

AD-A173 681

PERFORMANCE EVALUATION OF DECISIONMAKING ORGANIZATIONS

1/2

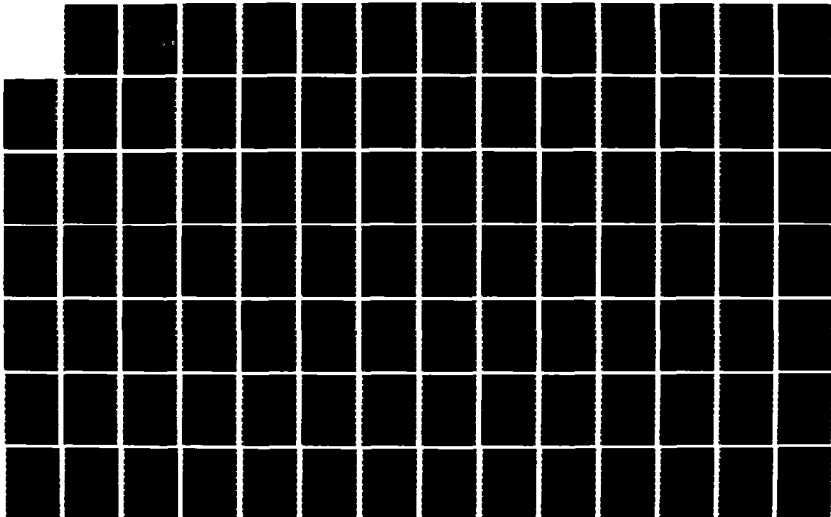
USING TIMED PETRI (U) MASSACHUSETTS INST OF TECH
CAMBRIDGE LAB FOR INFORMATION AND D H P HILLION

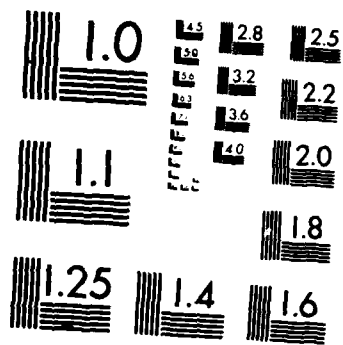
UNCLASSIFIED

AUG 86 LIDS-TH-1590 N00014-85-K-0782

F/G 9/2

NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

AD-A173 681

AUGUST, 1986

UDS-TH-1590

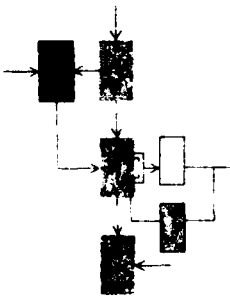
13

Research Supported By:

*Joint Directors of Laboratories
Office of Naval Research
Contract N00014-85-K-0782*

*Office of Naval Research
Contract N00014-84-K-0519*

*National Science Foundation
Grant No. 84-19885SES*



**PERFORMANCE EVALUATION OF
DECISIONMAKING ORGANIZATIONS
USING TIMED PETRI NETS**

Herve P. Hillion

DTIC
ELECTE
OCT 31 1986
S D

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

Laboratory for Information and Decision Systems
MASSACHUSETTS INSTITUTE OF TECHNOLOGY, CAMBRIDGE, MASSACHUSETTS 02139

DTIC FILE COPY

86 10 21 032

AUGUST 1986

LIDS-TH-1590

PERFORMANCE EVALUATION OF DECISIONMAKING ORGANIZATIONS
USING TIMED PETRI NETS

by

Herve P. Hillion

This report is based on the unaltered thesis of Herve P. Hillion, submitted to the Department of Mechanical Engineering in partial fulfillment of the requirements for the degree of Master of Science in Technology and Policy at the Massachusetts Institute of Technology in August 1986. The research was conducted at the MIT Laboratory for Information and Decision Systems with support provided primarily by the Joint Directors of Laboratories under Contract No. N00014-85-K-0782 and also by the Office of Naval Research under Contract No. N00014-84-K-0519 and the National Science Foundation under Grant No. 84-19885SES.



Laboratory for Information and Decision Systems
Massachusetts Institute of Technology
Cambridge, MA 02139

Accession For	
NTIS CRA&I	☒
DTIC TAB	☐
Unannounced	☐
Justification	
By <i>Dr. on file</i>	
Distribution /	
Availability Codes	
Dist	Avail and/or Special
A-1	

PERFORMANCE EVALUATION OF DECISIONMAKING ORGANIZATIONS
USING TIMED PETRI NETS

by

HERVE P. HILLION

Ingenieur de l'Ecole Polytechnique
(1983)

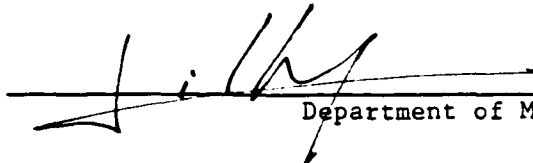
SUBMITTED TO THE
DEPARTMENT OF MECHANICAL ENGINEERING
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE
IN
TECHNOLOGY POLICY

at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

© Massachusetts Institute of Technology

Signature of Author



Department of Mechanical Engineering
August 15, 1986

Certified by



Alexander H. Levis, Thesis Supervisor

Accepted by

Richard de Neufville
Chairman, Technology and Policy Program

Accepted by

Ain A. Sonin, Chairman
Departmental Committee on Graduate Studies
Department of Mechanical Engineering

PERFORMANCE EVALUATION OF DECISIONMAKING ORGANIZATIONS
USING TIMED PETRI NETS

by

HERVE P. HILLION

submitted to the Department of Mechanical Engineering
on August 15, 1986
in partial fulfillment of the requirements for the degree of
Master of Science in Technology and Policy

ABSTRACT

Decisionmaking organizations are modeled as asynchronous concurrent systems, using Timed Petri Nets. The modeling allows for evaluating the time-related performances, with respect to the following measures: The maximum throughput rate, defined as the maximum processing rate achievable by the system, and the execution schedule, which determines the earliest instants at which the different operations can occur in the process. These measures of performance are analyzed and expressed as a function of the resource and time constraints that are specific to the organization. The characterization obtained makes it possible to compare different organizational forms and to modify existing designs so as to improve performance.

Thesis Supervisor: Alexander H. Levis
Title: Senior Research Scientist

ACKNOWLEDGEMENT

I would like to thank Dr. Alexander H. Levis for his guidance and support throughout the work on this thesis. Working with him was a very valuable experience.

I would like also to thank Lisa Babine for her excellent typing and her endless patience at completing the final document.

The suggestions of Dr. John Brode at the early stages of this thesis were very helpful in setting the direction of the work.

Finally, I thank the very good friends I have known at MIT and whose moral support has always been helpful to me.

The Petri Nets shown in the Figures were obtained with a customized beta version of "Design" provided by its developer, Mr. Robert Shapiro of Meta Software, Inc., Cambridge, MA.

This research was carried out at the MIT Laboratory for Information and Decision Systems, with support provided primarily by the Joint Directors of Laboratories under Contract No. N00014-85-K-0782 and also by the Office of Naval Research under Contract No. N00014-84-K-0519 and the National Science Foundation under Grant No. 84-19885SES.

TABLE OF CONTENTS

	<u>Page</u>
Abstract	2
Acknowledgement	3
List of Figures	8
List of Tables	10
 CHAPTER I: INTRODUCTION	 11
1.1 Background	11
1.2 Goals	12
1.3 Approach	13
1.4 The Thesis in Outline	15
 CHAPTER II: REVIEW OF PETRI NET THEORY	 17
2.1 Basic Definitions	17
2.1.1 Structure of a Petri Net	17
2.1.2 Incidence Matrix	18
2.1.3 Marked Petri Net	19
2.1.4 Rules of Operation	20
2.1.5 Forward Marking Class	22
2.1.6 Liveness and Boundedness	22
2.1.7 Application of the Petri Net Model	23
2.2 S and T-Invariants	24
2.2.1 Definitions	25
2.2.2 Example of S-Invariants	25
2.2.3 Properties of S-Invariants	27
2.2.4 T-Invariants	29

2.2	Petri Net Theory and Graph Theory	29
2.3.1	Definitions from Graph Theory	30
2.3.2	Event-Graph: Definitions and Properties	31
2.3.3	S-Invariants of Event-Graphs	34
2.4	Timed Petri Nets	38
2.4.1	Definitions	38
2.4.2	Timed Event-Graph	41
CHAPTER III: PETRI NET DESIGN OF DECISIONMAKING ORGANIZATIONS		45
3.1	Aggregated Model of the Interacting Decisionmaker	45
3.2	Model of the Interacting Decisionmaker with Limited Resources	47
3.3	Aggregated Model of the DMO	48
3.3.1	Model of the DMO Interacting with the Environment	48
3.3.2	Model of the DMO with Limited Resources	55
3.4	Modeling Time Constraints	57
3.5	Model of the DMO as a Strongly Connected Timed Event-Graph ..	59
CHAPTER IV: MAXIMUM THROUGHPUT RATE OF THE DMO		63
4.1	Properties of the Petri Net Model	63
4.2	Computation of the Maximum Throughput Rate	64
4.2.1	Average Cycle Time of Transitions	65
4.2.2	Average Circuit Processing Time	71
4.2.3	Maximum Average Circuit Processing Time	76
4.2.4	Maximum Throughput Rate	83
4.2.4.1	Deterministic System	84
4.2.4.2	Non-Deterministic System	86

4.3	Application to the Two Member Organization	89
4.3.1	Analysis of the Processing Rate Constraints	89
4.3.2	Evaluation of the Maximum Throughput Rate	95
4.3.2.1	Example 1	95
4.3.2.2	Example 2	97
4.3.2.3	Example 3	98
4.4	Conclusion	99
CHAPTER V: ANALYSIS OF EXECUTION SCHEDULES		101
5.1	Analysis of Sequential and Concurrent Operations	101
5.1.1	Representation of the Process as an Occurrence Net	102
5.1.2	Characterization of Lines and Slices	103
5.2	Execution Schedule of the DMO	106
5.2.1	Assumptions of the Model	106
5.2.2	Feasible Firing Schedule	107
5.2.3	Determination of the Execution Schedule	108
5.2.4	Properties of the Execution Schedule	113
5.3	Measures of Performance from the Execution Schedule	116
5.3.1	Execution Schedule of the Input Transition	117
5.3.2	Execution Schedule of the Output Transition	119
5.4	Conclusion	124
CHAPTER VI: APPLICATION		125
6.1	Presentation of the Model	125
6.2	Performance Evaluation of the Five DM Organization	129
6.2.1	Structural Characteristics of the Model	129

6.2.2	Measures of Performance	133
6.2.3	Numerical Results	136
6.2.3.1	First Case	136
6.2.3.2	Second Case	137
6.2.3.3	Third Case	142
6.2.3.4	Fourth Case	145
6.2.4	Summary Analysis of the Results	147
6.2.4.1	Comparison of the Performance Measures	147
6.2.4.2	Execution Schedule of the Concurrent Tasks ..	149
CHAPTER VII: CONCLUSIONS AND DIRECTIONS FOR FURTHER RESEARCH		151
7.1	Conclusions	151
7.2	Directions for Further Research	153
REFERENCES		155
APPENDIX A: ALGORITHM TO OBTAIN ALL THE CIRCUITS OF AN EVENT-GRAPH ...		157
A.1	Summary	157
A.2	Algorithm to Obtain All the Minimal Support S-Invariants of a General Petri Net	157
A.3	Determination of the Directed Circuits for an Event-Graph ...	159
A.4	Example	160
APPENDIX B: DEMONSTRATION OF THEOREM 5.1		163
APPENDIX C: ALGORITHM TO COMPUTE THE FIRING SCHEDULE		167
C.1	Summary Analysis of the Firing Schedule	167
C.2	Description of the Algorithm	171

LIST OF FIGURES

	<u>Page</u>
2.1 Example of a Petri Net	18
2.2 Example of a Marked Petri Net	19
2.3 Example of a Live Petri Net	23
2.4 Example of a Strongly Connected Net	30
2.5 Example of an Event-Graph	32
2.6 Example of a Live Event-Graph	32
2.7 An Example of an S-Component	34
2.8 Timed Petri Net Equivalence	40
2.9 Representation of a Self-Loop	42
2.10 Model of Transition Firing	43
3.1 Model of the Interacting Decisionmaker	46
3.2 Model of Interacting Decisionmaker with Limited Resources	49
3.3 Model of the Partitioning Operation	53
3.4 Model of the DMO Interacting with the Environment	54
3.5 Model of the DMO with Limited Resources	56
3.6 Petri Net Model of an n-Decision Switch	57
3.7 Two Member Organization (Structure (a))	61
3.8 Two Member Organization (Structure (b))	61
4.1 Directed Circuit Containing t_i and t_j	68
4.2 Directed Circuit	72
4.3 Interconnected Circuits	77
4.4 Non-Deterministic Example	87

5.1	Model of the Two Member Organization as an Occurence Net	102
5.2	Input Places and Transitions of t_j	110
5.3	K-Periodic Sequence with Period π	114
5.4	2-Periodic Firing Schedule	118
6.1	Petri Net Model of the Five DM Organization	126
6.2	Petri Net Representation of the SA Stage with Preprocessor (DM2) .	127
6.3	Model of the Five DM Organization With Limited Resources	128
6.4	Directed Circuit (Case 1)	139
6.5	Directed Circuit (Case 2)	142
6.6	Directed Circuit (Case 3)	144
A.1	Example of an Event-Graph	160
C.1	Input Places and Transitions of t_j	168
C.2	Flow-Chart of the Algorithm that Computes the Firing Schedule	172

LIST OF TABLES

	<u>Page</u>
6.1 Incidence Matrix	130
6.2 List of the Directed Circuits	131
6.3 List of Slices	133
6.4 Computation of the Execution Schedule: Case 1	138
6.5 Computation of the Execution Schedule: Case 2	140
6.6 Computation of the Execution Schedule: Case 3	143
6.7 Computation of the Execution Schedule: Case 4	146
6.8 Summary of the Results	148

CHAPTER I

INTRODUCTION

1.1 BACKGROUND

In most decisionmaking organizations (DMO), and especially in military organizations supported by C³ (command, control, and communications) systems, timeliness is of critical importance. The ability of an organization to perform the tasks and transmit information in a timely manner is indeed a determinant factor of effectiveness. There are two types of constraints which affect the time performance of a DMO. The first type is related to the internal organizational structure that determines how the various operations do occur in the process: some tasks are indeed processed one after the other, characteristic of sequential activities, and others are processed independently, which characterizes concurrent activities. The sequential and concurrent events are precisely determined by the communication and execution protocols among the individual organization members, as defined in [1]. The second type of constraints consists of time and resource constraints. More precisely, the time constraints derive from the task execution times, i.e., the amount of time necessary to perform each task. In addition, the organization controls specified resources, which are generally limited. Depending on which of the resources are free at a given instant, certain activities can take place and others must be delayed: Hence, the resource limitations also constrain the processing of inputs.

This set of constraints makes the decisionmaking process occur asynchronously and concurrently in real-time. Indeed, the coordination among the concurrent tasks occurs asynchronously and not at set times. At this point, the Petri Net formalism, introduced in [2] for modeling the DMO, provides a convenient tool for analyzing the behavior of systems that exhibit asynchronous and concurrent properties. The Petri Net framework

will therefore be used in this thesis to evaluate the performance of an organization, with respect to time-related measures.

1.2 GOALS

In earlier work [3], the evaluation of a particular measure of performance (MOP), the response time of the DMO, has been carried out, using the Petri Net representation. This MOP characterizes the time interval between the moment one external input or task is received by the organization and the moment a response can be made. The response time so computed is a static measure, in so far as the processing is supposed to take place for a single input arriving at any instant of time.

In the analysis developed here, we want to investigate the dynamic behavior of the DMO, for successively arriving inputs. More precisely, the objective of this research task is to evaluate the performance of a DMO, with respect to the following time-related measures:

- (a) Maximum Throughput Rate of the system. Clearly enough, if we assume that external inputs are arriving continuously at a rate which is low enough, the DMO will be able to handle all inputs as soon as they arrive. Then, the rate at which inputs are being processed, which characterizes the throughput rate of the organization, will precisely correspond to the arrival rate of inputs. However, beyond a certain arrival rate of inputs, the DMO will be overloaded: inputs will queue at the entry of the system and this queue will grow to infinity over time, meaning that the DMO will never catch up to all the inputs. This bound precisely determines the maximum throughput rate of the system, i.e., the maximum rate at which inputs can be processed. This MOP characterizes therefore the best performance that can be achieved by the organization, with respect to the processing rate.

- (b) The Execution Schedule of the organization. Let us assume that the processing starts at $\tau=0$ and that it occurs repetitively at its maximum throughput rate. We want to determine the earliest instants of time at which the various tasks can occur in the repetitive process. In our analysis, the DMO will be allowed to process simultaneously more than one input: we will therefore obtain dynamic measures of performance. In particular, if we assume that several inputs arrive simultaneously, it will be possible to determine the response time, which, in that case, corresponds to the time interval between the moment inputs were received and the moment a response to each of the inputs was made.

These MOPs, as described above, are important in the performance evaluation of an organization. Suppose, for instance, that the goal of a C³ system is to detect, track, and identify threats and allocate weapons to them. If several threats arrive simultaneously (or within a short period of time) it will be possible, from the execution schedule, to evaluate the ability of the system to respond to all these threats in a timely manner. Likewise, if a large number of threats arrive successively with a rate that is above the maximum throughput rate of the system, we know that the system will be overloaded.

1.3 APPROACH

The performance evaluation is based on the Petri Net modeling of decisionmaking organizations, as introduced in [2]. However, some extensions are introduced, in order to account for the time and resource constraints. The resource constraints actually occur for both individual decisionmakers (DMs) and the organization as a whole. In our approach, the resource limitation existing for individual DMs allows for modeling the limited capacity of information-processing that characterizes a human DM, as described in the information theoretic framework by the bounded rationality constraint [4]. More precisely, according to the analysis

carried out in [5], the resource constraint is derived from the limited ability of the human short-term memory to handle a certain amount of information at the same time. Therefore, the limitation of resources is expressed here as a bound on the total number of inputs that a DM is able to process simultaneously. Likewise, the amount of information being transmitted between the DMs is necessarily limited by the storage capacity of the organization. We should recall that the process occurs asynchronously, which implies, generally, that the information exchanged between the DMs has to be stored temporarily. Exactly as for individual DMs, this capacity constraint is modeled as a bound on the total number of inputs that the overall organization can handle at the same time.

As already stated, the time constraints correspond to the task execution times, i.e., the amount of time it takes to process each task. Consequently, arbitrary processing times are assigned to each organizational task, according to the Timed Petri Net model developed by Ramchandani [6]. The performance evaluation of the DMO is therefore carried out using the properties of Timed Petri Nets.

In this work, the maximum throughput rate is expressed directly as a function of both the task processing times and the resources available, given the structure of the organization. In addition, an algorithm is developed, that determines the execution schedule of the organization, when the processing of inputs occurs at its maximum throughput rate. These MOPs characterize in fact the best dynamic performance achievable by the organization, as determined by the maximum rate at which inputs can be processed and the earliest instants of time at which the various tasks can be executed. These measures provide, therefore, a useful tool to evaluate and compare different organizational forms. In particular, the precise characterization of the resource and time constraints makes it possible to investigate which of the constraints should be modified in the actual design, so as to improve the time performance of the organization.

1.4 THE THESIS IN OUTLINE

The thesis is organized as follows. Chapter 2 is a review of Petri Net theory and presents the extended Timed Petri Nets that are used in the analysis. In Chapter 3, the model of the DMO using Timed Petri Nets is discussed. Included in the modeling are the resource and time constraints, as described previously. In Chapter 4, the maximum throughput rate is determined as a function of the time and resource constraints. The analysis of the execution schedule is carried out in Chapter 5 and a method is developed to obtain the precise instants at which the various operations take place in the process. In Chapter 6, an application is developed for a five member organization and the corresponding measures of performance are computed. Finally, conclusions, as well as suggestions for further research are presented in Chapter 7.

CHAPTER II

REVIEW OF PETRI NET THEORY

Petri Net theory forms the framework for the model of the decisionmaking organization analyzed in this thesis. Accordingly, the basic definitions are first reviewed in this chapter, as they are developed in [7]. The important concepts of S and T-invariants [8] are also presented and will be used later to develop algorithms. In addition, the model of Timed Petri Nets [6] is described, since it is the approach taken in this thesis to carry out the performance evaluation of the DMO.

2.1 BASIC DEFINITIONS

2.1.1. Structure of a Petri Net

A Petri Net (denoted by PN) is a quadruple

$$PN = (P, T, I, O)$$

where:

$P = \{p_1, \dots, p_n\}$ is a finite set of places

$T = \{t_1, \dots, t_m\}$ is a finite set of transitions

I is a mapping: $P \times T \longrightarrow \{0,1\}$ corresponding to the set of directed edges from places to transitions

O is a mapping: $T \times P \longrightarrow \{0,1\}$ corresponding to the set of directed edges from transitions to places.

Figure 2.1 shows a simple example of PN.

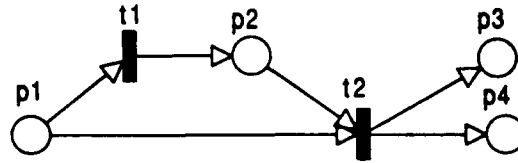


Figure 2.1 Example of a Petri Net

We denote by:

$t^+ = \{p \in P / O(t,p) = 1\}$ the set of all output places of transition t .

$t^- = \{p \in P / I(p,t) = 1\}$ the set of all input places of transition t .

p^+ and p^- are defined for each place p in a similar way.

In the example above: $t_1^- = \{p_1, p_2\}$, $t_1^+ = \{p_3, p_4\}$.

2.1.2 Incidence Matrix

The Incidence Matrix (also called Flow Matrix), denoted by C , characterizes the structure of a PN in the following way: The columns of the matrix correspond to the transitions of the net and the rows, to the places. C is therefore, a $n \times m$ matrix, such that:

$$C_{ij} = \begin{cases} -1 & \text{if } p_i \text{ is an input place of } t_j \\ +1 & \text{if } p_i \text{ is an output place of } t_j \\ 0 & \text{if there is no arc between } p_i \text{ and } t_j \end{cases}$$

We can easily verify that:

$$C_{ij} = O(t_j, p_i) - I(p_i, t_j) \quad (2.1)$$

The Petri Net of Figure 2.1 has, for example, the following incidence matrix:

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

2.1.3 Marked Petri Net

We call marking of a PN, denoted by M , a mapping: $P \longrightarrow \{0,1,2,\dots\}$ (set of integers) which assigns a non-negative integer number of so called tokens to each place of the net.

Figure 2.2 shows an example of marking for the PN considered in Section 2.1.1. Dots represent tokens. The marking is usually denoted as a n -dimensional integer vector, with each element corresponding to one of the places. For the example of Figure 2.2:

$$M = (2, 0, 1, 0)^T$$

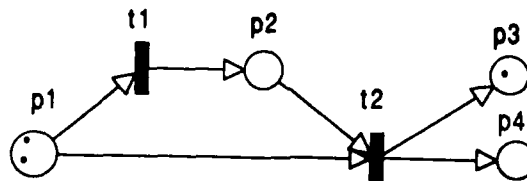


Figure 2.2 Example of a Marked Petri Net

2.1.4 Rules of Operation

- (i) A transition t is enabled by a marking M^0 if every input place

of this transition contains at least one token, i.e.,

for all $p \in t$, $M^0(p) \geq 1$

- (ii) Every transition enabled by a marking M^0 can fire. When a transition fires, a token is removed from each of its input places and is added in each of its output places. Therefore, the new marking M^1 , obtained by the firing of transition t , verifies, for each place p :

$$M^1(p) = \begin{cases} M^0(p) - 1 & \text{if } p \in t \\ M^0(p) + 1 & \text{if } p \in t^+ \\ M^0(p) & \text{otherwise} \end{cases} \quad (2.2)$$

For example, in the PN shown in Figure 2.2, t_1 is enabled, but not t_2 . After t_1 has fired once, the new marking is:

$$M^1 = (1, 1, 1, 0)^T$$

We denote by:

$$M^0 \xrightarrow{t} M^1$$

the new marking reached from M^0 by the firing of transition t .

Let us assume now that transition t_j ($j \in \{1, 2, \dots, m\}$) fires. We can easily verify that relation (2.2) can be written, for transition t_j and all places p_i ($i \in \{1, 2, \dots, n\}$), in the form:

$$M^1(p_i) = M^0(p_i) + O(t_j, p_i) - I(p_i, t_j)$$

or, using relation (2.1):

$$M^1(p_i) = M^0(p_i) + C_{ij} \quad \text{for } j \text{ and all } i=1,2,\dots,n$$

or, in a more compact form:

$$M^1 = M^0 + C X_j \quad (2.3)$$

where X_j is a m -dimensional vector with all components equal to zero except the j -th one, which equals 1:

$$X_j = [0, 0, \dots, 1, \dots, 0]^T$$

We can now consider a sequence of transition firings, denoted by:

$$\sigma_s = t_{j_1}, t_{j_2}, \dots, t_{j_s}$$

which means that transition t_{j_1} fires first, then transition t_{j_2} , and so on until transition t_{j_s} . We denote by

$$M^0 \xrightarrow{\sigma_s} M^s$$

the marking reached from the initial marking M^0 by firing the sequence σ_s . Let us construct the m -dimensional integer vector:

$$N_s = [n_1, n_2, \dots, n_m]^T$$

where n_j ($j=1,2,\dots,m$) denotes the number of occurrences of transition t_j in the sequence σ_s . Then relation (2.3) is generalized in the form [9]:

$$M^s = M^0 + C N_s \quad (2.4)$$

The algebraic equation (2.4) allows for computing directly the new marking reached by any sequence of transition firings. Let us remark however, that some information is lost when using the vector N_g , since the order of the firing sequence is not specified.

2.1.5 Forward Marking Class

Given an initial marking M^0 of the net, we call forward marking class, denoted by $\vec{M}^0$, the set of all possible reachable markings. In other words, a marking M belongs to $\vec{M}^0$ if there exists a firing sequence σ , such that:

$$M^0 \xrightarrow{\sigma} M$$

In the example shown in Figure 2.2, the forward marking class consists of the following markings:

$$\begin{aligned} M^0 &= (2, 0, 1, 0)^T && \text{(initial marking)} \\ M^1 &= (1, 1, 1, 0)^T && \text{(after firing of } t_1) \\ M^2 &= (0, 0, 2, 1)^T && \text{(after firing of } t_2) \\ M^3 &= (0, 2, 1, 0)^T && \text{(after two successive firings of } t_1) \end{aligned}$$

2.1.6 Liveness and Boundeness

Liveness: A marking M^0 is live if, for any transition t , and for every reachable marking M (i.e., $M \in \vec{M}^0$), there exists a firing sequence from M which fires t . In other words, this property guarantees the firing process to be deadlock free.

The example of Figure 2.2 is trivially not live: from M^3 , it is not possible to fire any more transitions. Figure 2.3 shows a simple example of a live Petri Net.

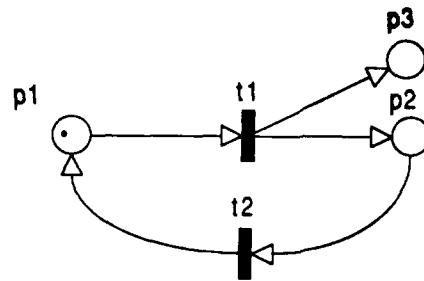


Figure 2.3 Example of a Live Petri Net

Boundedness: A marking M^0 is bounded, if there exists a positive integer N such that, for every reachable marking M , the number of tokens in each place is bounded by N . If N equals one, the marking is said to be safe.

The example of Figure 2.2 is trivially bounded (by 2). However, the example above (Figure 2.3) is not bounded. Each time that t_1 fires, one token is added in p_3 . Since t_1 can fire an infinite number of times, the marking of p_3 is not bounded.

Remark: There exists an extension to the definition of liveness (resp. boundedness), which is called structural liveness (resp. structural boundedness) [10]. In that case the properties are characteristics of the PN, independently of the initial marking. Here are the definitions:

- (1) A PN is (structurally) live if there exists at least one initial marking which is live.
- (2) A PN is (structurally) bounded if any initial marking is bounded.

2.1.7 Application of the Petri Net Model

Petri Nets are useful for modeling the flow of information and control

in systems, especially those which exhibit asynchronous and concurrent properties, as it is the case for the decision making process studied in this thesis. To model the dynamic behavior of a system, the execution of a process (or task) is represented by the firing of the corresponding transition. The flow of tokens represents the flow of information in the process and the marking of the net at any instant corresponds to a particular state of the system. The forward marking class determines the set of all possible states, given an initial state of the system. The two properties defined above, namely liveness and boundedness, characterize a well defined system. Indeed, they guarantee that first, the process is deadlock free (liveness) and second, that information does not accumulate at some steps of the process (boundedness). It becomes possible, therefore, to analyze the steady-state of the process.

In the next sections of this Chapter, we are going to present some extensions to the basic Petri Net model, that will be used later in our development.

2.2 S AND T-INVARIANTS

S and T-invariants are important concepts in Petri Net theory [8] and will be used later to describe algorithms. This is the reason why they are presented in this Chapter. S and T-invariants are dual in the following sense: Let us consider a Petri Net in which we replace all the places by transitions and conversely. The Petri Net obtained is called the dual [11] of the original Petri Net and it is easily proved that the incidence matrix of the new PN is the transpose of the incidence matrix of the original PN. We will show later that the S-invariants of a PN are the T-invariants of the dual and conversely. This is the reason why we will focus in this section on S-invariants only. The properties of T-invariants will be obtained directly from the analysis of S-invariants.

2.2.1 Definitions

We recall that C represents the incidence matrix of a PN, as defined in Section 2.1.2. C is a $n \times m$ matrix, where n is the number of places and m the number of transitions.

Definition: An n -dimensional positive integer vector X is called an S-invariant if and only if $X^T C = 0$.

Definition: The set of places whose corresponding components in X are strictly positive is called the support of X and is denoted by $||X||$.

Definition: The support $||X||$ of an S-invariant, X , is said to be minimal if and only if it does not contain the support of any other invariant, but itself and the empty set.

2.2.2 Example of S-Invariants

Let us consider again the example of Figure 2.1. We recall that the Incidence Matrix is:

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

An S-invariant: $X = [x_1, x_2, x_3, x_4]^T$ (where x_1, x_2, x_3 and x_4 are positive or null integer) satisfies:

$$X^T C = 0$$

which yields the following system of equations:

$$\begin{cases} -x_1 + x_2 = 0 \\ -x_1 - x_2 + x_3 + x_4 = 0 \end{cases}$$

Therefore, the general form of an S-invariant is:

$$X = [x, x, y, z]^T$$

such that:

$$y + z = 2x$$

In particular, it turns out that:

$$X_1 = [1 \ 1 \ 2 \ 0]^T$$

$$X_2 = [1 \ 1 \ 1 \ 1]^T$$

$$X_3 = [1 \ 1 \ 0 \ 2]^T$$

are all S-invariants, whose supports are respectively:

$$||X_1|| = \{p_1, p_2, p_3\}$$

$$||X_2|| = \{p_1, p_2, p_3, p_4\}$$

$$||X_3|| = \{p_1, p_2, p_4\}$$

Now, the support of any non-null S-invariant, $X = [x, x, y, z]$, contains necessarily p_1 and p_2 . Otherwise we would have $x=0$ and therefore $y = z = 0$ (since y and z are positive integers verifying: $y + z = 2x$). In addition, y and z cannot be both null simultaneously, which implies that the support of any non-null S-invariant also contains at least p_3 or p_4 . Accordingly,

$||X_1||$ and $||X_2||$ are minimal support of S-invariants. By construction, both are in fact the only minimal supports, since we have just proved that the support of any non-null S-invariant necessarily contains $\{p_1, p_2, p_3\}$ (i.e., $||X_1||$) or $\{p_1, p_2, p_4\}$ (i.e., $||X_2||$).

2.2.3 Properties of S-invariants

The fundamental property of an S-invariant, which justifies "a posteriori" its name, is the following one:

Theorem 2.1: X is an S-invariant iff for any initial marking M^0 and for any reachable marking M (i.e., $M \in \vec{M}^0$),

$$X^T M = X^T M^0 \quad (2.5)$$

The proof is straightforward from Equation (2.4) and from the definition of an S-invariant, i.e., $X^T C = 0$.

Relation (2.5) is very similar to an equation of conservation: it implies indeed the conservation of tokens for the places belonging to the support $||X||$ and weighted by certain coefficients.

For instance, in the example of Section 2.2.2, the three S-invariants yield the following equations of conservation:

$$M(p_1) + M(p_2) + 2M(p_3) = M^0(p_1) + M^0(p_2) + 2M^0(p_3) \\ \text{(for } X_1 = [1 \ 1 \ 2 \ 0]^T \text{)}$$

$$M(p_1) + M(p_2) + 2M(p_4) = M^0(p_1) + M^0(p_2) + 2M^0(p_4) \\ \text{(for } X_2 = [1 \ 1 \ 0 \ 2]^T \text{)}$$

$$\text{and} \quad \sum_{i=1}^4 M(p_i) = \sum_{i=1}^4 M^0(p_i) \quad \text{(for } X_3 = [1 \ 1 \ 1 \ 1]^T \text{)}$$

This last equation is especially easy to interpret: there is neither creation nor loss of tokens in the net.

The next theorem is the basis of algorithmic procedures to determine all the S-invariants of a Petri Net:

Theorem 2.2: (Memmi-Sifakis) [8]: Let I_0 be the support of an invariant and $I_1, \dots, I_k$ be the minimal supports contained in I_0 , then:

$$- I_0 = \bigcup_{i=1, k} I_i$$

- for every invariant X such that $||X|| = I_0$, there exists positive rational coefficients λ_i ($i=1, k$) such that:

$$X = \sum_{i=1}^k \lambda_i X_i$$

where X_i is an invariant whose support is I_i .

Corollary 2.2: Let $I_1, \dots, I_s$ be all the minimal supports of S-invariants of a PN (they are necessarily finite, since the number of places is finite). Let X_i be an invariant whose support is I_i , i.e., $||X_i|| = I_i$ (we can choose actually the minimal one, as determined by the partial ordering $\geq$ between integer vectors). Then the set $\{X_1, \dots, X_s\}$ forms a minimal set of generators of all invariants, i.e.,

-for every invariant X , there exists positive rational coefficients λ_i , such that

$$X = \sum_{i=1}^s \lambda_i X_i$$

-if one element is removed from the set - let us say X_k for example - the set $\{X_1, \dots, X_s\} - \{X_k\}$ is not generator.

In the example of Section 2.2.2, $I_1 = \{p_1, p_2, p_3\}$ and $I_2 = \{p_1, p_2, p_4\}$ are the minimal supports. $X_1 = [1 \ 1 \ 2 \ 0]^T$ and $X_2 = [1 \ 1 \ 0 \ 2]^T$ form the minimal set of generators and we can check for example that:

$$X_3 = [1 \ 1 \ 1 \ 1]^T = \frac{1}{2} X_1 + \frac{1}{2} X_2$$

$$\text{i.e., } \lambda_1 = \lambda_2 = \frac{1}{2}$$

2.2.4 T-Invariants

As stated earlier, T-invariants can be considered as duals of S-invariants. A T-invariant is therefore a m -dimensional positive integer vector Y such that

$$C Y = 0 \tag{2.6}$$

The set of transitions whose corresponding components in Y are strictly positive is called the support of Y and is denoted by $||Y||$. The definition of the minimal support is the same as for S-invariants, but it refers now to a set of transitions. Likewise, Theorem 2.2 holds also for T-invariants.

From equation (2.6), a T-invariant can be interpreted as a positive or null integer assignment to each transition of the net such that, at every place, the sum of integers assigned to its input transitions, equals the sum of integers assigned to its output transitions.

2.3 PETRI NET THEORY AND GRAPH THEORY

Formally, a Petri Net can be defined as a bipartite directed graph,

whose nodes are places (also called circle nodes) and transitions (also called bar nodes) and whose arcs correspond to the directed edges between places and transitions. We give first some definitions that come directly from Graph Theory.

2.3.1 Definition from Graph Theory

Connectivity: A Petri Net is said to be connected if and only if there exists a path (not necessarily directed) from any node (place or transition) to any other node.

Strong Connectivity: A Petri Net is said to be strongly connected if and only if there exists a directed path from any node to any other node.

Simple connectivity implies only that a PN cannot be split in two or more Petri Nets that are independent. All the examples of PNs presented previously are simply connected. Strong connectivity is however a much more stringent characteristic. The example of Figure 2.1 is obviously not strongly connected: there is, for example, no directed path between p_2 and p_1 . Figure 2.4 shows an example of a strongly connected net.

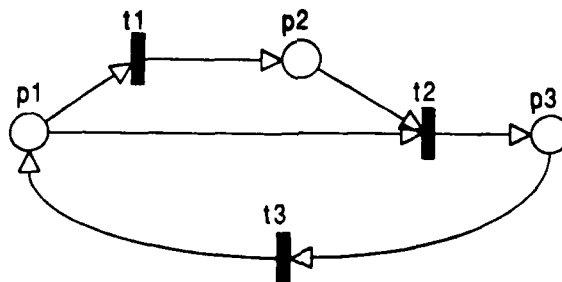


Figure 2.4 Example of a Strongly Connected Net

Directed Elementary Circuit: A directed elementary circuit is a directed path from one node back to itself such that none of the nodes are repeated.

Naturally, the condition that none of the nodes should be repeated refers to the fact that the circuit is elementary. A directed path from a node back to itself is, generally speaking, a directed circuit (not necessarily elementary). However, we will always consider, in the sequel, only the elementary circuits of a PN. For simplification, what will be called later directed circuits, shall always refer to elementary circuits.

In the example of Fig. 2.4, the sequence $\rho_1 = (p_1, t_1, p_2, t_2, p_3, t_3)$ constitutes a directed elementary circuit. However, the sequence $\rho = (p_1, t_1, p_2, t_2, p_3, t_3, p_1, t_1, p_2, t_2, p_3, t_3)$ is a directed circuit, which is not elementary.

2.3.2 Event-Graph: Definition and Properties

Event-Graphs constitute a special class of Petri Nets, whose properties are especially appropriate for analyzing the dynamic behavior of a system. In this section, we present the relevant definitions and properties that will be used later in this thesis.

Definition: An Event-Graph [9] (also known as Marked Graph [12]) is a connected Petri Net, in which each place has exactly one input and one output transition.

This means that tokens are always generated, at a given place, by a predefined transition (its only input transition) and consumed by a predefined transition (its only output transition). Neither the example of Figure 2.1, nor the example of Figure 2.4 are Event-Graphs. Indeed, place p_1 has, in both cases, two output transitions. Figure 2.5 is an example of an Event-Graph. We should point out that, in this particular example, the input transition (i.e., transition t_1) has no input places. This is allowable in Petri Net theory and simply means that the corresponding transition is always enabled. Otherwise stated, transition t_1 can fire any number of times at any instant.

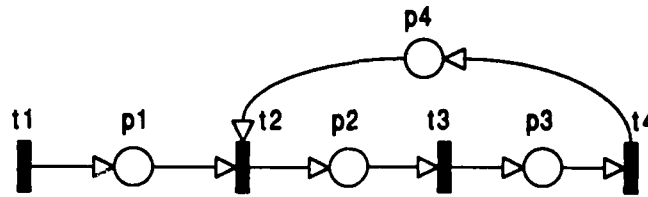


Figure 2.5 Example of an Event-Graph

The following two theorems, due to Commoner and Holt [12], describe useful properties of Event-Graphs:

Theorem 2.3: In an Event-Graph, the number of tokens in any directed elementary circuit (called the token content) remains invariant by transition firings.

Theorem 2.4: A marking of an Event-Graph is live if and only if the token content of every directed elementary circuit is strictly positive.

For instance, the marking of the event-graph shown in Figure 2.6 is live: The token content of the unique circuit $(t_2, p_2, t_3, p_3, t_4, p_4)$ is one.

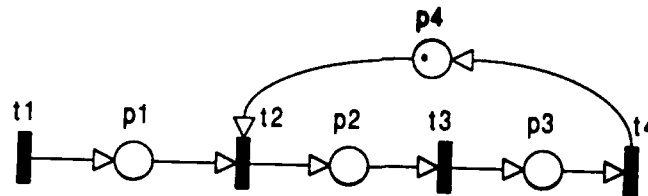


Figure 2.6 Example of a Live Event-Graph

The other interesting result relates the boundedness property to the condition of strong connectivity in the following way:

Theorem 2.5: An Event-Graph is bounded if and only if it is strongly connected.

In order to prove this theorem, we use a result obtained by Sifakis [10], stating that a Petri net, which is (structurally) live and bounded, is necessarily strongly connected. Let us now prove Theorem 2.5.

- (i) Let us assume that the Event-Graph is bounded. From Theorem 2.4, we know that an Event-Graph is always (structurally) live: Indeed, it is sufficient to choose an initial marking, such that the token content of every directed elementary circuit is strictly positive. Given the result stated above, the Event-Graph is therefore strongly connected, since it is live and bounded.
- (ii) Let us assume now that the Event-Graph is strongly connected and let p_i (resp. t_j) be any place (resp. any transition). There exists a directed path from p_i to t_j , that we denote by σ_{ij} , and a directed path from t_j to p_i , that we denote σ_{ji} . Now, the sequence of nodes belonging to the union of the two paths, $\sigma_{ij} + \sigma_{ji}$, determines a circuit in the net. The token content is invariant in the circuit (Theorem 2.3) and the marking of p_i is therefore bounded by this token content. Hence the Event-Graph is bounded. Q.E.D.

We can check, for instance, that the PN of Figure 2.6 is not bounded: t_1 can fire any number of times, as said previously, so that the marking of p_1 is never bounded. Indeed, the PN is not strongly connected.

It turns out that Theorem 2.4 and 2.5 provide a very easy way to check the two important properties - liveness and boundedness - that are characteristics of a well-defined system, if the system is modeled as an Event-Graph.

2.3.3 S-Invariants of Event-Graphs

Before characterizing the S-invariants of an Event-Graph, we need a further definition: S-components [10].

Definition: Let X be any S-invariant of a PN and $||X||$ its support. We recall that $||X||$ is a set of places. We call S-component the unique subnet whose set of places is precisely $||X||$ and whose set of transitions consists of all the transitions connected to the places of $||X||$. In other words, the corresponding S-component is the subnet $PN_S = (P_S, T_S, I_S, O_S)$ such that:

$$P_S = ||X||$$

$$T_S = \bigcup_{p \in P_S} (p \cdot \cup \cdot p)$$

(i.e., T_S contains all the input and output transitions of the places of P_S). I_S (res. O_S) is the restriction of I (resp. O) to $P_S \cup T_S$.

Let us take for example the PN shown in Figure 2.1. The corresponding S-component of the S-invariant $X_1 = [1 \ 1 \ 2 \ 0]^T$ is the subnet shown in Figure 2.7.

Given the definition of a minimal support, an S-component is said to be minimal if it corresponds to an S-invariant whose support is minimal, which means also that it does not contain any other S-component but itself and the empty set.

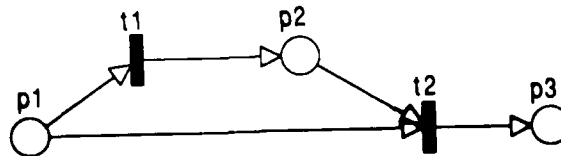


Figure 2.7 An Example of an S-component

The following new result, shows the relationship between the directed circuits and the S-invariants of an Event Graph.

Theorem 2.6: The minimal S-components of an Event-Graph are exactly the directed circuits.

Proof of Theorem 2.6

- (i) Let us prove first that a directed circuit is a minimal S-component. Assume then that ρ is any directed circuit and let us call P_ρ the set of places of the circuit, T_ρ the set of transitions of the circuit. We have trivially:

$$T_\rho = \bigcup_{p \in P_\rho} (p \cdot \cup \cdot p) \quad (2.7)$$

because each place has exactly one input and one output transition. From Theorem 2.3, we know that, for any initial marking M° of the net and for any reachable marking M :

$$\sum_{p \in P_\rho} M(p) = \sum_{p \in P_\rho} M^\circ(p) \quad (2.8)$$

Let X_ρ be the n-dimensional vector whose components corresponding to the places of P_ρ are equal to 1 and to other places equal to 0. Equation (2.8) can be written:

$$X_\rho^T M = X_\rho^T M^\circ \quad (2.9)$$

From Theorem 2.1, this means that X_ρ is an S-invariant. Because of relation (2.7), the corresponding S-component is the same as the directed circuit ρ . Let us prove now that the support of X_ρ , i.e.,

P_ρ , is necessarily minimal. If we consider the circuit ρ as a subnet (which is allowable since it is an S-component), we can construct its Incidence Matrix C_ρ , which, after an appropriate ordering of places and transitions, has the form (the columns correspond to the transition, $t_{\rho_1}, t_{\rho_2}, \dots, t_{\rho_k}$, and the rows to the places, $p_{\rho_1}, p_{\rho_2}, \dots, p_{\rho_k}$):

$$C_\rho = \begin{bmatrix} -1 & 0 & 0 & \dots & 0 & +1 \\ +1 & -1 & 0 & \dots & 0 & 0 \\ 0 & +1 & -1 & \dots & 0 & 0 \\ 0 & 0 & +1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & & +1 & -1 \end{bmatrix}$$

where ρ corresponds to the circuit:

$$\rho = (p_{\rho_1} \ t_{\rho_1} \ p_{\rho_2} \ \dots \ p_{\rho_k} \ t_{\rho_k})$$

Let $X_k = [a_1 \ \dots \ a_k]^T$ be any positive integer vector. Now X_k is an S-invariant of the subnet iff:

$$X_k^T C_\rho = 0$$

which yields immediately:

$$a_1 = a_2 = \dots = a_k$$

Otherwise stated, all the S-invariants of the subnet have the same support, which is actually the set of places of the circuit, i.e., P_ρ . Hence P_ρ is a minimal support of the Event-Graph: If it were not, it would strictly contain another minimal support, which would also be a support of invariant for the corresponding subnet. However, we have

showed that P_p cannot strictly contain another support of invariant.

Q.E.D.

- (ii) Let us prove now that a minimal S-component is a directed circuit. Assume then that PN_S is a minimal S-component corresponding to the S-invariant X_S and call P_S (resp. T_S) its set of places (resp. its set of transitions). By construction, PN_S is an Event-Graph. For any initial marking M^0 and for any reachable marking M , we deduce immediately from Theorem 2.1 that, for all the places p_i belonging to P_S :

$$M(p_i) \leq \frac{X_S^T M^0}{x_i}$$

where x_i denotes the i -th component of vector X_S . Hence, PN_S is a bounded net. From Theorem 2.5, we deduce that PN_S is a strongly connected Event-Graph. We can find therefore at least one directed circuit, ρ , in PN_S : we choose any two nodes and consider the directed path that goes from the first node to the second one and vice-versa. However, we know from (i) that ρ is in that case a minimal S-component of PN_S and consequently of the original Event-Graph PN . Since PN_S is by definition a minimal S-component, and since ρ is included in PN_S , PN_S is necessarily identical to ρ : Hence, PN_S is a directed circuit.

Q.E.D.

Theorem 2.6 will be used to determine all the circuits of an Event-Graph, from an algorithm that determines all the minimal support S-invariants of a net [13].

The last section of this chapter concerns the Timed Petri Net model, as it will be used to represent Decision Making Organizations. Indeed, Timed Petri Nets provide an appropriate model for describing the dynamic behavior of systems, especially those which exhibit asynchronous and concurrent properties.

2.4 TIMED PETRI NETS

The strength of the Petri Net formalism, as described in the previous sections, lies in the rather simple representation it provides for modeling complex systems, in which processes are both concurrent and sequential. However, the major weakness of ordinary Petri Net models comes from the lack of tools to analyze the real-time process. Indeed, no assumptions are made regarding the length of time it takes to complete the different processing operations. Tokens move in the system according to the transition firings, which are assumed instantaneous.

To handle this shortcoming, several types of Timed Petri Nets - in which the notion of time is explicitly introduced - have been developed in the literature. The main differences between the models come actually from the type of processing times considered: deterministic or random. Ramchandani [6] was the first to introduce a model of Timed Petri Net in which the processing times were assumed deterministic and assigned to the transition of the net. Sifakis [14] also considered deterministic processing times, but assigned to the places. He has proved, however, that both models are equivalent. Stochastic Timed Petri Nets, corresponding to the case where times are assumed random have been extensively analyzed by Wiley [15]. In this thesis, the type of TPN (Timed Petri Net) introduced by Ramchandani, will be used for modeling the DMO. In this section, the related definitions are presented.

2.4.1 Definitions

A Timed Petri Net (TPN) is a pair (PN, μ) , where PN is a Petri Net and μ is a firing time function that assigns a positive rational number to each transition of the net: $\mu : T \longrightarrow R^+$ (R^+ denotes the set of positive rational numbers). In a Timed Petri Net, each transition t takes the time $\mu(t)$ to fire. The reason why we require the firing times to be rational (and not merely real) is that we can discretize the processing times in units of time and precisely describe the state of the process at each

instant of time.

The rules of operation of a TPN remains the same as an ordinary PN, except that the firing of a transition consists of the three following phases:

- (1) The firing initiation, that can occur whenever the transition is enabled. When the firing of a transition is initiated, one token from each of its input places is removed.
- (2) The firing execution which occurs during the processing time $\mu(t)$.
- (3) The firing termination, which occurs at the end of the execution. At this moment one token is added in each of the output places of the transition.

The three phases of a transition firing can be visualized by imagining every transition as consisting of two transitions and an intermediate place, as shown in Figure 2.8. The firing time of the transition t can now be associated with the place p_t in the following manner: When transition t initiates, t_b fires instantaneously, a token is removed from each input place of t_b and a token is deposited on place p_t . This token stays in p_t for the interval $\mu(t)$, the firing time of t . At the end of this interval, transition t_e fires, corresponding to the termination of the transition t . The substitution described in Figure 2.8 is actually the transformation used by Sifakis [14] to prove the equivalence of both types of TPN (i.e., when processing times are either associated to transitions or places).

Remark: With this definition of a TPN, as given above, it should be clear that, at time τ , the state of the Petri Net is not completely described by its marking $M(\tau)$: some transitions are being processed at that time, which implies that some tokens are not taken into account in $M(\tau)$. If we substitute for each transition the subnet model described in Figure 2.8, the tokens being in the execution phase would be in the corresponding

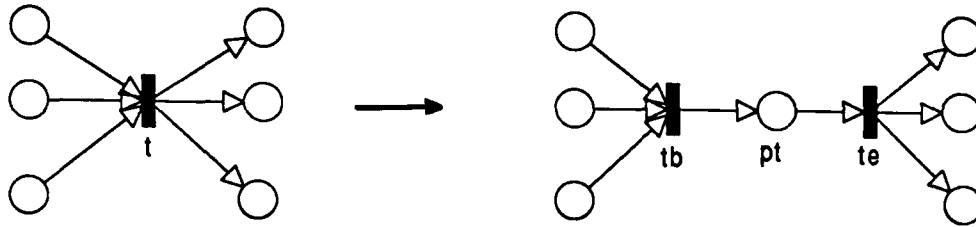


Figure 2.8 Timed Petri Net Equivalence

place p_t . This remains however a pure matter of convention, which does not affect the conservation of tokens in the net. In fact, in modeling the dynamic behavior of the system and, more precisely, analyzing how the sequence of transition firings occurs in real time, only the "free" tokens deserve attention; these tokens determine which transitions are enabled and, therefore, which transitions can fire at a given instant of time τ .

In order to have a complete description of the state of the system at any instant, Ramchandani has introduced the following two functions, which will be often referred to, in this thesis:

- $I_j(\tau)$, which denotes the number of initiations of transition t_j in the interval of time $[0, \tau]$.
- $T_j(\tau)$, which denotes the number of terminations of transition t_j in the interval $[0, \tau]$.

It becomes easy to determine the marking of the net at any instant, using these two functions. Let us write the incidence matrix:

$$C = C^+ - C^-$$

where C^+ characterizes the directed arcs that go from transitions to places, i.e.,

$$C_{ij}^+ = O(t_j, p_i)$$

and C^- characterizes the directed arcs that go from places to transitions, i.e.:

$$C_{ij}^- = I(p_i, t_j)$$

Let $M_i(\tau)$ denote the marking of place p_i ($i=1, \dots, n$) at the instant of time τ . Then:

$$M_i(\tau) = M_i^0 + \sum_{j=1}^m C_{ij}^+ T_j(\tau) - \sum_{j=1}^m C_{ij}^- I_j(\tau) \quad (2.10)$$

This equation describes the marking of the net at any instant, given the functions $T_j(\tau)$ and $I_j(\tau)$ ($j=1, \dots, m$) and we will make use of it in later sections.

In the next section, we are going to apply the model of TPN to the special class of Petri Nets, namely, Event-Graphs. The resulting Timed Event Graph will actually be the model used to carry out the performance evaluation of the DMO.

2.4.2 Timed Event-Graph

A Timed Event-Graph (TEG) is simply a model of TPN, as presented in the last section, i.e., a pair (PN, μ) where PN is in this case an Event-Graph. The model of TEG used in this thesis corresponds, however, to the one defined by Chretienne and Carlier [16], which includes an additional condition in the firing process of a transition. In the definition of a

TPN, a firing can be initiated as soon as the corresponding transition is enabled. This implies, in particular, that firings can be initiated even if the transition is already executing. In other words, no assumptions are made regarding the capacity of place p_t (Figure 2.8), which therefore can contain more than one token at a time. This is not allowable in the modeling of the decisionmaking process developed in this thesis. To the extent that tokens represent information messages and transitions model the different processing stages, no more than one information message can be processed in a certain stage at a time. In other words, inputs are processed one by one at any stage. This constraint implies, in fact, that a transition is not allowed to initiate, if it is already executing.

This constraint can actually be modeled in the Petri Net representation by adding a self-loop to each transition of the net, as shown in Fig. 2.9.

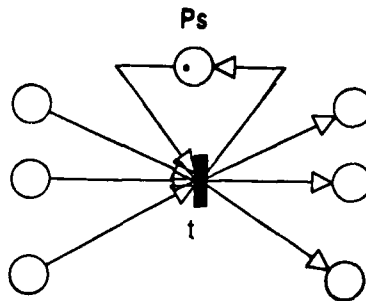


Figure 2.9 Representation of a Self-loop

The fact that there is only one token in the extra place p_s accounts for this constraint. The same kind of transformation, as used previously, describes clearly what happens in the firing process (Fig. 2.10). Once t_b has fired, one token is in p_t and place p_s is empty. Therefore, transition t_b cannot fire again (since it is not enabled) until t_e has fired, which occurs at the end of the interval $\mu(t)$ (i.e., the firing time of transition t).

Although self-loops allow for a clear representation of the firing constraint in the Petri Net model, they are not really convenient for two

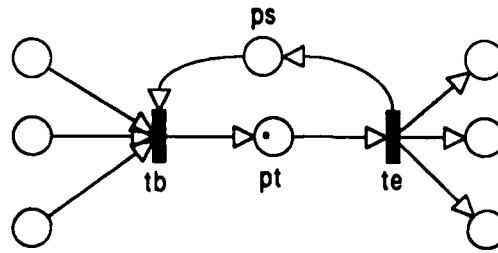


Figure 2.10 Model of Transition Firing

reasons: First, they do not change the structure of the system, but increase its complexity, since the number of places is considerably greater. Second, they cannot be included in the incidence matrix, because the structure of the incidence matrix only permits to represent the presence (or absence) of one directed arc between any two nodes. For these reasons, self-loops will not be represented in our Petri Net model of DMOs. They will be, however, implicit and it will be important to keep them in mind, when analyzing the real-time process. Indeed, we will see that the circuits of the net play a crucial role in the time-related performance measures of the system. At this point, each transition should also be considered as a circuit containing one token, precisely because of the implicit self-loop.

CHAPTER III

PETRI NET DESIGN OF DECISIONMAKING ORGANIZATIONS

In this chapter, we develop some extensions to the Petri Net representation of the DMO, as previously introduced in [2]. Included in our modeling are the resource and time constraints of the DMO. The resource constraints concern both individual DMs and the organization as a whole. This type of constraint is modeled as a limitation on the number of inputs that can be handled at the same time. Time constraints occur from the various task processing times. Firing times are therefore assigned to the transitions, so that the representation corresponds to a Timed Petri Net model.

3.1 AGGREGATED MODEL OF THE INTERACTING DECISIONMAKER

The modeling of decisionmaking organizations using Petri Nets has been introduced in [2]. Figure 3.1 shows the aggregated Petri Net model of the single interacting decisionmaker. The decision process occurs in four stages: Situation Assessment (SA), Information Fusion (IF), Command Interpretation (CI) and Response Selection (RS). Incoming inputs - either from the environment or from other DMs - are processed at the first stage to produce the situation assessment. This information is then fused at the IF stage with situation assessments communicated by other DMs. The resulting information is combined with commands received by the DM in the CI stage, so as to select a response in the RS stage. Each of the four stages, which corresponds to a particular task performed by the DM, is modeled by a transition. The firing of a transition represents the execution of the corresponding stage in the process. According to the rules of operation of a Petri Net, a transition can fire when it is enabled, i.e., when there is at least one token in each of its input places. When a transition fires, one token is removed from each of its input places and one token is added in each of its output places.

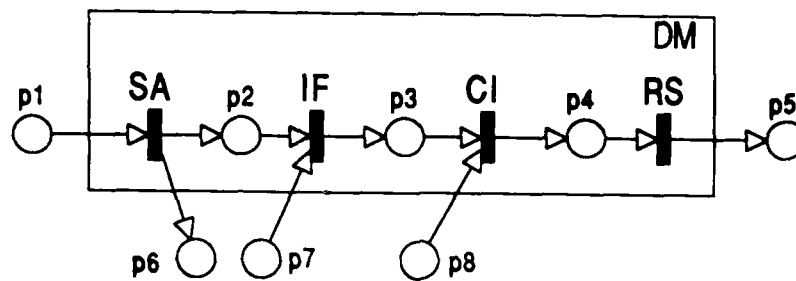


Figure 3.1 Model of the Interacting Decisionmaker

Assuming that tokens represent information messages, the flow of tokens models the flow of information in the process. Places receive and transmit the information that is exchanged between the different processing stages.

If the role of transitions is clearly defined, since they represent the various processing tasks, the role of places needs to be, however, further specified. From Figure 3.1, we can notice that places carry in fact two types of information:

- places like p_1 , p_5 , p_6 , p_7 , and p_8 carry the information being exchanged between the DM and the environment or other organizational members.
- places like p_2 , p_3 , and p_4 carry the information that is internally processed by the DM.

The first type of places allows for modelling the interactions between the DM and the environment or other DMs: they are either inputs to the DM (like p_1 , p_7 , p_8) or outputs of the DM (like p_5 , p_6), but in no case can they be both at the same time. If we consider these places for all the DMs involved in the organization, they determine in fact the organizational structure, i.e., the internal interactions between the DMs and the interactions between the DMs and the environment. This is the reason why we will call them structural places, in contrast to the other places.

The second type of places, together with the four tasks defined above, model the internal structure of the interacting DM. These places carry the information that is produced at one stage of the decisionmaker process and used at the next stage. In contrast to the structural places, these are both input and output places of the same DM. At this point, we should note that the processing of a specific input takes place in an asynchronous manner, i.e., delays generally occur between the different processing stages, precisely because of the interactions with the other DMs or the environment. For example, once the situation assessment stage is completed, one token is in p_2 , but the information fusion cannot take place before one token is also in p_7 , which is the information transmitted from the SA stage of another DM. Such delays imply that information has to be stored temporarily in places p_2 , p_3 , or p_4 until the processing of the corresponding stage can occur. This is the reason why we call these places memory places: they model the internal (short-term) memory of the DM, where the information being processed has to be stored temporarily until the different processing stages are completed.

3.2 MODEL OF THE INTERACTING DECISIONMAKER WITH LIMITED RESOURCES

The model of the Interacting Decisionmaker does not take into account the limited capacity for information processing, that characterizes the human DM. Indeed, as long as information messages are present, i.e., tokens are available in place p_1 , the processing can start, i.e., the transition corresponding to the SA stage can fire. However, the information processing of human DMs is subject to the bounded rationality constraint, as defined in [4]. In the information-theoretic approach, the amount of information processed is measured by the total activity, G , of the DM, which also characterizes the DM's workload. It is assumed that there exists an upper bound G_r , above which the DM becomes overloaded and his performance degrades:

$$G \leq G_r \quad (3.1)$$

When the analysis is carried out for the steady-state process, the above constraint takes the form:

$$G \leq F \tau = G_r \quad (3.2)$$

where F is the processing rate constraint that characterizes the human DM, and $1/\tau$ (resp. τ) is the average arrival rate of inputs (resp. average interarrival time). This constraint implies that the DM must process inputs at a rate at least equal to the rate at which they arrive.

In our work, we deal with time-related performance measures and do not address the accuracy of the response, which is based on the comparison between the actual organization's response and the ideal or desired response [1]. In particular, the way a DM reacts to information overload (which occurs when $G > F \tau$) and the extent to which it affects his performance are not a matter of concern here. This remains however allowable, in so far as we also include in our modelling the actual processing constraint, that can be expressed by writing inequality (3.2) in another form:

$$1/\tau \leq \frac{F}{G} \quad (3.3)$$

Using relation (3.3), the bounded rationality limitation turns out to be, in that case, a constraint on the allowable rate of incoming inputs, i.e., on the maximum rate of inputs that can be handled by the DM, without being overloaded. In other words, rather than assuming that inputs are processed at a rate at least equal to the rate with which they arrive, and derive a bound on the total activity allowable, G , (given F), we assume that no information overload does occur in the process and derive the corresponding constraint on the allowable rate of incoming inputs.

Let us see now how it is possible to model this constraint using the Petri Net framework. In fact, the limited processing capabilities of a

human DM come from the limited (cognitive) resources available to perform the various processing tasks. In particular, the bounded rationality constraint is very much related to the limited capacity of the human short-term memory, defined as the memory in which the information is held temporarily (in contrast to the long-term memory, in which the information is stored permanently). Indeed, this bound means that a DM cannot handle properly too much information at the same time. Interestingly enough, the analysis of the human short-term memory carried out in [5] has shown that a maximum of six or seven units of information can only be held in this memory (typically for a few seconds), without any loss of information. This is of much importance here, because the DM has precisely to handle different information messages during the time necessary to complete the various processing tasks. This is the reason why we have identified, in the previous section, the role of places p_2 , p_3 , and p_4 (Figure 3.1) as short-term memory: these places contain the information messages (represented by tokens), that are being processed internally by the DM. Quite naturally, the limited capacity of the short-term memory can therefore be modeled, using the Petri Net formalism, as a capacity constraint on the corresponding places p_2 , p_3 , and p_4 . Accordingly, we extend the model of Interacting Decisionmaker as shown in Figure 3.2.

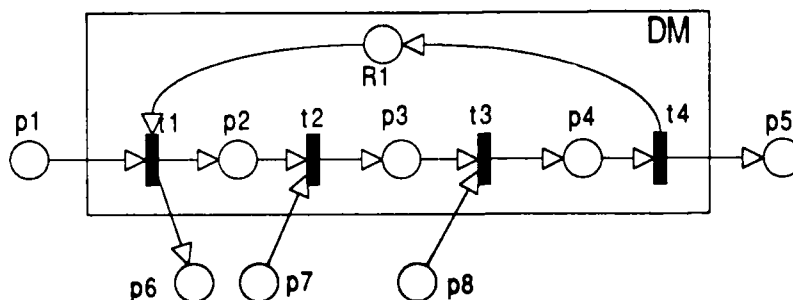


Figure 3.2 Model of Interacting Decisionmaker with Limited Resources

Now the added place R_1 , whose input transition corresponds to the RS stage and output transition to the SA stage, allows for modeling the information-processing constraint. We shall call it the resource place, because the number of tokens put initially in this place represents the resources available for processing. Indeed, the directed path:

$$\rho = (R_1, t_1, p_2, t_2, p_3, t_3, p_4, t_4)$$

determines a circuit in the net, in which the token content remains invariant by transition firings (Chapter 2, Theorem 2.3), i.e., for any reachable marking M :

$$M(R_1) + \sum_{i=2}^4 M(p_i) = M^0(R_1) + \sum_{i=2}^4 M^0(p_i) = M^0(R_1) \quad (3.4)$$

It is assumed here that places p_2 , p_3 , and p_4 contain initially no tokens, which means that, initially, there are no information messages being processed. Now, the constraint on the memory capacity is trivially satisfied. Assuming that there are n resources available (i.e., $M^0(R_1) = n$) which may represent the amount of memory space available (in other words, the cognitive resources), we deduce indeed from the marking equation (3.4), that, at any reachable state of the process:

$$M(p_2) + M(p_3) + M(p_4) \leq n$$

which precisely models the short-term memory limitation, as described earlier.

At this point, it is possible to extend further our analysis of the model, especially with respect to the processing rate constraint. Let us specify now how the processing of inputs does occur, holding the same assumptions about the initial marking of each place, i.e.

$$M^0(R_1) = n \quad \text{and} \quad M^0(p_2) = M^0(p_3) = M^0(p_4) = 0$$

The processing of any new input starts by the firing of transition t_1 , (corresponding to the SA stage), which consumes therefore one token (i.e., one resource) from place R_1 . Likewise, when the processing is completed, transition t_4 (corresponding to the RS stage) fires, which produces one token back in R_1 . In other words, one resource is engaged at the beginning of the process and is released at the end of the process. Clearly enough, the DM can only process at most n different inputs at the same time. If n inputs are currently being processed, the resource place R_1 is empty (which follows directly from the balance equation (3.4)) and transition t_1 is consequently not enabled. Otherwise stated, the resource constraint turns out to bound the number of inputs that the DM can process simultaneously.

To analyze deeper how it affects the processing rate, it is necessary to introduce the processing times of the four stages SA, IF, CI, and RS, that will be denoted respectively by μ_1 , μ_2 , μ_3 , and μ_4 . Let us point out that a complete description of the model, with firing times associated to the transitions, will be presented later in this chapter. Our intention here is only to make clear how the resource limitation actually bounds the processing rate. Now, if the decision process were fully synchronous, which means that no delays would occur between the different processing stages, it would take the amount of time

$$\mu_0 = \mu_1 + \mu_2 + \mu_3 + \mu_4$$

to complete the processing of any input. Since the DM can handle at most n inputs at the same time, the processing rate is necessarily bounded by:

$$f = \frac{n}{\mu_0} = \frac{n}{\mu_1 + \mu_2 + \mu_3 + \mu_4}$$

There is, however, an additional constraint to include: the execution of any processing stage can only take place for only one input at a time as discussed in Chapter 2, Section 2.4.2. Therefore, the actual bound is determined by:

$$\phi = \min \left(f, \frac{1}{\mu_1}, \frac{1}{\mu_2}, \frac{1}{\mu_3}, \frac{1}{\mu_4} \right) \quad (3.5)$$

We will show in the next chapter that ϕ determines precisely the maximum rate of information processing that characterizes the DM. This rate constraint is derived from the limited capabilities of a human DM, which include both the resource limitations (as specified by n) and the processing time-constraints (as specified by the amount of time required to perform the various tasks). ϕ is in fact similar to the processing rate constraint F , introduced in the information theoretic framework.

Remark: We have extended the Petri Net model of the Interacting Decisionmaker, so as to include the (cognitive) resource constraints, derived from the limited capacity of the human short-term memory. However, the model is also suitable for any kind of resource limitations that constrain the processing of inputs. Suppose, for example, that the same specific resource is needed to complete the processing of any input. Putting initially only one token in the resource place R_1 allows for modeling the corresponding constraint.

Moreover, as explained earlier, the resource limitation turns out to be a constraint on the number of inputs that can be handled at the same time. Therefore, the resources available to the DM may have in reality various forms, but it is only necessary to take into account, in the modeling, the most stringent resource limitation. If, for instance, the DM

has the capacity to handle several inputs simultaneously, but needs a unique specified resource, the corresponding processing constraint will be modeled by putting only one token in R_1 , as explained above.

3.3 AGGREGATED MODEL OF THE DMO

After having analyzed the model of the simple Interacting Decision-maker, we present in this section the aggregated Petri Net model for the overall organization. At first, we deal with the interactions between the organization and the environment.

3.3.1 Model of the DMO Interacting with the Environment

The interactions between a DMO and the external environment has been described in [1]. The first processing stage of the DMO consists of the partitioning of the external signals or messages into a set of inputs that are assigned to different organization members. This particular type of allocation has been addressed in [17]. Assuming, for example, that DM1, 2, and 3 are the DMs that interact directly with the environment, the corresponding Petri Net representation is shown on Figure 3.3.

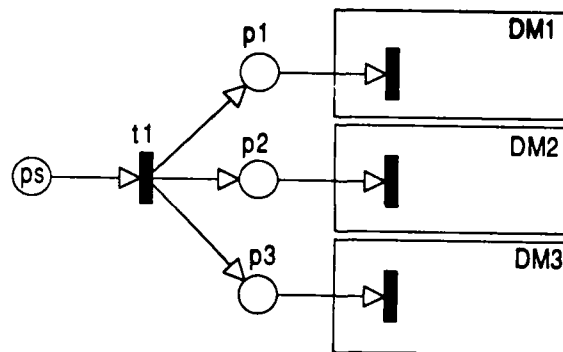


Figure 3.3 Model of the Partitioning Operation

The place p_s represents the source of signals or messages, and the transition t_1 models the partitioning operation. Since it is the first processing stage, t_1 will be called the input transition of the process. Each time that t_1 fires, one token is sent in the places p_1 , p_2 , and p_3 , which are the input places of DM1, 2 and 3. It should be clear that this model implies an overall synchronization between the individual inputs received by each of the DMs. In the following stages, however, the processing of the inputs by each DM becomes asynchronous and concurrent, as emphasized in the introduction.

Likewise, the Petri Net representation of the DMO has an output transition, which characterizes the last processing stage. The output place of this transition contains the organizational responses. For the moment, we will denote this transition by t_m , where m is the total number of transitions in the net. p_r will denote this output place. Finally, the aggregated representation of the DMO interacting with the environment is shown in Figure 3.4.

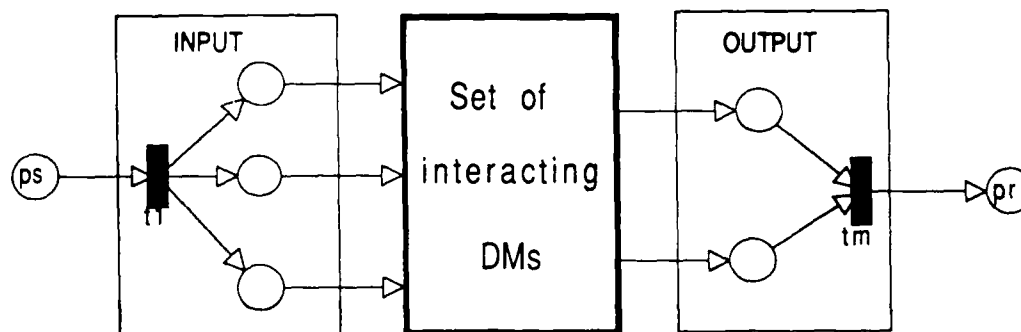


Figure 3.4 Model of DMO Interacting with the Environment

Interestingly enough, we can imagine the overall process as taking place through a sequence of three major stages. Each of the stages would correspond to a particular action, as defined in [3], as follows:

- the input action, corresponding to the partitioning operation of

the external messages;

- the processing of the information, carried out by the different DMs involved in the organization;
- the output action, corresponding to the production of the overall response that characterizes the organizational response.

3.3.2 Model of the DMO with Limited Resources

The analysis of the resource constraint, carried out in Section 3.2 for the Interacting Decisionmaker, can be applied in fact to the organization as a whole. Indeed, the resources used by the overall organization may have various forms, but there exists always at least a processing constraint that comes from the limited capacity of the structural places. This constraint is very similar to the one existing for the memory places of the individual DMs.

As described in Section 3.1, the structural places receive and transmit the information between the DMs and between the DMs and the environment; Now, since the process is asynchronous, information is to be stored temporarily in these places, exactly as what occurs for the internal decision process of a DM. These places carry in fact the role of buffer storage within the organization, where some information is stored temporarily. It is therefore important to make sure that at no instant does the amount of information stored exceed the buffer capacity of the system, i.e., the capacity of the structural places.

Let us assume, for instance, that the arrival rate of external inputs exceeds the maximum processing rate of any one of the DMs (defined as ϕ in Section 3.2). With the present Petri net model of the DMO, it follows that t_1 will fire at the same rate with which inputs arrive: Tokens will therefore necessarily accumulate in the system and, in particular, the token content of some structural places of the net will eventually grow to

infinity over time: in such case, we will say that the DMO is overloaded.

To handle this constraint, we modify the Petri net model of the DMO by adding an extra place, exactly as we did for the single Interacting Decisionmaker. The resulting model is shown on Figure 3.5. R_0 is the resource of the overall organization: the number of tokens put initially in this place bounds the number of inputs that the DMO can process at the same time. Indeed, each time that a new input is processed (i.e., whenever t_1 fires), a token is removed from R_0 and comes back again in R_0 once the processing is completed (i.e., when t_m fires). Assuming that R_0 contains initially n tokens and that n inputs are currently being processed, then R_0 is empty and no more inputs can be processed, since t_1 is not enabled.

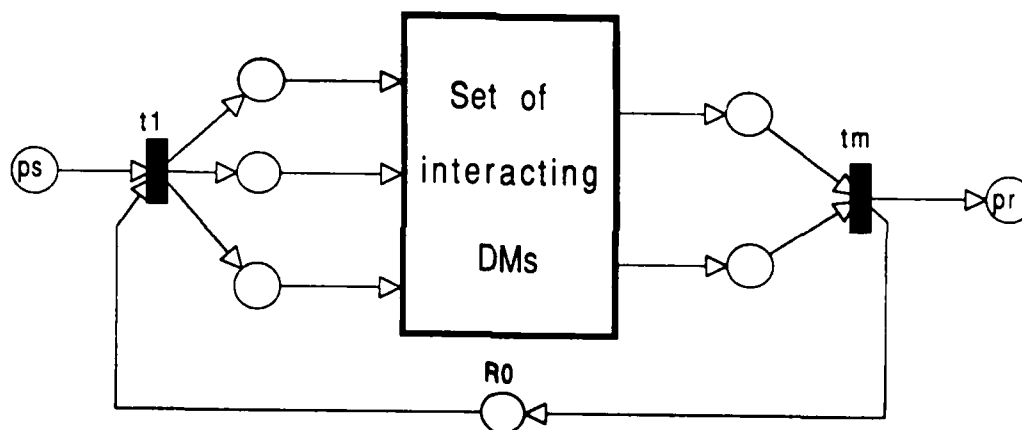


Figure 3.5 Model of the DMO with Limited Resources

Let us point out that the same remark, made in Section 3.2, applies here: it is only worth taking into account the type of resource limitation which is the most stringent. If, for instance, the same specific resource is needed to complete the processing of any organizational input, putting one token in R_0 will account for this processing constraint.

In the last two sections, we have dealt with a specific type of processing constraints derived from the limitation of resources. In the next section, we will address time constraints, as determined by the

various task processing times.

3.4 MODELING TIME CONSTRAINTS

Apart from the resource constraints, which were treated in Section 3.2 and 3.3, the other type of processing constraints is time constraints, which arise from the task execution times. As already introduced in Section 3.2, the task processing times are included in the model by assigning to each transition a corresponding firing time. At this point, however, the model of Timed Petri Net described in Chapter 2, Section 2.4 where the firing times are assumed deterministic, cannot be applied directly to the aggregated model of the DMO developed so far. The problem arises from switches, whose function is more complex than ordinary transitions.

In the aggregated model of the Interacting Decisionmaker described in Section 3.1, all four processing stages SA, IF, CI and RS were represented by transitions. In reality, the representation of both the SA and RS stages includes a decision node or switch as defined in [1], which is a more complex form of transition. An n-decision switch is in fact a subnet that contains n (simple) transitions, as shown on Figure 3.6.

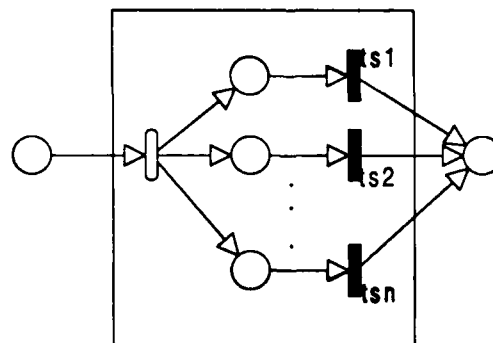


Figure 3.6 Petri Net Model of an n-Decision Switch

Each transition of a switch ($t_{s1}, t_{s2}, \dots, t_{sn}$) represents an alter-

native process or algorithm. The selection of one algorithm, i.e., the firing of one of these transitions, occurs for every input (i.e., token) processed. It turns out that each algorithm may have a different processing time, which implies that the processing time of the SA (or RS) stage is not necessarily unique, but depends on the algorithm selected. In fact, if in our model we represented explicitly all the transitions of a switch, it would be possible to assign to each transition of the net a deterministic firing time, according to the Timed Petri Net model presented in Chapter 2.

However, as it will be clear later, the performance analysis carried out in this thesis depends crucially on the fact that the DMO can be modeled as an Event-Graph, i.e., each place has exactly one input and one output transition; this condition would be violated, if the representation shown in Figure 3.6 is used. This is why the aggregated model is very convenient for our purpose. Now, to handle the problem of switches, we will introduce the following extension to the Timed Petri Net model: The firing time of a switch will be considered as a discrete random variable, having a finite number of possible values (corresponding to the processing times of the different algorithms), according to a certain probability distribution. Naturally, the probability distribution will depend on the decision-rule that determines the selection among the different algorithms. Because tokens are indistinguishable in a Petri Net, the probability distribution will be assumed independent from one token to the next (i.e., independent from one input to the next).

In the next section, we summarize the key points that have been developed so far. In particular, we show that the model of the DMO, which includes resource and time constraints, corresponds to a strongly connected Timed Event-Graph (following the definitions given in Chapter 2). We will illustrate the development with a two decisionmaker organization.

3.5 MODEL OF THE DMO AS A STRONGLY CONNECTED TIMED EVENT-GRAPH

Let us recall that an Event-Graph is a Petri Net such that each place has exactly one input and one output transition. The strong connectivity means that there exists a directed path between any two nodes (places or transitions). Now, our aggregated model of the DMO is clearly an Event-Graph since, as seen in Section 3.2 and 3.3, each place of the net — whether it is a structural, memory or resource place — has indeed one input and one output transition. For the memory and resource places, this is due directly to the internal structure of the Interacting Decisionmaker (Figure 3.2). It is also clear for the structural places, because the information messages (i.e., tokens) are transmitted from a certain processing stage to another stage (both modeled by transitions), as determined by the execution protocol of the organization.

Naturally, we are interested in analyzing the performances of the admissible organizational forms, as formulated in [18]. Two constraints should in particular be satisfied:

- (1) the information structure should have no loops
- (2) a directed path should exist from the input node to every node and a directed path should exist from any node to the output node.

At this point, it is quite important to clarify what the first constraint implies in our model. This constraint means that the information structure should be acyclical, i.e., there should be no information loops. Of course, in our Petri Net model of the DMO, there are loops (or circuits), because of the resource places. But the loops that exist because of the resource places, i.e., the loops that include a least one of the resource places, do not actually violate the constraint. As far as our model is concerned, the precise formulation of this constraint is the following:

- (1) Only circuits which contain one or more of the resource places are allowed. In other words, if we delete from the Petri Net all the resource places, the resulting net should have no circuits (the net so obtained models the information structure, as developed in [1]).

Now, the second constraint makes sure that our Petri Net model is strongly connected, precisely because of the resource place of the overall organization, R_0 . As we have seen, this resource place connects the output transition to the input transition of the net (Figure 3.5). Let us now consider any two nodes of the net. There exists a directed path from the first node to the output transition and therefore to the input transition (via the resource place). There exists also a directed path from the input transition to the second node. By appending these two paths, we have constructed a directed path between the two nodes.

Finally, given the processing times assigned to the transitions, as described in Section 3.4, the Petri net model of the DMO turns out to be a strongly connected Timed Event-Graph. We illustrate the model for two different structures of a two member organization, as shown in Figures 3.7 and 3.8. The example is derived from the analysis carried out in [18]. Transitions t_1 , t_2 , t_4 , and t_5 (resp. t_6 , t_7 , t_8 , and t_9) correspond respectively to the SA, IF, CI, and RS stages of DM1 (resp. DM2). In both cases, DM2 transmits the result of his SA stage to DM1, which constitutes information sharing among them. However, in structure (a), information is transmitted from the RS stage of DM1 to the IF stage of DM2 and this interaction is of the result sharing type, as defined in [18]. In structure (b) the same information is transmitted to the CI stage of DM2, which corresponds in that case to a command. This implies that there is, in that case, a hierarchical relationship between the two DMs. Naturally, we have added to the Petri Net model all the resource places, i.e., R_0 for the organization and R_1 (resp. R_2) for DM1 (resp. DM2), in accordance with the analysis carried out in this chapter. Both models will be used to illustrate the performance analysis that will be developed in the next chapter.

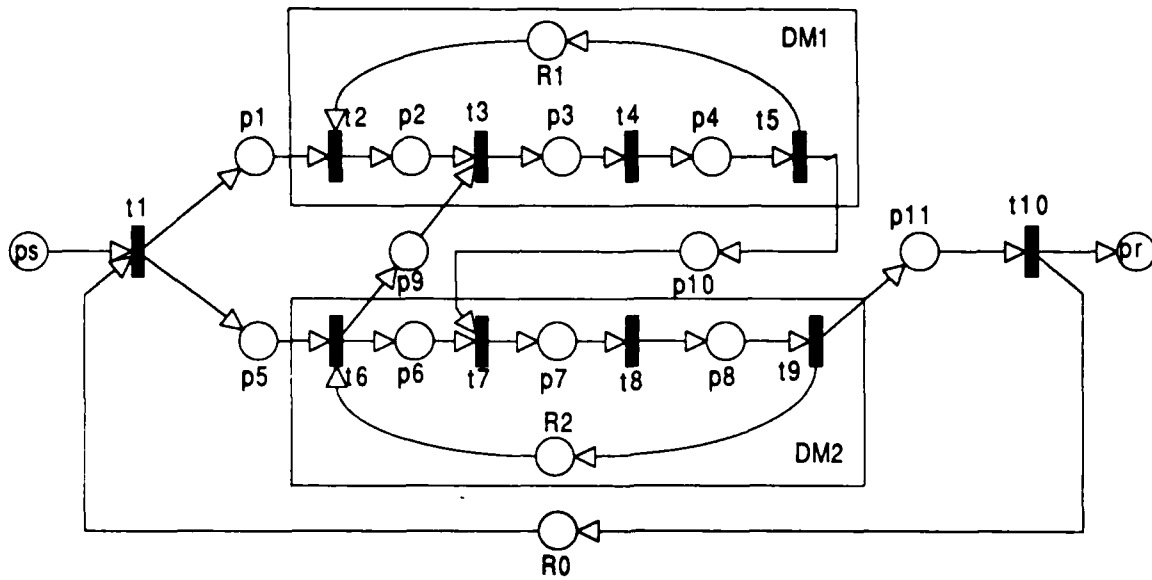


Figure 3.7 Two Member Organization (Structure (a))

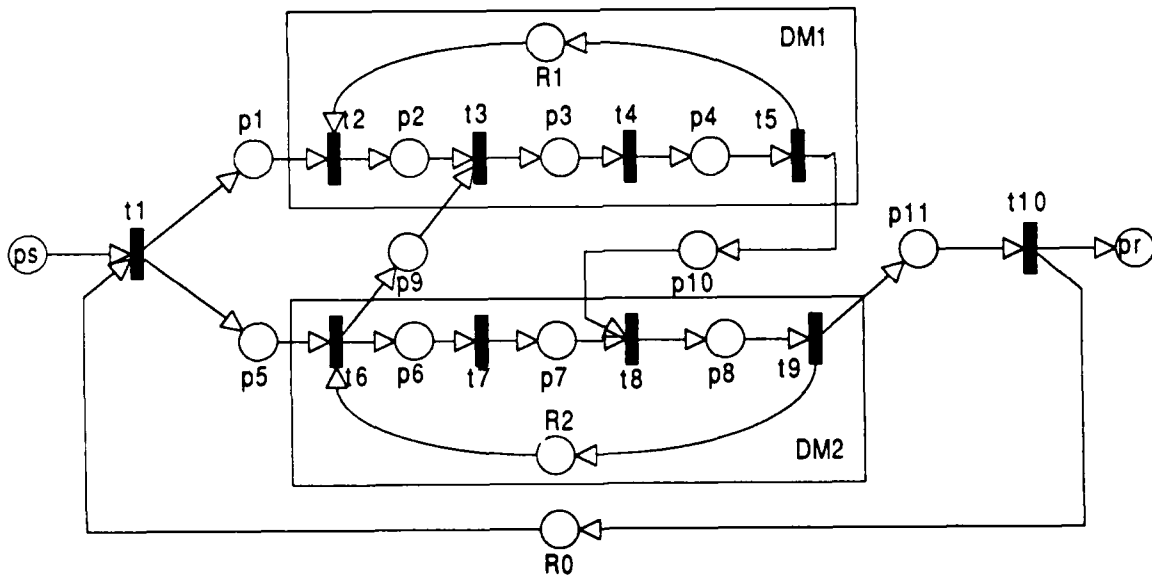


Figure 3.8 Two Member Organization (Structure (b))

CHAPTER IV

MAXIMUM THROUGHPUT RATE OF THE DMO

Using the model of the DMO developed in the previous chapter, we present in this chapter a method for computing the maximum throughput rate of the DMO. This MOP characterizes the maximum rate of processing of the overall organization. This measure is especially important because it bounds the allowable rate of external inputs that can be handled by the DMO. Given the internal structure of the system, the maximum throughput rate is characterized as a function of both the time constraints (i.e., the various task processing times) and the resource constraints. It will therefore be possible to investigate which constraints should be relaxed in order to improve this performance. Additionally, different organizational structures can be compared with respect to their maximum throughput rate. These types of performance analysis will be illustrated in this chapter using the example of the two decisionmaker organization presented in Chapter 3.

4.1 PROPERTIES OF THE PETRI NET MODEL

Before computing the throughput rate, it is first necessary to verify that the Petri Net model of the DMO satisfies the properties of a well-defined system, namely liveness and boundedness. As stated in Chapter 2, Section 2.1.6, these properties guarantee the system to be deadlock free (liveness) and that information does not accumulate in the system (boundedness), in other words that the flow of outputs equals the flow of inputs.

Let us recall that an Event-Graph is live, iff the token content of every directed circuit is strictly positive (Theorem 2.4) and it is bounded, iff the net is strongly connected (Theorem 2.5). Now, we have assumed in the previous chapter, that the model of the DMO is admissible,

which implies, that there should not exist any directed circuit which does not contain any of the resource places (otherwise the information structure would not be acyclical). Therefore, every directed circuit contains at least one resource place and, consequently, its token content is strictly positive. Since the model is also strongly connected, as seen in Chapter 3, Section 3.5, the system is indeed live and bounded.

The liveness and boundedness properties are very important for analyzing the steady-state process, because these characteristics imply also that the system is consistent. Actually, an Event-Graph is said to be consistent, if there exists a cyclic firing sequence, i.e., a sequence which fires each transition once and brings the marking back to its initial state. Thus, if a system is consistent, it goes back to its initial state after each cycle and then repeats itself. If the system is inconsistent, it can be shown that either it produces an infinite number of tokens (i.e., needs infinite resources) or consumes tokens and eventually comes to a stop. Consistency is, therefore, a critical property of systems which should function continuously, with a finite amount of resources, as it is the case for the model of the DMO studied here. As proved in [6], a live and bounded Event-Graph is consistent. Having assured that our Petri Net model of the DMO is consistent, we can now analyze the throughput rate of the corresponding system.

4.2 COMPUTATION OF THE MAXIMUM THROUGHPUT RATE

Once we are sure that the model of the DMO under consideration corresponds to a live and consistent system, and therefore each transition can fire repeatedly, we would like to determine the maximum rate the transitions can actually fire. Indeed, this will yield the maximum throughput rate. Of course, this supposes that we take explicitly into account the transition firing times. At this point, our analysis will be based on the model of Timed Petri Nets and more precisely on Timed Event-Graph described in Chapter 2, Section 2.4. Some of the results obtained here have been originally studied by Ramchandani [6], but in a more

restrictive case. In his work, it was assumed that the steady-state process was deterministic (i.e., all the transition firing times were deterministic) and strongly periodic, which means that every transition fires at regular intervals of time, with the same period. No such assumptions are made in our analysis. We can handle both cases: firing times that are deterministic or discrete random variables (with a finite set of possible values), according to the operating rules of switches, as defined in Chapter 3, Section 3.4. In addition, we will deal in our analysis with average firing rates, rather than fixed rates. Indeed, as it will be shown in the next chapter, the steady-state process is generally not strongly periodic even for the deterministic case: the periodicity is of a more complex type, that will be described later as the K-periodicity introduced by Chretienne [16]. Finally, the only assumption made in our analysis is that inputs are supposed to queue at the entry of the system so that the process occurs repetitively over time.

4.2.1. Average Cycle Time of Transitions

In this section, we present the first important and rather intuitive result that the average firing rate is the same for all the transitions. The average rate considered here is very precisely defined. We assume that the processing starts at $\tau = 0$ and then occurs repetitively, as said earlier. We then denote by S_1^n the instant of time at which the transition t_1 initiates its n-th firing, corresponding to the n-th occurrence of this particular processing task. Naturally, $(S_1^n - S_1^{n-1})$ represents the time that has elapsed between the (n-1)-th occurrence and the n-th occurrence of transition t_1 . This can also be interpreted as a cycle time, since the system is assumed to function cyclically, each cycle corresponding in fact to the complete processing of one input. The definition of the average cycle time is therefore as follows:

Definition: The average cycle time, denoted by θ_1 , of transition t_1 is defined as:

$$\theta_1 = \lim_{n \rightarrow +\infty} \frac{\sum_{k=1}^n (S_1^k - S_1^{k-1})}{n} \quad (4.1)$$

This is naturally an average measure, since the time interval between two successive occurrences of transition t_1 is averaged over the total number of occurrences (from the initial instant). Relation (4.1) can actually be written in a more simple form:

$$\theta_1 = \lim_{n \rightarrow +\infty} \frac{S_1^n - S_1^0}{n} = \lim_{n \rightarrow +\infty} \frac{S_1^n}{n} - \lim_{n \rightarrow +\infty} \frac{S_1^0}{n}$$

and S_1^0 being a constant value:

$$\lim_{n \rightarrow +\infty} \frac{S_1^0}{n} = 0$$

so that:

$$\theta_1 = \lim_{n \rightarrow +\infty} \frac{S_1^n}{n} \quad (4.2)$$

This will be the expression used later in the developments. Naturally,

$$\bar{\theta}_1 = \frac{1}{\theta_1}$$

represents the average firing rate of transition t_1 .

We are now going to prove the following result:

Theorem 4.1: All the transitions have the same average cycle time and consequently the same average firing rate.

As it will be shown, this result holds because it is possible to find, for any two transitions, a directed circuit that includes them. The result is therefore crucially dependent on the fact that the Petri net model of the DMO corresponds to a strongly connected Event-Graph, as stated in Chapter 3, Section 3.5.

Proof: Let t_i and t_j be any two transitions. There exists at least one directed circuit that contains both transitions, because the net is strongly connected: Indeed, we can find a directed path from t_i to t_j and from t_j to t_i . Let us point out that the directed circuit is not necessarily elementary (as defined in Chapter 2, Section 2.3.1), but this would not change the nature of the proof given here: In that case we could split the circuit into a set of elementary circuits. Without loss in generality, we can therefore denote by

$$p = (t_{i_1} \ p_{i_2} \ t_{i_2} \ \dots \ t_{i_k} \ p_{i_k} \ \dots \ p_{i_r})$$

the corresponding circuit, where:

$$t_{i_1} = t_i \quad \text{and} \quad t_{i_k} = t_j$$

The circuit is shown on Figure 4.1. Because the net is an Event-Graph, each place of the circuit has exactly one input and one output transition and the token content of the circuit is invariant by transition firings (Theorem 2.3, Chapter 2, Section 2.3.2). Let M_k^0 denote the initial marking of place p_k and $M_{i_j}^0$ (resp. $M_{j_i}^0$) the number of tokens in the places belonging to the path from t_i and t_j (resp. from t_j to t_i), i.e.,

$$M_{ij}^{\circ} = \sum_{p=1}^{k-1} M_{ip}^{\circ} \quad \text{and} \quad M_{ji}^{\circ} = \sum_{p=k}^r M_{ip}^{\circ}$$

Let us recall from Chapter 2, Section 2.4.1, that we denote by:

$I_i(\tau)$ (resp. $I_j(\tau)$), the number of initiations of transition t_i (resp. t_j) in the interval of time $[0, \tau]$.

$T_i(\tau)$ (resp. $T_j(\tau)$), the number of terminations of transition t_i (resp. t_j) in the interval $[0, \tau]$.

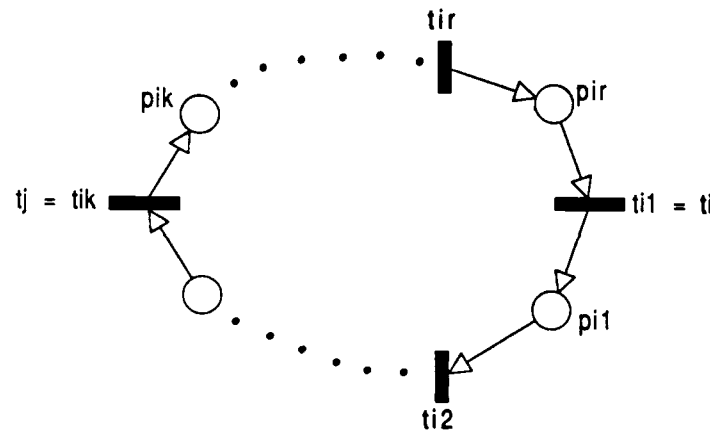


Figure 4.1 Directed Circuit Containing t_i and t_j

The processing is assumed to start at $\tau=0$ (no transitions are therefore executing at $\tau=0$) so that, at any instant τ , we have trivially:

$$\begin{aligned} T_i(\tau) &\leq I_i(\tau) \\ T_j(\tau) &\leq I_j(\tau) \end{aligned} \tag{4.3}$$

Now, since in the firing process the initiation of a transition "consumes"

one token from its input places and the termination "produces" one token in its output places, we have trivially, at any instant τ , for the transitions t_i and t_j considered:

$$I_i(\tau) \leq M_{ji}^0 + T_j(\tau) \quad (4.4)$$

$$I_j(\tau) \leq M_{ij}^0 + T_i(\tau) \quad (4.5)$$

Using inequality (4.3), (4.4) and (4.5) yields:

$$I_i(\tau) - M_{ji}^0 \leq I_j(\tau) \quad (4.6)$$

$$I_i(\tau) + M_{ij}^0 \geq I_j(\tau) \quad (4.7)$$

Consider the instant of time

$$\tau = S_i^{n+M_{ji}^0}$$

where n is any positive integer (this is the instant at which the $(n+M_{ji}^0)$ -th firing initiation of transition t_i occurs). By definition, at this instant:

$$I_i(\tau) = n + M_{ji}^0$$

Replacing in (4.5) yields:

$$I_j(\tau) \geq n$$

Otherwise stated, the number of initiations of transition t_j in $[0, \tau]$ is at least equal to n , which implies:

$$S_j^n \leq \tau = S_i^{n+M_{ji}^{\circ}}$$

Likewise, considering the instant of time:

$$\tau = S_i^{n-M_{ij}^{\circ}}$$

where n is assumed greater than M_{ij}° , then:

$$I_i(\tau) = n - M_{ij}^{\circ}$$

and replacing in (4.7) yields:

$$n \geq I_j(\tau)$$

which implies this time:

$$S_j^n \geq \tau = S_i^{n-M_{ij}^{\circ}}$$

Finally, for any positive integer n , such that $n > M_{ij}^{\circ}$, we have:

$$S_i^{n-M_{ij}^{\circ}} \leq S_j^n \leq S_i^{n+M_{ji}^{\circ}}$$

Hence, dividing by n :

$$\frac{n - M_{ij}^{\circ}}{n} \frac{S_i^{n-M_{ij}^{\circ}}}{n - M_{ij}^{\circ}} \leq \frac{S_j^n}{n} \leq \frac{S_i^{n+M_{ji}^{\circ}}}{n + M_{ji}^{\circ}} \frac{n + M_{ji}^{\circ}}{n}$$

By taking the limits as n goes to infinity:

$$\lim_{n \rightarrow +\infty} \frac{n - M_{ij}^{\circ}}{n} \frac{S_i^{n-M_{ij}^{\circ}}}{n - M_{ij}^{\circ}} \leq \lim_{n \rightarrow +\infty} \frac{S_j^n}{n} \leq \lim_{n \rightarrow +\infty} \frac{S_i^{n+M_{ji}^{\circ}}}{n + M_{ji}^{\circ}} \frac{n + M_{ji}^{\circ}}{n}$$

which means exactly:

$$\theta_i \leq \theta_j \leq \theta_i$$

and therefore: $\theta_i = \theta_j$. Q.E.D.

In the next sections, we will denote by θ the unique average cycle time and

$$\bar{\Phi} = \frac{1}{\theta}$$

will denote the average firing rate of all the transitions. $\bar{\Phi}$ characterizes the average processing rate of the overall system and for that reason, will be called the throughput rate.

4.2.2. Average Circuit Processing Time

In the previous section, we have proved in that any two transitions of a directed circuit have the same average cycle time. It is now possible to determine what would this average cycle time be, assuming that the circuit were by itself, i.e., could function independently of the remaining system.

Let us consider a directed circuit, as shown in Figure 4.2, that we denote, without loss of generality by $\rho = (t_1 p_1 t_2 \dots t_k p_k)$. Let us assume now that the circuit has only one token in it, which is initially in place p_1 . Let $\mu(\rho)$ denote the sum of the transition firing times, i.e.,

$$\mu(\rho) = \sum_{i=1}^k \mu_i$$

where $\mu_i = \mu(t_i)$ is the firing time of t_i , as defined in Chapter 2, Section 2.4.1. We will begin by assuming that the firing times are deterministic and will then extend the analysis to the random case. Now, in the steady state process, the token fires every transition in the circuit in turn and reappears in p_1 every $\mu(\rho)$ seconds, assuming that no time is allowed to elapse between a transition being enabled and being initiated. Quite naturally, we shall call $\mu(\rho)$ the circuit processing time corresponding to the amount of time it takes to complete the processing operations of the circuit, under the previous assumption. In that case, every transition fires at intervals of $\mu(\rho)$ seconds: therefore, the cycle time of any transition of the circuit is precisely $\mu(\rho)$, i.e., $\theta = \mu(\rho)$.

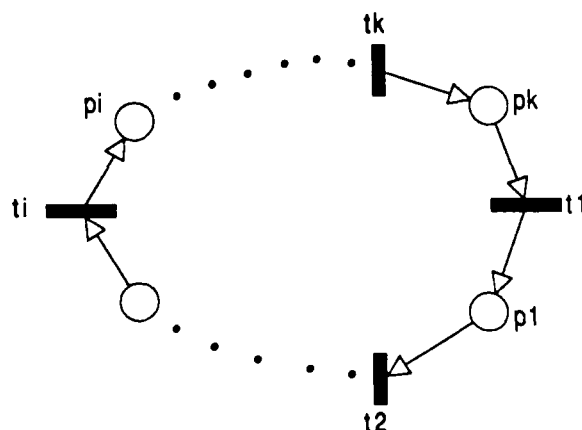


Figure 4.2 Directed Circuit

Now, suppose that the circuit has n tokens instead of 1, assuming that

all the tokens are identical (which is always implicit in ordinary Petri Nets). Then, the average cycle time of every transition of the circuit becomes:

$$\theta = \frac{\mu(\rho)}{n}.$$

Clearly enough,

$$\frac{\mu(\rho)}{n}$$

represents the average amount of time between a token moving from p_i and a token reappearing in p_i . Extending the previous analysis, we shall call this ratio the average circuit processing time. At this point, it is interesting to recall that the token content of a circuit represents, in our model, the resources available for processing the corresponding operations of the circuit. Hence, the average circuit processing time is determined as the sum of the task processing times divided by the number of resources available. Since the resources (i.e., the token content) are specified by the initial marking of the places of the circuit, the precise definition is finally the following one:

Definition: Let ρ be any directed circuit that we denote (without loss of generality) by $\rho = (t_1 p_1 \dots t_1 p_1 \dots t_k p_k)$. The average circuit processing time, denoted by $\alpha(\rho)$, is defined as:

$$\alpha(\rho) = \frac{\mu(\rho)}{M^0(\rho)} = \frac{\sum_{i=1}^k \mu_i}{\sum_{i=1}^k M_i^0} \quad (4.8)$$

where:

$\mu_i = \mu(t_i)$ denotes the firing time of transition t_i .

M_i^0 denotes the initial marking of place p_i , and

$M^0(\rho)$ characterizes the token content of the circuit. Furthermore, the quantity

$$\frac{1}{a(\rho)}$$

will be called the average circuit processing rate.

Now, for the random process, corresponding to the case where the transition firing times may be discrete random variables (with a finite set of values) with a fixed probability distribution, this definition is easily extended by taking the expected firing time for each transition. For instance, let us suppose that the firing time of transition t_i is a random variable that can take the values

$$\{\mu_{i,1}, \mu_{i,2}, \dots, \mu_{i,u}\}$$

according to the probability distribution

$$\{\gamma_{i,1}, \gamma_{i,2}, \dots, \gamma_{i,u}\}$$

where u denotes the number of possible processing times for transition t_i . The probability distribution satisfies, as usual, the equality

$$\sum_{j=1}^{j=u} \gamma_{i,j} = 1 \quad (4.9)$$

For simplicity, we assume that the processing times of all the other transitions but t_i are deterministic. The results can be extended

trivially to the case where more than one transition has a non-deterministic processing time. We also assume, for clarity, that the token content of the circuit is one. Let us denote then by $\bar{\mu}_1$ the expected firing time of t_1 , i.e.,

$$\bar{\mu}_1 = \sum_{j=1}^{j=u} \gamma_{1,j} \mu_{1,j} \quad (4.10)$$

It turns out that the circuit processing time, i.e., the amount of time it takes for the token to complete the processing operation of the circuit, takes the value

$$\mu_{1,j}(\rho) = \mu_1 + \mu_2 + \dots + \mu_{1,j} + \dots + \mu_k$$

whenever the firing time of transition t_1 takes the value: $\mu(t_1) = \mu_{1,j}$. If we consider a large number of repetitions of the circuit, i.e., assuming that the token runs cyclically in the circuit, the expected circuit processing time will be:

$$\bar{\mu}(\rho) = \sum_{j=1}^u \gamma_{1,j} (\mu_1 + \mu_2 + \dots + \mu_{1,j} + \dots + \mu_k)$$

which can be written simply, given (4.9) and (4.10)

$$\bar{\mu}(\rho) = \mu_1 + \mu_2 + \dots + \bar{\mu}_1 + \dots + \mu_k$$

Naturally, in the general case, we would obtain

$$\bar{\mu}(\rho) = \bar{\mu}_1 + \bar{\mu}_2 + \dots + \bar{\mu}_1 + \dots + \bar{\mu}_k$$

assuming that all the transition firing times are discrete random variables. Likewise, if we assume that there are n tokens, instead of one, given the assumption that the probability distribution is independent from one token to the next, as stated in Chapter 3, Section 3.4, the average circuit processing time would be:

$$\alpha(\rho) = \frac{\bar{\mu}(\rho)}{n} \quad (4.11)$$

Hence, in the non-deterministic case, the average circuit processing time, as defined by (4.11) determines the average cycle time of all the transitions. Otherwise stated, (4.8) is simply extended by taking, in that case, the expected firing time of each transition.

In brief, we have showed that, under the assumption that the circuit is by itself, and that no time would elapse between a transition being enabled and being initiated, the average circuit processing time, as defined previously, determines the average cycle time of every transition of the circuit.

4.2.3 Maximum Average Circuit Processing Time

In the previous section, we have considered a circuit of the net, as if it were isolated from the remaining system and we have determined in that case what would be the average cycle time of each transition of the circuit. The fundamental assumption was that there was no time delays between the different processing operations. Considering now the system as a whole, it is clear that the different circuits are in fact interconnected, as shown for instance in Figure 4.3.

It turns out that the interconnected circuits will affect each others processing time. For instance, in the example of Figure 4.3, transition t_1 belongs to both circuit ρ_1 and ρ_2 . Clearly enough, the average cycle time of this transition cannot to lower than the maximum of $\alpha(\rho_1)$ and $\alpha(\rho_2)$,

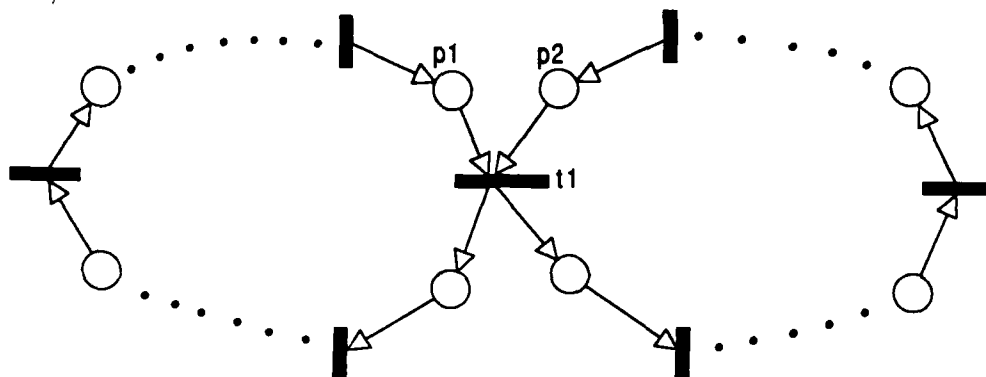


Figure 4.3 Interconnected Circuits

which are the average processing time of each of the circuit. Indeed, in order for t_1 to be initiated, both places p_1 and p_2 should contain at least one token. Since we have proved, in Section 4.2.1., that the average cycle time, θ , is the same for all the transitions of the net, it should intuitively be clear that θ , in the, general case, cannot be lower than:

$$\alpha = \max (\alpha(\rho_1), \alpha(\rho_2), \dots, \alpha(\rho_r))$$

where $\rho_1, \rho_2, \dots, \rho_r$ denote all the directed circuits of the net. That is what we are now going to prove rigorously.

Definition: Let $\rho_1, \rho_2, \dots, \rho_r$ denote all the directed circuits of the net. We call maximum average circuit processing time, denoted by α , the value:

$$\alpha = \max (\alpha(\rho_1), \alpha(\rho_2), \dots, \alpha(\rho_r)) \quad (4.12)$$

where $\alpha(\rho_1)$ is the average processing time of circuit ρ_1 , as defined by relation (4.8). Naturally,

$$\frac{1}{\alpha} = \min \left(\frac{1}{\alpha(\rho_1)}, \dots, \frac{1}{\alpha(\rho_r)} \right)$$

will be called the minimum average circuit processing rate. At this point, it is emphasized that the minimum is taken over all the directed circuits of the net.

We have now the following result:

Theorem 4.2: The maximum average circuit processing time is a lower bound of the average cycle time, i.e., $\theta \geq \alpha$.

Let us first point out that this result holds only for the type of Petri Nets used here to modeling the DMO, i.e., strongly connected Event-Graphs. As already stated in the introduction of this chapter, this result was first obtained by Ranchandami [6] but with restrictive assumptions: In his work, the transition firing times were assumed deterministic and, moreover, the steady-state process was assumed strongly periodic (i.e., transitions fire at regular intervals of time). We extend the result to the non-deterministic case, where firing times may be discrete random variables. In addition, we do not make any assumption regarding the periodicity of the deterministic system, which will be shown in the next chapter to be of a more complex type (as already mentioned before, the deterministic process is in fact K-periodic).

In order to prove Theorem 4.2, it is sufficient to prove that, for any directed circuit ρ of the net, we have necessarily: $\theta \geq \alpha(\rho)$. This is actually the proof given below.

Proof: Let ρ be any directed circuit that we denote (without loss of generality) $\rho = (t_1 p_1 t_2 \dots t_k p_k)$. Let us consider, for instance, the sequence S_1^n ($n = 1, 2, 3, \dots$) corresponding to the instant at which the firings of transition t_1 are initiated successively. We first assume that

all the transition firing times are deterministic.

Now, let us consider any couple of transitions (t_i, t_{i+1}) of the circuit that are connected by place p_i as shown in Figure 4.2 (p_i is the output place of t_i and the input place of t_{i+1} in the circuit). The inequalities (4.4) - (4.5) established previously for (t_i, t_j) can be applied here for (t_i, t_{i+1}) , so that, at any instant τ :

$$I_{i+1}(\tau) \leq M_i^{\circ} + T_i(\tau) \quad (4.13)$$

where M_i° denotes the initial marking (at $\tau=0$) of place p_i .

Let us consider then the instant:

$$\tau = S_i^{n-M_i^{\circ}} + \mu_i$$

where n is any positive integer, such that $n > M_i^{\circ}$ and μ_i is the firing time of t_i . τ is the instant of the $(n-M_i^{\circ})$ -th firing termination of transition t_i , hence at this instant:

$$T_i(\tau) = n - M_i^{\circ}$$

Using relation (4.13) yields:

$$I_{i+1}(\tau) \leq n$$

which implies:

$$S_{i+1}^n \geq \tau = S_i^{n-M_1^0} + \mu_1 \quad (4.14)$$

Now, by iterating (4.14) from $i=1$ to $i=k$, we obtain:

$$S_1^n = S_{k+1}^n \geq S_1^{n-M^0(\rho)} + \sum_{i=1}^k \mu_i$$

where

$$M^0(\rho) = \sum_{i=1}^k M_i^0$$

Finally, for any positive integer n , such that $n > M^0(\rho)$:

$$S_1^n \geq S_1^{n-M^0(\rho)} + \mu(\rho) \quad (4.15)$$

where $M^0(\rho)$ denotes the token content of the circuit and $\mu(\rho)$ the circuit processing time.

Let us now take a place of the circuit which has a non-null initial marking (i.e., contains initially at least one token). We will assume, without loss in generality, that this is the case for place p_1 . If we consider p_1 as the origin for the circuit, we shall say that the circuit has been repeated exactly K times ($K=1,2,3,\dots$) when all the tokens have gone through p_1 exactly K times (i.e., all the tokens have been processed K times through the circuit). In that case, the number, n , of firing occurrences of transition t_1 is determined by:

$$\begin{cases} n = K M^{\circ}(\rho) + r \\ 0 \leq r < M^{\circ}(\rho) \end{cases}$$

where r denotes the number of tokens that are initially not in place p_1 . Indeed, all these tokens must be fired by transition t_1 once to reach place p_1 (the origin) for the first time.

By iterating (4.15) for $n = r, M^{\circ}(\rho) + r, \dots, K M^{\circ}(\rho) + r$, we obtain:

$$S_1^n \geq S_1^r + K \mu(\rho) \quad (4.16)$$

Now, dividing by n , we obtain:

$$\frac{S_1^n}{n} \geq \frac{S_1^r}{K M^{\circ}(\rho) + r} + \frac{K}{K M^{\circ}(\rho) + r} \mu(\rho)$$

Finally, when n goes to infinity:

$$\lim_{n \rightarrow +\infty} \frac{S_1^n}{n} \geq \underbrace{\lim_{K \rightarrow +\infty} \frac{S_1^r}{K M^{\circ}(\rho) + r}}_{= 0} + \underbrace{\lim_{K \rightarrow +\infty} \frac{K}{K M^{\circ}(\rho) + r} \mu(\rho)}_{= \frac{\mu(\rho)}{M^{\circ}(\rho)}}$$

which means, given (4.2) and (4.8):

$$\theta = \lim_{n \rightarrow +\infty} \frac{S_1^n}{n} \geq \alpha(\rho) = \frac{\mu(\rho)}{M^{\circ}(\rho)} \quad \text{Q.E.D.}$$

In case where the firing times are discrete random variables, it is

not difficult to extend the proof. Suppose, for instance, that the firing time of t_i is a random variable that can take the values

$$\{\mu_{i,1}, \mu_{i,2}, \dots, \mu_{i,u}\}$$

according to the probability distribution

$$\{\gamma_{i,1}, \gamma_{i,2}, \dots, \gamma_{i,u}\}$$

In addition, all the other processing times are deterministic, exactly as we have assumed in the previous section. Recall that the circuit processing time takes the value

$$\mu_{ij}(\rho) = \mu_1 + \mu_2 + \dots + \mu_{i,j} + \dots + \mu_k$$

with probability $\gamma_{i,j}$ when $\mu(t_i) = \mu_{i,j}$. Clearly enough, (4.16) becomes now:

$$S_1^n \geq S_1^r + \sum_{j=1}^u K_j \mu_{i,j}(\rho) \quad (4.17)$$

where K_j denotes the number of times that the circuit processing time was equal to $\mu_{i,j}(\rho)$ for the K occurrences of the circuit. By the definition of the probability distribution:

$$\lim_{K \rightarrow +\infty} \frac{K_j}{K} = \gamma_{i,j} \quad \text{for } j = 1, 2, \dots, u.$$

From (4.17), we deduce:

$$\frac{S_1^n}{n} \geq \frac{S_1^r}{K M^o(\rho) + r} + \sum_{j=1}^u \frac{K_j}{K M^o(\rho) + r} \mu_{1,j}(\rho)$$

When n goes to infinity, we obtain the same result given in (4.11):

$$\theta = \lim_{n \rightarrow \infty} \frac{S_1^n}{n} \geq \sum_{j=1}^u \frac{\gamma_{1,j}}{M^o(\rho)} \mu_{1,j}(\rho) = \frac{\bar{\mu}(\rho)}{M^o(\rho)} = \alpha(\rho)$$

Remark: In the determination of the maximum average circuit processing time (as defined by (4.12)), it is necessary to include all the transition processing times. Indeed, in our model of the DMO, each transition contains in fact a self-loop with one token, as discussed in Chapter 2, Section 2.4.2. Therefore, each transition constitutes by itself a circuit, whose token content is one. In that case, the average circuit processing time is trivially the processing time of the transition. In order to differentiate later the self-loops from the other directed circuits, we will refer to the former as the trivial circuits.

4.2.4 Maximum Throughput Rate

In the previous section, we have found that the maximum average circuit processing time, as determined by (4.12), is a lower bound of the average cycle time, θ . Naturally, this result can also be stated this way:

The throughput rate of the system, $\bar{\Phi} = 1/\theta$, as defined in Section 4.2.1, is bounded from above by the minimum average circuit processing rate, i.e., by $1/\alpha$

The problem before us now is to know whether or not this bound is actually achievable. In other words, we have to determine whether the minimum average circuit processing rate characterizes the maximum

throughput rate of the system. It is clear that if the arrival rate of external inputs is low enough, the DMO will be able to handle all the arriving inputs and the throughput rate will be precisely the rate at which inputs arrive. However, there is a rate above which the DMO will be overloaded, in the sense that inputs will queue at the entry of the system and this queue will grow to infinity over time. This rate characterizes the maximum throughput rate of the DMO, i.e., the maximum rate at which inputs can be processed in the organization. From the analysis carried out so far, we know that the maximum throughput rate is necessarily lower or equal to the minimum average circuit processing rate, as determined by $1/\alpha$. Indeed, the average rate at which transitions can fire, i.e., θ , cannot exceed $1/\alpha$ (Theorem 4.2): Therefore, if the arrival rate of inputs exceeds $1/\alpha$, we are sure that the queue of inputs will grow to infinity over time. As we are going to see, $1/\alpha$ characterizes in fact the maximum throughput rate for the deterministic system, i.e., when all the transition firing times are deterministic. It will be shown that it is generally not true for the non-deterministic case.

4.2.4.1 Deterministic System

For the deterministic system, the following result holds:

Theorem 4.3: The minimum average circuit processing rate determines the maximum throughput rate.

As stated earlier, we have proved that the minimum average circuit processing rate, as determined by $1/\alpha$ (relation (4.12)), bounds the maximum throughput rate. We will not prove here that $1/\alpha$ corresponds effectively to the maximum throughput rate. This result will be derived from the analysis carried out in Chapter 5, which concerns the execution schedule of the system. In Chapter 5, we will determine the precise instants of time at which transitions fire, assuming that the process occurs repetitively and that the firings are initiated as soon as the transitions are enabled. From the execution schedule obtained, it will be shown that the (average)

rate at which the transition fires is determined by the minimum average circuit processing rate.

Theorem 4.3 allows the maximum throughput rate to be computed very easily, once we have determined all the directed circuits of the net (which only depend on the structure of the Petri Net). In order to determine all the directed circuits, we have used an algorithm developed by Alaiwan and Toudic [19] that determines the S-invariants (Chapter 2, Section 2.2) of an ordinary Petri Net. This algorithm is described in Appendix A of the thesis.

Let us analyze further how we can relate the evaluation of the maximum throughput rate to the resource and time constraints. To that extent, we need a further definition characterizing the critical circuits.

Definition: A circuit ρ_i is said to be critical, if its average processing time is maximal over all the directed circuits, i.e.,

$$\alpha = \max (\alpha(\rho_1) , \dots , \alpha(\rho_r)) = \alpha(\rho_1)$$

The following corollary of Theorem 4.3 is straightforward:

Corollary 4.3: The average processing rate of the critical circuits determines the maximum throughput rate (for the deterministic system).

This corollary is very interesting because it provides a way for determining which resource and time constraints are actually binding, i.e., bound the throughput rate. Indeed, these will be the task processing times (resp. resources) that correspond to the transitions (resp. token content) of the critical circuits. Hence, the problem of modifying the right constraints so as to improve the maximum throughput rate becomes straightforward. In addition, by determining the critical circuits of different organizational structures, a comparative study of performance can

be achieved, with respect to the throughput rate. This type of performance evaluation will be carried out in the last section of this chapter for the example of the two DM organization.

4.2.4.2 Non-Deterministic System

Let us recall that the characteristic of the non-deterministic system studied here is the following: The transition firing times are discrete random variables, that can take a finite number of possible values, according to a fixed probability distribution (i.e., independent from one token to the next). As discussed in Chapter 3, Section 3.4, these assumptions allow for modeling the decision nodes or switches. Recall also that the average circuit processing time is computed by taking the expected (or mean) firing time of each transition. In that case, it turns out that, in general, the maximum throughput rate is lower than the minimum average circuit processing rate, as defined by relation (4.12). Let us see why on a simple example.

Let us consider the Petri Net shown on Figure 4.4, which is a strongly connected Event-Graph with the following two circuits:

$$\rho_1 = (t_0 \ p_1 \ t_1 \ p_2)$$

$$\rho_2 = (t_0 \ p_3 \ t_2 \ p_4)$$

Now, suppose that:

- the token content of both circuit is one
- the firing time of t_0 is deterministic: $\mu_0=1$.
- the set of possible firing times for transition t_1 is $\mu_1 = \{1,3\}$ according to the probability distribution: $\{\gamma_{1,1} = 0.5, \gamma_{1,3} = 0.5\}$

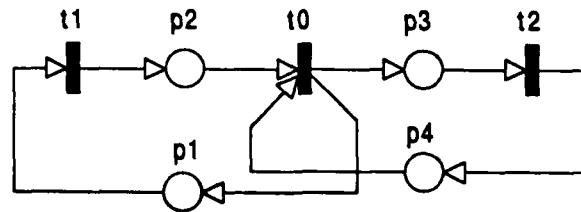


Figure 4.4 Non Deterministic Example

- the set of possible firing times for t_2 is $\mu_2 = \{2, 4\}$, according to the probability distribution: $\{\gamma_{2,1} = 0.5, \gamma_{2,2} = 0.5\}$.

There are four deterministic cases that correspond to the random system. In each case, we can easily compute the best cycle time achievable, as determined by the maximum average circuit processing time (Theorem 2.3). These are:

- for $\mu_1 = 1, \mu_2 = 2$ then $\alpha_1 = \alpha(p_2) = \mu_2 + \mu_0 = 3$
- for $\mu_1 = 1, \mu_2 = 4$ then $\alpha_2 = \alpha(p_2) = \mu_2 + \mu_0 = 5$
- for $\mu_1 = 3, \mu_2 = 2$ then $\alpha_3 = \alpha(p_1) = \mu_1 + \mu_0 = 4$
- for $\mu_1 = 3, \mu_2 = 4$ then $\alpha_4 = \alpha(p_1) = \mu_1 + \mu_0 = 5$

Given the probability distribution, the average cycle time of the system (taken over a large number of repetitions of the process) is clearly:

$$\begin{aligned} \theta &= \gamma_{1,1}\gamma_{2,1}\alpha_1 + \gamma_{1,1}\gamma_{2,2}\alpha_2 + \gamma_{1,2}\gamma_{2,1}\alpha_3 + \gamma_{1,2}\gamma_{2,2}\alpha_4 \\ &= 0.25 (3+5+4+5) = 4.25 \end{aligned}$$

As said earlier, the maximum throughput rate corresponds to the average rate at which transitions fire, assuming that the system works continuously and that the firings are initiated as soon as the transitions are enabled.

By construction, $\bar{q} = 1/4.25$ determines the maximum throughput rate in that case. If we compute now the maximum average circuit processing time of the random system, by taking the expected firing time of t_1 and t_2 , we obtain:

$$\alpha(\rho_1) = \bar{\mu}_1 + \mu_0 = 0.5(1+3) + 1 = 3$$

$$\alpha(\rho_2) = \bar{\mu}_2 + \mu_0 = 0.5(2+4) + 1 = 4$$

$$\alpha = \max(\alpha(\rho_1), \alpha(\rho_2)) = 4$$

and therefore the minimum average circuit processing rate equals:

$$\frac{1}{\alpha} = \frac{1}{4}$$

Finally, it turns out that:

$$\bar{q} = \frac{1}{4.25} < \frac{1}{\alpha} = \frac{1}{4}$$

This example proves that, in contrast to the deterministic system, the maximum throughput rate can be lower than the minimum average circuit processing rate, as determined by $1/\alpha$. The reason should be clear from this example: the critical circuits are not necessarily the same among the deterministic systems obtained by setting a deterministic value (among those possible) for each transition firing time. In our example, when $\mu_1=1$ and $\mu_2=2$, ρ_2 is the critical circuit and when $\mu_1 = 3$, $\mu_2 = 2$, the critical circuit is ρ_1 . By averaging the corresponding circuit processing times over the probability distribution, we do not necessarily obtain $\alpha(\rho)$, where ρ is the critical circuit obtained when the different circuit processing times are computed by taking the expected firing times of each transition.

In the non-deterministic case, we can obtain a lower and an upper bound for the maximum throughput rate. From the analysis above, the upper bound is given by $1/\alpha$, the minimum average circuit processing rate. If the arrival rate of inputs exceeds $1/\alpha$, we are sure that the DMO will be overloaded, i.e., the queue of inputs will grow to infinity over time. Now, the lower bound is trivially obtained by assigning to each transition its worst, i.e., longest, firing time possible and computing the minimum average circuit processing rate for the corresponding deterministic system. For arrival rates of inputs that do not exceed this lower bound, we are sure that the DMO will be able to handle all inputs. Interestingly enough, if the upper and lower bound turned out to be the same, which is the case, when in our model of the DMO the critical circuits do not contain switches, we have then completely characterized the maximum throughput rate. Let us point out that, as for the deterministic system, the lower and upper bound so determined are directly expressed as a function of both the time and resource constraints. Therefore, the performance evaluation, based on the circuits that are critical can also be applied here.

4.3 APPLICATION TO THE TWO MEMBER ORGANIZATION

In this section, we illustrate the results obtained so far using the example of the two member organization presented in Chapter 3, Section 3.5. We recall that the first structure, shown on Fig. 3.7, includes an interaction of the result sharing type between the two DMs, while the second one, shown on Fig. 3.8, implies a hierarchical relationship between them.

4.3.1 Analysis of the Processing Rate Constraints

In order to compute the maximum throughput rate, it is first necessary to determine all the directed circuits of the net. The algorithm developed by Alawain and Toudic [19], that is described in Appendix A, has been used. In this simple example, it is however easy to check the directed circuits graphically. Given the labeling of transitions and places shown in Figs.

3.7 and 3.8, there are three directed circuits that have the same structure in both cases. These are:

$$\rho_1 = (t_2, p_2, t_3, p_3, t_4, p_4, t_5, R_1)$$

$$\rho_2 = (t_6, p_6, t_7, p_7, t_8, p_8, t_9, R_2)$$

$$\rho_3 = (t_1, p_1, t_6, p_6, t_7, p_7, t_8, p_8, t_9, p_{11}, t_{10}, R_0)$$

Additionally, for structure (a), Fig. 3.7,

$$\rho_{4a} = (t_6, p_1, t_3, p_2, t_4, p_4, t_5, p_{10}, t_7, p_7, t_8, p_8, t_9, R_2)$$

$$\rho_{5a} = (t_1, p_1, t_2, p_2, t_3, p_3, t_4, p_4, t_5, p_{10}, t_7, p_7, t_8, p_8, t_9, p_{11}, t_{10}, R_0)$$

$$\rho_{6a} = (t_1, p_1, t_6, p_6, t_7, p_7, t_8, p_8, t_9, p_{11}, t_{10}, R_0)$$

and for structure (b), Fig. 3.8,

$$\rho_{4b} = (t_6, p_1, t_3, p_2, t_4, p_4, t_5, p_{10}, t_7, p_7, t_8, p_8, t_9, R_2)$$

$$\rho_{5b} = (t_1, p_1, t_2, p_2, t_3, p_3, t_4, p_4, t_5, p_{10}, t_7, p_7, t_8, p_8, t_9, p_{11}, t_{10}, R_0)$$

$$\rho_{6b} = (t_1, p_1, t_6, p_6, t_7, p_7, t_8, p_8, t_9, p_{11}, t_{10}, R_0)$$

If we denote by μ_1 the processing time of transition t_1 and by n_0 (resp. n_1, n_2) the amount of resources (i.e., the number of tokens that is initially in the corresponding resource place) available for processing the organizational inputs (resp. the inputs to DM1 and DM2), then the average circuit processing times are:

$$\alpha(\rho_1) = \frac{\mu_2 + \mu_3 + \mu_4 + \mu_5}{n_1}$$

$$\alpha(\rho_2) = \frac{\mu_6 + \mu_7 + \mu_8 + \mu_9}{n_2}$$

$$\alpha(\rho_3) = \frac{\mu_1 + \mu_6 + \mu_7 + \mu_8 + \mu_9 + \mu_{10}}{n_0}$$

$$\alpha(\rho_{4a}) = \frac{\mu_6 + \mu_3 + \mu_4 + \mu_5 + \mu_7 + \mu_8 + \mu_9}{n_2}$$

$$\alpha(\rho_{5a}) = \frac{\mu_1 + \mu_2 + \mu_3 + \mu_4 + \mu_5 + \mu_7 + \mu_8 + \mu_9 + \mu_{10}}{n_0}$$

$$\alpha(\rho_{6a}) = \frac{\mu_1 + \mu_6 + \mu_3 + \mu_4 + \mu_5 + \mu_7 + \mu_8 + \mu_9 + \mu_{10}}{n_0}$$

$$\alpha(\rho_{4b}) = \frac{\mu_6 + \mu_3 + \mu_4 + \mu_5 + \mu_8 + \mu_9}{n_2}$$

$$\alpha(\rho_{5b}) = \frac{\mu_1 + \mu_2 + \mu_3 + \mu_4 + \mu_5 + \mu_8 + \mu_9 + \mu_{10}}{n_0}$$

$$\alpha(\rho_{6b}) = \frac{\mu_1 + \mu_6 + \mu_3 + \mu_4 + \mu_5 + \mu_8 + \mu_9 + \mu_{10}}{n_0}$$

We recall that, in order to determine the maximum throughput rate, it is also necessary to take into account the processing times of the individual transitions, which constitute the trivial circuits of the net. Let us denote by:

$$\mu_{\max} = \max_{i=1, \dots, 10} \mu_i$$

Then, the maximum average circuit processing time is, for structure (a):

$$\alpha_a = \max (\alpha(\rho_1), \alpha(\rho_2), \alpha(\rho_3), \alpha(\rho_{4a}), \alpha(\rho_{5a}), \alpha(\rho_{6a}), \mu_{\max})$$

and, for structure (b):

$$\alpha_b = \max (\alpha(\rho_1), \alpha(\rho_2), \alpha(\rho_3), \alpha(\rho_{4b}), \alpha(\rho_{5b}), \alpha(\rho_{6b}), \mu_{\max})$$

However, we have seen in Chapter 3, Section 3.2, (relation (3.5)) that:

$$\phi_1 = \min \left(\frac{1}{\alpha(\rho_1)}, \frac{1}{\mu_2}, \frac{1}{\mu_3}, \frac{1}{\mu_4}, \frac{1}{\mu_5} \right)$$

describes the maximum processing rate that characterizes DM1. Identically, for DM2, it is:

$$\phi_2 = \min \left(\frac{1}{\alpha(\rho_2)}, \frac{1}{\mu_6}, \frac{1}{\mu_7}, \frac{1}{\mu_8}, \frac{1}{\mu_9} \right)$$

Moreover, it turns out that:

$$\alpha(\rho_{6a}) = \alpha(\rho_3) + \frac{\mu_3 + \mu_4 + \mu_5}{n_0}$$

Hence $\alpha(\rho_{6a}) \geq \alpha(\rho_3)$, whatever the processing times and the resources may be. Similarly, we have always:

$$a(\rho_{\epsilon b}) \geq a(\rho_{\epsilon})$$

We deduce finally that the minimum average circuit processing rate is determined for structure (a), by:

$$\bar{\Phi}_a = \frac{1}{a_a} = \min \left(\phi_1, \phi_2, \frac{1}{a(\rho_{\epsilon a})}, \frac{1}{a(\rho_{s a})}, \frac{1}{a(\rho_{\epsilon a})}, \frac{1}{\mu_1}, \frac{1}{\mu_{10}} \right)$$

and for structure (b):

$$\bar{\Phi}_b = \frac{1}{a_b} = \min \left(\phi_1, \phi_2, \frac{1}{a(\rho_{\epsilon b})}, \frac{1}{a(\rho_{s b})}, \frac{1}{a(\rho_{\epsilon b})}, \frac{1}{\mu_1}, \frac{1}{\mu_{10}} \right)$$

Now, the expressions above are especially interesting to interpret. It turns out that the throughput rate of the DMO is bounded by three types of constraints:

- (i) The constraint that comes from the maximum processing rate of each DM, as defined by ϕ_1 and ϕ_2 .
- (ii) The constraints due to the structure of the organization, as determined by the interactions between the DMs and between the DMs and the environment. In the example, these constraints are derived from the average processing rates of the circuits $\rho_{\epsilon a}$, $\rho_{s a}$ and $\rho_{\epsilon a}$ (resp. $\rho_{\epsilon b}$, $\rho_{s b}$ and $\rho_{\epsilon b}$). We will refer to these constraints as structural constraints.
- (iii) The time constraint μ_1 , (resp. μ_{10}), specific to the first (resp. last) processing stage, which corresponds to the allocation of inputs between the DMs (resp. the production of the organizational response). These tasks are indeed different from the four processing stages that characterize the DM, i.e., the

SA, IF, CI, and RS stages, as described in Chapter 3, Section 3.1.

However, it should be clear that all these constraints are actually strongly interrelated; For example, the resources available to DM2 (i.e., n_2) are used to compute ϕ_2 but also to compute the average processing rate of the circuit ρ_{4a} . Likewise, the average processing rate of the circuits ρ_{3a} and ρ_{6a} depends on μ_1 and μ_{10} . In particular, the critical circuits which, as said earlier, determine the maximum throughput rate, may be at the same time in the three categories described above. In other words, there is not necessarily only one type of constraint which is the binding constraint, given the values of the various processing times and resources available.

The differentiation is nevertheless very useful in carrying out the performance evaluation, especially when the point is to compare different protocols of operations for the same organization. Indeed, in that case, it is sufficient to look at the structural constraint only. For instance, assuming that the processing times and resources available are identical for both organizations (structure (a) and (b)), it turns out that the following inequalities are always verified:

$$\alpha(\rho_{4b}) \leq \alpha(\rho_{4a})$$

$$\alpha(\rho_{3b}) \leq \alpha(\rho_{3a})$$

$$\alpha(\rho_{6b}) \leq \alpha(\rho_{6a})$$

Therefore, $\bar{Q}_b \geq \bar{Q}_a$.

In fact, the equality between $\bar{Q}_b$ and $\bar{Q}_a$ is only verified if the binding constraint comes from the processing rate of one of the DMs (i.e., $\bar{Q}_a = \phi_1$ or $\bar{Q}_a = \phi_2$) or from the processing time of the input or

output task, i.e.:

$$\bar{\tau}_a = \frac{1}{\mu_1} \quad \text{or} \quad \bar{\tau}_b = \frac{1}{\mu_{10}}$$

In any case, it turns out that the performance of the hierarchical structure (structure (b)) is always at least as good as that of the result sharing type (structure (a)). We are now going to analyze further how the performance of the organization is related to the different types of constraints previously described, by assuming various numerical values for the resource and time constraints. To that extent, we will consider only one organizational form, structure (b) of the two member organization.

4.3.2 Evaluation of the Maximum Throughput Rate

In this section, we consider the two member organization, with structure (b), as shown in Figure 3.8. For simplicity, all the processing times are assumed deterministic. Indeed, our objective here is to make clear how to carry out the performance evaluation depending on the type of binding constraints. To that extent, and according to the development presented in the previous section, we will consider three cases: The first one is an example where the throughput rate is bounded by the maximum processing rate of one DM. In the second one, the binding processing rate constraint comes from a structural constraint. In the last one, the processing time of the partitioning operation (which corresponds to μ_1) is the binding constraint.

Remark: For clarity, the circuits previously denoted by ρ_{ab} , ρ_{sb} and ρ_{sb} will be simply denoted by ρ_a , ρ_s , and ρ_b .

4.3.2.1 Example 1

Let us assume that it takes one unit of time to complete any one of

the tasks, except for the SA stage of DM1, which takes four units of time, i.e., $\mu_1 = 1$ for $i = 1, 3, 4, \dots, 10$ and $\mu_2 = 4$. In addition, DM1 and DM2 can only process one input at a time, i.e.,

$$n_1 = n_2 = 1$$

and the overall organization can handle at most eight inputs at a time, i.e.,

$$n_0 = 8$$

In that case, the average processing times of the different circuits are:

$$\alpha(\rho_1) = 7$$

$$\alpha(\rho_2) = 4$$

$$\alpha(\rho_3) = \frac{6}{8} = \frac{3}{4}$$

$$\alpha(\rho_4) = 6$$

$$\alpha(\rho_5) = \frac{11}{8}$$

$$\alpha(\rho_6) = \frac{8}{8} = 1$$

$$\text{and } \mu_{\max} = \max_{i=1, \dots, 10} \mu_i = \mu_2 = 4$$

The maximum throughput rate is therefore determined by:

$$\Phi = \min \left(\frac{1}{7}, \frac{1}{4}, \frac{4}{3}, \frac{1}{6}, \frac{8}{11}, 1 \right) = \frac{1}{7}$$

The maximum processing rate of each DM is:

$$\phi_1 = \min \left(\frac{1}{a(\rho_1)} , \frac{1}{\mu_2} , \frac{1}{\mu_3} , \frac{1}{\mu_4} , \frac{1}{\mu_5} \right) = \frac{1}{7}$$

$$\phi_2 = \min \left(\frac{1}{a(\rho_2)} , \frac{1}{\mu_6} , \frac{1}{\mu_2} , \frac{1}{\mu_8} , \frac{1}{\mu_9} \right) = \frac{1}{4}$$

The maximum processing rate of DM1 is clearly the binding constraint to the throughput rate of the overall organization. Indeed, ρ_1 is the unique critical circuit of the net.

Now, there are only two possible alternatives so as to improve the performance of the organization: either increase the resources available to DM1 or reduce the processing time of one its processing stages (or both), which are indeed the only ways to reduce $a(\rho_1)$. For instance, it would be possible to improve the performance, if DM1 was better trained, so that he could handle more information at the same time (which means that he has more cognitive resources) or execute the tasks faster (or both). In any case, the only way to improve the organizational performance is by improving the individual performance of DM1.

4.3.2.2 Example 2

Let us keep the same values as before, except that μ_2 equals now one unit of time like the other task processing times. In this example:

$$a(\rho_1) = a(\rho_2) = 4, a(\rho_3) = \frac{3}{4}, a(\rho_4) = 6, a(\rho_5) = a(\rho_6) = 1$$

$$\mu_{\max} = 1$$

Hence:

AD-A173 681

PERFORMANCE EVALUATION OF DECISIONMAKING ORGANIZATIONS

272

USING TIMED PETRI (U) MASSACHUSETTS INST OF TECH

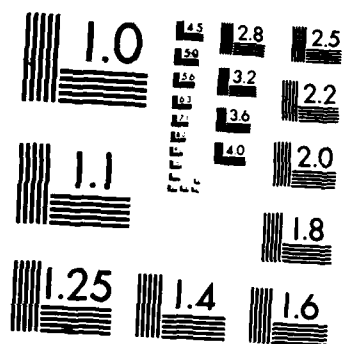
CANBRIDGE LAB FOR INFORMATION AND D H P HILLION

UNCLASSIFIED

AUG 86 LIDS-TH-1590 N00014-85-K-0782

F/G 9/2

Ni



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

$$\bar{\phi} = \min \left(\frac{1}{4}, \frac{4}{3}, \frac{1}{6}, 1 \right) = \frac{1}{6}$$

and the maximum processing rate of each DM is the same:

$$\phi_1 = \phi_2 = \frac{1}{4}$$

The maximum throughput rate is determined by a structural constraint, since ρ_4 is now the unique critical circuit. Given the structure of ρ_4 , the only way to improve the throughput rate is, in that case, by either increasing the resources available to DM2 (i.e., n_2) or reducing the processing times of the corresponding tasks of the circuits (or both). Interestingly enough, it will have no effect to either increase the resources of DM1 (i.e., n_1) or the processing time of its SA stage (i.e., μ_1), in contrast to the previous case.

4.3.2.3 Example 3

At last, let us assume that:

$$\mu_i = 1 \quad \text{for } i = 2, \dots, 10$$

$$\mu_1 = 2 \quad (\text{processing time of the input transition, corresponding to the partitioning operation})$$

$$n_0 = 8, n_1 = n_2 = 4 \quad (\text{Both DMs can now handle four inputs simultaneously})$$

then, in that case:

$$\alpha(\rho_1) = \alpha(\rho_2) = 1, \alpha(\rho_3) = \frac{7}{8}, \alpha(\rho_4) = \frac{3}{2}, (\rho_5) = \alpha(\rho_6) = \frac{9}{8}$$

$$\mu_{\max} = \mu_1 = 2$$

Hence:

$$\bar{Q} = \min \left(1, \frac{8}{7}, \frac{2}{3}, \frac{8}{9}, \frac{1}{2} \right) = \frac{1}{2}$$

In that case, the processing time of t_1 , corresponding to the partitioning operation, determines the maximum throughput rate. The way to improve the performance of the organization is very simple in that case, since the only alternative possible is to reduce the amount of time it takes to allocate the external inputs to the DMs interacting directly with the environment.

4.4 CONCLUSION

We have showed in this chapter that the maximum throughput rate is bounded from above by the minimum average circuit processing rate. In addition, this bound characterizes the maximum throughput rate for deterministic systems. In the next chapter, we will analyze the execution schedule, i.e., the instant of times at which the various operations take place in the process.

CHAPTER V

ANALYSIS OF EXECUTION SCHEDULES

In the previous chapter, we have carried out the computation of the maximum throughput rate as a function of both the time and resource constraints, specific to the DMO. This throughput rate bounds the rate with which inputs can arrive without overloading the organization. In this chapter, we want to determine the execution schedule (also called firing schedule, by reference to the transition firings), i.e., the instant of times at which the various operations occur in the process, for allowable rates of incoming inputs. It is emphasized that the firing schedule computed here will characterize the dynamic behavior of the DMO: starting from an initial state, the process will be assumed to occur repetitively, which supposes that there are always inputs available at the entry of the system. By doing so, we will obtain the best performance, with respect to real-time processing, that can be achieved by the organization.

Naturally, the precise firing schedule can only be obtained for the deterministic process, in which all the processing times are deterministic. For the non-deterministic case, where processing times may be discrete random variables (as defined for switches), it will be possible to obtain the earliest (resp. latest) firing schedule by taking the shortest (resp. longest) firing time of each transition.

5.1 ANALYSIS OF SEQUENTIAL AND CONCURRENT OPERATIONS

Before modeling the dynamic behavior of the DMO, taking into account the time and resource constraints, it is first necessary to analyze further how the different operations take place in the process. Indeed, the precise sequence of operations in the process is completely determined by the structure of the net. In other words, the structure of the Petri Net characterizes the causality of the system, i.e., the fact that certain

transitions must fire before others, corresponding to the sequential tasks, while others can fire independently, corresponding to the concurrent tasks. Apart from the task processing times and the resources available, this partial order in the different operations, as determined by the structure of the net, is clearly a determinant factor of the execution schedule. This is the reason why our objective here is to find a convenient representation that could account for the causal dependence and independence between the various processing tasks.

5.1.1 Representation of the Process as an Occurrence Net

In order to model the precise sequence of activities in the process, we first consider the net that is obtained by deleting all the resource places, including the resource place of the overall organization (R_o) and of each DM. The corresponding net for the example of the two member organization described in Chapter 3, Section 3.5 (structure (a)) is shown in Figure 5.1.

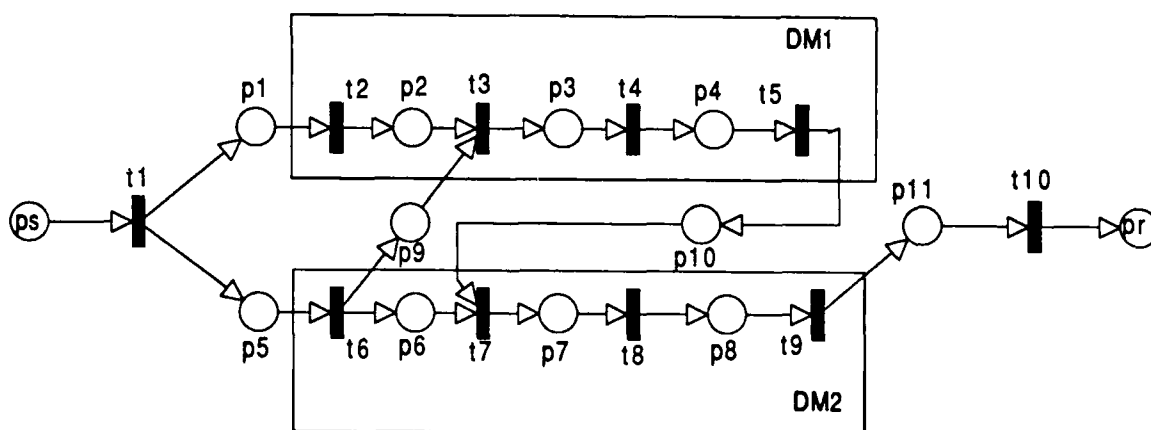


Figure 5.1 Model of the Two Member Organization as an Occurrence Net

Naturally, the net so constructed is exactly the Petri Net model of the DMO, as introduced in [2], which represents the information structure within the organization. Let us recall that, as long as the model corresponds to an admissible organizational structure, we are sure that

this net will contain no loops (or directed circuits), as discussed in Chapter 3, Section 3.5. At this point, a Petri Net which satisfies the two properties:

- (1) it is acyclic (i.e., contains no loops)
- (2) each place has at most one input transition and at most one output transition

is called an Occurrence Net [20]. The Petri Net obtained is therefore a model of an occurrence net. The second property is trivially satisfied, since our model of the DMO is an Event-Graph (each place has exactly one input and one output transition). If we identify the reachable markings as the states of the process and the transitions as the steps, then property 1 implies that the process is completed in a finite sequence of steps and property 2 implies that the passage from one state to another one is uniquely defined (and can be predicted).

5.1.2 Characterization of Lines and Slices

We call marked places the places of the occurrence net that contain one token. We assume that initially, there are no marked places in the net. Let us denote by $\bar{T}_1 = \{t_1\}$, where t_1 is the input transition of the occurrence net. Assume now that t_1 fires and let $\bar{P}_1$ be the resulting set of marked places, which, in that case, are also the output places of t_1 . In the example of Figure 5.1, $\bar{P}_1 = \{p_1, p_2\}$. Let $\bar{T}_2$ be the set of transitions that are enabled by $\bar{P}_1$: $\bar{T}_2 = \{t_2, t_3\}$. Assume now that all the transitions of $\bar{T}_2$ fire and let $\bar{P}_2$ be the new set of marked places: $\bar{P}_2 = \{p_2, p_4, p_5\}$. Similarly $\bar{T}_3$ will be the set of transitions that are enabled by $\bar{P}_2$ and so on. By iterating this procedure, we will eventually construct a sequence $\bar{T}_1, \bar{T}_2, \dots, \bar{T}_S$ such that: $\bar{T}_S = \{t_m\}$, where t_m denotes the output transition.

In the example of Figure 5.1:

$$\begin{array}{ll}
\bar{T}_1 = \{t_1\} & \bar{P}_1 = \{p_1, p_6\} \\
\bar{T}_4 = \{t_4\} & \bar{P}_4 = \{p_4, p_6\} \\
\bar{T}_5 = \{t_5\} & \bar{P}_5 = \{p_6, p_{10}\} \\
\bar{T}_6 = \{t_6\} & \bar{P}_6 = \{p_7\} \\
\bar{T}_7 = \{t_7\} & \bar{P}_7 = \{p_8\} \\
\bar{T}_8 = \{t_8\} & \bar{P}_8 = \{p_{11}\} \\
\bar{T}_9 = \{t_{10}\} &
\end{array}$$

Let us analyze the properties of the sequence $(\bar{T}_i)_{i=1, \dots, s}$ so constructed. We first give the definition of a slice and a line of an occurrence net, as described in [21]. Let $<$ denote the partial order relation, defined as follows. Let a and b be any two nodes (place or transition) of the net, then:

$a < b$ iff there exists a directed path that goes from a to b .

Naturally the definition is consistent, since an occurrence net is acyclical (and therefore there could not exist a directed path from a to b and from b to a). The definitions of a line and a slice are:

Line: A line is a set B of nodes (places or transitions) such that:

- (1) for any two nodes $a \in B, b \in B$:

$$a < b \quad \text{or} \quad b < a$$

- (2) B is maximal: there does not exist a node c in the net which satisfies (1) but does not belong in B .

Slice: A slice is a set D of nodes (places or transitions) such that:

- (3) for any two nodes $a \in D, b \in D$:

$$a \nless b \quad \text{and} \quad b \nless a$$

- (4) D is maximal: there does not exist a node c in the net which satisfies (3) but does not belong in D .

Clearly, the elements of a slice are unordered (independent with respect to $\langle \rangle$): the elements of a slice characterize the concurrent activities of the process. In contrast, the elements of a line are strictly ordered (causally dependent): these elements characterize the activities that take place sequentially in the process.

In our analysis, we are only interested in the slices that are set of transitions, since we want to determine the tasks that are concurrent in the process. By construction, each $\bar{T}_i$, as previously defined, is a slice. Assume, for instance, that it would take exactly one unit of time for each transition to fire. In that case, the transitions of $\bar{T}_i$ are all the transitions that fire concurrently at the i -th instant. In fact, the sequence $\bar{T}_1, \bar{T}_2, \dots, \bar{T}_S$ represents the steps of the process with the transitions that fire concurrently at each step. The order of these slices determines precisely the partial order between the various processing tasks, in the following sense: If we consider a line (a sequential subprocess), then each transition of the line belongs to a different slice (a line cuts necessarily any slice at most once, as it can be deduced straightforwardly from the definitions) and the order of the corresponding slices corresponds to the order in which the transitions fire in the sequential subprocess. This property of $\bar{T}_1, \dots, \bar{T}_S$ is fundamental and will be used in the next section for determining the precise firing schedule.

Remark: In our model, where there is an input transition (t_i) and an output transition (t_m), the lines are exactly the directed paths that go from t_i to t_m . The determination of the lines is straightforward from the development in Chapter 4. By adding the resource place (R_0) of the overall organization to a line, we obtain a directed circuit. Therefore, once we have determined all the directed circuits of a net (so as to compute the maximum throughput rate), we can obtain all the lines by selecting the circuits that contain R_0 as unique resource place and deleting it.

5.2 EXECUTION SCHEDULE OF THE DMO

In this section, we are going to carry out the computation of the firing schedule and analyze its properties. As stated previously, it is only possible to predict the precise firing schedule for the deterministic process. Therefore, in this section, all the transition firing times are therefore assumed deterministic. We will see later how to use the results when processing times are discrete random variables (corresponding to the switches) so as to obtain the earliest execution schedule and the latest one. We also assume here that external inputs are always available at the entry of the system: In other words, there is a queue of inputs, such that the process occurs repetitively. Given this assumption, we will be able to characterize the execution schedule as a function of both the time and resource constraints specific to the organization, independently of the arrival rate of inputs. As for the maximum throughput rate, the execution schedule obtained will characterize the dynamic performance of the DMO. Let us point out that, here again, the analysis is based on the fact that the Petri Net Model corresponds to a strongly connected Event-Graph.

5.2.1 Assumptions of the Model

Let us recall that S_1^n denotes the instant of the n-th initiation firing of transition t_1 and μ_1 the firing time of t_1 . From now on, we will use the sequence (S_1^n) for $n=1,2,3,\dots,N$ and for all the transitions ($i=1,\dots,m$) to characterize the execution schedule of the system. For clarity, we will denote the places differently: since each place has exactly one input and one output transition, it is possible to specify precisely a place by its unique input and unique output transition. Accordingly, we will denote by p_{ij} the place whose input transition is t_i and output transition t_j . Likewise, M_{ij}^0 (res. $M_{ij}(\tau)$) will denote the marking of place p_{ij} initially (resp. at instant τ).

At this point, it is emphasized that our objective here is to analyze the dynamic behavior of the DMO. Otherwise stated, starting from an

initial state, we want to compute the sequence of initiation firings (S_1^n), assuming that the process occurs repetitively. Quite naturally, we will assume that the processing starts at $\tau = 0$, so that initially $M_{1j}^0 = 0$ for all the places p_{1j} , except for the resource places, meaning that there are, at the initial instant, no inputs being processed. Naturally, the initial marking of the resource places will denote the resources available for processing (and this initial marking is consequently strictly positive).

5.2.2 Feasible Firing Schedules

The underlying assumption concerning the firing schedule that we are interested in computing, is that the firing initiations occur as soon as the corresponding transitions are enabled. Otherwise stated, the tasks are executed as soon as the corresponding information is available. This will give us indeed the best time performance of the DMO, since the instant of times at which the various tasks are executed are clearly the earliest ones. However, we could imagine that some tasks are arbitrarily delayed and, in particular, that the firing schedule is set up externally for the system. At this point, in treating (S_1^n) as parameters for the system, the problem necessarily arises as to whether or not this firing schedule is realizable (or feasible), given the transition firing times (time constraints) and the initial marking (resource constraints). A firing schedule would be feasible, if every transition was actually enabled at the instant the firing should be initiated. If it were not the case, the firing schedule would be infeasible, i.e., the firing initiation of at least one transition is required to occur before it is enabled.

Ramchandani [6] has given a precise characterization of a feasible firing schedule, in case of an Event-Graph:

Theorem 5.1: A firing schedule (S_1^n), $i=1, \dots, m$ (where m is the number of transitions) is feasible if and only if, for any place p_{1j} (whose input transition is, by definition, t_i and output transition t_j) and for $n=1, 2, 3, \dots$:

$$S_j^{n+M_{1j}^0} \geq S_1^n + \mu_1 \quad (5.1)$$

The above result is intuitively clear. If we consider the instant of the n -th termination firing of transition t_1 , i.e., the instant

$$\tau_1 = S_1^n + \mu_1$$

then, transition t_1 has exactly "produced" n tokens in place p_{1j} (since t_1 has fired n times). Now

$$\tau_2 = S_j^{n+M_{1j}^0}$$

is the instant at which the firing of transition t_j is initiated for the $(n+M_{1j}^0)$ -th time. At this instant, transition t_j has "consumed" $(n+M_{1j}^0)$ tokens in p_{1j} . Because there were initially only M_{1j}^0 tokens in p_{1j} (i.e., at $\tau=0$), τ_2 should necessarily occur after τ_1 , i.e., after the instant at which transition t_1 has fired n times. The inequality

$$\tau_2 \geq \tau_1$$

corresponds exactly to relation (5.1). It is noted that a rigorous demonstration of Theorem 5.1 is provided in Appendix B.

5.2.3 Determination of the Execution Schedule

The result stated in Theorem 5.1 provides an easy way to compute the earliest firing schedule, i.e., the one in which the firings are initiated as soon as the transitions are enabled. Let t_j be any transition and let us denote by

$$p_{i_1,j}, p_{i_2,j}, \dots, p_{i_r,j}$$

the input places of t_j . Following the notation defined earlier, $t_{i_1}, t_{i_2}, \dots, t_{i_r}$ correspond to the input transitions of each of these places, as shown in Figure 5.2. Applying now the previous result to each of these places yields:

$$S_j^{n+M_{i_k,j}^0} \geq S_{i_k}^n + \mu_{i_k} \quad \text{for } k = 1, 2, \dots, r \quad (5.2)$$

At this point, we recall that our model of the DMO as a Timed Event-Graph includes an additional constraint, when compared to the model of Timed Petri Nets introduced by Ramchandani [6]. As discussed in Chapter 2, Section 2.4.2, a transition is not enabled, if it is already executing, which can be modeled by adding a self-loop to each transition. The corresponding place should be denoted by p_{jj} (which means that its input and output transition is t_j) with an initial marking $M_{jj}^0 = 1$. Applying the inequality (5.1) to this place yields:

$$S_j^{n+1} \geq S_j^n + \mu_j \quad (5.3)$$

This is therefore the additional constraint that should be included in our modeling, for specifying a feasible firing schedule.

Let us now assume that n , the number of repetitions, is large enough, so that:

$$n > M_{i_k,j}^0 \quad \text{for } k = 1, \dots, r. \quad (5.4)$$

The previous inequalities (5.2) and (5.3) can be written:

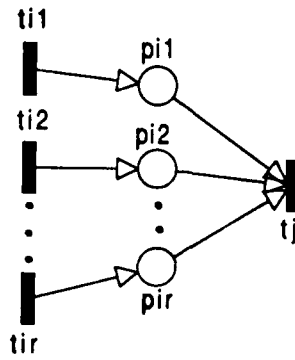


Figure 5.2 Input Places and Transitions of t_j

$$S_j^n \geq S_{i_k}^{n-M_{i_k}^0} + \mu_{i_k} \quad \text{for } k = 1, 2, \dots, r. \quad (5.5)$$

and

$$S_j^n \geq S_j^{n-1} + \mu_j \quad (5.6)$$

The earliest n -th firing initiation of transition t_j is then determined by:

$$S_j^n = \max_{k=1, \dots, r} (S_{i_k}^{n-M_{i_k}^0} + \mu_{i_k}, S_j^{n-1} + \mu_j) \quad (5.7)$$

By construction, such a firing schedule is feasible and it is necessarily the earliest one. The instant, as defined in (5.7), corresponds to the instant at which the transition is enabled for the n -th time. As expected, the places p_{i_j} for which the maximum is obtained have received a token at the latest instant of time, therefore enabling transition t_j .

Now if,

$$n \leq M_{i_k j}^0 \quad \text{for some } k \quad (k \in \{1, 2, \dots, r\})$$

the equality (5.7) still holds, except that, in the determination of the maximum, we do not take into account the corresponding values of k . Indeed, it means that the corresponding places $p_{i_k j}$ have still a marking strictly positive (from the initial marking) and therefore these places do not disable transition t_j .

Relation (5.7) turns out to be an implicit equation, which permits the computation of the firing schedule by iteration on n , the number of firing repetitions. If the initial marking of all the places were strictly positive (i.e., $M_{ij}^0 > 0$ for all the places p_{ij}), S_j^n (for $j=1, 2, \dots, m$) could be easily computed, using (5.7), from the determination of (S_i^k) , for $k=1, 2, \dots, n-1$ and $i=1, 2, \dots, m$. This is however not the case in the model, given the assumptions made in Section 5.2.1; As already stated, only the resource places have a non-null initial marking. For a pair of transitions (t_i, t_j) connected by the place p_{ij} such that $M_{ij}^0 = 0$, we obviously need to determine first S_i^n to compute S_j^n , using (5.7). This condition implies that the computation of S_j^n for the various transitions ($j = 1, 2, \dots, m$) should be carried out following a precise order. As it will be discussed now, this order is characterized by the slices of the net, as introduced in Section 5.1.2.

Given our assumptions that the processing starts at $\tau=0$, the places p_{ij} for which $M_{ij}^0 = 0$, as stated in Section 5.2.1, are all the places of the net, except the resource places. In other words, these places are the places of the occurrence net that describes the precise sequence of activities in the process, as constructed in Section 5.1.1. Let us consider then the sequence of slices $\bar{T}_1, \dots, \bar{T}_s$, which has allowed us to characterize the sequential and concurrent activities of the process.

Any pair of transitions (t_i, t_j) , (connected by place p_{ij}) belongs to

a line (since trivially $t_i < t_j$) and both transitions are therefore elements of two different slices, let us say $\bar{T}_a$ and $\bar{T}_b$. As we have explained, the order of the slices reflects the partial order between the transitions, so that we have necessarily: $a < b$. It turns out that, if we have determined the instants of the n -th firing initiation for all the transitions belonging to $\bar{T}_1, \dots, \bar{T}_{k-1}$, then we will be able to derive, using relation (5.7), the corresponding instant for the transitions belonging to $\bar{T}_k$. Now, S_1^n , corresponding to $\bar{T}_1 = \{t_1\}$ (input transition) is never a problem, since t_1 has a unique input place, which is the resource place, R_0 , of the overall organization. Hence, M_{m1}^0 , where p_{m1} denotes the resource place R_0 , given the notation defined previously (t_m being the output transition), is always strictly positive and a direct iteration on n generates S_1^n , i.e., using (5.7):

$$S_1^n = \max (S_m^{n-M_{m1}^0} + \mu_m, S_1^{n-1} + \mu_1)$$

The order in which the computation of S_j^n , for $j = 1, \dots, m$, should be carried out, is determined by the partial order defined by the slices $\bar{T}_1, \bar{T}_2, \dots, \bar{T}_s$. Let us point out that, for the transitions belonging to the same slice, the order has strictly no importance since the transitions fire concurrently.

In brief, relation (5.7) allows the determination of the execution schedule by iteration on n , which corresponds to the number of firing repetitions from the initial instant. At each iteration, however, the instants at which the different tasks are executed should be determined, following the order with which the operations occur in the process, i.e., the partial order defined by the slices. A detailed description of the corresponding algorithm is provided in Appendix C.

Finally, we have been able to determine the best execution schedule (i.e., the earliest one) of the DMO, as a function of both the time constraints and resource constraints. Indeed, in relation (5.7), μ_i are

the task processing times and, as we have said, the initial marking $M_{1,j}^0$ of the net models the resource constraints. The execution schedule so obtained characterizes the dynamic behavior of the DMO: starting from the initial state at $\tau = 0$ (where no inputs are processed and all the resources are available), the process is assumed to occur repetitively (external inputs are assumed queueing). Naturally, the results developed here are only valid for the deterministic process, as we have said in the introduction. Now, in the case where processing times are discrete random variable, it is not possible to predict precisely the instants of time at which the various operations take place. However, by considering on the one hand the shortest processing time for each random variable and on the other hand the longest processing time, we obtain two deterministic models. Computing the firing schedule yields in the first case the earliest execution schedule possible and in the second case the latest one. Naturally, the real-time firing schedule is in between these two bounds. Accordingly, we can say that the earliest execution schedule characterizes the best possible performance of the DMO and the latest one, the worst possible performance. In addition, we can easily compute the probability of each occurring. From the probability distribution assigned to the possible processing times, we know the probability of the shortest (resp. longest) processing time for each transition. By computing the product of these probabilities for all the transitions, we obtain precisely the probability that the best (resp. worst) performance occurs.

5.2.4 Properties of the Execution Schedule

In this section, we carry out the analysis of the firing schedule and investigate the properties of the steady state process. The fundamental result presented here, which characterizes the periodicity of the steady-state process, has been obtained by Chretienne [16]. Let us first give the definition of the K-periodicity that he introduced in [16].

Definition: A sequence S_n ($n = 1, 2, 3, \dots$) is said to be K-periodic, with period π , iff for all $n \geq 1$:

$$\begin{cases} n = K q + r \\ 0 \leq r < K \end{cases} \implies S_n = S_r + q \pi \quad (5.8)$$

If we identify the sequence S_n as the instant of times at which the same operation occurs successively, it means that the K successive occurrences repeat periodically, as it can be visualized on Figure 5.3.

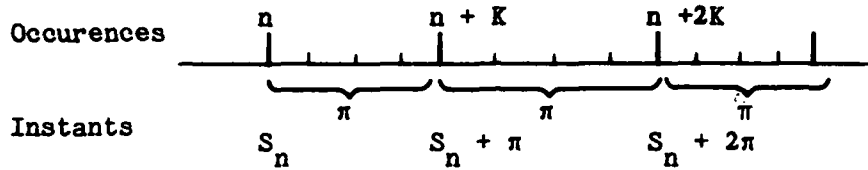


Figure 5.3 K-Periodic Sequence with period π

When $K = 1$, it corresponds to the usual periodicity (also called strong periodicity by Ramchandani [6]). The ratio K/π represents the (average) rate (or frequency) with which the operation occurs. We will denote this rate by:

$$\bar{\Phi} = \frac{K}{\pi}$$

so that (5.8) can be rewritten:

$$S_n = S_r + q \frac{K}{\bar{\Phi}} \quad (5.9)$$

The fundamental property that characterizes the dynamic behavior of the system is the following one:

Theorem 5.2 [16]: The firing schedule (as determined previously) becomes K-periodic after a finite number of firing repetitions, which means that

there exist K and N_0 such that, for $n \geq N_0$ and for $j = 1, 2, \dots, m$ (where m is the number of transitions):

$$\begin{cases} n - N_0 = Kq + r \\ 0 \leq r < K \end{cases} \Rightarrow S_j^n = S_j^{N_0+r} + q \frac{K}{\bar{q}} \quad (5.10)$$

The rate $\bar{q}$ is equal to the minimum average circuit processing rate as defined in Chapter 4, Section 4.2.2.1.

Let us point out first that this Theorem holds only for the type of Petri Nets used for modeling the DMO, i.e., strongly connected Event-Graphs, and for deterministic processing times. Let us now comment on the result.

This result completes the development carried out in the previous chapter, concerning the maximum throughput rate. At it was discussed there, the minimum average circuit processing rate bounds the allowable rate at which inputs can arrive, in the sense that above this rate, the queue of inputs at the entry of the system will necessarily grow to infinity over time. This theorem makes sure that, in the deterministic case, this rate corresponds effectively to the maximum throughput rate that can be achieved by the DMO, as stated in Chapter 4, Section 4.2.4.1.

In addition, we have a further result: We know that this throughput rate is actually achieved within a finite amount of time or, in other words, after a finite number of repetitions of the process. This result is not surprising; A Petri Net which is live and bounded (which is the case of our model) has necessarily a finite number of reachable markings, i.e., the forward marking class, as defined in Chapter 2, Section 2.1.5 is finite. Therefore, there is at least one state, s_0 , (i.e., one marking) which occurs cyclically in the repetitive process. Furthermore, in case of an Event-Graph, the set of all reachable markings from any initial marking is uniquely defined and can be predicted, as stated in Section 5.1.1.

Consequently, all the states that take place between the successive occurrences of s_0 also repeat cyclically. It means that, once the state s_0 is reached for the first time, the reachable states repeat periodically, determining the steady-state process. However, as Theorem 5.2 proves it, the period in real time (i.e., given the transition firing times) turns out to be of a more complex type than the simple (or strong) periodicity.

From the determination of the firing schedule (S_1^n) and of the rate $\bar{\mu}$, computed as the minimum average circuit processing rate, it is now straightforward to deduce K , the periodicity factor. However, an interesting result further obtained in [16], states that if all the critical circuits, as defined in Chapter 4, Section 4.2.2.2, contain only one token, then $K = 1$. In that case, it means that the steady-state process is strongly periodic, i.e., each transition fires at the same regular interval of time. Otherwise stated, the processing rate is constant (for the steady-state). Interestingly enough, for the model of the DMO analyzed here, it would be this case if:

- either the critical circuits contain only one resource place and this resource place has initially one token (meaning that the corresponding DM or the overall organization can only process one input at time)
- or the critical circuits correspond to individual transitions (recall that a transition contain in fact a self loop with one token).

5.3 MEASURES OF PERFORMANCE FROM THE EXECUTION SCHEDULE

In the first two sections, we have been able to characterize the dynamic behavior of the DMO, having first determined the sequential and concurrent operations of the process and then computed the precise execution schedule for any number of repetitions of the process. In addition, we know that the steady state process is K -periodic, with a processing rate $\bar{\mu}$ which corresponds to the minimum average circuit processing rate. Recall that this analysis is applicable under the

following assumptions:

- (i) the process is deterministic, which implies that all the task processing times are deterministic,
- (ii) the process occurs repetitively from the initial instant $\tau = 0$.
- (iii) The tasks are executed as soon as the corresponding information is available, which means that, in the Petri Net model, the firing initiations occur as soon as the transitions are enabled. The execution schedule so obtained is the best possible, in the sense that the instants of time at which the various tasks are processed are the earliest ones.

In the case where the processing times are discrete random variable, as it is the case for switches, we can extend this analysis by considering the deterministic process associated with the shortest processing times and the one associated with the longest processing times of each transition. In the first case, we obtain the earliest execution schedule possible and in the second case the latest execution schedule possible. Naturally, the real-time process is in between these two bounds.

It is now interesting to analyze qualitatively the results obtained and derive related measures of performance. There are actually two measures that are particularly interesting to analyze: The execution schedule (S_1^n) of the input transition and the execution schedule (S_m^n) of the output transition.

5.3.1 Execution Schedule of the Input Transition

Each time that the firing of the input transition t_1 is initiated, it means that the DMO starts processing a new input. Given the assumptions made for determining the firing schedule, S_1^n represents the earliest instant at which the n-th input can be processed; i.e., whatever may be,

the instant of times at which inputs arrive successively, the schedule (S_1^n) ($n = 1, 2, 3, \dots$) determines the earliest instants at which the DMO is able to handle these inputs. Because this schedule is not necessarily strongly periodic but only K -periodic in its steady state, we do not generally obtain the best performance by regulating inputs at the entry of the system, such that the arrival rate be constant. Suppose for example that $K=2$ and that

$$S_1^{N_0+1} = S_1^{N_0} + \frac{1}{2\bar{\theta}}, \quad S_1^{N_0+2} = S_1^{N_0} + \frac{2}{2\bar{\theta}}, \dots \quad (5.11)$$

as shown on Figure 5.4 (Recall that, after N_0 , repetitions the process is in its steady-state).

Occurrences

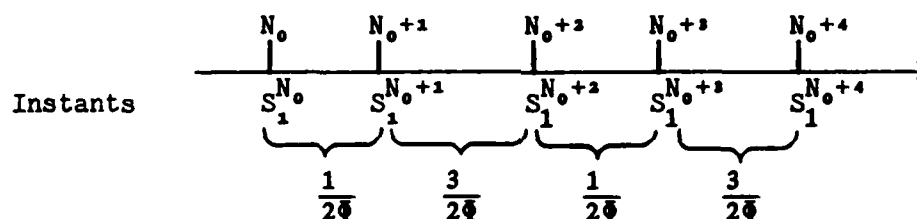


Figure 5.4 2-Periodic Firing Schedule

If the interarrival time of inputs was constant and equal to $1/\bar{\theta}$ ($\bar{\theta}$ being the maximum throughput rate), the processing of every two inputs would be delayed by $1/2\bar{\theta}$, compared to what it could be if, for example, two inputs were arriving simultaneously at regular interval of time $2/\bar{\theta}$. In fact, so as to reduce at the minimum possible the processing delays, K inputs should arrive simultaneously at regular intervals of time duration $K/\bar{\theta}$. Naturally, this supposes that the buffer capacity at the entry of the system, is at least equal to K .

5.3.2 Execution Schedule of the Output Transition

Let us denote by:

$$T_m^n = S_m^n + \mu_m$$

where μ_m is the processing time of the output transition t_m . T_m^n is the instant of the n -th firing termination of t_m , the output transition. Each time that the firing of t_m is terminated, it means that the DMO has completed the processing of an input and has produced the corresponding response. Now, let us assume that n inputs arrive simultaneously at $\tau = 0$. Then T_m^k ($k = 1, 2, \dots, n$) represents the instant of time at which the processing of the k -th input (defined by the order in which inputs are handled) is completed: For that reason, we shall say that the schedule (T_m^n) represents the dynamic response time of the DMO. At this point, we should emphasize the difference existing between the time-delay of the DMO, as computed in [3] and this measure. In [3], the time-delay measures the duration from input entering the system to the response taking place. This time-delay is a static measure, in the sense that the system is supposed to process one input at a time. Quite obviously, this measure would correspond to $T_m^1 = S_m^1 + \mu_1$ in our model, i.e., the amount of time it takes to process the first input, since the processing is supposed to start at $\tau=0$. Likewise, it would also correspond to the difference:

$$d_n = T_m^n - S_1^n = S_m^n - S_1^n + \mu_m \quad (5.12)$$

for $n \geq N_0$ (where N_0 determines the number of repetitions above which the process is in its steady state) and assuming that the resource place of the overall organization, R_0 , contains only one token. As we have discussed in Chapter 3, Section 3.3.2, it means indeed that the DMO can only handle, in that case, one input at a time. This intuitive result can actually be proved rigorously. Let us consider all the lines (or directed path of the

occurrence net, as defined in Section 5.1, for which the sum of the transition firing times is maximal. By adding the resource place R_0 , we would trivially obtain all the critical circuits, precisely because R_0 contains only one token. In that case, the maximum average circuit processing time, i.e., $1/\bar{\theta}$, is equal to the maximum time-delay over all the possible directed paths, denoted by $D_{\max}$. Now, since the token content of the critical circuits is one, the periodicity factor K equals one (Section 5.2.4), which means that the steady state is strongly periodic. In particular, for $n \geq N_0$:

$$S_1^{n+1} = S_1^n + \frac{1}{\bar{\theta}} = S_1^n + D_{\max} \quad (5.13)$$

Applying relation (5.7) to the place p_{m1} , which, following the notation used previously, denotes the resource place R_0 of the overall organization, yields:

$$S_1^{n+1} = \max (S_m^n + \mu_m, S_1^n + \mu_1) \quad (5.14)$$

since $M_{m1}^0 = 1$, by assumption.

Now, the expression, for any $n \geq 1$:

$$S_m^n + \mu_m - (S_1^n + \mu_1)$$

represents the amount of time it takes to complete the processing of the n -th input, once the input operation (modeled by t_1) is completed. Hence, this quantity is always positive, so that in fact, (5.14) can be written:

$$S_1^{n+1} = S_m^n + \mu_m = T_m^n$$

Using relation (5.13), we deduce:

$$S_1^n + D_{Max} = T_m^n$$

and therefore, given the definition (5.12):

$$d_n = T_m^n - S_1^n = D_{Max} \quad \text{Q.E.D.}$$

In that case, the amount of time it takes for the DMO to complete the processing of n inputs is clearly equal to nD_{Max} . However, it should be clear that, in the general case, where the DMO can handle more than one input at a time, the amount of time it takes to complete the processing of n inputs may actually be much less. With the assumption that the processing starts at $\tau=0$, this amount of time is precisely given by T_m^n , the dynamic response time.

Now, we can define:

$$RT = \lim_{n \rightarrow +\infty} \frac{T_m^n}{n} = \lim_{n \rightarrow +\infty} \frac{S_m^n + \mu_m}{n} \quad (5.15)$$

as the average response time of the DMO. From Chapter 4, Section 4.2.1, we know that this limit will precisely be equal to α , which is also equal to $1/\bar{\mu}$. Hence the average response time of the DMO corresponds to the maximum average circuit processing time.

At this point we should make a clear distinction between what we have defined as the average response time and what would correspond to the average processing time (the equivalent of the time-delay defined in [3]), that is:

$$d = \lim_{n \rightarrow +\infty} d_n = \lim_{n \rightarrow +\infty} (T_m^n - S_1^n) \quad (5.16)$$

which can also be written as:

$$d = \frac{d_n + d_{n+1} + \dots + d_{n+K-1}}{K} \quad \text{for any } n \geq N_0$$

because of the K-periodicity of the steady-state process. d is an average measure of the amount of time it takes to complete the processing of any individual input. In the general case, the value of d is necessarily greater or equal to the maximum time-delay of the directed paths, previously denoted by D_{Max} , i.e., the corresponding value when the DMO handles only one input at a time. It may be indeed greater precisely because more than one input can be processed at the same time, so that additional delays occur, due to the unavailability of the limited resources. In fact, d can be characterized quite easily. In the steady-state process, M_{m1}^0 inputs are processed simultaneously (M_{m1}^0 is the initial marking of the resource place, R_0 , of the overall organization). Since d is the average processing time of each input, every transition fires at the average rate:

$$\frac{M_{m1}^0}{d}$$

Since $\bar{\sigma}$ is precisely the processing rate:

$$\frac{M_{m1}^0}{d} = \bar{\sigma}$$

so that, finally:

$$d = \frac{M_{m1}}{\bar{\Phi}} \quad (5.17)$$

Let us now prove that we have necessarily: $d \geq D_{Max}$. As stated earlier, by adding the resource place R_0 to the directed paths (i.e., lines) for which the time-delay is maximal and therefore equal to D_{Max} , we obtain a circuit. The token content of such a circuit is M_{m1}^0 , the corresponding average circuit processing rate is:

$$\frac{M_{m1}^0}{D_{Max}}$$

By definition, $\bar{\Phi}$ is the minimum average circuit processing rate, therefore

$$\bar{\Phi} \leq \frac{M_{m1}^0}{D_{Max}}$$

Given (5.17), we deduce immediately:

$$d \geq D_{Max} \quad \text{Q.E.D.}$$

In brief, we have defined the dynamic response time as the amount of time it takes to complete the processing of n inputs, which would arrive simultaneously at the initial instant (i.e., $\tau=0$). For large number of inputs (when n goes to infinity), dividing by n yields the average response time of the organization and it was proved that the average response time is precisely determined by the maximum average circuit processing time. However, the average processing time, d , for each individual input, i.e., the average time-delay from one input entering the system to the response taking place is determined by:

$$d = \frac{M^0(R_0)}{\bar{t}}$$

In other words, d is equal to the total number of inputs being processed simultaneously times the average response time, as would be expected.

5.4 CONCLUSION

In this chapter, we have carried out the analysis of the execution schedule for the DMO model. We have first described the sequential and concurrent activities of the process, using the slices of a Petri Net. A method has been developed to compute the firing schedule, for the case where all the transition processing times are deterministic. The property of K -periodicity that characterizes the steady-state process has also been presented. Finally, we have described different MOPs derived from the execution schedule. This analysis will now be illustrated in the next chapter with a specific example of DMO.

CHAPTER VI

APPLICATION

In this chapter, an example of a five DM organization is used to illustrate the results presented in Chapter 4 and 5. The structure of the organization is fixed, but the resource and time constraints vary. In each case considered, the execution schedule and the maximum throughput rate are computed. The measures of performance obtained are then analyzed and compared. The task processing times considered in the examples are all deterministic.

6.1 PRESENTATION OF THE MODEL

The Petri Net representation of the organizational structure considered is shown in Figure 6.1. The five DM organization models a generalized submarine ship control party performing emergency control tasks. This particular organization has been extensively described and analyzed in [22], using the information theoretic framework as a way to evaluate the performance of the organization. More precisely, the performance evaluation carried out in [22] is based on measures of the workload of the organization members and the accuracy of the response [4]. The goal here is to evaluate the time-related performance of the organization.

Let us now comment on the structure of the organization. Interestingly enough, the organization has both hierarchical and parallel aspects, where the DMs play in fact different roles: DM1 acts as a supervisor, DM2 acts as a coordinator and DM3, DM4 and DM5 act as subordinates. Indeed, DM3, DM4 and DM5 transmit their situation assessments to DM2 and receive commands inputs from this DM. However, before issuing these command, DM2 fuses the information transmitted by these three DMs with his own assessment and transmits the resulting

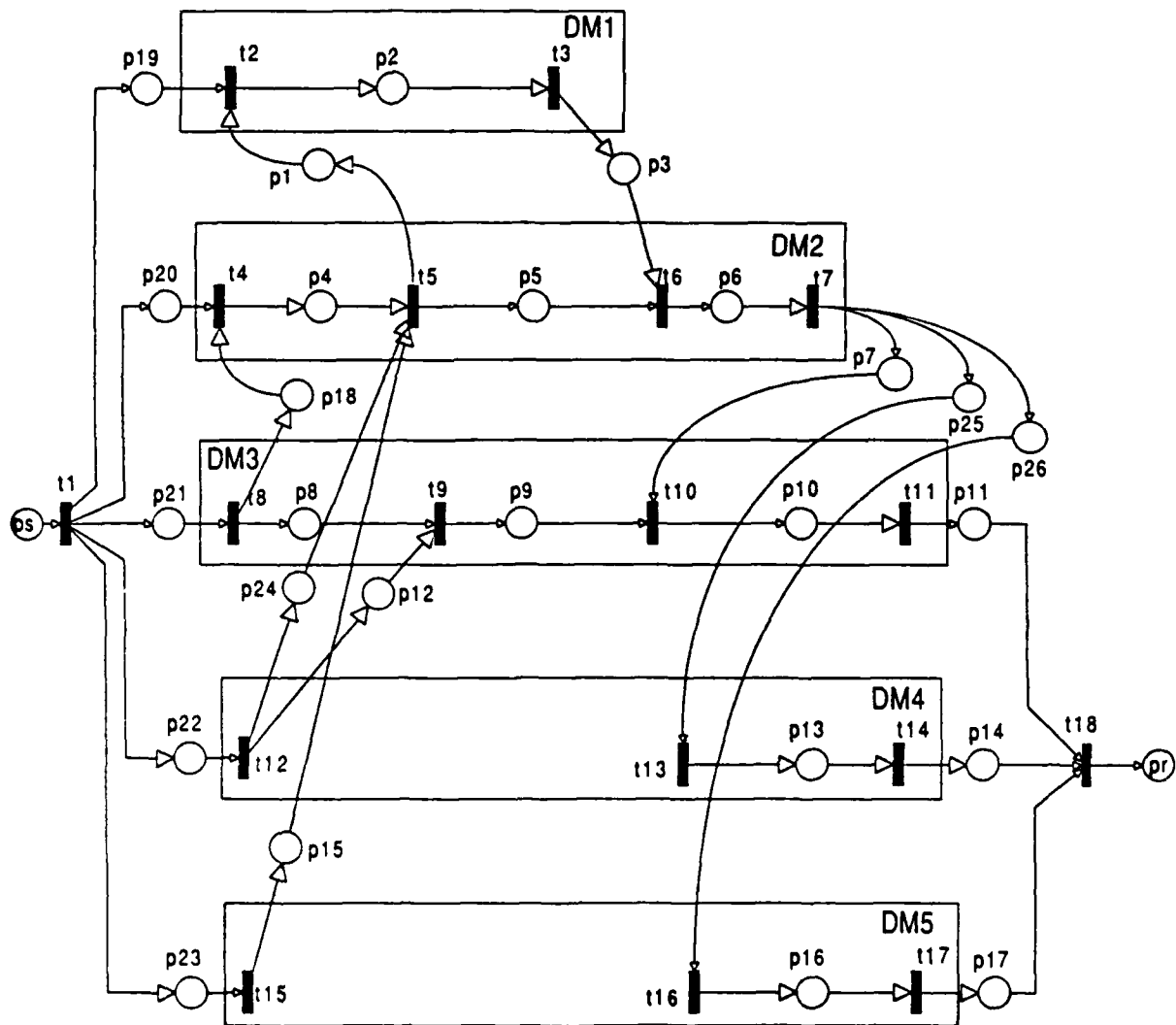


Figure 6.1 Petri Net Model of the Five DM Organization

information to DM1, the supervisor. DM1 combines this information with his own situation assessment and then produces a command to DM2. DM2 proceeds to select a response that he then transmits back to the subordinates (DM3, 4, and 5) as a command. Finally, DM3, 4 and 5 select in turn a response and the three produce the output which constitutes the organization's response. At this point, we should note in particular two characteristics

of the organizational structure. The first one is that DM4 and DM5 directly select a response from the command inputs received by DM2, without processing the information relative to their situation assessment. The second one is that the situation assessment transmitted by DM3 is not combined with the other assessments (from DM4 and DM5) at the IF (Information Fusion) stage of DM2. What actually occurs is that the situation assessment of DM3 is preprocessed by DM2. In fact, the transition t_4 that models the SA stage of DM2 corresponds to an aggregate representation of the subnet shown on Figure 6.2.

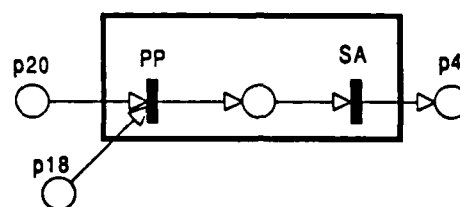


Figure 6.2 Petri Net Representation of the SA Stage
with Preprocessor (DM2)

In the representation of Fig. 6.2, the first transition corresponds to the preprocessor task and the second transition models the usual Situation Assessment stage. Now, the external inputs received by DM2 (in place p_{20}) are first preprocessed together with the situation assessment transmitted (in place p_{18}) by DM3. The resulting output from the preprocessor is then processed at the SA stage of DM2. In that case, the preprocessor acts as a decision aid to DM2. Note that, in the aggregated Petri Net model of the DMO, as shown on Figure 6.1, some transitions include in reality switches, where a selection does occur among different algorithms possible. The corresponding transitions are transition t_4 (resp. t_7) modeling the SA stage (resp. RS stage) of DM2, and transition t_{11} , modeling the RS stage of DM3.

Finally, the Petri Net representation of this DMO, that corresponds to the model developed in Chapter 3 of the thesis, is shown on Figure 6.3.

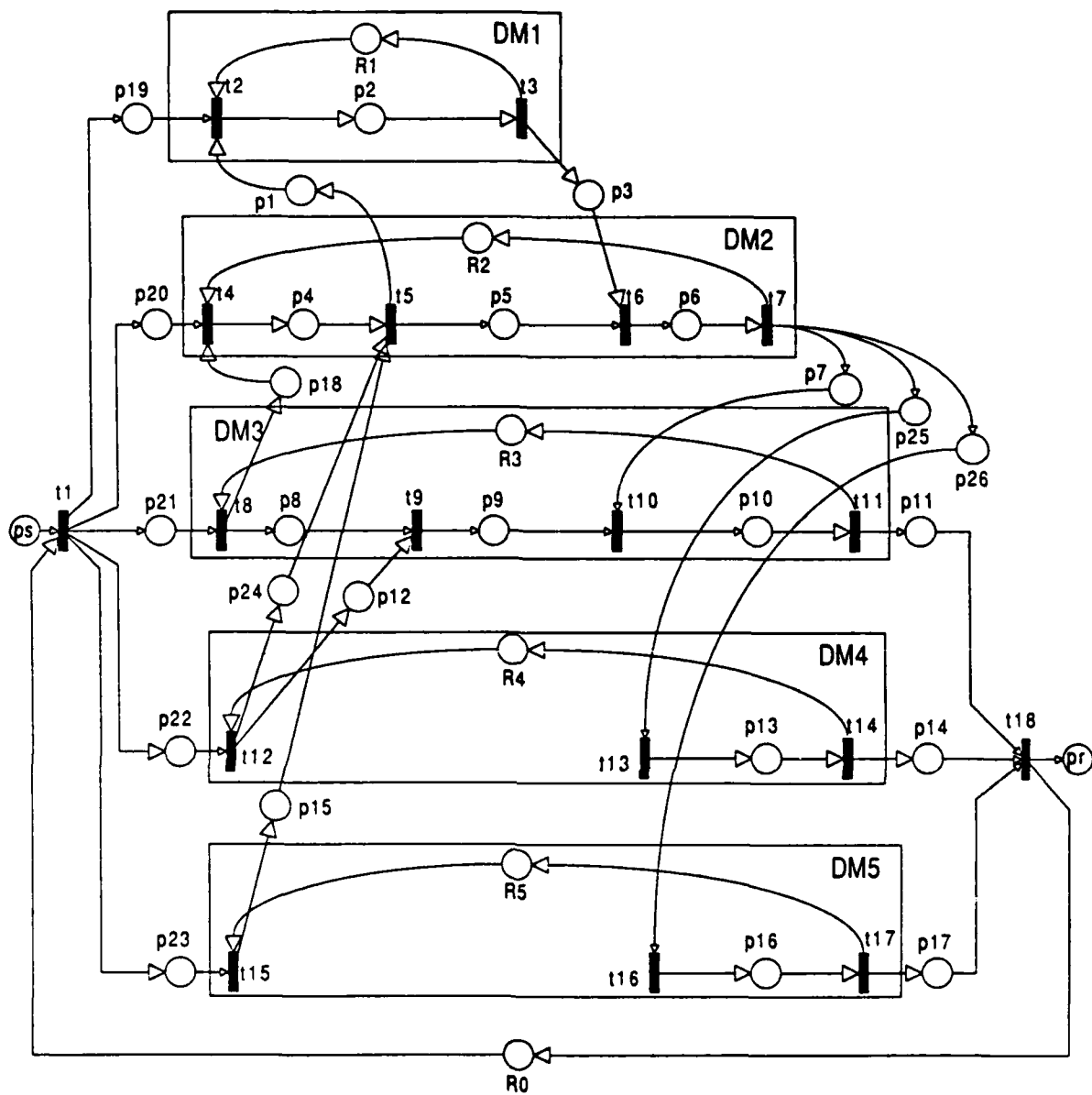


Figure 6.3 Model of the Five DM Organization with Limited Resources

The different resource places have been added, i.e., the resource place of each DM, between the RS stage and the SA stage, and the resource place of the overall organization, between the output transition and the input transition of the net. Based on this model, the performance evaluation of

the organization will now be conducted, following the analysis carried out in Chapter 4 and Chapter 5.

6.2 PERFORMANCE EVALUATION OF THE FIVE DM ORGANIZATION

In order to evaluate the performances of the organization, given the resource and time constraints, it is first necessary to describe the structural characteristics of the model, as determined by the directed circuits and the slices of the Petri Net. As discussed in Chapter 4, Section 4.2.4, the determination of all the directed elementary circuits is required so as to compute the maximum throughput rate. Likewise, the determination of the slices which, as said in Chapter 5, Section 5.1, precisely describe the concurrent and sequential activities of the process, is necessary to compute the execution schedule of the organization. Accordingly, we will first describe these characteristics for the model studied here.

6.2.1 Structural Characteristics of the Model

Both the directed circuits and the slices are determined from the Incidence Matrix. Recall that the Incidence Matrix, C , characterizes the structure of the Petri Net in the following manner: The columns of the matrix correspond to the transitions of the net and the rows to the places. The element C_{ij} of the matrix is such that:

$$C_{ij} = \begin{cases} -1 & \text{if } p_i \text{ is an input place of } t_j \\ +1 & \text{if } p_i \text{ is an output place of } t_j \\ 0 & \text{otherwise} \end{cases}$$

The Incidence Matrix corresponding to the Petri Net model of Figure 6.3 is shown on Table 6.1. From the Incidence Matrix, it is possible to determine all the directed circuits of the net, using the algorithm developed by Alawain and Toudic [19], that is presented in Appendix A of the thesis. The list of the directed circuits obtained is provided in Table 6.2. The

TABLE 6.1. INCIDENCE MATRIX

	t1	t2	t3	t4	t5	t6	t7	t8	t9	t10	t11	t12	t13	t14	t15	t16	t17	t18
p1	0	-1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
p2	0	1	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
p3	0	0	1	0	0	-1	0	0	0	0	0	0	0	0	0	0	0	0
p4	0	0	0	1	-1	0	0	0	0	0	0	0	0	0	0	0	0	0
p5	0	0	0	0	1	-1	0	0	0	0	0	0	0	0	0	0	0	0
p6	0	0	0	0	0	1	-1	0	0	0	0	0	0	0	0	0	0	0
p7	0	0	0	0	0	0	1	0	0	-1	0	0	0	0	0	0	0	0
p8	0	0	0	0	0	0	0	1	-1	0	0	0	0	0	0	0	0	0
p9	0	0	0	0	0	0	0	0	1	-1	0	0	0	0	0	0	0	0
p10	0	0	0	0	0	0	0	0	0	1	-1	0	0	0	0	0	0	0
p11	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	-1
p12	0	0	0	0	0	0	0	0	-1	0	0	1	0	0	0	0	0	0
p13	0	0	0	0	0	0	0	0	0	0	0	0	1	-1	0	0	0	0
p14	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	-1
p15	0	0	0	0	-1	0	0	0	0	0	0	0	0	0	1	0	0	0
p16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	-1	0
p17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	-1
p18	0	0	0	-1	0	0	0	1	0	0	0	0	0	0	0	0	0	0
p19	1	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
p20	1	0	0	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
p21	1	0	0	0	0	0	0	-1	0	0	0	0	0	0	0	0	0	0
p22	1	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	0	0	0
p23	1	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0
p24	0	0	0	0	-1	0	0	0	0	0	0	1	0	0	0	0	0	0
p25	0	0	0	0	0	0	1	0	0	0	0	0	-1	0	0	0	0	0
p26	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	-1	0	0
R0	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
R1	0	-1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R2	0	0	0	-1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
R3	0	0	0	0	0	0	0	-1	0	0	1	0	0	0	0	0	0	0
R4	0	0	0	0	0	0	0	0	0	0	0	-1	0	1	0	0	0	0
R5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	1	0

TABLE 6.2. LIST OF THE DIRECTED CIRCUITS

- 1: p2-t3-R1-t2-p2
- 2: p4-t5-p5-t6-p6-t7-R2-t4-p4
- 3: p1-t2-p2-t3-p3-t6-p6-t7-R2-t4-p4-t5-p1
- 4: p4-t5-p5-t6-p6-t7-p7-t10-p10-t11-R3-t8-p18-t4-p4
- 5: p1-t2-p2-t3-p3-t6-p6-t7-p7-t10-p10-t11-R3-t8-p18-t4-p4-t5-p1
- 6: p8-t9-p9-t10-p10-t11-R3-t8-p8
- 7: p2-t3-p3-t6-p6-t7-p7-t10-p10-t11-p11-t18-R0-t1-p19-t2-p2
- 8: p4-t5-p5-t6-p6-t7-p7-t10-p10-t11-p11-t18-R0-t1-p20-t4-p4
- 9: p1-t2-p2-t3-p3-t6-p6-t7-p7-t10-p10-t11-p11-t18-R0-t1-p20-t4-p4-t5-p1
- 10: p4-t5-p5-t6-p6-t7-p7-t10-p10-t11-p11-t18-R0-t1-p21-t8-p18-t4-p4
- 11: p1-t2-p2-t3-p3-t6-p6-t7-p7-t10-p10-t11-p11-t18-R0-t1-p21-t8-p18-t4-p4-t5-p1
- 12: p8-t9-p9-t10-p10-t11-p11-t18-R0-t1-p21-t8-p8
- 13: p5-t6-p6-t7-p7-t10-p10-t11-p11-t18-R0-t1-p22-t12-p24-t5-p5
- 14: p1-t2-p2-t3-p3-t6-p6-t7-p7-t10-p10-t11-p11-t18-R0-t1-p22-t12-p24-t5-p1
- 15: p9-t10-p10-t11-p11-t18-R0-t1-p22-t12-p12-t9-p9
- 16: p5-t6-p6-t7-p25-t13-p13-t14-R4-t12-p24-t5-p5
- 17: p1-t2-p2-t3-p3-t6-p6-t7-p25-t13-p13-t14-R4-t12-p24-t5-p1
- 18: p4-t5-p5-t6-p6-t7-p25-t13-p13-t14-R4-t12-p12-t9-p9-t10-p10-t11-R3-t8-p18-t4-p4
- 19: p1-t2-p2-t3-p3-t6-p6-t7-p25-t13-p13-t14-R4-t12-p12-t9-p9-t10-p10-t11-R3-t8-p18-t4-p4-t5-p1
- 20: p2-t3-p3-t6-p6-t7-p25-t13-p13-t14-p14-t18-R0-t1-p19-t2-p2
- 21: p4-t5-p5-t6-p6-t7-p25-t13-p13-t14-p14-t18-R0-t1-p20-t4-p4
- 22: p1-t2-p2-t3-p3-t6-p6-t7-p25-t13-p13-t14-p14-t18-R0-t1-p20-t4-p4-t5-p1
- 23: p4-t5-p5-t6-p6-t7-p25-t13-p13-t14-p14-t18-R0-t1-p21-t8-p18-t4-p4
- 24: p1-t2-p2-t3-p3-t6-p6-t7-p25-t13-p13-t14-p14-t18-R0-t1-p21-t8-p18-t4-p4-t5-p1
- 25: p5-t6-p6-t7-p25-t13-p13-t14-p14-t18-R0-t1-p22-t12-p24-t5-p5
- 26: p1-t2-p2-t3-p3-t6-p6-t7-p25-t13-p13-t14-p14-t18-R0-t1-p22-t12-p24-t5-p1
- 27: p4-t5-p5-t6-p6-t7-p25-t13-p13-t14-p14-t18-R0-t1-p22-t12-p12-t9-p9-t10-p10-t11-R3-t8-p18-t4-p4
- 28: p1-t2-p2-t3-p3-t6-p6-t7-p25-t13-p13-t14-p14-t18-R0-t1-p22-t12-p12-t9-p9-t10-p10-t11-R3-t8-p18-t4-p4-t5-p1

TABLE 6.2. (Cont'd)

- 29: p2-t3-p3-t6-p6-t7-p25-t13-p13-t14-R4-t12-p12-t9-p9-t10-p10-t11-p11-t18-R0-t1-p19-t2-p2
- 30: p4-t5-p5-t6-p6-t7-p25-t13-p13-t14-R4-t12-p12-t9-p9-t10-p10-t11-p11-t18-R0-t1-p20-t4-p4
- 31: p1-t2-p2-t3-p3-t6-p6-t7-p25-t13-p13-t14-R4-t12-p12-t9-p9-t10-p10-t11-p11-t18-R0-t1-p20-t4-p4-t5-p1
- 32: p4-t5-p5-t6-p6-t7-p25-t13-p13-t14-R4-t12-p12-t9-p9-t10-p10-t11-p11-t18-R0-t1-p21-t8-p18-t4-p4
- 33: p1-t2-p2-t3-p3-t6-p6-t7-p25-t13-p13-t14-R4-t12-p12-t9-p9-t10-p10-t11-p11-t18-R0-t1-p21-t8-p18-t4-p4-t5-p1
- 34: p5-t6-p6-t7-p7-t10-p10-t11-p11-t18-R0-t1-p23-t15-p15-t5-p5
- 35: p1-t2-p2-t3-p3-t6-p6-t7-p7-t10-p10-t11-p11-t18-R0-t1-p23-t15-p15-t5-p1
- 36: p5-t6-p6-t7-p25-t13-p13-t14-p14-t18-R0-t1-p23-t15-p15-t5-p5
- 37: p1-t2-p2-t3-p3-t6-p6-t7-p25-t13-p13-t14-p14-t18-R0-t1-p23-t15-p15-t5-p1
- 38: p5-t6-p6-t7-p25-t13-p13-t14-R4-t12-p12-t9-p9-t10-p10-t11-p11-t18-R0-t1-p23-t15-p15-t5-p5
- 39: p1-t2-p2-t3-p3-t6-p6-t7-p25-t13-p13-t14-R4-t12-p12-t9-p9-t10-p10-t11-p11-t18-R0-t1-p23-t15-p15-t5-p1
- 40: p5-t6-p6-t7-p26-t16-p16-t17-R5-t15-p15-t5-p5
- 41: p1-t2-p2-t3-p3-t6-p6-t7-p26-t16-p16-t17-R5-t15-p15-t5-p1
- 42: p2-t3-p3-t6-p6-t7-p26-t16-p16-t17-p17-t18-R0-t1-p19-t2-p2
- 43: p4-t5-p5-t6-p6-t7-p26-t16-p16-t17-p17-t18-R0-t1-p20-t4-p4
- 44: p1-t2-p2-t3-p3-t6-p6-t7-p26-t16-p16-t17-p17-t18-R0-t1-p20-t4-p4-t5-p1
- 45: p4-t5-p5-t6-p6-t7-p26-t16-p16-t17-p17-t18-R0-t1-p21-t8-p18-t4-p4
- 46: p1-t2-p2-t3-p3-t6-p6-t7-p26-t16-p16-t17-p17-t18-R0-t1-p21-t8-p18-t4-p4-t5-p1
- 47: p5-t6-p6-t7-p26-t16-p16-t17-p17-t18-R0-t1-p22-t12-p24-t5-p5
- 48: p1-t2-p2-t3-p3-t6-p6-t7-p26-t16-p16-t17-p17-t18-R0-t1-p22-t12-p24-t5-p1
- 49: p4-t5-p5-t6-p6-t7-p26-t16-p16-t17-p17-t18-R0-t1-p22-t12-p12-t9-p9-t10-p10-t11-R3-t8-p18-t4-p4
- 50: p1-t2-p2-t3-p3-t6-p6-t7-p26-t16-p16-t17-p17-t18-R0-t1-p22-t12-p12-t9-p9-t10-p10-t11-R3-t8-p18-t4-p4-t5-p1
- 51: p5-t6-p6-t7-p26-t16-p16-t17-p17-t18-R0-t1-p23-t15-p15-t5-p5
- 52: p1-t2-p2-t3-p3-t6-p6-t7-p26-t16-p16-t17-p17-t18-R0-t1-p23-t15-p15-t5-p1

total number of circuits amounts to fifty-two (52) and we can check that every circuit contains at least one of the resource place (i.e., R_0, R_1, R_2, R_3, R_4 or R_5): we are sure therefore that the organizational form is admissible, i.e., the information structure is acyclical, as discussed in Chapter 4, Section 4.1. Finally, the characterization of the slices is presented in Table 6.3. The procedure described in Chapter 5, Section 5.1, has been used to determine the slices. There are eleven (11) slices that model the different steps of the complete process. The transitions belonging to the i -th slice correspond to the tasks that are performed concurrently at the i -th processing step.

We are now going to present the performance measures obtained, for different values of the resource and time constraints. In each case, we will determine the maximum throughput rate and the execution schedule of the organization. Recall that the task processing times considered are all deterministic.

TABLE 6.3. LIST OF THE SLICES

Slice 1:	t_1
Slice 2:	t_8, t_{12}, t_{15}
Slice 3:	t_9, t_4
Slice 4:	t_5
Slice 5:	t_2
Slice 6:	t_3
Slice 7:	t_6
Slice 8:	t_7
Slice 9:	t_{10}, t_{13}, t_{16}
Slice 10:	t_{11}, t_{14}, t_{17}
Slice 11:	t_{18}

6.2.2 Measures of Performance

The results obtained for each case considered will be presented on a Table that contains the following information:

- The transition firing times, corresponding to the task processing times, and the initial marking of the resource places, corresponding to the resources available for processing.
- The firing schedule obtained for a total number of N repetitions of the process which corresponds to the number of inputs processed (the choice of N is arbitrary and can be as large as desired). Accordingly, the firing schedule will correspond to the sequence (S_1^n) , for $i = 1, 2, \dots, m$ (total number of transitions in the net) and $n=1, 2, \dots, N$.
- The resulting period π of the steady-state process and the periodicity factor K , as defined in Chapter 5, Section 5.2.4.
- The critical circuits obtained, as defined in Chapter 4, Section 4.2.4.1.

From this information, the following measures of performance will be computed:

- (a) The maximum throughput rate, defined as the ratio (Chapter 5, Section 5.2.4):

$$\bar{\Phi} = \frac{K}{\pi} \quad (6.1)$$

It will be checked that $\bar{\Phi}$ is equal to the average circuit processing rate of the critical circuits, as determined by $1/a$ (Chapter 4, Section 4.2.3).

- (b) The dynamic response time corresponding to the complete processing of N inputs as defined in Chapter 5, Section 5.3.2:

$$T_m^N = S_m^N + \mu_m \quad (6.2)$$

where t_m is the output transition of the net. The ratio

$$RT_N = \frac{T_m^N}{N} \quad (6.3)$$

will give an approximate value of the average response time RT (defined as the limit of RT_N when N goes to infinity). Recall that:

$$RT = \frac{1}{\Phi} \quad (6.4)$$

(c) The average processing time (of any individual input), defined as:

$$d = \frac{d_n + d_{n+1} + \dots + d_{n+K-1}}{K} \quad (6.5)$$

where:

$$d_n = T_m^n - S_1^n = S_m^n - S_1^n + \mu_m$$

Recall that the following equality holds (Chapter 5, Section 5.3.2):

$$d = \frac{M^0(R_0)}{\Phi} \quad (6.6)$$

where $M^0(R_0)$ denotes the initial marking of the resource place of the overall organization.

These measures will be analyzed and compared for the different cases considered.

6.2.3 Numerical Results

Four different cases of resource and time constraints will be analyzed. The first example corresponds to the case where the DMO as a whole and each of the five DMs can only handle one input at a time. All the task processing times are assumed identical and equal to one unit of time. In the second case, there are more resources available for processing: the DMO can handle four external inputs at the same time and each DM is able to process at most two inputs simultaneously. The task processing times remain the same as in the first case (one unit of time). In the third case, the resources available are assumed to be the same as in the second example, but the time constraints now differ. The task processing times of DM4 are assumed to be twice as long as the other processing times, i.e., these processing times are equal to two units of time. Finally, the fourth case is an example where the processing times are arbitrary. The resources available are identical and set to three (the DM and the organization can handle three inputs at a time).

In each case, the firing schedule has been computed for a total of ten (10) repetitions of the process, i.e., $N=10$. Since the total number of transitions in the net amounts to $m=18$, the firing schedule obtained corresponds to sequence (S_1^n) for $i = 1, 2, \dots, 18$ and $n = 1, 2, \dots, 10$. The algorithm described in Appendix C has been used to determine this sequence. Let us now present the results obtained in each case.

6.2.3.1 First Case

The results obtained for the first case are shown on Table 6.4. The period π is 11 units of time and the periodicity factor K equals one. As expected, the steady-state process is strongly periodic, since the token content of the elementary directed circuits is one (see Chapter 5, Section

5.2.4). According to (6.1), the maximum throughput rate is:

$$\bar{Q} = \frac{1}{11} = 0.091 \text{ inputs processed per unit of time}$$

We can check that $\bar{Q} = 1/\alpha$, where α is the maximum average circuit processing time, which is also the average circuit processing time of the critical circuits. The three critical circuits obtained have indeed eleven (11) transitions and the token content is one. Since the transition firing time equal one unit of time, we immediately obtain:

$$\alpha = 11 \text{ units of time}$$

Furthermore, all the critical circuits have one unique resource place, which is R_0 . The first such critical circuit is represented in Figure 6.4. By deleting R_0 , we obtain the directed paths (or lines) for which the processing time (called time-delay in [3]) is maximal, as discussed in Chapter 5, Section 5.3.2. Accordingly, this time-delay is:

$$d = D_{\max} = 11 \text{ units of time}$$

and the dynamic response time, corresponding to the processing of the ten inputs is (relation (6.2))

$$T_{1s}^{10} = S_{1s}^{10} + 1 = 110 \text{ units of time.}$$

6.2.3.2 Second Case

The results obtained in the second case are shown on Table 6.5. It turns out that the steady-state process is 2-periodic (i.e., the periodicity factor K equals 2) with a period $\pi = 9$ units of time. Accordingly, the maximum throughput rate is:

$$\bar{Q} = \frac{2}{9} = 0.222 \text{ inputs processed per unit of time}$$

TABLE 6.4. COMPUTATION OF THE EXECUTION SCHEDULE: CASE 1

* RESOURCE and TIME CONSTRAINTS *

Marking of the Resource Places : R0 R1 R2 R3 R4 R5
1 1 1 1 1 1

Transition Firing Times :

t1	t2	t3	t4	t5	t6	t7	t8	t9	t10	t11	t12	t13	t14	t15	t16	t17	t18
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

* EXECUTION SCHEDULE *

t1 :	0.0	11.0	22.0	33.0	44.0	55.0	66.0	77.0	88.0	99.0
t8 :	1.0	12.0	23.0	34.0	45.0	56.0	67.0	78.0	89.0	100.0
t12:	1.0	12.0	23.0	34.0	45.0	56.0	67.0	78.0	89.0	100.0
t15:	1.0	12.0	23.0	34.0	45.0	56.0	67.0	78.0	89.0	100.0
t9 :	2.0	13.0	24.0	35.0	46.0	57.0	68.0	79.0	90.0	101.0
t4 :	2.0	13.0	24.0	35.0	46.0	57.0	68.0	79.0	90.0	101.0
t5 :	3.0	14.0	25.0	36.0	47.0	58.0	69.0	80.0	91.0	102.0
t2 :	4.0	15.0	26.0	37.0	48.0	59.0	70.0	81.0	92.0	103.0
t3 :	5.0	16.0	27.0	38.0	49.0	60.0	71.0	82.0	93.0	104.0
t6 :	6.0	17.0	28.0	39.0	50.0	61.0	72.0	83.0	94.0	105.0
t7 :	7.0	18.0	29.0	40.0	51.0	62.0	73.0	84.0	95.0	106.0
t10:	8.0	19.0	30.0	41.0	52.0	63.0	74.0	85.0	96.0	107.0
t13:	8.0	19.0	30.0	41.0	52.0	63.0	74.0	85.0	96.0	107.0
t16:	8.0	19.0	30.0	41.0	52.0	63.0	74.0	85.0	96.0	107.0
t11:	9.0	20.0	31.0	42.0	53.0	64.0	75.0	86.0	97.0	108.0
t14:	9.0	20.0	31.0	42.0	53.0	64.0	75.0	86.0	97.0	108.0
t17:	9.0	20.0	31.0	42.0	53.0	64.0	75.0	86.0	97.0	108.0
t18:	10.0	21.0	32.0	43.0	54.0	65.0	76.0	87.0	98.0	109.0

Period : 11.0 Periodicity Factor : 1

Critical Circuits :

11: p1-t2-p2-t3-p3-t6-p6-t7-p7-t10-p10-t11-p11-t18-R0-t1-p21-t8-p18-t4-p4-t5-p1
24: p1-t2-p2-t3-p3-t6-p6-t7-p25-t13-p13-t14-p14-t18-R0-t1-p21-t8-p18-t4-p4-t5-p1
46: p1-t2-p2-t3-p3-t6-p6-t7-p26-t16-p16-t17-p17-t18-R0-t1-p21-t8-p18-t4-p4-t5-p1

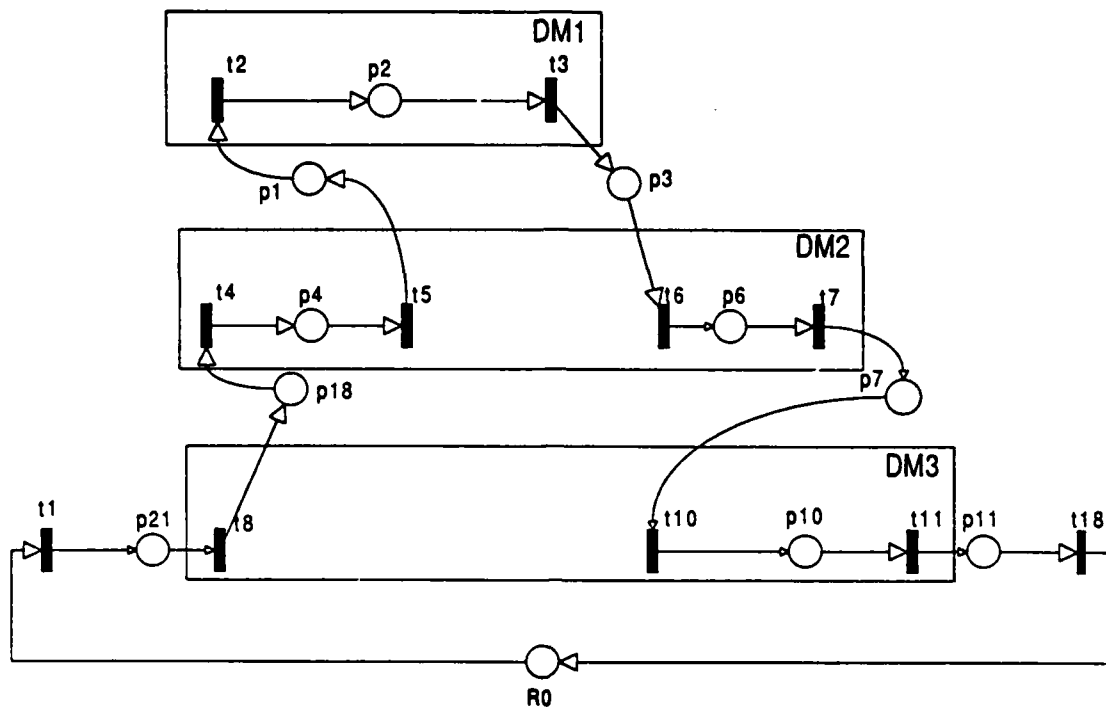


Figure 6.4 Critical Circuit (Case 1)

The process reaches its steady-state after the fourth repetition ($N=4$). For $n > 2$ and $i = 1, 2, \dots, 18$, we have:

$$\begin{cases} S_1^{3n+1} = S_1^{3n} + 8 \text{ (units of time)} \\ S_1^{3n+2} = S_1^{3n+1} + 1 \text{ (unit of time)} \end{cases}$$

Inputs are therefore processed at successive intervals of eight (8) units of time and then one (1) unit of time.

In that case, the dynamic response time is given by:

$$T_{18}^{10} = S_{18}^{10} + \mu_{18} = 47.0 + 1.0 = 48.0 \text{ units of time}$$

It is interesting to note that the ratio:

TABLE 6.5. COMPUTATION OF THE EXECUTION SCHEDULE: CASE 2

* RESOURCE and TIME CONSTRAINTS *

Marking of the Resource Places : R0 R1 R2 R3 R4 R5
4 2 2 2 2 2

Transition Firing Times :

t1	t2	t3	t4	t5	t6	t7	t8	t9	t10	t11	t12	t13	t14	t15	t16	t17	t18
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

* EXECUTION SCHEDULE *

t1 :	0.0	1.0	2.0	3.0	11.0	12.0	20.0	21.0	29.0	30.0
t8 :	1.0	2.0	10.0	11.0	19.0	20.0	28.0	29.0	37.0	38.0
t12:	1.0	2.0	10.0	11.0	19.0	20.0	28.0	29.0	37.0	38.0
t15:	1.0	2.0	10.0	11.0	19.0	20.0	28.0	29.0	37.0	38.0
t9 :	2.0	3.0	11.0	12.0	20.0	21.0	29.0	30.0	38.0	39.0
t4 :	2.0	3.0	11.0	12.0	20.0	21.0	29.0	30.0	38.0	39.0
t5 :	3.0	4.0	12.0	13.0	21.0	22.0	30.0	31.0	39.0	40.0
t2 :	4.0	5.0	13.0	14.0	22.0	23.0	31.0	32.0	40.0	41.0
t3 :	5.0	6.0	14.0	15.0	23.0	24.0	32.0	33.0	41.0	42.0
t6 :	6.0	7.0	15.0	16.0	24.0	25.0	33.0	34.0	42.0	43.0
t7 :	7.0	8.0	16.0	17.0	25.0	26.0	34.0	35.0	43.0	44.0
t10:	8.0	9.0	17.0	18.0	26.0	27.0	35.0	36.0	44.0	45.0
t13:	8.0	9.0	17.0	18.0	26.0	27.0	35.0	36.0	44.0	45.0
t16:	8.0	9.0	17.0	18.0	26.0	27.0	35.0	36.0	44.0	45.0
t11:	9.0	10.0	18.0	19.0	27.0	28.0	36.0	37.0	45.0	46.0
t14:	9.0	10.0	18.0	19.0	27.0	28.0	36.0	37.0	45.0	46.0
t17:	9.0	10.0	18.0	19.0	27.0	28.0	36.0	37.0	45.0	46.0
t18:	10.0	11.0	19.0	20.0	28.0	29.0	37.0	38.0	46.0	47.0

Period : 9.0 Periodicity Factor : 2

Critical Circuits :

5: p1-t2-p2-t3-p3-t6-p6-t7-p7-t10-p10-t11-R3-t8-p18-t4-p4-t5-p1

$$RT_{10} = \frac{T_{10}^{10}}{10} = 4.8 \text{ units of time per input}$$

is already quite close to the average response time (given by 6.4):

$$RT = \frac{1}{\bar{Q}} = 4.5 \text{ units of time per input}$$

The average response time, determined by (6.5), is in that case:

$$d = \frac{d_9 + d_{10}}{2} = \frac{(T_{10}^9 - S_1^9) + (T_{10}^{10} - S_1^{10})}{2} = \frac{18 + 18}{2} = 18$$

We can check that relation (6.6) is satisfied, since:

$$\frac{M^0(R_0)}{\bar{Q}} = 4 * 4.5 = 18 \text{ units of time}$$

Finally, there is a unique critical circuit, which is represented on Figure 6.5. This circuit includes nine (9) transitions and its token content is two (2), corresponding to the initial marking of the resource place R_0 . The maximum average circuit processing time is therefore:

$$\alpha = \frac{9}{2} = 4.5 \text{ units of time}$$

which is equal to $1/\bar{Q}$, as expected. It should be noted that, in contrast to the first case, the critical circuit obtained here is internal to the organizational structure. It is clear, from the structure of the circuit, that the throughput rate constraints come from the interactions between DM1, DM2 and DM3. In addition, only the resources of DM3 are critical.

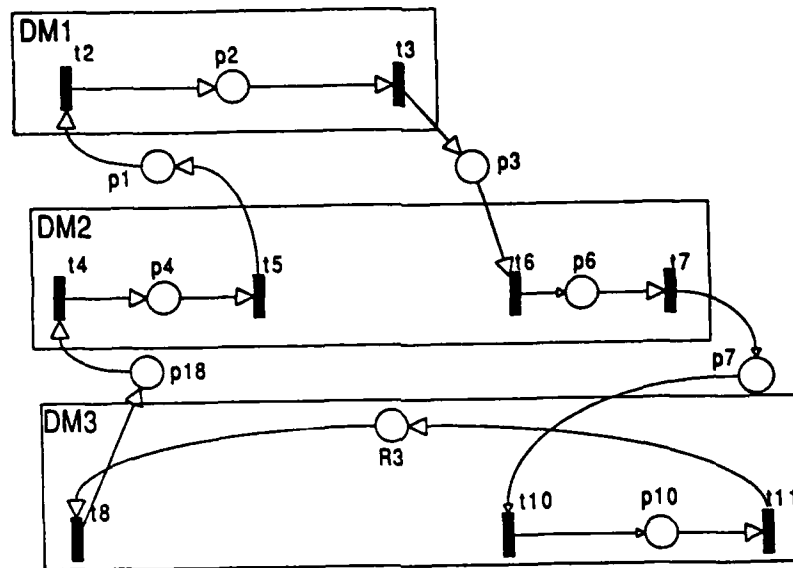


Figure 6.5 Critical Circuit (Case 2)

6.2.3.3 Third Case

The results obtained in the third case are shown on Table 6.6. The steady-state process is 2-periodic with a period of $\pi = 11$ units of time. The maximum throughput rate is therefore:

$$\bar{Q} = \frac{2}{11} = 0.182 \text{ inputs processed per unit of time}$$

The process reaches its steady-state after the fifth repetition ($N=5$). For $n > 3$ and $i = 1, 2, \dots, 18$, we have:

$$\begin{cases} S_i^{2n} = S_i^{2n-1} + 2 & (\text{units of time}) \\ S_i^{2n+1} = S_i^{2n} + 7 & (\text{units of time}) \end{cases}$$

The dynamic response time is, in that case:

$$T_{18}^{10} = S_{18}^{10} + \mu_{18} = 58.0 + 1.0 = 59.0 \text{ units of time}$$

TABLE 6.6. COMPUTATION OF THE EXECUTION SCHEDULE: CASE 3

* RESOURCE and TIME CONSTRAINTS *

Marking of the Resource Places : R0 R1 R2 R3 R4 R5
4 2 2 2 2 2

Transition Firing Times :

t1	t2	t3	t4	t5	t6	t7	t8	t9	t10	t11	t12	t13	t14	t15	t16	t17	t18
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	2.0	2.0	2.0	1.0	1.0	1.0	1.0

* EXECUTION SCHEDULE *

t1 :	0.0	1.0	2.0	3.0	13.0	15.0	24.0	26.0	35.0	37.0
t8 :	1.0	2.0	10.0	12.0	21.0	23.0	32.0	34.0	43.0	45.0
t12:	1.0	3.0	12.0	14.0	23.0	25.0	34.0	36.0	45.0	47.0
t15:	1.0	2.0	10.0	12.0	21.0	23.0	32.0	34.0	43.0	45.0
t9 :	3.0	5.0	14.0	16.0	25.0	27.0	36.0	38.0	47.0	49.0
t4 :	2.0	3.0	11.0	13.0	22.0	24.0	33.0	35.0	44.0	46.0
t5 :	3.0	5.0	14.0	16.0	25.0	27.0	36.0	38.0	47.0	49.0
t2 :	4.0	6.0	15.0	17.0	26.0	28.0	37.0	39.0	48.0	50.0
t3 :	5.0	7.0	16.0	18.0	27.0	29.0	38.0	40.0	49.0	51.0
t6 :	6.0	8.0	17.0	19.0	28.0	30.0	39.0	41.0	50.0	52.0
t7 :	7.0	9.0	18.0	20.0	29.0	31.0	40.0	42.0	51.0	53.0
t10:	8.0	10.0	19.0	21.0	30.0	32.0	41.0	43.0	52.0	54.0
t13:	8.0	10.0	19.0	21.0	30.0	32.0	41.0	43.0	52.0	54.0
t16:	8.0	10.0	19.0	21.0	30.0	32.0	41.0	43.0	52.0	54.0
t11:	9.0	11.0	20.0	22.0	31.0	33.0	42.0	44.0	53.0	55.0
t14:	10.0	12.0	21.0	23.0	32.0	34.0	43.0	45.0	54.0	56.0
t17:	9.0	11.0	20.0	22.0	31.0	33.0	42.0	44.0	53.0	55.0
t18:	12.0	14.0	23.0	25.0	34.0	36.0	45.0	47.0	56.0	58.0

Period : 11.0 Periodicity Factor : 2

Critical Circuits :

17: p1-t2-p2-t3-p3-t6-p6-t7-p25-t13-p13-t14-R4-t12-p24-t5-p1

and the average processing time:

$$d = \frac{d_0 + d_{10}}{2} = \frac{(T_{10}^0 - S_1^0) + (T_{10}^{10} - S_1^{10})}{2} = \frac{22+22}{2} = 22$$

As expected (relation 6.6):

$$\frac{M^0(R_0)}{2} = 4 * 5.5 = 22 = d$$

As in the previous case, we obtain a unique critical circuit, which is internal to the organizational structure. The corresponding circuit is shown on Figure 6.6. In that case, however, the interactions between DM1, DM2 and DM4 are the one that constrain the maximum throughput rate. The resources of DM4 are now critical.

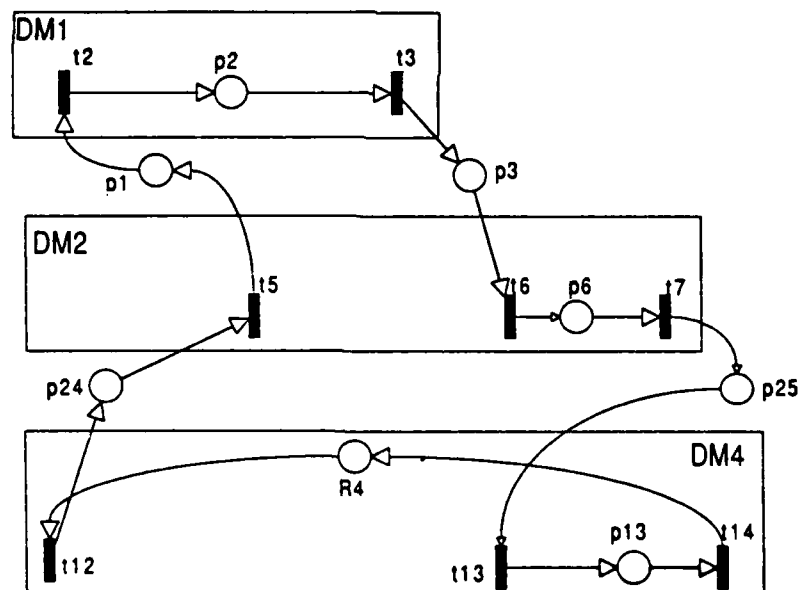


Figure 6.6. Critical Circuit (Case 3)

6.2.3.4 Fourth Case

The results obtained in the fourth case are shown in Table 6.7. In this example, we have taken totally arbitrary transition firing times and set the initial marking of all the resource places to three (3) tokens. It turns out that the steady-state process is 3-periodic with a period $\pi=25.5$ units of time. Accordingly, that maximum throughput rate is:

$$\bar{q} = \frac{3}{25.5} = 0.118 \quad \text{inputs processed per units of time}$$

We note that the process reaches its steady-state after the fourth repetition. For $n \geq 2$ and $i = 1, 2, \dots, 18$, we obtain:

$$\begin{cases} S_i^{n-1} = S_i^{n-2} + 3.5 & (\text{units of time}) \\ S_i^n = S_i^{n-1} + 3.5 & (\text{units of time}) \\ S_i^{n+1} = S_i^n + 18.5 & (\text{units of time}) \end{cases}$$

The dynamic response time is in that case:

$$T_{18}^{10} = S_{18}^{10} + \mu_{18} = 100.0 + 2.0 = 102.0 \text{ units of time}$$

and the average processing time:

$$\begin{aligned} d &= \frac{d_s + d_p + d_{10}}{3} \\ &= \frac{(T_{18}^8 - S_1^8) + (T_{18}^9 - S_1^9) + (T_{18}^{10} - S_1^{10})}{3} \\ &= \frac{25.5 + 25.5 + 25.5}{3} = 25.5 \text{ units of time} \end{aligned}$$

TABLE 6.7. COMPUTATION OF THE EXECUTION SCHEDULE: CASE 4

*** RESOURCE and TIME CONSTRAINTS ***

Marking of the Resource Places : R0 R1 R2 R3 R4 R5
3 3 3 3 3 3

Transition Firing Times :

t1	t2	t3	t4	t5	t6	t7	t8	t9	t10	t11	t12	t13	t14	t15	t16	t17	t18
2.0	1.5	3.0	3.5	2.5	3.0	2.0	1.5	3.0	2.0	2.5	1.5	1.0	2.0	3.5	2.0	1.0	2.0

*** EXECUTION SCHEDULE ***

t1 :	0.0	2.0	4.0	25.5	29.0	32.5	51.0	54.5	58.0	76.5
t8 :	2.0	4.0	6.0	27.5	31.0	34.5	53.0	56.5	60.0	78.5
t12:	2.0	4.0	6.0	27.5	31.0	34.5	53.0	56.5	60.0	78.5
t15:	2.0	5.5	9.0	27.5	31.0	34.5	53.0	56.5	60.0	78.5
t9 :	3.5	6.5	9.5	29.0	32.5	36.0	54.5	58.0	61.5	80.0
t4 :	3.5	7.0	10.5	29.0	32.5	36.0	54.5	58.0	61.5	80.0
t5 :	7.0	10.5	14.0	32.5	36.0	39.5	58.0	61.5	65.0	83.5
t2 :	9.5	13.0	16.5	35.0	38.5	42.0	60.5	64.0	67.5	86.0
t3 :	11.0	14.5	18.0	36.5	40.0	43.5	62.0	65.5	69.0	87.5
t6 :	14.0	17.5	21.0	39.5	43.0	46.5	65.0	68.5	72.0	90.5
t7 :	17.0	20.5	24.0	42.5	46.0	49.5	68.0	71.5	75.0	93.5
t10:	19.0	22.5	26.0	44.5	48.0	51.5	70.0	73.5	77.0	95.5
t13:	19.0	22.5	26.0	44.5	48.0	51.5	70.0	73.5	77.0	95.5
t16:	19.0	22.5	26.0	44.5	48.0	51.5	70.0	73.5	77.0	95.5
t11:	21.0	24.5	28.0	46.5	50.0	53.5	72.0	75.5	79.0	97.5
t14:	20.0	23.5	27.0	45.5	49.0	52.5	71.0	74.5	78.0	96.5
t17:	21.0	24.5	28.0	46.5	50.0	53.5	72.0	75.5	79.0	97.5
t18:	23.5	27.0	30.5	49.0	52.5	56.0	74.5	78.0	81.5	100.0

Period : 25.5 Periodicity Factor : 3

Critical Circuits :

11: p1-t2-p2-t3-p3-t6-p6-t7-p7-t10-p10-t11-p11-t18-R0-t1-p21-t8-p18-t4-p4-t5-p1

As expected (relation (6.6)):

$$\frac{M^{\circ}(R_0)}{\Phi} = 3 \cdot \frac{25.5}{3} = 25.5 = d$$

There is also a unique critical circuit, which is the same as the first such circuit obtained in Case 1 (see Figure 6.4). The maximum average circuit processing time is:

$$\begin{aligned} \alpha &= \frac{\mu_2 + \mu_3 + \mu_6 + \mu_7 + \mu_{10} + \mu_{11} + \mu_{12} + \mu_1 + \mu_8 + \mu_4 + \mu_5}{M^{\circ}(R_0)} \\ &= \frac{1.5 + 3.0 + 3.0 + 2.0 + 2.0 + 2.5 + 2.0 + 2.0 + 1.5 + 3.5 + 2.5}{3} \\ &= \frac{25.5}{3} = 8.5 \text{ units of time} \end{aligned}$$

As expected, we obtained $\Phi = 1/\alpha$.

6.2.4 Summary Analysis of the Results

6.2.4.1 Comparison of the Performance Measures

The performance measures obtained in each of the four cases analyzed previously are summarized in Table 6.8. We note first that the performance obtained in the first case is the worst one, compared to the other cases. The maximum throughput rate Φ is the lowest and the dynamic response time T_m^N (resp. the average response time RT) is the longest. In the second case, where there are more resources available for processing, the performance is greatly improved. The maximum throughput rate is more than twice the one obtained in the first case while the response time is less. However, the average processing time, d , which represents the average amount of time it takes to process any individual input, is now larger, as emphasized in Chapter 5, Section 5.3.2. In the third case, the resources

available for processing are the same as in the second case, but the processing times of DM4 are twice the other ones. As expected, the performance obtained is slightly degraded, when compared to the second case, but is still much better than in the first case. Recall that in the fourth case, the task processing times are arbitrary (but are greater or equal to one unit of time) and there are more resources than in the first three cases. It turns out that the performance obtained is worse than in case 2 and 3, but is still better than in case 1.

TABLE 6.8. SUMMARY OF THE RESULTS

$\backslash$	CASE 1	CASE 2	CASE 3	CASE 4
Π	11.0	9.0	11.0	25.5
K	1	2	2	3
ϕ	0.091	0.222	0.182	0.118
α	11.0	4.5	5.5	8.5
T_m^N	110.0	48.0	59.0	102.0
RT_N	11.0	4.8	5.9	10.2
RT	11.0	4.5	5.5	8.5
d	11.0	18.0	22.0	25.5

6.2.4.2 Execution Schedule of the Concurrent Tasks

In the previous section; we have been able to evaluate and compare the performance of the DMO by computing different time-related measures from the execution schedule obtained. The performance measures were computed, given the time and resource constraints set in each case. We want to analyze further how the constraints can be modified so as to improve the performance of the organization. In Chapter 4, this problem has already been addressed for the maximum throughput rate. It was said that the critical circuits, such as those obtained in the four cases, determine precisely which of the time and resource constraints bound the throughput rate. We want to investigate now how the time-delays in the organization are related to the various task processing times and, in particular, how it is possible to improve the dynamic response time.

At this point, the execution schedule obtained for the concurrent tasks, i.e., the instants of time at which the transitions within a slice fire, is very interesting to analyze. In the four cases described previously, we can note that, for cases 1 and 2, all the transitions within a slice fire at the same instants of time (at each repetition of the process), which is not true for case 3 and 4 (recall that the list of the slices is provided on Table 6.3). This result is not surprising: as stated in Chapter 5, Section 5.1.2, when all the transition firing times are equal to one unit of time, the slice $\bar{T}_i$ ($i = 1, 2, \dots, s$) determines the transitions that fire concurrently at the i -th instant. However, when the transition firing times are arbitrary, there is no reason why the concurrent operations should occur at the same instant of time. This is actually what makes the process to occur asynchronously in real-time.

From the execution schedule it is possible to identify which operations take place (repeatedly) at the latest instants, compared to the other concurrent operations (i.e., compared to the other transitions of the same slice). For instance, in case 3, the transitions that fire the latest are $t_{1,2}$ (compared to t_s and $t_{1,1}$, which are the other transitions of

$\bar{T}_2$) and t_{14} (compared to t_{11} and t_{17} , which are the other transitions of $\bar{T}_{10}$). It turns out that these particular transitions are critical for the processing delays. We are sure that, by reducing their processing times, the time-delay of the organization will improve. By determining, from the execution schedule obtained, the concurrent tasks that fire at the latest instants, we therefore know which time constraints are critical for the processing delays. It becomes easier to modify the right constraints that can improve the related performance of the organization.

CHAPTER VII

CONCLUSIONS AND DIRECTIONS FOR FURTHER RESEARCH

7.1 CONCLUSIONS

In this thesis, the time-related performances of a DMO have been analyzed and evaluated. The computation of two measures of performance has been carried out: the maximum throughput rate, which characterizes the maximum processing rate achievable by the organization, and the execution schedule, determining the earliest instants at which the various tasks can be executed. In order to evaluate these MOPs, the DMO has been modeled as an asynchronous concurrent system using Timed Petri Nets.

The modeling addresses two types of constraints. On the one hand, there are the time constraints derived from the various task processing times. On the other hand, the resource constraints, which affect both individual DMs and the DMO as a whole. For the human DMs in the organization, the resource limitation corresponds to the limited capacity of the human short-term memory, which bounds the amount of information that can be handled at the same time. For the overall organization, the resource constraint derives from the limited capacity of the system to store the information currently processed. In both cases, the resource limitation is modeled as a constraint on the total number of inputs that can be processed simultaneously.

The maximum throughput rate is expressed as a function of the resource and time constraints in the following manner. The inclusion of the resource constraints in the Petri Net modeling the DMO results in directed circuits (or loops), which are characterized by:

- the circuit processing time, μ , defined as the sum of the different task processing times of the circuit. μ represents the amount of time

it takes for one input to complete the processing operations of the circuit.

- The amount of resources available, n , which bounds the total number of inputs that can be processed at the same time in the circuit.

For a given circuit, the ratio n/μ characterizes the average circuit processing rate. As shown in Chapter 4, the minimum average circuit processing rate, taken over all the directed circuits of the net, determines the maximum throughput rate of the deterministic system, i.e., when all the task processing times are deterministic. For the non-deterministic system, where processing times are discrete random variables with a certain probability distribution (to model decision switches), we only obtain an upper bound to the maximum throughput rate: In that case, the average circuit processing time is computed, taking the expected (or mean) processing time of each task. The determination of the critical circuits, for which the corresponding average processing rate is precisely minimal, is very useful for evaluating and comparing different organizations. Indeed, the critical circuits characterize the particular time and resource constraints that actually bound the throughput rate. The way with which the different constraints affect the performance of the organization can now be completely specified. In particular, the problem of modifying the right constraints so as to improve the performance of the actual design becomes transparent.

In Chapter 5, we have developed a method for obtaining and analyzing the exact execution schedule of the deterministic system. In particular, we have described a representation, as defined by the slices of the net, that allows for a clear characterization of the causality of the system. The causality refers to the partial ordering of the different operations that take place in the process. The execution schedule so obtained determines the earliest instants at which the various tasks can be executed in real time for a process that occurs repetitively. An interesting aspect of the steady-state process has also been highlighted, namely the condition

of K-periodicity, according to the results obtained by Chretienne [16].

Once we have determined the maximum rate with which inputs can arrive without overloading the system, as specified by the maximum throughput rate, the execution schedule becomes useful for evaluating the time-delays associated with the various processing tasks. Two useful MOPs can be computed from the execution schedule: (a) the earliest instants at which the processing of the incoming inputs can start, and (b) the dynamic response time of the organization, characterized as the minimum amount of time it takes to complete the processing of any large number of external inputs. In our modeling, both the DMO as a whole and individual DMs are allowed to handle more than one input at a time, depending on the type of resource constraints. In such cases, the response time necessarily depends on the number of inputs being processed simultaneously.

Finally, the evaluation of the maximum throughput rate, together with the precise execution schedule, completely characterize the dynamic behavior of the system. These measures describe entirely the time-related performance of the DMO.

7.2 DIRECTIONS FOR FURTHER RESEARCH

The evaluation of the maximum throughput rate and of the execution schedule has been obtained, in this work, for deterministic systems, where all the task processing times are assumed deterministic. However, as discussed in Chapter 3, the Petri Net representation of the DMO also includes switches, which model the particular processing stages where decisions take place. Because the algorithms selected by the decisions may have in reality different processing times, the processing time of these particular tasks was treated as a discrete random variable, with a probability distribution that depends on the decision rules. The probability distribution was assumed, however, to be independent from one input to the next, since tokens (which model the inputs in the Petri Net framework) are indistinguishable in ordinary Petri Nets. In such a case,

i.e., when the system is non-deterministic, we have only obtained a lower and an upper bound for the performance measures.

Further research is needed to obtain more accurate measures of performance, by relaxing some of the assumptions made about the decision switches. In particular, it appears promising to extend the modeling of the DMO using a more sophisticated type of Petri Net, called Colored Petri Nets [23]. In contrast to ordinary Petri Nets, the tokens, in this type of Petri Nets, are marked. More precisely, each token is identified by a certain color, so that it is now possible to differentiate the tokens. Clearly enough, Colored Petri Nets provide a useful tool to account for the different type of inputs that an actual organization receives and processes. As developed in [1], external inputs can be treated as elements of an alphabet and the decision that rules the selection of a particular algorithm depends now on the specific element in the alphabet. Therefore, we can easily foresee how Colored Petri Nets would be relevant for evaluating more accurately the time performance of a DMO: In that case, the probability distribution that rules the selection among the possible algorithms (and consequently the corresponding processing time), should depend on the color of the token, rather than being independent from one token to the next. The Petri Net model so analyzed, would now fully account for the assumptions that are made in the information theoretic framework.

REFERENCES

- [1] Levis, A. H. (1984). "Information Processing and Decisionmaking Organizations: A Mathematical Description," Large Scale Systems, Vol. 7, pp. 151-163.
- [2] Tabak, D. and A. H. Levis (1985). "Petri Net Representation of Decision Models," IEEE Trans. on Systems, Man, and Cybernetics, Vol. SMC-15, No. 6, pp. 812-818.
- [3] Jin, Y.-Y. V., A. H. Levis, and P. Remy (1986). "Delays in Acyclical Distributed Decisionmaking Organizations," Proc. 4th IFAC/IFORS Symposium on Large Scale Systems, Zurich, Switzerland.
- [4] Boettcher, K. L., and A. H. Levis (1983). "Modeling and Analysis of Teams of Interacting Decisionmakers with Bounded Rationality," Automatica, No. 6, pp. 703-709.
- [5] Bayley, R. W. (1982). Human Performance Engineering: A Guide for System Designers, Prentice-Hall, Englewood Cliffs, NJ.
- [6] Ramchandani, C., (1974). "Analysis of Asynchronous Concurrent Systems by Timed Petri nets," Technical Report No. 120, Laboratory for Computer Science, M.I.T., Cambridge, MA.
- [7] Peterson, J. L. (1981). Petri Net Theory and the Modeling of Systems, Prentice-Hall, Englewood Cliffs, NJ.
- [8] Memmi, G., and G. Roucairol (1979). "Linear Algebra in Net Theory," Net Theory and Application, Lecture Notes in Computer Science, No. 84, Springer-Verlag, Hamburg, FRG, pp. 213-223.
- [9] Brams, G. W. (1983). Reseaux de Petri: Theorie et Pratique, Masson, Paris, France.
- [10] Sifakis, J., (1978). "Structural Properties of Petri Nets," Mathematical Foundations of Computer Science, Lecture Notes in Computer Science, No. 64, Springer-Verlag, Berlin, FRG, pp. 474-483.
- [11] Memmi, G., (1979). "Notion de Dualite et de Symmetrie dans les Reseaux de Petri," Semantics of Concurrent Computations, Lecture Notes in Computer Science, No. 70, Springer-Verlag, Berlin, FRG, pp. 91-108.
- [12] Commoner, F., and A. Holt (1971). "Marked Directed Graphs," Journal of Computer and System Science, Vol. 5, No. 5, pp. 511-523.

- [13] Martinez, J., and M. Silva (1980). "A Simple and Fast Algorithm to Obtain all Invariants of a Generalized Petri Net," Lecture Notes in Computer Science, No. 52, Springer-Verlag, Berlin, FRG, pp. 302-310.
- [14] Sifakis, J. (1980). "Performance Evaluation of Systems Using Nets," Net Theory and Applications, Lecture Notes in Computer Science, Springer-Verlag, Berlin, FRG, pp. 307-319.
- [15] Wiley, R. P. (1986) "Performance Analysis of Stochastic Timed Petri Nets," Ph.D. Thesis, Report LIDS-TH-1525, Laboratory for Information and Decision Systems, MIT, Cambridge, MA.
- [16] Chretienne P., and J. Carlier (1984). "Modeling Scheduling Problems with Timed Petri Nets," Advances in Petri Nets, Lecture Notes in Computer Science, No. 188, Springer-Verlag, Berlin, FRG, pp. 62-82.
- [17] Stabile, D.A., and A. H. Levis (1981). "The Design of Information Structures: Basic Allocation Strategies for Organizations," Large Scale Systems, Vol. 6, pp. 123-132.
- [18] Remy, P., A. H. Levis and V.Y.-Y. Jin (1986). "On the Design of Distributed Organizational Structures," LIDS-P-1581, Laboratory for Information and Decision Systems, MIT, Cambridge, MA.
- [19] Alaiwan, H., and J. M. Toudic (1985). "Recherche des Semi-Flots, des Verrous et des Trappes dans les Reseaux de Petri," Technique et Science Informatiques, Vol. 4, No. 1, Dunod, France, pp. 103-112.
- [20] Best, E. (1979). "Atomicity of Activities," Net Theory and Applications, Lecture Notes in Computer Science, No. 84, Springer-Verlag, Hamburg, FRG, pp. 223-250.
- [21] Fernandez, C., and P.S. Thiagarajan, (1984). "A Lattice Theoretic View of K-Density," Advances in Petri Nets, Lecture Notes in Computer Science, No. 188, Springer-Verlag, Berlin, FRG, pp. 139-153.
- [22] Weingaertner, S. (1986). "A Model of Submarine Emergency Decision-making and Decision Aiding," SM Thesis, Laboratory for Information and Decision Systems, MIT, Cambridge, MA (in preparation).
- [23] Jensen, K. (1981). "Colored Petri Nets and the Invariant Method," Theoretical Computer Science, No. 14, North-Holland, pp. 317-336.

APPENDIX A

ALGORITHM TO OBTAIN ALL THE CIRCUITS OF AN EVENT-GRAPH

A.1 SUMMARY

This appendix presents the algorithm that has been used to obtain all the circuits of an Event-Graph. As discussed in Chapter 4, it is necessary to determine all the circuits of the net, so as to compute the maximum throughput rate. The algorithm described here has been developed by Alaiwan and Toudic [19] and determines, in fact, all the minimal support S-invariants of an ordinary Petri Net. As we have proved (Theorem 2.6), the directed circuits of an Event-Graph correspond to the minimal S-component of the net. This is why this algorithm can be used directly.

A.2 ALGORITHM TO OBTAIN ALL THE MINIMAL SUPPORT S-INVARIANTS OF A GENERAL PETRI NET

The complete proof of the algorithm can be found in Martinez and Silva [13]. An improved and more detailed version is given in [19]. We first present the description of the algorithm and then roughly explain how it works.

In the following description, C is assumed to be the Incidence Matrix of the Petri Net, of dimensions $n \times m$, where n is the number of places and m the number of transitions. C_{ij} denotes, as usual, the element corresponding to the i th row and j -th column of C and I_n denotes the $n \times n$ identity matrix.

Description of the Algorithm

Step 1: $A = C$; $D = I_n$ {initialization}

Step 2: REPEAT for $j = 1$ UNTIL $j = m$ {main loop}

Step 2.1: Determine the sets I_1 and I_2 such that:

$$I_1 = \{i_1 / A_{i_1,j} > 0\} \text{ and } I_2 = \{i_2 / A_{i_2,j} < 0\}$$

Step 2.2: For all pairs $(i_1, i_2) \in I_1 \times I_2$ do

Step 2.2.1: Append to the Matrix A the row vector:

$$A_{i_1,j} * (i_2\text{-th row of A}) - A_{i_2,j} * (i_1\text{-th row of A})$$

Step 2.2.2: Append to the Matrix D the row vector:

$$A_{i_1,j} * (i_2\text{-th row of D}) - A_{i_2,j} * (i_1\text{-th row of D})$$

Step 2.3: Eliminate from A and D all the rows with index $i \in I_1 \cup I_2$.

Step 2.4: Eliminate from D all the rows whose support is non-minimal with respect to the other rows of D. If two rows have the same support eliminate one of them.

Step 2.5: Eliminate from A the rows corresponding to the rows eliminated from D.

Step 3: The rows of D determine all the minimal support S-invariant of the net.

Remark: In the algorithm, we call support of a n-positive integer vector v , the set of elements, $i \in \{1, 2, \dots, n\}$, such that the i -th component of v is non-null. If $G = \{v_1, \dots, v_k\}$ is a set of vectors, v_i ($i \in \{1, 2, \dots, k\}$) will be a vector whose support is minimal in G iff there does not exist in G a non-null vector v_j such that its support is strictly

included in the support of v_i .

Let us make a brief comment on how the algorithm works. The underlying idea is to proceed in m steps (where m is the number of transitions), by finding the minimal support S -invariants of the R^i net ($i=0,1,\dots,m-1$), which results from the elimination of the transitions $\{i+1,\dots,m\}$. The Matrix D is constructed so that at the i -th iteration, the row vectors of D are precisely the minimal support S -invariants of the net R^i . Initially $D = I_n$ because R^0 corresponds to the net without transitions and therefore each single place constitutes a minimal support invariant. Once the minimal support S -invariants of the net R^i are obtained, those of the net R^{i+1} are generated in the following manner: As they obviously are also invariants for the net R^i , they can be expressed as a positive linear combination of the minimal support S -invariants of R^i (Chapter 2, Section 2.2, Theorem 2.2), i.e., as the rows of D . However, the invariants generated are not all minimal support S -invariants and it is necessary to eliminate the ones that are not by comparing their supports.

A.3 DETERMINATION OF THE DIRECTED CIRCUITS FOR AN EVENT-GRAPH

We have proved that the directed circuits of an Event-Graph are the minimal S -components (Theorem 2.6). Let us recall that an S -component is a subnet constructed as follows:

- the set of places P_s is the support of the corresponding minimal S -invariant,
- the set of transitions T_s are all the transitions of the PN connected to the places of P_s , i.e.,

$$T_s = \bigcup_{p \in P_s} (p \cup p')$$

Once we have obtained all the minimal support S -invariants of the Event-

Graph, using the algorithm of Alawain and Toudic [19], the determination of the circuits becomes straightforward: For each minimal support, we then determine the unique output (or input) transition of each place of the support and we immediately obtain the corresponding S-component, which is a circuit.

A.4 EXAMPLE

Let us illustrates the algorithm with a simple example. The example considered is shown on Figure A.1.

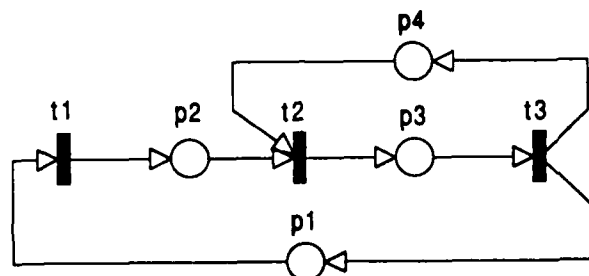


Figure A.1 Example of an Event-Graph

The incidence matrix is:

$$C = \begin{matrix} & t_1 & t_2 & t_3 \\ \begin{matrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{matrix} & \begin{bmatrix} -1 & 0 & 1 \\ 1 & -1 & 0 \\ 0 & 1 & -1 \\ 0 & -1 & 1 \end{bmatrix} \end{matrix}$$

Let us now carry out the steps of the algorithm described previously.

Initially (Step 1):

$$A, D = \begin{bmatrix} -1 & 0 & 1 \\ 1 & -1 & 0 \\ 0 & 1 & -1 \\ 0 & -1 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

For $j = 1$

Step 2.1: $I_1 = \{2\}$ $I_2 = \{1\}$

Therefore, we append the row: row 1 + row 2.

Step 2.2:

$$A, D = \begin{bmatrix} -1 & 0 & 1 \\ 1 & -1 & 0 \\ 0 & 1 & -1 \\ 0 & -1 & 1 \\ 0 & -1 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{bmatrix}$$

At the next step, we eliminate row 1 and row 2:

Step 2.3:

$$A, D = \begin{bmatrix} 0 & 1 & -1 \\ 0 & -1 & 1 \\ 0 & -1 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{bmatrix}$$

The rows of D are obviously minimal support so no elimination occurs at Step 2.4 and Step 2.5.

For $j = 2$

Step 2.1: $I_1 = \{1\}$ $I_2 = \{2,3\}$

Step 2.2 and 2.3: we append the two rows formed by (row 1 + row 2) and (row 1 + row 3) and eliminate afterwards row 1, row 2, and row 3, so that the new matrices are:

$$A, D = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

Once again, the two rows of D are minimal supports. Now the last iteration, corresponding to $j = 3$ is useless, since the third column of A is already null. The two rows of D are therefore the minimal support S-invariants of the net, which correspond to the set of places: $\{p_1, p_4\}$ and $\{p_1, p_2, p_3\}$. The two directed circuits are immediately obtained:

$$\rho_1 = (p_1, t_1, p_4, t_2, p_3)$$

$$\rho_2 = (p_1, t_1, p_2, t_2, p_3, t_3, p_1)$$

APPENDIX B

DEMONSTRATION OF THEOREM 5.1

In this appendix, we present the proof of Theorem 5.1, stated in Chapter 5, Section 5.2.2. The result was obtained by Ramchandani [6], but the approach taken here is a different version of the proof.

Let us recall the result stated in Theorem 5.1: A firing schedule (S_i^n) is feasible iff, for any pair of transitions (t_i, t_j) connected by place p_{ij} and for $n = 1, 2, 3, \dots$:

$$S_j^{n+M_{ij}^0} \geq S_i^n + \mu_i \quad (B.1)$$

where:

S_i^n denotes the instant of the n -th firing initiation of transition t_i

μ_i is the firing of time of t_i

M_{ij}^0 is the initial marking (at $\tau=0$) of place p_{ij} (i.e., the unique place whose input transition is t_i and output transition t_j).

Let us recall the notation introduced in Chapter 2, Section 2.4.1:

$I_j(\tau)$ denotes the number of initiations of transition t_j during the interval of time $[0, \tau]$.

$T_j(\tau)$ denotes the number of terminations of transition t_j during the interval of time $[0, \tau]$.

The marking of place p_{ij} at any instant τ is given by (Equation 2.10):

$$M_{ij}(\tau) = M_{ij}^{\circ} + T_i(\tau) - I_j(\tau) \quad (B.2)$$

From what has been discussed in Chapter 5, Section 5.2.1, it should be clear that a firing schedule is feasible if and only if the marking of each place remains non-negative at any instant of time (assuming that if a firing should be initiated before the transition is enabled, the marking of the input places becomes strictly negative). In order to prove the theorem, we need therefore to prove the equivalence between (B.1) and the fact that, at any instant τ , $M_{ij}(\tau) \geq 0$.

Proof:

- (1) Suppose that the firing schedule is feasible and let us consider the instant τ , such that:

$$\tau = S_j^{n+M_{ij}^{\circ}}$$

where n is any positive integer. This is the instant of the $(n+M_{ij}^{\circ})$ -th firing initiation of transition t_j . Accordingly:

$$I_j(\tau) = n + M_{ij}^{\circ}$$

Hence:

$$M_{ij}(\tau) = M_{ij}^{\circ} + T_i(\tau) - n - M_{ij}^{\circ} = T_i(\tau) - n$$

from (B.2).

By assumption $M_{ij}(\tau) \geq 0$ (since the firing schedule is feasible) and therefore: $T_i(\tau) \geq n$. This implies that, during the interval of time $[0, \tau]$, at least n firing terminations of transition t_i have occurred.

Since $S_1^n + \mu_1$ is the instant of the n -th firing termination of t_1 , we deduce:

$$\tau = S_j^{n+M_{1j}^0} \geq S_1^n + \mu_1$$

- (2) Let us assume now that the set of inequalities described by (B.1) are verified. If we consider any instant τ such that:

$$S_j^{n+M_{1j}^0} \leq \tau < S_j^{n+1+M_{1j}^0}$$

where n is any positive integer, then the number of initiation firings of transition t_j during the interval $[0, \tau]$ is precisely:

$$I_j(\tau) = n + M_{1j}^0$$

By assumption, we have also:

$$\tau \geq S_1^n + \mu_1 \quad (\text{from B.1})$$

which implies that, during $[0, \tau]$, there have been at least n firing terminations of transition t_1 . Hence:

$$T_1(\tau) \geq n$$

Applying equation (B.2) yields:

$$M_{1j}(\tau) = M_{1j}^0 + T_1(\tau) - n - M_{1j}^0 \geq 0$$

If we consider now any instant τ such that

$$0 < \tau \leq S_j^{M_{ij}^{\circ}}$$

It means that, during the interval $[0, \tau]$, there have been less than M_{ij}° firing initiations of transition t_j . Accordingly:

$$I_j(\tau) \leq M_{ij}^{\circ}$$

and we deduce immediately that:

$$M_{ij}(\tau) = M_{ij}^{\circ} + T_i(\tau) - I_j(\tau) \geq T_i(\tau) \geq 0.$$

Q.E.D.

The firing schedule is therefore feasible.

APPENDIX C

ALGORITHM TO COMPUTE THE FIRING SCHEDULE

In this Appendix, we present the algorithm that has been developed to obtain the execution schedule of the DMO, according to the analysis carried out in Chapter 5. We first recall the key points of the analysis and then describe the corresponding algorithm.

C.1 SUMMARY ANALYSIS OF THE FIRING SCHEDULE

The execution schedule that is determined here characterizes the earliest instants at which the various tasks can be executed, assuming that the processing starts at $\tau=0$ and occurs repetitively. We denote by:

μ_i the (deterministic) firing time of t_i

S_i^n the instant of the n-th firing initiation of transition t_i
(corresponding to the n-th repetition of the process)

p_{ij} the unique place whose input transition is t_i and output transition t_j

M_{ij}^0 the initial marking (at $\tau=0$) of the corresponding place p_{ij}

Recall that $M_{ij}^0 = 0$, except for the resource places of the organization, in which case the initial marking corresponds to the resources available for processing. This assumption means that at $\tau=0$, there are no inputs already being processed.

For each transition t_j , we consider now all its input places, that we denote by $p_{i_1 j}, p_{i_2 j}, \dots, p_{i_r j}$ and the input transitions of each of these places, i.e., $t_{i_1}, t_{i_2}, \dots, t_{i_r}$, as shown on Figure C.1.

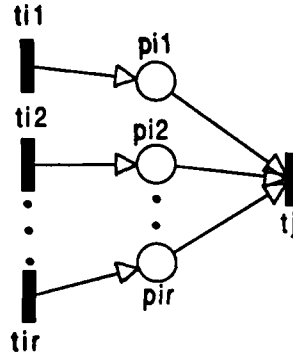


Figure C.1 Input Places and Input Transitions of t_j

In the algorithm, we will denote by:

$$\overleftarrow{t_j} = \{t_{i_1}, t_{i_2}, \dots, t_{i_r}\}$$

the corresponding set of transitions. The backward arrow means that we consider all the transitions that precede transition $\overleftarrow{t_j}$. Naturally, t_j is uniquely determined by the structure of the Petri Net model of the DMO.

For each transition t_j , S_j^n is now determined from the schedule of the transitions t_i that belong to $\overleftarrow{t_j}$, by the relation (see relation (5.7)):

$$S_j^n = \max (S_i^{n-M_{ij}^0} + \mu_i, S_j^{n-1} + \mu_j) \quad (C.1)$$

In the relation (C.1), the maximum is computed for all i such that $t_i \in \overleftarrow{t_j}$.

Let us recall briefly from where this relation comes. Let us consider any transition t_i belonging to $\overleftarrow{t_j}$, i.e., any of the transitions $t_{i_1}, \dots, t_{i_r}$, as shown on Figure C.1. At the instant of the n -th termination firing of transition t_i , i.e., at the instant:

$$\tau = S_i^n + \mu_i$$

transition t_i has exactly "produced" n tokens in place p_{ij} (since t_i has exactly fired n times from the initial instant). Since there were initially M_{ij}^0 tokens in this place, the total number of tokens available in p_{ij} during the interval of time $[0, \tau]$, is precisely $n + M_{ij}^0$. Now, each time that the firing of transition t_j was initiated, one token was consumed in p_{ij} . Accordingly, the firing of transition t_j could not have been initiated more than $(n + M_{ij}^0)$ times in the time interval $[0, \tau]$, meaning that:

$$S_j^{n+M_{ij}^0} \geq \tau = S_i^n + \mu_i \quad (C.2)$$

This intuitive result is proved in the previous Appendix (Appendix B). Applying (C.2) for all the transitions t_i belonging to $\bar{t}_j$ yields:

$$S_j^{n+M_{ij}^0} \geq \max (S_i^n + \mu_i) \quad (C.3)$$

However, in our modeling, the firing of a transition cannot be initiated if the transition is already executing, meaning that a transition cannot fire more than one token at a time (see Chapter 2, Section 2.4.2). Accordingly, we have also the inequality:

$$S_j^{n+1} \geq S_j^n + \mu_j \quad (C.4)$$

A firing schedule (S_j^n) verifying (C.3) and (C.4) for $n = 1, 2, 3, \dots$ is said to be feasible, as defined in Chapter 5, Section 5.2.2. Let us now assume that n , the number of repetitions, is large enough, so that:

$$n > 1 \quad \text{and} \quad n > M_{ij}^0 \quad \text{for all } i \text{ such that } t_i \in \bar{t}_j$$

Then, (C.3) and (C.4) can be rewritten as:

$$S_j^n \geq \max (S_i^{n-M_{ij}^0} + \mu_i, S_j^{n-1} + \mu_j) \quad (C.5)$$

It should be clear from (C.3) that relation (C.1) determines the earliest firing schedule, such that the firing initiations occur as soon as the transitions are enabled.

In case where n is less than some M_{ij}^0 , we simply do not take into account the corresponding value of i in the determination of the maximum: it means that the corresponding transition t_i does not need to fire in order to enable t_j , because the marking of p_{ij} is still positive (i.e., still contains initial tokens). In particular, (C.1) can be used to compute the instant of the first occurrence of each transition, i.e., the schedule S_j^1 ($j=1,2,\dots,m$), if we adopt the convention that:

$$S_j^0 = -\mu_j \quad \text{for } j = 1,2,\dots,m$$

(where m denotes, as usual, the total number of transitions). In that case, there are only two alternatives:

- either the initial marking of all the input places p_{ij} of transition t_j is strictly positive, implying that transition t_j is enabled at the initial instant and therefore starts firing at $\tau = 0$. Using relation (C.1) for $n = 1$, given the convention, yields effectively:

$$S_j^1 = 0$$

- or there is at least one place p_{ij} with a null marking, i.e., $M_{ij}^0=0$. Then transition t_j will therefore be enabled only after the corresponding transition t_i has fired, which is what

we obtain, using (C.1):

$$S_j^1 = S_1^1 + \mu_1$$

Finally, it turns out that relation (C.1) provides a way to compute the firing schedule (S_j^n) by iteration on n , the number of firing repetitions. However, in order to be able to compute S_j^n , we need to determine S_1^n , for all the transitions t_1 that belong to $\bar{t}_j$ and for which $M_{1j}^0 = 0$ (p_{1j} is any one of the places that is not a resource place). The order in which (S_j^n) for $j = 1, 2, \dots, m$ (number of transitions) should be computed is therefore defined by the partial order between the transitions, as determined by the slices $\bar{T}_1, \bar{T}_2, \dots, \bar{T}_S$ (see Chapter 5, Section 5.2.3). Let us describe now the corresponding algorithm.

C.2 DESCRIPTION OF THE ALGORITHM

The algorithm is described by means of a flow-chart shown on Figure C.2. The analysis of the Flow-chart is straightforward, from the explanations given previously. Let us note that N represent the desired number of simulations, i.e., of firing repetitions. It may also be seen as the total number of inputs processed by the organization. It is emphasized that there are in fact two steps to complete, before using the algorithm:

- (1) determine the slices $\bar{T}_1, \bar{T}_2, \dots, \bar{T}_S$ of the net, following the procedure described in Chapter 5, Section 5.1.2.
- (2) determine $\bar{t}_j$ for each transition t_j , as defined previously.

These steps are easily implemented, using the Incidence Matrix, that characterizes the structure of the Petri Net (see Chapter 2, Section 2.1.2).

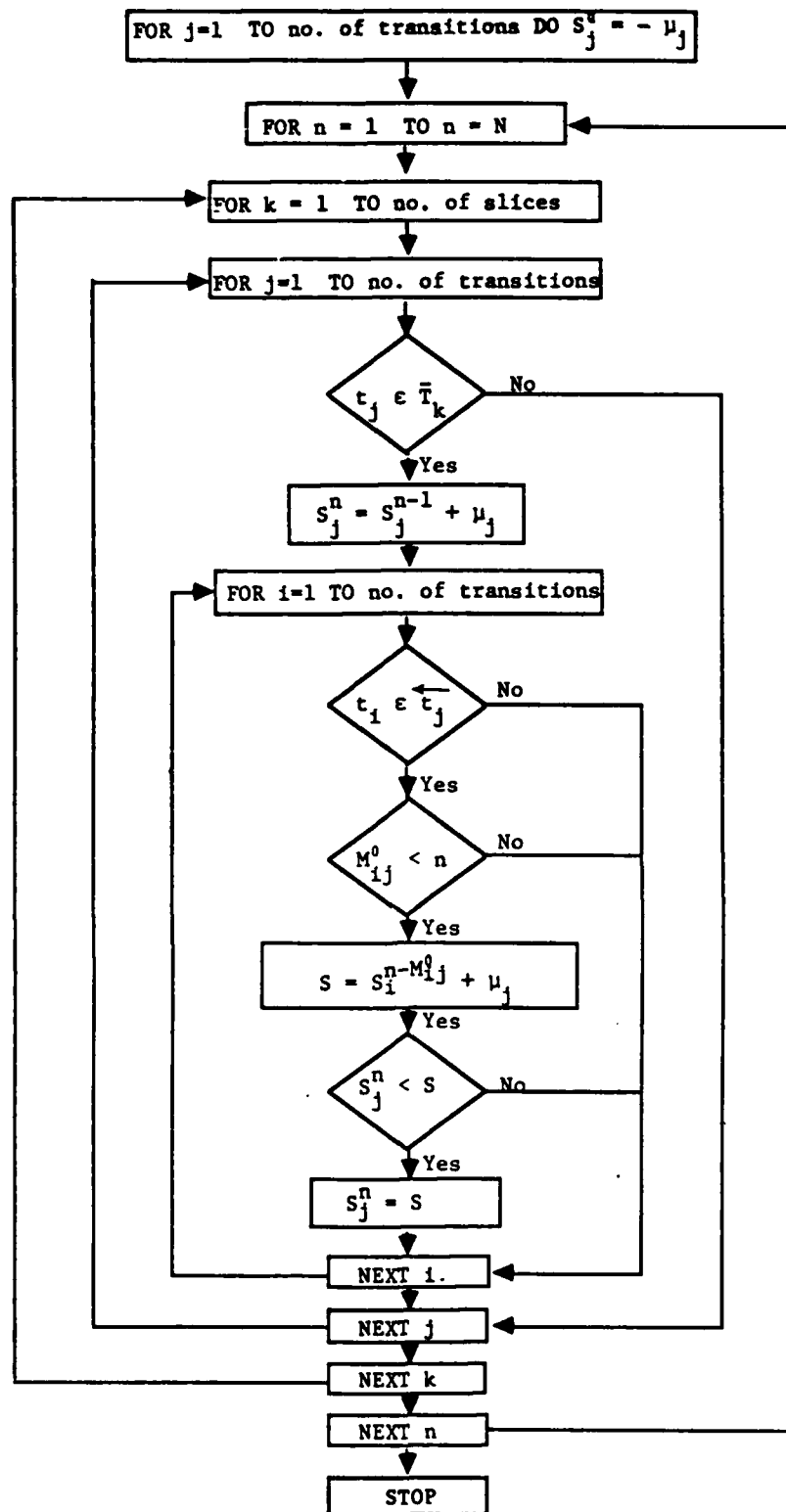


Figure C.2 Flow-Chart of the Algorithm Computing the Firing Schedule

END

12-86

DTIC