	Raw material part #3 is available.
29	Robot 2 is loading part #3 onto machine B.
30	Part #3 is done processing on machine B.
31	Robot 4 is unloading part #3 from machine B.
32	Buffer for part #3.
33	Part #3 is being inspected.
34	Buffer for accepted part #3.
35	Part #2 is processing on machine C.
36	Part #1 is processing on machine A.
37	Part #2 is processing on machine A.
38	Part #2 is processing on machine B.
39	Part #3 is processing on machine B.

4 EVALUATION OF THE FMC

4.1 Current vs. Upgraded Equipment

The budget set aside to improve this manufacturing cell is enough only to purchase two major pieces of equipment. Choices include: upgrading machines A, B, or C, automating the inspection procedure, or purchasing an additional robot. Of these five choices, the two which best improve throughput must be determined.

The new CNC machines are capable of processing three times faster than the current machines. An automated inspection is expected to be ten times faster than the current manual method (see Table 1). An additional robot (robot #3) would assume half of the duties of robot #2. The current manual inspection process reveals a defect probability of five percent and is not expected to change with the arrival of new equipment.

4.2 Software Packages

There are numerous software packages available for modeling and evaluating flexible manufacturing systems. SPNP, GRAMAN, SIMAN, SLAM, and SIMSCRIPT are some of the more popular packages. SIMAN, SLAM, and SIMSCRIPT are better suited for simulation. Whereas SPNP and GRAMAN were developed around and intended to be used for Petri nets. Languages such as FORTRAN, PASCAL, and C can be utilized; however, this requires much more programming time and should only be done if all existing software has been explored and found to be insufficient. A

UNIX based version of SPNP (Stochastic Petri Net Package) has been found to be appropriate to model and evaluate the FMC under investigation. SPNP is written in C and is also available for VMS systems (VAX).

4.3 Requirements and Terminology of SPNP

Table 5 - SPNP Terminology

place("p1");	establishes a place p_1.
init("p1",1);	defines initial marking of p_1 as 1.
trans("t1");	establishes a transition t_1.
rateval("t1",2.0);	defines firing rate of t_1 = 2.0.
iarc("p1","t2");	defines an input arc to t_2 from p_1.
oarc("p1","t2");	defines an output arc from t_2 to p_1.
miarc("p3","t5",2);	defines an input arc to t_5 from p_3 with arc weight of 2.
oarc("p4","t1",3);	defines an output arc from t_1 to p_4 with arc weight of 3.

In order to run the model with the SPNP software, a source code must be written. The purpose of this code is to provide the necessary input and also request certain output. The input consists of the set of places along with their initial markings, a set of transitions along with their firing rates, the placement of the input and output arcs, and any variables that the user intends to change from run to run; such as machine process times. The terminology is described in Table 5. Refer to the SPNP manual for more information.

4.4 Net Reduction

The SPNP software was used to execute the Petri net model; however, due to the complexity of the model, calculations were extremely time consuming or impossible, depending on the initial marking. The model of Figure 5E was run on SPNP with a

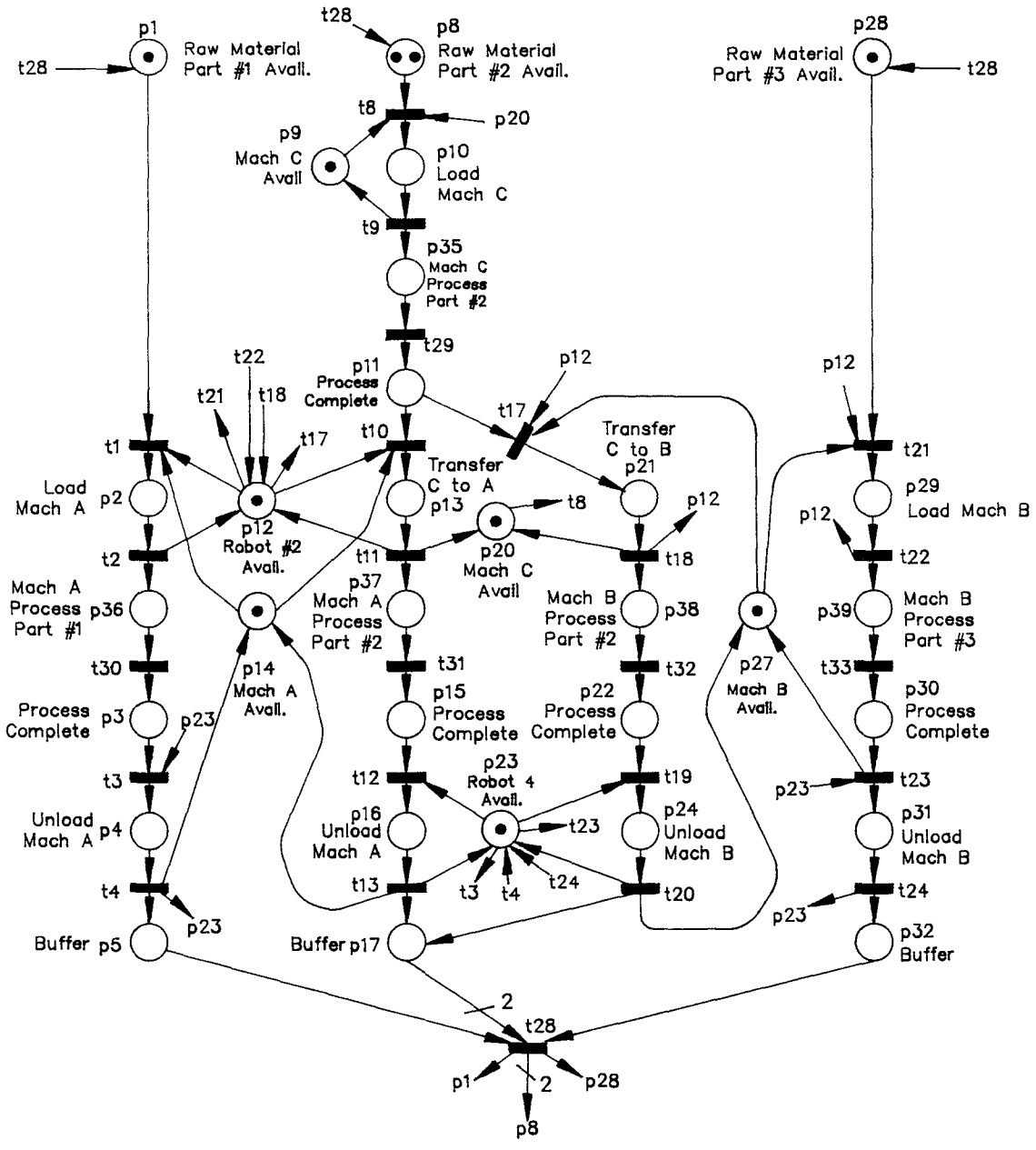

Figure 6 - Process Subsystem

minimal amount of tokens and was found to have an unmanageable number of states. This was expected since the input and output matrices are very large (39 x 33). For this reason the model was split into two subsystems: a Process Subsystem (Figure 6) and an Inspection Subsystem (Figure 7).

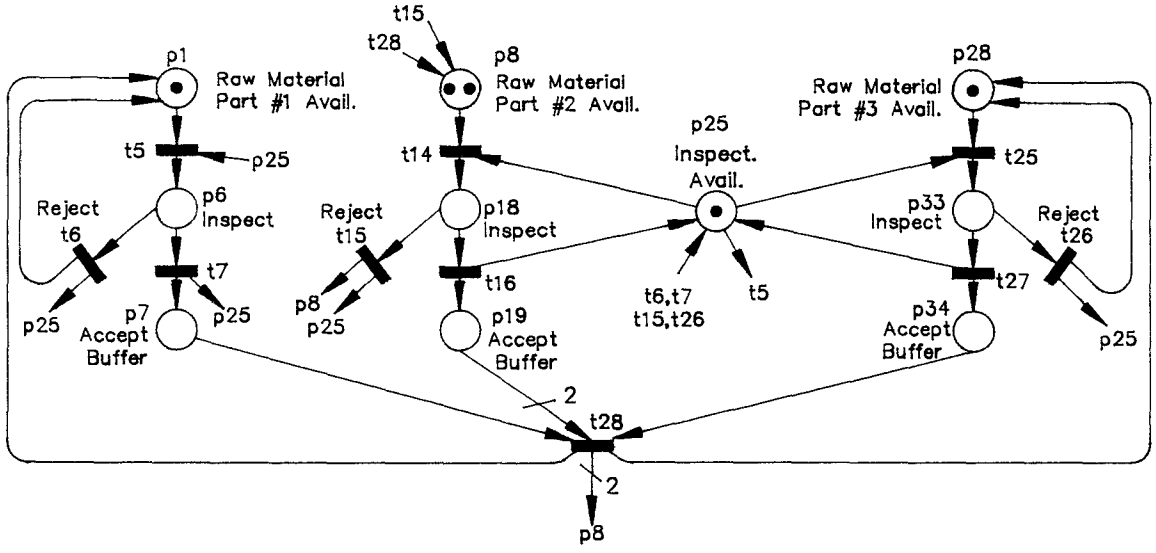

Figure 7 - Inspection Subsystem

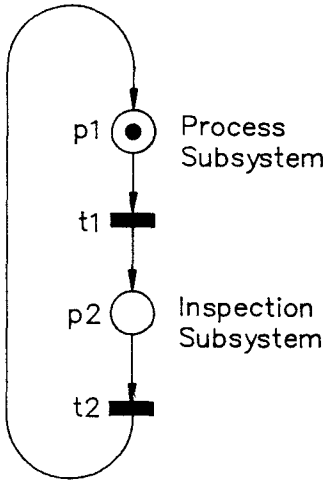

Figure 8 - Main net

These subsystems have a much smaller number of states compared to the complete net, thus greatly reducing the computational requirements. The use of subsystems or subnets makes it possible to evaluate large systems. Once the subnets are constructed and evaluated they can be represented by a single place or transition and then combined to form the main net for the overall system. Figure 8 shows the process and inspection subsystems as single places in the main net. It is apparent what a drastic

reduction this is when compared to the net of Figure 5E. Performance measures such as system throughput and subsystem utilization can be derived from the main net. Whereas, measures such as machine utilization and average buffer levels can be found from the individual subnets. As will be discussed in later sections, this subnet approach not only reflects the hierarchical structure of most flexible manufacturing systems, but it is also well suited for process control of the individual subnets as well as the main net.

4.5 Results of Evaluation

Table 6 - Results of Evaluation

Run #	Mach A	Mach B	Mach C	Robot 3	Inspect.	Process Subsys. Thrput.	Inspect. Subsys. Thrput.	Main Net Thrput.
1	NEW	NEW	OLD	NO	MANUAL	0.0154	0.0681	0.0126
2	NEW	OLD	NEW	NO	MANUAL	0.0335	0.0681	0.0224
3	NEW	OLD	OLD	NO	AUTO	0.0148	0.6796	0.0145
4	OLD	NEW	NEW	NO	MANUAL	0.0356	0.0681	0.0234
5	OLD	NEW	OLD	NO	AUTO	0.0149	0.6796	0.0146
6	OLD	OLD	NEW	NO	AUTO	0.0305	0.6796	0.0292
7	NEW	OLD	OLD	YES	MANUAL	0.0148	0.0681	0.0122
8	OLD	NEW	OLD	YES	MANUAL	0.0149	0.0681	0.0122
9	OLD	OLD	NEW	YES	MANUAL	0.0306	0.0681	0.0211
10	OLD	OLD	OLD	YES	AUTO	0.0144	0.6796	0.0141

The output requested from SPNP includes machine, robot, and inspection utilizations as well as system throughput. Notice that throughput is the average rate at which t_{28} fires. The SPNP source code for the complete system (Figure 5E), the process subsystem (Figure 6), and inspection subsystem (Figure 7) can be found in Appendices 2, 3, and 4, respectively. Each subsystem was executed for all possible choices of

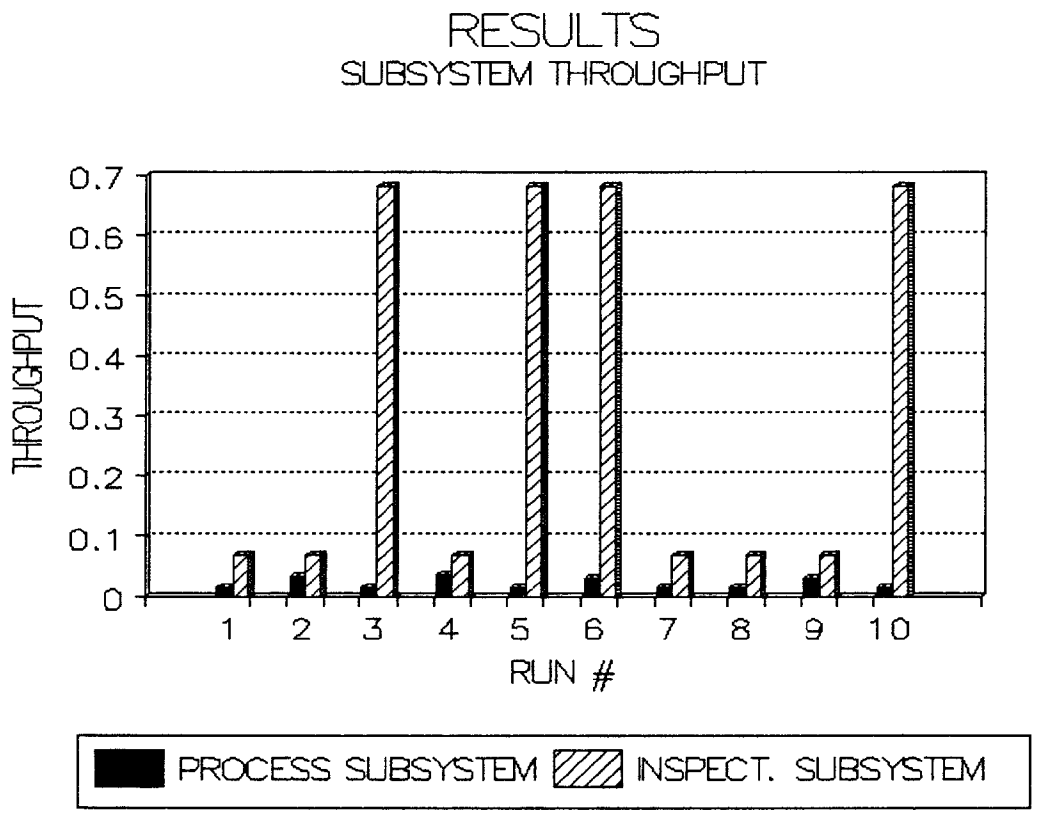

Figure 9A - Results of Evaluation (Subsystems)

upgrades. There are 10 possible combinations of upgrades when choosing 2 of the 5 candidates. The results are shown in Table 6. Details are in Appendix 5. The subsystem results in Table 6 are also graphically represented in Figure 9A.

The subsystem throughputs are equivalent to the firing rate for that subsystem. These throughput rates were applied to the main net (Figure 8) and SPNP was used to evaluate the overall throughput. The highest overall throughput rate (0.0292) was obtained from run #6. This indicates that the inspection station and machine C should be upgraded in order to maximize throughput. The throughput results for the main net are shown in Figure 9B.

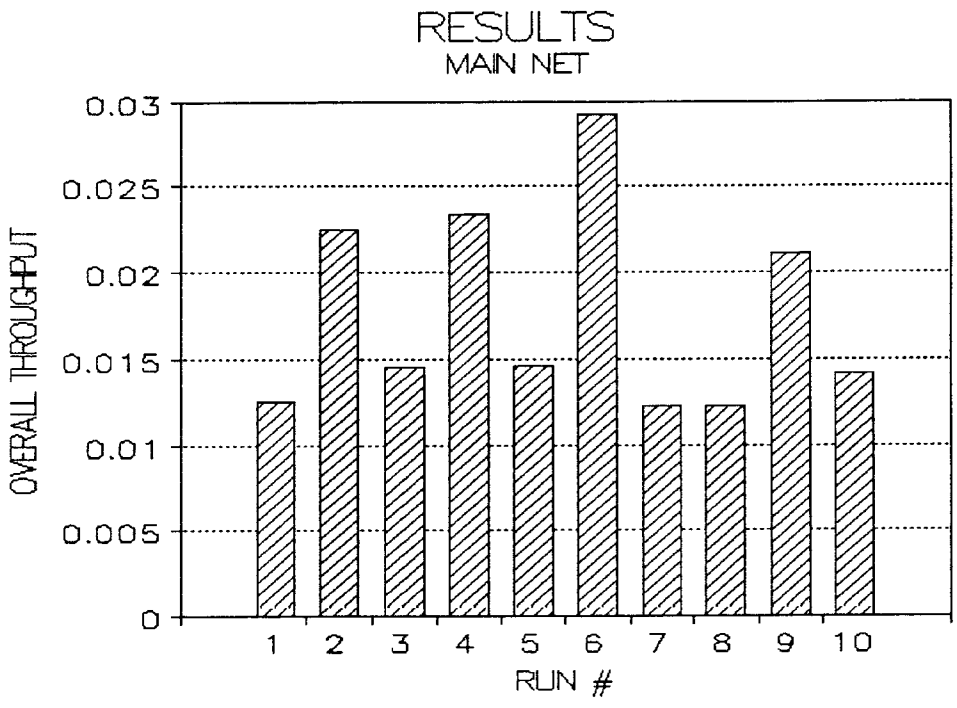

Figure 9B - Results of Evaluation (Main Net)

5 CONTROL OF A FMC

5.1 History of Control

In the past several decades the demands placed on manufacturing control systems has increased dramatically. Due to the trend towards greater flexibility and the increasing size and complexity of manufacturing systems, the control hardware and software must be more reliable, efficient, and flexible. During the 1960's control was accomplished through mechanical relays, timers, and counters. These were bulky, slow, and unreliable. In 1970 the Programmable Logic Controller (PLC) was introduced. The PLC has several improvements over its mechanical predecessors. It is more reliable, easier to maintain, and reprogrammable [16]. In addition, it requires far less space and is much more flexible [16]. Most PLC languages are based on *ladder logic* and *boolean algebra* [5],[21]. Intended for more complicated applications, some PLC's utilize more universal languages based on graphical formalism such as state diagrams, GRAFCET, and algorithmic state machines. GRAFCET was developed from Petri net theory and is well suited for controlling concurrence. It is commonly used in industrial applications [21]. Although PLC's were a vast improvement over former methods, programs using ladder logic or procedural languages such as ASSEMBLER can become overwhelming for complex systems. They are difficult to interpret making them inflexible and hard to maintain. Programming concurrent events can be a tedious, if not impossible task. In an attempt to deal with concurrence, concurrent PASCAL has been developed. Although GRAFCET is capable of handling concurrence, it is only a modeling technique and cannot be executed directly [18]. During the 1980's

Petri net theory was generally used in conjunction with PLC's [26],[25]. Today Petri net based controllers are usually developed around a computer such as a VAX 11/780 [12], an IBM PC/XT [12] or even an INTEL 8031 microprocessor [19]. Recently several Petri net based control languages and controllers have been the subject of research and development. These include: Station Controller (SCR) [18]; Petri net Descriptive Language (PNDL) [26],[25]; Petri net System Supervisor (PNSS) [26],[25]; Marked Flow Graph (MFG) [18]; Control net (C-net) [18],[15]; Petri net Controller (PNC) [11]; and the Token Player [1],[24],[19]. Petri net theory has also been combined with knowledge based or rule based systems [2].

5.2 Attributes of an Effective Control System

A control system should have the following attributes:

1) A controller must have the inherent ability to deal with asynchronous, stochastic, discrete-time events occurring concurrently as well as sequentially. The system controller must be equipped to resolve conflicts.
2) A controller should share much of the methodology, terminology, and definitions used throughout the design and evaluation phase. This provides for a smooth transition from one stage to the next and provides a common language in which people involved in each particular phase can communicate [21].
3) The control scheme should be applicable to all levels of control ranging from the host computer at the global level to the workcell controller at the local level. This facilitates communication of the software at different levels [21].

4) A control system should have a simple, easy to understand, graphical representation with a solid mathematical foundation.

5) The hardware and software should be expandable and maintainable [15]. The software should be flexible without being overly complex. See [21] for a discussion of flexibility versus optimality.

6) The processor or computer needs to have rapid processing speed and the software should be efficient to allow the system to be controlled in real time.

7) The cost of the design and implementation should be proportionally low.

8) The control system should be arranged hierarchically. It should be decentralized to enable the different levels and subsystems to operate as independently as possible [15].

9) The control system should be capable of detecting faults, diagnosing them, and even learning from them as discussed in [8].

The remaining portion of this paper will describe the *Control net* and will demonstrate its use in monitoring a flexible manufacturing cell. The Control net has most of the attributes listed above (the exception being the ability to learn) thus providing a solid foundation for a control system.

5.3 The Control Net

The basis for the following Control net (*C-net*) was developed from a Petri net based controller presented by Tomohiro Murato *et al* [18]. The C-net is a bounded net which is designed to be as choice-free as possible. The C-net has all of the characteristics of

the Petri net previously described for modeling and evaluation. However, several aspects have been added to facilitate the control function. As was described in Section 2.2, a Petri net is defined as:

$$Z=\{P,T,I,O,m\} \qquad (3)$$

The expanded description of the C-net is as follows:

$$C=\{P,T,I,O,\delta,\varphi,\eta,U,V,m\} \qquad (6)$$

The new additions are split into two groups: the *process i/o functions*, δ, φ, and η; and the *process status symbols,* U and V. Definitions for the original members P,T,I,O, and m remain unchanged.

The process i/o functions, as the name implies, provide an i/o interface between the actual system and the controller thus enabling the C-net controller to communicate with the system in real time. Let A represent a set of output (control) signals x_i and let E be a set of input (observable) signals y_i. Each place and transition has a particular set of signals associated with it and they are defined as follows:

$$\delta(p_i) = x_i, \{x_i \in A, p_i \in P\} \qquad (7)$$

$$\varphi(p_i) = (y_{i,1}, y_{i,2}, \ldots, y_{i,n}), \{y_{i,j} \in E, p_i \in P\} \qquad (8)$$

$$\eta(t_j) = (y_{i,k}, \ldots y_{m,n}), \{y_{i,k} \in E, t_j \in T, {}^{\bullet}t_j = (p_i, \ldots, p_m)\} \qquad (9)$$

The process status symbols, $U(p_i)$ and $V(t_j)$, monitor the progress of the activities occurring throughout the system. $U(p_i)$ is directly dependent on $\delta(p_i)$ and $\varphi(p_i)$. $U(p_i)$ is defined for each place and is set to zero until the action associated with p_i has been

completed, at which time $U(p_i)$ is set to a value other than zero ($U(p_i) \neq 0$). $V(t_j)$ is dependent on $\eta(t_j)$ and determines whether or not transition t_j is opened or closed. If an input $y_{i,k}$ defined by $\eta(t_j)$ has been detected $V(t_j)$ is set to 1 (open); otherwise, $V(t_j)$ is set to 0 (closed) [18].

The C-net terminology refers to places as *boxes* and transitions as *gates*. With the additions incorporated into the C-net, our previous form of the transition firing rule is no longer adequate. An expanded version of this rule is called the *gate firing rule* and includes $U(p_i)$ and $V(t_j)$. The gate firing rule for a k-bounded C-net states that a gate $t_i \epsilon T$ is enabled at marking m_1 if and only if, $V(t_j)=1$, $U(p_i) \neq 0$, $1 \leq m_1(p_i) \leq k$, and $0 \leq m_1(p_h) \leq (k-1)$ for all $p_i \epsilon {}^{\bullet}t_j$ and $p_h \epsilon\ t_j^{\bullet}$. As with the transition firing rule, the requirements placed upon the input and output boxes remain intact for the gate firing rule. One purpose of the gate firing rule is to insure that the C-net remains k-bounded. This is captured in $1 \leq m_1(p_i) \leq k$ and $0 \leq m_1(p_h) \leq (k-1)$. $V(t_j)=1$ insures that the required input signal defined by $\eta(t_j)$ has been received and the gate is open. The completion of the action modeled by p_i is acknowledged by $U(p_i) \neq 0$.

5.4 C-net Symbols

In an effort to make the graphical representation of a C-net more descriptive, several different box types will be used which have been developed as part of the Petri Net Controller (PNC) which is presented by D. Crockett *et al* in [11]. The five box types are *semaphore, simple, action, switching*, and *macro*. Figure 10 contains graphical representations of each type. The first type, a *semaphore box*, is an elementary box

Figure 10 - C-net Symbols

which is generally used to acknowledge that a specific condition exists. For example, the availability of a resource would be represented through a marked semaphore box. A *simple box* is employed to describe a relatively short procedure in which a very small amount of time is required to perform a task and receive the result. This may involve sending a message to a workstation and receiving a confirmation of its receipt. The third type of box is the *action box* which is used to represent more lengthy procedures such as machining processes or the robotic loading of workpieces. Notice the action box has two general areas: the larger open area within the square and the area inside of the small circle. When a token first arrives in an action box it resides in the larger area and the action associated with this box begins. Upon completion of the action, the token moves into the small circle signifying that the action has been completed and the token is ready to continue through the net. The *switching box* resolves conflicts or makes decisions based on the current state of the system. As with the action box, a token is held in the larger area within the square. The state of the system is evaluated and depending on the result the token moves into circle *a* or *b* where it is available to continue through the net. The fifth box type, a *macro box*, is used in conjunction with a subprogram or a subnet. When the larger circle within this

box is marked, the subprogram or subnet is initialized. After the subprogram or subnet has been executed successfully, the token moves into the smaller circle. These five symbols are useful for the real time graphical representation of the C-net.

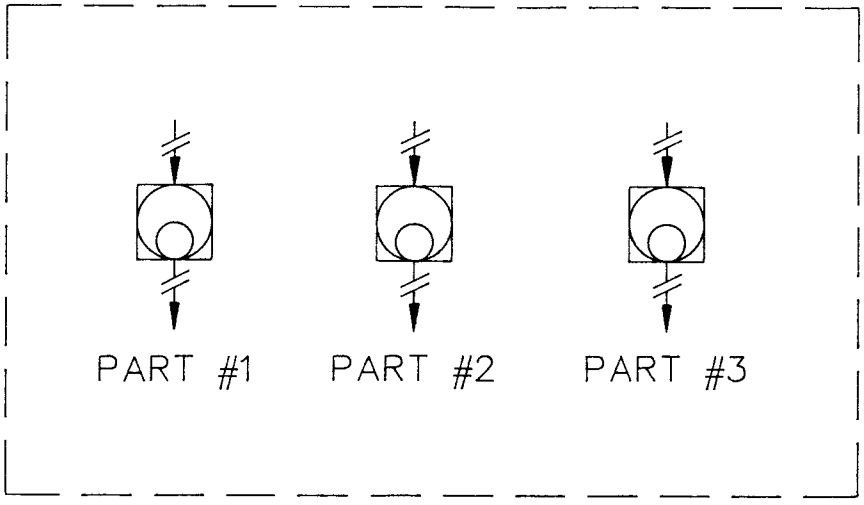

Figure 11 - Main Net With 3 Part Families

An example of how these new box types would be used to model the hierarchical structure of a flexible manufacturing system is shown in Figure 12. Here, a host computer executes the main C-net algorithm. Two particular macro boxes contained within the main net represent workcells A and B. These workcell macro boxes engage subnets of the individual workcells. In turn, these workcell subnets contain macro boxes representing the individual machines within the cell. This hierarchical structure distributes the computational burden between the host, workcell, and machine computers.

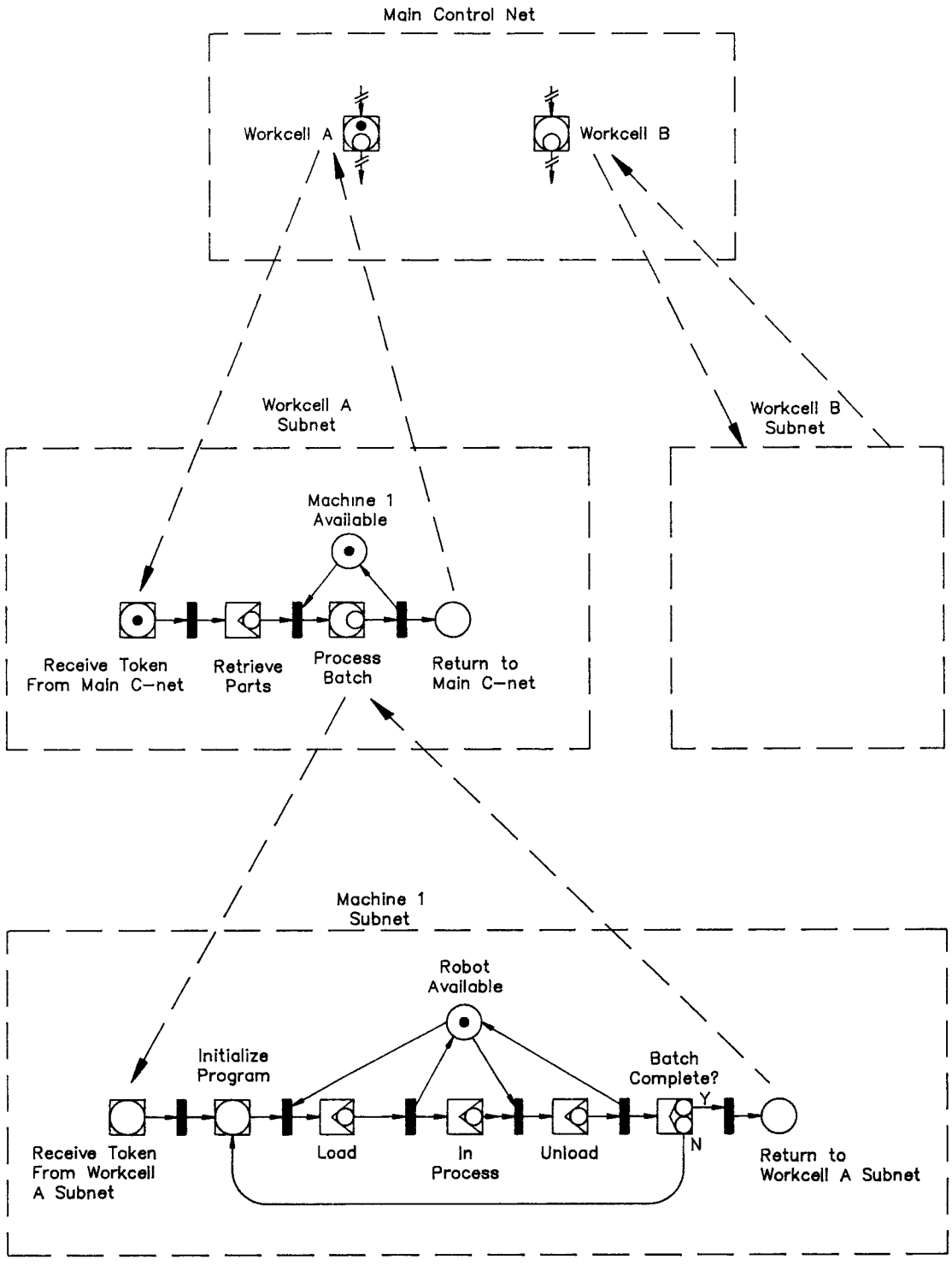

Figure 12 - Hierarchically Structured C-net

The FMC (Figure 4) modeled in earlier sections can be viewed as a subsystem within a larger Flexible Manufacturing System (FMS). The Main Control Net for the FMS (Figure 11) would contain three macro boxes (among other boxes) representative of each of the three part families. Once one of the macro boxes of the main control net become marked, control would be initiated in the subnet (Figure 13) representing the FMC. In Figure 13 these new box types have been employed to better represent the net of Figure 5E for control purposes. This C-net is 2-bounded. When the required quantities of parts 1, 2, and 3 have been processed, t_{28} fires and returns control to the main net.

In order to expedite the defining of the process i/o functions, basic definitions have been developed for $\delta(p_i)$ and $\varphi(p_i)$ as they apply to each box type. These definitions can be found in Table 7. In addition to $\delta(p_i)$ and $\varphi(p_i)$, a basic definition for $\eta(t_j)$ must be developed which is appropriate for all gates of the C-net. Since all gates have at least one input box and all input boxes have a minimum of two input (observable) signals associated with them, $\eta(t_j)$ will be defined as the *desired response* from the system. For a gate following an action box the desired response would be the proper completion of the action. Similarly the desired response for a macro box is that the subnet or subprogram has been executed correctly. For all of the box types, except the switching box, the desired responses are $y_{i,1}$. The *error* response, $y_{i,2}$, is not desirable. Therefore, $\eta(t_j) = y_{i,1}$ where *j* is the gate number and *i* is the number of the input box. In the event that the gate has a switching box as an input, the desired response may be either $y_{i,1}$ or $y_{i,2}$ ($y_{i,3}$ is the *error* response). If a gate t_j has two input boxes, *a* and

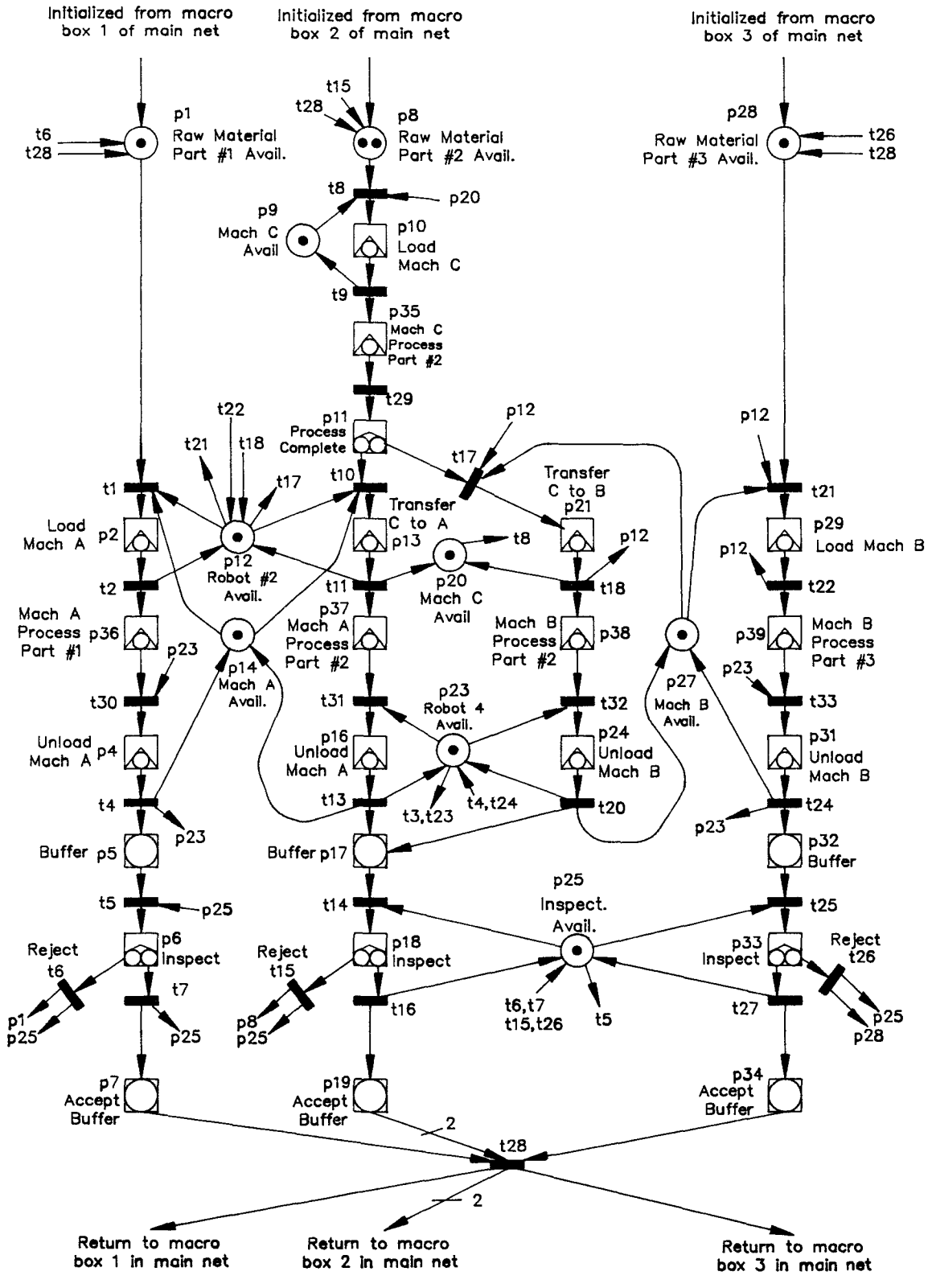

Figure 13 - C-net for the FMC

Table 7 - Process i/o Function Definitions

Semaphore Box, i -	$\delta(p_i)$ =	x_1 → No output required.
	$\varphi(p_i)$ =	$y_{i,1}$ → Resource is ready.
		$y_{i,2}$ → Error (Resource is down).
Simple Box, i -	$\delta(p_i)$ =	x_1 → Send message (if required).
	$\varphi(p_i)$ =	$y_{i,1}$ → Acknowledge receipt of message.
		$y_{i,2}$ → Error
Action Box, i -	$\delta(p_i)$ =	x_1 → Send message to start action.
	$\varphi(p_i)$ =	$y_{i,1}$ → Action completed properly.
		$y_{i,2}$ → Error
Switching Box, i -	$\delta(p_i)$ =	x_1 → Evaluate current state of the system.
	$\varphi(p_i)$ =	$y_{i,1}$ → State of system warrants choice A.
		$y_{i,2}$ → State of system warrants choice B.
		$y_{i,3}$ → Error
Macro Box, i -	$\delta(p_i)$ =	x_1 → Execute a subnet or subprogram.
	$\varphi(p_i)$ =	$y_{i,1}$ → Execution completed properly.
		$y_{i,2}$ → Error

b, then $\eta(t_j) = \{y_{a,1}, y_{b,1}\}$, with ${}^{\bullet}t_j = (p_a, p_b)$. These basic process i/o function definitions are now linked to the C-net symbols to completely describe the specific i/o requirements of the FMC.

6 CONCLUSION

6.1 Contributions

It has been shown that Petri nets are an extremely versatile tool which are useful for the modeling, evaluation, and control of flexible manufacturing systems. Their simple yet powerful graphical representation allows manufacturing engineers to model existing or proposed systems. Petri nets have proven to be flexible and maintainable. The graphical model is easily developed via the hybrid method. Input and output matrices, used for mathematical analysis, are derived directly from the graphical model. The system characteristics and performance measures can be evaluated mathematically or through simulation. The model and its parameters can be altered and evaluated repeatedly until the results are optimal. Petri nets bridge the gap between modeling a system, evaluating a system, and designing a control net for that system. As a result, the overall development time is reduced.

In particular this paper has made several contributions to the application of Petri nets for Flexible Manufacturing Systems.

1) Thorough research has resulted in a list of attributes for an effective control system.
2) Strong graphical C-net symbols have been combined with process i/o function definitions to provide a smooth transition from a model to a functional system controller. The Petri net is transformed into a Control net through the use of special C-net box symbols. Basic i/o requirements for the system are inherent in the Control net since they are defined for each box symbol.

3) The original definition of a C-net [18] has been expanded to allow a structurally bounded rather than a strictly safe net. In addition, the definition of $\eta(t_j)$ has been expanded to include desired responses from multiple input places, ${}^{\bullet}t_j$.
4) This paper has demonstrated the use of Petri nets for the complete engineering cycle of a Flexible Manufacturing Cell. Several ideas and methodologies have been combined to provide a workable framework for the modeling, evaluation and control phases.

6.2 Limitations

Two significant limitations were encountered during this research. The number of states which can evolve from a relatively simple model require a large amount of computational power. Also, in trying to incorporate flexibility into the system the model can become overly complex and unmanageable.

6.3 Future Research

Future research could involve developing a C-net software package to be used for design, simulation, and control of flexible manufacturing systems. A crucial first step would be the creation of an effective control algorithm. This software would enable the user to build a graphical C-net interactively on a personal computer screen and simulate its operation. In addition it might assist the user in defining the process i/o functions. Also it should be capable of controlling the system in real time through some i/o interface.

APPENDIX 1

Equivalent Firing Rates
for Inspection Related Transitions

P1 Probability of acceptance.

P2 Probability of rejection.

$\lambda 1$ Firing rate of inspection acceptance transition discounting probabilities.

$\lambda 2$ Firing rate of inspection rejection transition discounting probabilities.

$eq\lambda 1$ Equivalent firing rate of acceptance transition combining inspection time delay τ with probability of acceptance P1.

$eq\lambda 2$ Equivalent firing rate of rejection transition combining inspection time delay τ with probability of acceptance P2.

$$P1 = .95 \qquad P2 = .05 \qquad \tau = .75$$

$$\lambda 1 = 1/\tau = 1.333 \qquad \lambda 2 = 1/\tau = 1.333$$

$$eq\lambda 1 = P1 \cdot (\lambda 1 + \lambda 2) = 2.533$$

$$eq\lambda 2 = P2 \cdot (\lambda 1 + \lambda 2) = 0.133$$

APPENDIX 2

SPNP Source Code for Complete System With Robot #3

```
# include        "user.h"

/* Glenn Thornley, 203-58-4911*/

/* Analysis of FMS with Robot #3 */


float   S,T,U,V,MA1,MC2,MA2,MB2,MB3,IA, IR;

parameters() {
        iopt(IOP_PR_FULL_MARK, VAL_YES);
        iopt(IOP_PR_MC,VAL_YES);
        iopt(IOP_PR_RGRAPH,VAL_YES);
        iopt(IOP_PR_PROB,VAL_YES);
       S = input("Machine A Old(3) or New(1):");
       T = input("Machine B Old(3) or New(1):");
       U = input("Machine C Old(3) or New(1):");
       V = input("Inspection - Automatic(10) or Manual(1):");
       MA1 = .4/S;
       MC2 = .1/U;
       MA2 = .667/S;
       MB2 = .667/T;
       MB3 = .227/T;
       IA = .253*V;
       IR = .013*V;
}
```

```
net() {
        place("p1"); init("p1",1);
        place("p2");
        place("p3");
        place("p4");
        place("p5");
        place("p6");
        place("p7");
        place("p8"); init("p8",1);
        place("p9"); init("p9",1);
        place("p10");
        place("p11"); init("p11",1);
        place("p12"); init("p12",1);
        place("p13");
        place("p14"); init("p14",1);
        place("p15");
        place("p16");
        place("p17");
        place("p18");
        place("p19");
        place("p20"); init("p20",1);
        place("p21");
        place("p22");
        place("p23"); init("p23",1);
        place("p24");
        place("p25"); init("p25",1);
        place("p26"); init("p26",1);
```

```
place("p27"); init("p27",1);
place("p28");
place("p29");
place("p30");
place("p31");
place("p32"); init("p32",1);
place("p33");
place("p34");
place("p35");
place("p36");
place("p37");
place("p38");
place("p39");

trans("t1");        rateval("t1",1000.0);
trans("t2");        rateval("t2",2.0);
trans("t3");        rateval("t3",1000.0);
trans("t4");        rateval("t4",1.428);
trans("t5");        rateval("t5",1000.0);
trans("t6");        rateval("t6",IR);
trans("t7");        rateval("t7",IA);
trans("t8");        rateval("t8",1000.0);
trans("t9");        rateval("t9",2.0);
trans("t10");       rateval("t10",1000.0);
trans("t11");       rateval("t11",1.25);
trans("t12");       rateval("t12",1000.0);
trans("t13");       rateval("t13",1.428);
```

```
trans("t14");        rateval("t14",1000.0);
trans("t15");        rateval("t15",IR);
trans("t16");        rateval("t16",IA);
trans("t17");        rateval("t17",1000.0);
trans("t18");        rateval("t18",1.25);
trans("t19");        rateval("t19",1000.0);
trans("t20");        rateval("t20",1.428);
trans("t21");        rateval("t21",1000.0);
trans("t22");        rateval("t22",2.0);
trans("t23");        rateval("t23",1000.0);
trans("t24");        rateval("t24",1.428);
trans("t25");        rateval("t25",1000.0);
trans("t26");        rateval("t26",IR);
trans("t27");        rateval("t27",IA);
trans("t28");        rateval("t28",1000.0);
trans("t29");        rateval("t29",MC2);
trans("t30");        rateval("t30",MA1);
trans("t31");        rateval("t31",MA2);
trans("t32");        rateval("t32",MB2);
trans("t33");        rateval("t33",MB3);

iarc("t1","p1"); iarc("t1","p12"); iarc("t1","p14");
oarc("t1","p2"); iarc("t2","p2"); oarc("t2","p36");
oarc("t2","p12"); iarc("t3","p3"); iarc("t3","p23");
oarc("t3","p4"); iarc("t4","p4"); oarc("t4","p5");
oarc("t4","p14"); oarc("t4","p23"); iarc("t5","p5");
iarc("t5","p25"); oarc("t5","p6"); iarc("t6","p6");
```

```
oarc("t6","p1"); oarc("t6","p25"); iarc("t7","p6");
oarc("t7","p7"); oarc("t7","p25"); iarc("t8","p8");
iarc("t8","p9"); iarc("t8","p20"); oarc("t8","p10");
iarc("t9","p10"); oarc("t9","p9"); oarc("t9","p35");
iarc("t10","p11"); iarc("t10","p12"); iarc("t10","p14");
oarc("t10","p13"); iarc("t11","p13"); oarc("t11","p12");
oarc("t11","p37"); oarc("t11","p20"); iarc("t12","p15");
iarc("t12","p23"); oarc("t12","p16"); iarc("t13","p16");
oarc("t13","p14"); oarc("t13","p17"); oarc("t13","p23");
iarc("t14","p17"); iarc("t14","p25"); oarc("t14","p18");
iarc("t15","p18"); oarc("t15","p8"); oarc("t15","p25");
iarc("t16","p18"); oarc("t16","p19"); oarc("t16","p25");
iarc("t17","p11"); iarc("t17","p26"); iarc("t17","p27");
oarc("t17","p21"); iarc("t18","p21"); oarc("t18","p20");
oarc("t18","p38"); oarc("t18","p26"); iarc("t19","p22");
iarc("t19","p23"); oarc("t19","p24"); iarc("t20","p24");
oarc("t20","p17"); oarc("t20","p23"); oarc("t20","p27");
iarc("t21","p26"); iarc("t21","p27"); iarc("t21","p28");
oarc("t21","p29"); iarc("t22","p29"); oarc("t22","p26");
oarc("t22","p39"); iarc("t23","p23"); iarc("t23","p30");
oarc("t23","p31"); iarc("t24","p31"); oarc("t24","p23");
oarc("t24","p27"); oarc("t24","p32"); iarc("t25","p25");
iarc("t25","p32"); oarc("t25","p33"); iarc("t26","p33");
oarc("t26","p25"); oarc("t26","p28"); iarc("t27","p33");
oarc("t27","p25"); oarc("t27","p34"); iarc("t28","p7");
miarc("t28","p19",2); iarc("t28","p34"); oarc("t28","p1");
moarc("t28","p8",2); oarc("t28","p28"); iarc("t29","p35");
```

```
        oarc("t29","p11"); iarc("t30","p36"); oarc("t30","p3");
        iarc("t31","p37"); oarc("t31","p15"); iarc("t32","p38");
        oarc("t32","p22"); iarc("t33","p39"); oarc("t33","p30");

/*The  net is defined, and the analysis will be conducted */
}

/* the following lines should appear in all programs */

assert() {return(RES_NOERR);}
ac_init() {}
ac_reach() {fprintf(stderr,"/nThe reachibility graph has been generated/n/n");}

/* User-defined output functions */

reward_type ef0() {return(rate("t28"));}
/* throughput */

reward_type ef1() {return(mark("p10"));}
reward_type ef2() {return(mark("p2")+mark("p13"));}
reward_type ef3() {return(mark("p21")+mark("p29"));}
reward_type ef4() {return(1.0-mark("p23"));}
/* robot utilization */

reward_type ef5() {return(1.0-mark("p14"));}
reward_type ef6() {return(1.0-mark("p27"));}
reward_type ef7() {return(1.0-mark("p20"));}
```

```
/* machine utilization */

reward_type ef8() {return(1.0-mark("p25"));}
/* inspection utilization */

/*Output*/
ac_final() {pr_expected("throughput = ",ef0);
            pr_expected("Robot 1 Utilization = ",ef1);
            pr_expected("Robot 2 Utilization = ",ef2);
            pr_expected("Robot 3 Utilization = ",ef3);
            pr_expected("Robot 4 Utilization = ",ef4);
            pr_expected("Machine A Utilization = ",ef5);
            pr_expected("Machine B Utilization = ",ef6);
            pr_expected("Machine C Utilization = ",ef7);
            pr_expected("Inspection Utilization = ",ef8);
            pr_std_average();}
```

APPENDIX 3

SPNP Source Code for the Inspection Subsystem

```
# include        "user.h"

/* Glenn Thorniley, 203-58-4911*/

/* Analysis of Inspection Subsystem */


float   V,IA, IR;

parameters() {
        iopt(IOP_PR_FULL_MARK, VAL_YES);
        iopt(IOP_PR_MC,VAL_YES);
        iopt(IOP_PR_RGRAPH,VAL_YES);
        iopt(IOP_PR_PROB,VAL_YES);
       V = input("Inspection - Manual(10) or Automatic(1):");
       IA = 2.53/V;
       IR = .13/V;
}

net() {
       place("p1"); init("p1",2);
       place("p6");
       place("p7");
       place("p8"); init("p8",4);
       place("p18");
       place("p19");
       place("p25"); init("p25",1);
```

```
place("p28"); init("p28",2);
place("p33");
place("p34");

trans("t5");          rateval("t5",1000.0);
trans("t6");          rateval("t6",IR);
trans("t7");          rateval("t7",IA);
trans("t14");         rateval("t14",1000.0);
trans("t15");         rateval("t15",IR);
trans("t16");         rateval("t16",IA);
trans("t25");         rateval("t25",1000.0);
trans("t26");         rateval("t26",IR);
trans("t27");         rateval("t27",IA);
trans("t28");         rateval("t28",1000.0);

iarc("t5","p1");
iarc("t5","p25"); oarc("t5","p6"); iarc("t6","p6");
oarc("t6","p1"); oarc("t6","p25"); iarc("t7","p6");
oarc("t7","p7"); oarc("t7","p25");
iarc("t14","p8"); iarc("t14","p25"); oarc("t14","p18");
iarc("t15","p18"); oarc("t15","p8"); oarc("t15","p25");
iarc("t16","p18"); oarc("t16","p19"); oarc("t16","p25");
iarc("t25","p25");
iarc("t25","p28"); oarc("t25","p33"); iarc("t26","p33");
oarc("t26","p25"); oarc("t26","p28"); iarc("t27","p33");
oarc("t27","p25"); oarc("t27","p34"); iarc("t28","p7");
miarc("t28","p19",2); iarc("t28","p34"); oarc("t28","p1");
```

```
        moarc("t28","p8",2); oarc("t28","p28");

/*The  net is defined, and the analysis will be conducted */
}

/* the following lines should appear in all programs */

assert() {return(RES_NOERR);}
ac_init() {}
ac_reach() {fprintf(stderr,"/nThe reachibility graph has been generated/n/n");}

/* User-defined output functions */

reward_type ef0() {return(rate("t28")+(rate("t6")/2)+(rate("t15")/4)+(rate("t26")/2));}
/* throughput */

reward_type ef1() {return(rate("t6"));}
reward_type ef4() {return(rate("t7"));}
reward_type ef2() {return(rate("t15"));}
reward_type ef5() {return(rate("t16"));}
reward_type ef3() {return(rate("t26"));}
reward_type ef6() {return(rate("t27"));}
/* actual rejection rate */

reward_type ef8() {return(1.0-mark("p25"));}
/* inspection utilization */
```

```
/*Output*/
ac_final() {pr_expected("throughput ",ef0);
            pr_expected("Reject rate of part #1 ",ef1);
            pr_expected("Accept rate of part #1 ",ef4);
            pr_expected("Reject rate of part #2 ",ef2);
            pr_expected("Accept rate of part #2 ",ef5);
            pr_expected("Reject rate of part #3 ",ef3);
            pr_expected("Accept rate of part #3 ",ef6);
            pr_expected("Inspection Utilization  ",ef8);
            pr_std_average();}
```

APPENDIX 4

SPNP Source Code for
the Process Subsystem
Without Robot #3

```
# include        "user.h"

/* Glenn Thornley, 203-58-4911*/

/* Analysis of process subsystem without Robot #3 */


float   S,T,X,MA1,MC2,MA2,MB2, MB3;

parameters() {
        iopt(IOP_PR_FULL_MARK, VAL_YES);
        iopt(IOP_PR_MC,VAL_YES);
        iopt(IOP_PR_RGRAPH,VAL_YES);
        iopt(IOP_PR_PROB,VAL_YES);
       S = input("Machine A Old(3) or New(1):");
       T = input("Machine B Old(3) or New(1):");
       X = input("Machine C Old(3) or New(1):");
       MA1 = .4/S;
       MC2 = .1/X;
       MA2 = .667/S;
       MB2 = .667/T;
       MB3 = .227/T;
}

net() {
       place("p1"); init("p1",1);
       place("p2");
```

```
place("p3");
place("p4");
place("p5");
place("p8"); init("p8",2);
place("p9"); init("p9",1);
place("p10");
place("p11");
place("p12"); init("p12",1);
place("p13");
place("p14"); init("p14",1);
place("p15");
place("p16");
place("p17");
place("p20"); init("p20",1);
place("p21");
place("p22");
place("p23"); init("p23",1);
place("p24");
place("p27"); init("p27",1);
place("p28"); init("p28",1);
place("p29");
place("p30");
place("p31");
place("p32");
place("p35");
place("p36");
place("p37");
```

```
place("p38");
place("p39");

trans("t1");        rateval("t1",1000.0);
trans("t2");        rateval("t2",2.0);
trans("t3");        rateval("t3",1000.0);
trans("t4");        rateval("t4",1.428);
trans("t8");        rateval("t8",1000.0);
trans("t9");        rateval("t9",2.0);
trans("t10");       rateval("t10",1000.0);
trans("t11");       rateval("t11",1.25);
trans("t12");       rateval("t12",1000.0);
trans("t13");       rateval("t13",1.428);
trans("t17");       rateval("t17",1000.0);
trans("t18");       rateval("t18",1.25);
trans("t19");       rateval("t19",1000.0);
trans("t20");       rateval("t20",1.428);
trans("t21");       rateval("t21",1000.0);
trans("t22");       rateval("t22",2.0);
trans("t23");       rateval("t23",1000.0);
trans("t24");       rateval("t24",1.428);
trans("t28");       rateval("t28",1000.0);
trans("t29");       rateval("t29",MC2);
trans("t30");       rateval("t30",MA1);
trans("t31");       rateval("t31",MA2);
trans("t32");       rateval("t32",MB2);
trans("t33");       rateval("t33",MB3);
```

```
iarc("t1","p1"); iarc("t1","p12"); iarc("t1","p14");
oarc("t1","p2"); iarc("t2","p2"); oarc("t2","p36");
oarc("t2","p12"); iarc("t3","p3"); iarc("t3","p23");
oarc("t3","p4"); iarc("t4","p4"); oarc("t4","p5");
oarc("t4","p14"); oarc("t4","p23");
iarc("t8","p8");
iarc("t8","p9"); iarc("t8","p20"); oarc("t8","p10");
iarc("t9","p10"); oarc("t9","p9"); oarc("t9","p35");
iarc("t10","p11"); iarc("t10","p12"); iarc("t10","p14");
oarc("t10","p13"); iarc("t11","p13"); oarc("t11","p12");
oarc("t11","p37"); oarc("t11","p20"); iarc("t12","p15");
iarc("t12","p23"); oarc("t12","p16"); iarc("t13","p16");
oarc("t13","p14"); oarc("t13","p17"); oarc("t13","p23");
iarc("t17","p11"); iarc("t17","p12"); iarc("t17","p27");
oarc("t17","p21"); iarc("t18","p21"); oarc("t18","p20");
oarc("t18","p38"); oarc("t18","p12"); iarc("t19","p22");
iarc("t19","p23"); oarc("t19","p24"); iarc("t20","p24");
oarc("t20","p17"); oarc("t20","p23"); oarc("t20","p27");
iarc("t21","p12"); iarc("t21","p27"); iarc("t21","p28");
oarc("t21","p29"); iarc("t22","p29"); oarc("t22","p12");
oarc("t22","p39"); iarc("t23","p23"); iarc("t23","p30");
oarc("t23","p31"); iarc("t24","p31"); oarc("t24","p23");
oarc("t24","p27"); oarc("t24","p32");
iarc("t28","p5");
miarc("t28","p17",2); iarc("t28","p32"); oarc("t28","p1");
moarc("t28","p8",2); oarc("t28","p28"); iarc("t29","p35");
```

```
        oarc("t29","p11"); iarc("t30","p36"); oarc("t30","p3");
        iarc("t31","p37"); oarc("t31","p15"); iarc("t32","p38");
        oarc("t32","p22"); iarc("t33","p39"); oarc("t33","p30");

/*The  net is defined, and the analysis will be conducted */
}

/* the following lines should appear in all programs */

assert() {return(RES_NOERR);}
ac_init() {}
ac_reach() {fprintf(stderr,"/nThe reachibility graph has been generated/n/n");}

/* User-defined output functions */

reward_type ef0() {return(rate("t28"));}
/* throughput */

reward_type ef1() {return(mark("p10"));}
reward_type ef2() {return(mark("p2")+mark("p13")+mark("p21")+mark("p29"));}
reward_type ef4() {return(1.0-mark("p23"));}
/* robot utilization */

reward_type ef5() {return(1.0-mark("p14"));}
reward_type ef6() {return(1.0-mark("p27"));}
reward_type ef7() {return(1.0-mark("p20"));}
/* machine utilization */
```

```
/*Output*/
ac_final() {pr_expected("throughput ",ef0);
            pr_expected("Robot 1 Utilization ",ef1);
            pr_expected("Robot 2 Utilization ",ef2);
            pr_expected("Robot 4 Utilization ",ef4);
            pr_expected("Machine A Utilization ",ef5);
            pr_expected("Machine B Utilization ",ef6);
            pr_expected("Machine C Utilization ",ef7);
            pr_std_average();}
```

APPENDIX 5

SPNP Results

LOW INITIAL MARKING

$P_1 = 1 \quad P_3 = \_ \quad P_{23} = 1$

INPUT: Machine A Old(3) or New(1): = 1

INPUT: Machine B Old(3) or New(1): = 1

INPUT: Machine C Old(3) or New(1): = 3

EXPECTED: throughput = 0.0153808857286

EXPECTED: Robot 1 Utilization = 0.0153808857286

EXPECTED: Robot 2 Utilization = 0.0399903028944

EXPECTED: Robot 4 Utilization = 0.0430837135255

EXPECTED: Machine A Utilization = 0.105935328175

EXPECTED: Machine B Utilization = 0.131742184266

EXPECTED: Machine C Utilization = 0.965249957578

INPUT: Machine A Old(3) or New(1): = 1

INPUT: Machine B Old(3) or New(1): = 3

INPUT: Machine C Old(3) or New(1): = 1

EXPECTED: throughput = 0.033523984539

EXPECTED: Robot 1 Utilization = 0.033523984539

EXPECTED: Robot 2 Utilization = 0.0871623598013

EXPECTED: Robot 4 Utilization = 0.0939047185966

EXPECTED: Machine A Utilization = 0.267211275694

EXPECTED: Machine B Utilization = 0.60548304163

EXPECTED: Machine C Utilization = 0.781018565647

INPUT: Machine A Old(3) or New(1): = 1

INPUT: Machine B Old(3) or New(1): = 3

INPUT: Machine C Old(3) or New(1): = 3

EXPECTED: throughput = 0.0148311008485

EXPECTED: Robot 1 Utilization = 0.0148311008485

EXPECTED: Robot 2 Utilization = 0.038560862206

EXPECTED: Robot 4 Utilization = 0.0415436998557

EXPECTED: Machine A Utilization = 0.108750410419

EXPECTED: Machine B Utilization = 0.285968420606

EXPECTED: Machine C Utilization = 0.932036107568

INPUT: Machine A Old(3) or New(1): = 3

INPUT: Machine B Old(3) or New(1): = 1

INPUT: Machine C Old(3) or New(1): = 1

EXPECTED: throughput = 0.0355997644126

EXPECTED: Robot 1 Utilization = 0.0355997644126

EXPECTED: Robot 2 Utilization = 0.0925593874728

EXPECTED: Robot 4 Utilization = 0.0997192280466

EXPECTED: Machine A Utilization = 0.489275868275

EXPECTED: Machine B Utilization = 0.327518060752

EXPECTED: Machine C Utilization = 0.839018893752

INPUT: Machine A Old(3) or New(1): = 3

INPUT: Machine B Old(3) or New(1): = 1

INPUT: Machine C Old(3) or New(1): = 3

EXPECTED: throughput = 0.0149488162929

EXPECTED: Robot 1 Utilization = 0.0149488162929

EXPECTED: Robot 2 Utilization = 0.0388669223616

EXPECTED: Robot 4 Utilization = 0.0418734349942

EXPECTED: Machine A Utilization = 0.213531804902

EXPECTED: Machine B Utilization = 0.132967052135

EXPECTED: Machine C Utilization = 0.941071239519

INPUT: Machine A Old(3) or New(1): = 3

INPUT: Machine B Old(3) or New(1): = 3

INPUT: Machine C Old(3) or New(1): = 1

EXPECTED: throughput = 0.0305320799866

EXPECTED: Robot 1 Utilization = 0.0305320799866

EXPECTED: Robot 2 Utilization = 0.0793834079652

EXPECTED: Robot 4 Utilization = 0.0855240335759

EXPECTED: Machine A Utilization = 0.483125145075

EXPECTED: Machine B Utilization = 0.592565884829

EXPECTED: Machine C Utilization = 0.762865258787

INPUT: Machine A Old(3) or New(1): = 1

INPUT: Machine B Old(3) or New(1): = 3

INPUT: Machine C Old(3) or New(1): = 3

EXPECTED: throughput = = 0.0148379717715

EXPECTED: Robot 1 Utilization = = 0.0148379717715

EXPECTED: Robot 2 Utilization = = 0.0216126580076

EXPECTED: Robot 3 Utilization = = 0.0169660685984

EXPECTED: Robot 4 Utilization = = 0.0415629461388

EXPECTED: Machine A Utilization = = 0.108758870116

EXPECTED: Machine B Utilization = = 0.286175859058

EXPECTED: Machine C Utilization = = 0.932249554797

INPUT: Machine A Old(3) or New(1): = 3

INPUT: Machine B Old(3) or New(1): = 1

INPUT: Machine C Old(3) or New(1): = 3

EXPECTED: throughput = = 0.0149540230703

EXPECTED: Robot 1 Utilization = = 0.0149540230703

EXPECTED: Robot 2 Utilization = = 0.0185075990661

EXPECTED: Robot 3 Utilization = = 0.0203728609166

EXPECTED: Robot 4 Utilization = = 0.0418880198047

EXPECTED: Machine A Utilization = = 0.213635509528

EXPECTED: Machine B Utilization = = 0.132993115568

EXPECTED: Machine C Utilization = = 0.94114634447

INPUT: Machine A Old(3) or New(1): = 3

INPUT: Machine B Old(3) or New(1): = 3

INPUT: Machine C Old(3) or New(1): = 1

EXPECTED: throughput = = 0.0306376646557

EXPECTED: Robot 1 Utilization = = 0.0306376646557

EXPECTED: Robot 2 Utilization = = 0.0441953150773

EXPECTED: Robot 3 Utilization = = 0.0354626130275

EXPECTED: Robot 4 Utilization = = 0.0858197889515

EXPECTED: Machine A Utilization = = 0.484862153642

EXPECTED: Machine B Utilization = = 0.594532795797

EXPECTED: Machine C Utilization = = 0.763895210177

EXPECTED: throughput = = 0.0143798216136

EXPECTED: Robot 1 Utilization = = 0.0143798216136

EXPECTED: Robot 2 Utilization = = 0.0197886653038

EXPECTED: Robot 3 Utilization = = 0.0175988708916

EXPECTED: Robot 4 Utilization = = 0.0402796123631

EXPECTED: Machine A Utilization = = 0.220153063019

EXPECTED: Machine B Utilization = = 0.285937480366

EXPECTED: Machine C Utilization = = 0.912949295163

HIGH INITIAL MARKING

INPUT: Machine A Old(3) or New(1): = 1

INPUT: Machine B Old(3) or New(1): = 1

INPUT: Machine C Old(3) or New(1): = 3

EXPECTED: throughput = = 0.0159255608857

EXPECTED: Robot 1 Utilization = = 0.0159255608665

EXPECTED: Robot 2 Utilization = = 0.0414064584304

EXPECTED: Robot 4 Utilization = = 0.0446094146301

EXPECTED: Machine A Utilization = = 0.109906268726

EXPECTED: Machine B Utilization = = 0.136449069121

EXPECTED: Machine C Utilization = = 0.999968077062

INPUT: Machine A Old(3) or New(1): = 1

INPUT: Machine B Old(3) or New(1): = 3

INPUT: Machine C Old(3) or New(1): = 1

EXPECTED: throughput = = NaN

EXPECTED: Robot 1 Utilization = = NaN

EXPECTED: Robot 2 Utilization = = NaN

EXPECTED: Robot 4 Utilization = = NaN

EXPECTED: Machine A Utilization = = NaN

EXPECTED: Machine B Utilization = = NaN

EXPECTED: Machine C Utilization = = NaN

INPUT: Machine A Old(3) or New(1): = 1

INPUT: Machine B Old(3) or New(1): = 3

INPUT: Machine C Old(3) or New(1): = 3

EXPECTED: throughput = 0.0158931062717

EXPECTED: Robot 1 Utilization = 0.0158931733636

EXPECTED: Robot 2 Utilization = 0.0413222663171

EXPECTED: Robot 4 Utilization = 0.0445186515121

EXPECTED: Machine A Utilization = 0.118644558228

EXPECTED: Machine B Utilization = 0.302724058355

EXPECTED: Machine C Utilization = 0.99980055532

INPUT: Machine A Old(3) or New(1): = 3

INPUT: Machine B Old(3) or New(1): = 1

INPUT: Machine C Old(3) or New(1): = 1

EXPECTED: throughput = NaN

EXPECTED: Robot 1 Utilization = NaN

EXPECTED: Robot 2 Utilization = NaN

EXPECTED: Robot 4 Utilization = NaN

EXPECTED: Machine A Utilization = NaN

EXPECTED: Machine B Utilization = NaN

EXPECTED: Machine C Utilization = NaN

INPUT: Machine A Old(3) or New(1): = 3

INPUT: Machine B Old(3) or New(1): = 1

INPUT: Machine C Old(3) or New(1): = 3

EXPECTED: throughput = 0.0158665191804

EXPECTED: Robot 1 Utilization = 0.0158665190803

EXPECTED: Robot 2 Utilization = 0.0412529507526

EXPECTED: Robot 4 Utilization = 0.0444440335113

EXPECTED: Machine A Utilization = 0.224146547102

EXPECTED: Machine B Utilization = 0.142606145763

EXPECTED: Machine C Utilization = 0.999963219362

INPUT: Machine A Old(3) or New(1): = 3

INPUT: Machine B Old(3) or New(1): = 3

INPUT: Machine C Old(3) or New(1): = 1

EXPECTED: throughput = NaN

EXPECTED: Robot 1 Utilization = NaN

EXPECTED: Robot 2 Utilization = NaN

EXPECTED: Robot 4 Utilization = NaN

EXPECTED: Machine A Utilization = NaN

NPUT: Machine B Old(3) or New(1): = 3

INPUT: Machine C Old(3) or New(1): = 3

EXPECTED: throughput = = 0.015895312463

EXPECTED: Robot 1 Utilization = = 0.015895377829

EXPECTED: Robot 2 Utilization = = 0.0236782355169

EXPECTED: Robot 3 Utilization = = 0.0176497620586

EXPECTED: Robot 4 Utilization = = 0.0445248276866

EXPECTED: Machine A Utilization = = 0.118649442048

EXPECTED: Machine B Utilization = = 0.302778247778

EXPECTED: Machine C Utilization = = 0.999804066883

INPUT: Machine A Old(3) or New(1): = 3

INPUT: Machine B Old(3) or New(1): = 1

INPUT: Machine C Old(3) or New(1): = 3

EXPECTED: throughput = = 0.0158687182592

EXPECTED: Robot 1 Utilization = = 0.0158687181678

EXPECTED: Robot 2 Utilization = = 0.019284116768

EXPECTED: Robot 3 Utilization = = 0.0219745515453

EXPECTED: Robot 4 Utilization = = 0.0444501932988

EXPECTED: Machine A Utilization = = 0.224184477505

EXPECTED: Machine B Utilization = = 0.142617109799

EXPECTED: Machine C Utilization = = 0.999963416156

INPUT: Machine A Old(3) or New(1): = 3

INPUT: Machine B Old(3) or New(1): = 3

INPUT: Machine C Old(3) or New(1): = 1

EXPECTED: throughput = = NaN

EXPECTED: Robot 1 Utilization = = NaN

EXPECTED: Robot 2 Utilization = = NaN

EXPECTED: Robot 3 Utilization = = NaN

EXPECTED: Robot 4 Utilization = = NaN

EXPECTED: Machine A Utilization = = NaN

EXPECTED: Machine B Utilization = = NaN

EXPECTED: Machine C Utilization = = NaN

INPUT: Machine A Old(3) or New(1): = 3

INPUT: Machine B Old(3) or New(1): = 3

INPUT: Machine C Old(3) or New(1): = 3

EXPECTED: throughput = = 0.0156868843794

EXPECTED: Robot 1 Utilization = = 0.0156868838995

EXPECTED: Robot 2 Utilization = = 0.0217322149873

EXPECTED: Robot 3 Utilization = = 0.0190536861088

EXPECTED: Robot 4 Utilization = = 0.0439408573187

EXPECTED: Machine A Utilization = = 0.241411791994

EXPECTED: Machine B Utilization = = 0.310999122732

EXPECTED: Machine C Utilization = = 0.999874828192

Inspection

INPUT: Inspection - Manual(10) or Automatic(1): = 10

EXPECTED: throughput = 0.0681096183864

EXPECTED: Reject rate of part #1 = 0.0032492663644

EXPECTED: Accept rate of part #1 = 0.0632357188129

EXPECTED: Reject rate of part #2 = 0.00649853272994

EXPECTED: Accept rate of part #2 = 0.126471437648

EXPECTED: Reject rate of part #3 = 0.0032492663644

EXPECTED: Accept rate of part #3 = 0.0632357188129

EXPECTED: Inspection Utilization = 0.999774245428

INPUT: Inspection - Manual(10) or Automatic(1): = 1

EXPECTED: throughput = 0.679715156704

EXPECTED: Reject rate of part #1 = 0.0324267773512

EXPECTED: Accept rate of part #1 = 0.631074990614

EXPECTED: Reject rate of part #2 = 0.0648535546739

EXPECTED: Accept rate of part #2 = 1.26214998068

EXPECTED: Reject rate of part #3 = 0.0324267773512

EXPECTED: Accept rate of part #3 = 0.631074990614

EXPECTED: Inspection Utilization = 0.997747031799

REFERENCES

1. Atabakhche, H., D. S.Barbalho, R. Valette, and M. Courvoisier. "From Petri Net Based PLC's to Knowledge Based Control." *IECON*, (1986): 817-821.

2. Bako, B., and R. Valette, "Towards a Decentralization of Rule-Based Systems Controlled by Petri Nets: an Application to F.M.S.." *Fourth International Symposium on Knowledge Engineering, Technical Sessions, Barcelona*, L.A.A.S. C.N.R.S, Technical Report #90123, Toulouse, France (May 7-11, 1990).

3. Bako, B., and R. Valette, "Software Implementation of Petri Nets and Compilation of Rule-Based Systems." *Lecture Notes in Computer Science 524, Advances in Petri Nets 1991*, G. Rozenberg (editor). Springer-Verlag: 296-316.

4. Boucher, T., M. Jafari, and G. Meredith. "Petri Net Control of an Automated Manufacturing Cell." *Industrial Engineering*, Vol. 17. Nos. 1-4 (1989): 459-463.

5. Brand, K., and J. Kopainsky. "Principles and Engineering of Process Control with Petri Nets." *IEEE Transactions on Automatic Control*, Vol. 33, No. 2 (February, 1988): 138-149.

6. Cardoso, J., R. Vallete, and D. Dubois. "Petri Nets with Uncertain Markings." *Lecture Notes in Computer Science 483, Advances in Petri Nets 1990*, G. Rozenberg (editor). Springer-Verlag: 64-78.

7. Cofrancesco, P., A. Cristoforetti, R. Scattolini. " Petri Nets Based Approach to Software Development for Real Time Control." *IEE Proceedings-D*, Vol. 138, No. 5 (September, 1991): 474-478.

8. Combacau, M., and M. Courvoisier. "Process Failure Diagnosis in F.M.S. Real-Time Control: An Approach Combining Rule-Based Systems and Petri Nets." *IEEE Proceedings, Second Annual Conference on AI, Simulation and Planning in High Autonomy Systems*, Cocoa Beach, Florida. April 1-2, 1991. pp. 174-180.

9. Courvoisier, M., J. M. Bigou, R. Valette, C. Desclaux and K. Benzakour. "The SECOIA Project." *European Conference on Computers in Communication and Control* (Septemper, 1984).

10. Courvoisier, M., R. Valette, A. Sahraoui, and M. Combacau. "Specification and Implementation Techniques for Multilevel Control and Monitoring of F.M.S.." *Computer Applications in Production Engineering*, F. Kimura and A Rolstadas (editors), Elsevier Science Publishers B.V. IFIP (1989): 509-516.

11. Crockett, D., A. Desrochers, F. DiCesare, and T.Ward. "Implementation of a Petri Net Controller for a Machining Workstation." *Proceedings of IEEE International Conference on Robotics and Automation* (March 30 - April 3, 1987).

12. Desrochers, A. "Modeling and Control Using Petri Nets." Modeling and Control of Manufacturing Systems, *IEEE Computer Society Press* (1990): 239-251.

13. Hatono, I., K. Yamagata, and H. Tamura. "Modeling and On-Line Scheduling of Flexible Manufacturing Systems Using Stochastic Petri Nets." *IEEE Transactions on Software Engineering*, Vol. 17, No. 2 (February, 1991).

14. Kasturia, E., F. DiCesare, and A. Desrochers. "Real Time Control of Multilevel Manufuacturing Systems Using Colored Petri Nets." *IEEE International Conference on Robotics and Automation* (April, 1988): 1114-1119.

15. Komoda, N., K. Kera, and T. Kubo. "An Autonomous Decentralized Control System for Factory Automation." *Computer* (December, 1984): 73-83.

16. Konat, H. "Series One Programmable Controllers." *GE Fanuc Automation*, GEK-90842A(1987): 1.

17. Murata, T. "Petri Nets: Properties, Analysis and Applications." *IEEE*, Vol. 17, No. 4 (April, 1989): 541-580.

18. Murata, T., N. Komoda, K. Matsumoto, and K. Haruna. "A Petri Net Based Controller for Flexible and Maintainable Sequence Control and its Applications in Factory Automation." *IEEE Transactions on Industrial Electronics*, Vol. IE-33, No. 1 (February, 1986): 1-8.

19. Nketsa, J., and M. Courvoisier. "A Petri Net Based Single Chip Programmable Controller for Distributed Local Controls." *IEEE, IECON '90, Volume I, Processing and System Control Factory Automation* (November 27-30, 1990): 542-547.

20. Sahraoui, A., and L. Gilhodes. "State Charts, Temporal Logic and Petri Nets to Specify Discrete Event Controllers: A Comparative Study on Descriptive Power." *European Control Conference (ECC) Proceedings*, Vol. 1 (July 2-5, 1991): 1690-1695.

21. Silva, M., and R. Valette. "Petri Nets and Flexible Manufacturing." *E.T.S. Ingenieros Industriales*, Technical Report #E50015, Zaragoza, Spain (January, 1990): 1-43.

22. Thuriot, C., and M. F. Valax. "Interactive Algorithm for Scheduling Based on a Temporal Logic Under Resource Constraints." L.A.A.S. C.N.R.S, Technical Report #89182, Toulouse, France (May 7-11, 1990).

23. Tsukimoto, H. "A Model Matching Design Method for Sequence Control Systems." *IEEE,* ISSN# 0-7803-0233-8/91 (1991): 501-506.

24. Valette, R., J. Cardoso, H. Atabakhche and T. Lemaire. "Petri Nets and Production Rules for Decision levels in FMS Control." *Artificial Intelligence in Scientific Computation: Towards Second Generation Systems*. R. Huber et al. (editors). J.C. Baltzer AG, Scientific Publishing Company. IMACS (1989): 301-305.

25. Zhou, M. C., F. Dicesare, and D. Rudolph. "Control of a Flexible Manufacturing System Using Petri Nets." *IFAC World Congress*, Vol. 9 (August 13-17, 1990): 43-48.

26. Zhou, M.C., F. DiCesare, and D. L. Rudolph. "Design and Implementation of a Petri Net Based Supervisor for a Flexible Manufacturing System." To appear in *Automatica* (November, 1992).

27. Zhou, M. C., K. McDermott, P. Patel, and T. Tang. "Construction of Petri Net Based Mathematical Models of an FMS Cell." *Proceedings of IEEE International Conference on Systems, Man, and Cybernetics*, Charlottesville, Virginia (October 13-16, 1991).