MODELING AND EVALUATION OF VARIABLE STRUCTURE ORGANIZATIONS

by

JEAN-MARC MONGUILLET

Ingénieur de l' Ecole Polytechnique (1984)
Ingénieur des Ponts et Chaussées (1986)

SUBMITTED TO THE DEPARTEMENT OF CIVIL ENGINEERING
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

IN TECHNOLOGY AND POLICY

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

December 1987

Signature of Author _____
Departement of Civil Engineering
December 28, 1987

Certified by _____
Dr. Alexander H. Levis
Thesis Supervisor

Accepted by _____
Professor Richard de Neufville, Chairman
Technology and Policy Program

Accepted by _____
Professor Ole S. Madsen, Chairman
Departmental Committee on Graduate Students
Department of Civil Engineering

MODELING AND EVALUATION OF VARIABLE STRUCTURE ORGANIZATIONS

by

JEAN-MARC MONGUILLET

Submitted to the Department of Civil Engineering
on December 28, 1987
in partial fulfillment of the requirements of the degree of
Master of Science in Technology and Policy

ABSTRACT

Variable structure organizations are defined as decisionmaking organizations in which the pattern of interactions between decisionmakers can vary. A quantitative methodology is developed to evaluate their Effectiveness in the achievement of their mission. A model of these organizations using Predicate Transition Nets is presented, in which the decisionmakers are treated as an ordinary resource. The decisionmakers are modeled by tokens that move from one pattern of interactions to the other, depending on designer defined protocols. An example of a three member variable structure organization carrying out an air defense task is presented. In that organization, the Headquarters sets the pattern of interactions of the Field Units according to the characteristics of the incoming signals. Ranges of mission requirements are computed in which this variable organization is the most effective, when compared to corresponding organizations with a fixed structure of interactions.

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

ACKNOWLEDGEMENTS

I wish to express my gratitude to Dr. Alexander Levis for his guidance and professional example.

There is an Indian story about an Englishman who, having been told that the world rested on a platform which rested on the back of an elephant, which rested in turn on the back of a turtle, asked:

-"what did the turtle rest on ?

-"Another turtle.

-"And that turtle ?

-"Ah, Sahib, after that, it is turtle all the way down".

Working with Dr. Levis has been a valuable experience in learning how to stay on the back of the elephant.

I would like to thank Dr. John Brode from Metasoftware Corporation for his helpful suggestions at the early stages of this Thesis, and for his advice on the software Design.

I would also like to thank Lisa Babine for her help and kindness, and for her uninterrupted supply of masa and chocolate cookies.

I thank Jim, Vicky, Jean-Louis, Didier, Stamos, and all the good friends I met in MIT, for their moral support throughout this endeavour. I am especially grateful for the help of Jim, Vicky and Stamos in the use of the various modules of the software package CAESAR.

Finally, I wish to express my gratitude to my family, and Jessica.

This research was carried out at the MIT Laboratory for Information and Decision Systems, with support provided by the Office of Naval Research under Contract No N00014-84-K-0519 (NR 649-003).

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER I

INTRODUCTION

1.1 PROBLEM DEFINITION

Most of the developments in decision and control theory have addressed the problem of analyzing the performance of a given organizational form, or of designing an organization whose performance would meet some specific set of requirements. The models of organization which had been then obtained had always had a fixed structure. Some changes in the topology of the interactions between their components may have been proposed, but they have always remained incremental.

There is indeed a need to investigate the whole set that the variable decisionmaking organizations constitute. They are organizations in which the interactions between the decisionmakers can change, or which can process the same task with different combination of resources. Variable structure organizations could be a possible design solution when no fixed structure organization can meet the requirements of the mission. The concept of variability in such a context could also lead, later on, to the investigation of properties such as robustness or survivability. The modeling of variability in organizations constitutes then another step towards the representation of more realistic decisionmaking organizations.

Three main problems need to be addressed in order to fully assess the property of variability.

First, a framework needs to be developed which will specify the class of organizations under consideration, and which will allow the mathematical formulation of the problem of the comparison of organizational designs. Such a framework should include both fixed and variable structure organizations.

Second, the concept of variability has to be sharpened; a distinction between different types of variability should be made based on which parts of the organization vary, and which do not, and on when they do so. The evaluation tools have then to be adapted to each kind of variability.

Third, a modeling tool needs to be developed to represent variable organizations. It should have high enough modeling power to account for significant features of the organizations, but also convenient enough to provide models that are easy to understand, and figures that are easy to interpret. A compromise between modeling power and illustrative power has therefore to be found.

1.2 THEORETICAL BACKGROUND

A quantitative methodology for the modeling, evaluation, and design of decisionmaking organizations has been developed at the MIT Laboratory for Information and Decision Systems (Boettcher and Levis, 1982; Andreadakis and Levis, 1987; Remy et al, 1987). The organization has been depicted as a system performing a task in order to achieve a mission. The extent to which it does so is assessed by using the formalism of the System Effectiveness Analysis (SEA) methodology (Martin and Levis, 1987).

From a structural point of view, the processing of the task is achieved through the execution of procedures or algorithms that the decisionmakers have. These algorithms are connected together with a relation of precedence which is conveniently represented with Petri Nets (Tabak and Levis, 1985). The internal processing of a given decisionmaker is modeled so that it has a four stage structure, which allows to differentiate the types of interactions that two decisionmakers can have.

The mathematical formulation of the problem of the modeling of variable structure organizations is based on the theory of Predicate Transition Nets, which is an extension of the Petri Net Theory using the language of first order predicate logic (Genrich and Lautenbach, 1981).

1.3 GOALS AND CONTRIBUTION

In this Thesis, the Predicate Transition Net formalism as presented in Genrich and Lautenbach (1981) is adapted to account for the particularities of variable structure organizations: their resources, their patterns of interactions, and their switching protocols. In particular, the connectors are defined in an original manner as sets of combinations of individual tokens, instead of a formal sum of variables. A new formulation of the conflict resolution rules is also proposed, and then applied in the context of the modeling of variable

organizations.

The decisionmaking organizations are then viewed from a new perspective. The types of interactions which can exist between the decisionmakers are first considered without taking into account the identity of the decisionmakers themselves. The latter are represented by individual tokens (instead of subnets of a Petri Net) moving from one interaction to the other, and as such, are treated in the same manner as any other resources needed for the processing of a task. Interactions, resources, and tasks are then modeled independently, and this new way of describing decision making organizations allows the development of a modeling methodology with a modular architecture. By modular is meant that the representation of the basic components of the information processing (interactions, resources, and tasks) is done separately in separate modules, and that modifications in one module can be made without affecting the others.

The System Effectiveness Analysis is extended in order to be applicable to variable structure organizations. The concept of variability is made specific by distinguishing different types of variability, depending on whether the organization adapts its pattern of interactions to changes in the tasks it processes, or in the environment, or in the nature of its components. A Measure of Effectiveness for variable organization is then defined for each case.

The overall procedure is illustrated by an example of a three member decisionmaking organization carrying out an air defense task.

1.4 THE THESIS IN OUTLINE

The Thesis is organized as follows: chapter II defines what is meant by decisionmaking organization and limits the scope of the Thesis. It reviews the main features of System Effectiveness Analysis, and gives to it a mathematical formulation. It adapts these concepts to the case of variable structure organizations.

Chapter III and chapter IV develop the Predicate Transition Net formalism so that it can be used for quantitative modeling. The first of these two chapters presents the primitives used in that formalism, whereas the second one addresses some more advanced topics, such as the problem of conflict resolution, and the linear algebra representation.

Chapter V first illustrates the problems associated with the modeling of variable organizations with Petri Nets with switches, and then integrates the concepts of the previous chapters by developing a modular methodology for the modeling of variable structure organizations. Chapter VI provides illustrative examples of the application of the methodology.

Chapter VII illustrates the whole procedure with an example of a set of three design candidates for a given mission, one of which is variable. It develops a convenient representation of the effectiveness of these candidates, which allows to select the most effective candidate for a specific mission.

Finally, chapter VIII concludes the Thesis by summarizing the results and suggesting some developments for further research.

As it befits a Thesis on organizations with variable structure, an illustration of the articulation of this Thesis in different chapters is provided in Figure 1.1. In accordance with conventions used in such representations, a line from chapter A to chapter B means that chapter A has to be read before chapter B.

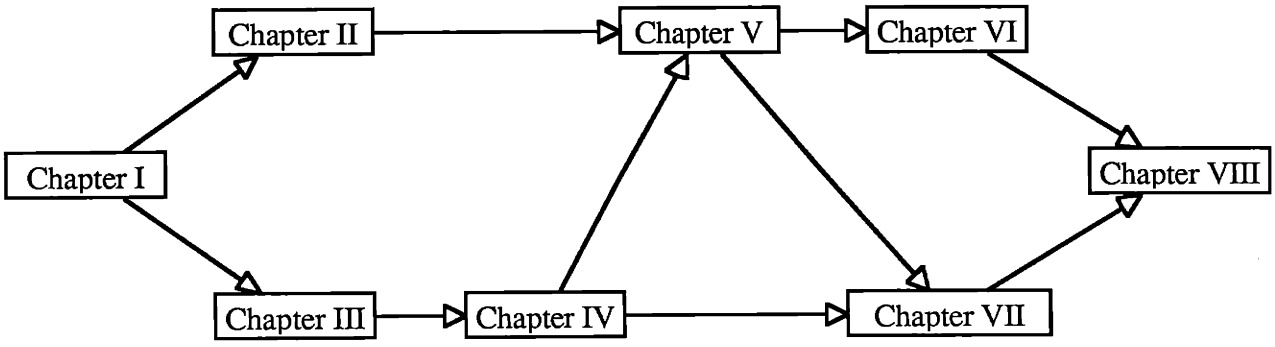


Figure 1.1 Structure of the Thesis.

CHAPTER II

EVALUATION OF VARIABLE STRUCTURE ORGANIZATIONS

This chapter provides a review of the methodology used for the evaluation of decisionmaking organizations (DMO's). First it defines the class of organizations which are considered in this Thesis. The concept of **variability** in organizations is then introduced, and three different types of variability are distinguished, depending on what feature of the organization varies. A relation is established between these three types of variability and the properties of flexibility, reconfigurabilty, and survivability that an organization may exhibit. Finally, the methodology for assessing the effectiveness of DMO's is adapted to the class of variable DMO's.

2.1 DECISIONMAKING ORGANIZATIONS

2.1.1 The System

The concepts which characterize DMO's are introduced in this section.

A **system** (from the Greek 'standing together') is a 'set of objects together with the relationships between them, and between them and the environment, so as to form a whole'. A **set** refers here to a well-defined collection of elements where it is possible to tell beyond doubt whether a given object belongs to the set.

The **objects** are the components of the system. They are considered from a static viewpoint. They are the physical, technological or human components which receive, manipulate, generate, and transmit information. They include the decisionmakers, the physical communication links and the related devices, the computers, displays and other decision-aids.

The **relationships** are the links which tie the objects together. They can be of different kinds, such as symbiotic, if the connected parts of the system can not continue to function separately; they can be synergistic, if the cooperative action produces a greater output than the sum of the outputs of the separate parts alone; they can be redundant when they simply are a

duplication of existing links. These relationships can be thought at three different levels: they may address the physical arrangements of the components, or the relations linking the components, or the rules and protocols describing the interactions between the components. The demarcation line between the relationships and the components is sometimes difficult to sketch: the chips in a computer are an example of that; they are components like any other element of hardware, but they may work like pieces of software and interact with the relationships between components.

The system forms a whole because of these relationships, and also because it carries out a **function**. In the case of an organization (conceived as a system with human components), the extent to which it carries out that function depends mainly on the commonality of goals of its decision makers, or in other words, on the extent to which they constitute a team.

A **boundary** must be drawn which defines what is to be included in the system, and what is to be excluded. It sets the limits of the part of the world which has to be structured. Outside the boundary lie the **environment** and the **context**. The system is included in an environment, which in turn is part of a context. The environment can act upon the system, and the system has some effect on the environment. The context denotes the set of conditions and assumptions within which system and environment exist (Bouthonnier and Levis, 1982) (Fig. 2.1). Drawing the boundaries of the system may mean to isolate it, but certainly not to ignore what lies beyond. The environment is also modeled in the sense that the system has its own representation of it.

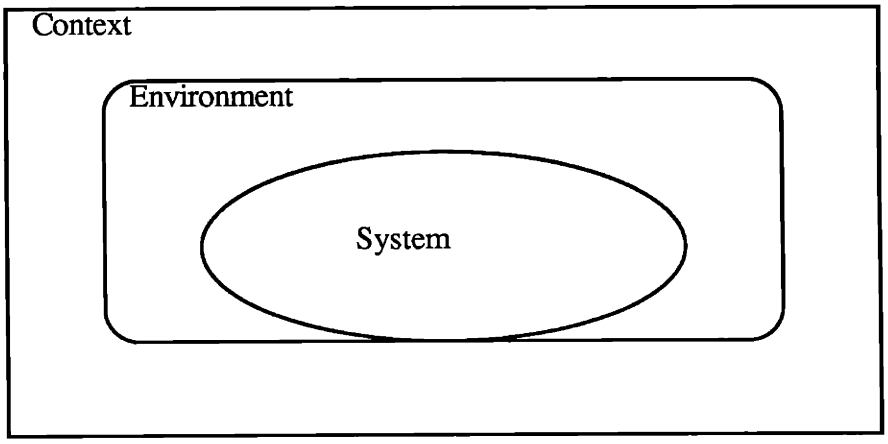


Figure 2.1 System, Environment and Context.

The interface between the system and the environment is composed of the **sensors** and of the **effectors**. The sensors sense the environment and send information as inputs to the system. Their dual parts, the effectors, achieve the responses that the system has selected for the inputs it has processed, and in doing so, may modify the environment by transforming these responses into real and tangible actions. There is in fact no precise distinction between sensors and effectors, except when they are simple devices; but when they are themselves complex subsystems such as aircraft or submarines they may be sensors or effectors depending on the mission they have been assigned. The question of including the sensors and the effectors inside the boundaries of the system is mission-dependent as well. This issue is still debatable and must be considered in each particular case.

2.1.2 Decisionmaking Organizations

The organizations under consideration in this Thesis are restricted to the class of teams of boundedly rational decisionmakers (DM's) (Boettcher and Levis, 1982). Each DM is well trained and memoryless. He processes the information he receives with his algorithms which are deterministic, i.e., their output is a deterministic function of their input.

A DM may possess a set of alternatives, i.e., a set of algorithms among which he chooses one to process his input. In that case, he has knowledge, in a probabilistic sense, of the decision ruling the choice among these algorithms. The probability distribution of that choice is the DM's strategy; it can be conditioned on the input he processes. Each DM's strategy is called an individual strategy.

The organization functions in a hostile environment where the tempo of operations is fast. The DM's have therefore to perform under time pressure. Inputs or observations are generated independently and repetitively, to which the DMO is supposed to respond in a timely manner. Typical examples of such DMO's can be found in the C^3 (Command, Control, and Communications) area, such as a fire support system or an air defense organization.

The organization is described with a set of **parameters**, which are independent quantities specifying the system. For example, system parameters can be communication delays between components, failure probabilities associated with the links or with the elements of the organization, or characteristics of sensors and effectors.

The DMO is performing a **mission** which is also specified by a set of parameters; these parameters can be the tempo of operations, or the type of threats.

2.1.3 Measures of Performance

Measures of Performance (MOP's) are quantities which describe the system properties. In the military environment, an attempt has been made (Rona, 1977) to develop a unified conceptual framework for the evaluation of C^3 systems; under the transformation of these systems into a so-called canonical form, where sensors, decisionmaking units, effectors and boundaries are clearly distinguished, a set of MOP's can be proposed, allowing these systems to be compared, and eventually improved. This unification of the interpretation of the basic concepts (as stated in sections 2.1.1 and 2.1.2) is indeed the basis of the evaluation process for the DMO's.

The MOP's are functions of the system parameters and of the organizational strategy adopted by the organization. If we denote by

$(par_j)_{j=1,...,p}$ the system parameters,
D the organizational strategy,

then m MOP's can be defined as

$$MOP_i = f_i(D, par_1, ..., par_p), \qquad \text{for } i = 1,..., m.$$

Two MOP's will be considered in this Thesis, namely **Accuracy** and **Timeliness**.

Accuracy, denoted by J, is a measure of the degree to which the actual response of the organization to a given input matches the ideal response for that same input.

If we denote by

X the alphabet of inputs x_i: $X = \{x_1, x_2, ..., x_n\}$,
Y the alphabet of outputs y_j: $Y = \{y_1, y_2,..., y_q\}$,
$p(x_i)$ the probability of occurence of the input x_i, with $\sum p(x_i) = 1$,
$y_d(x_i)$ the ideal (or desired) response to x_i,

$y_{aj}(x_i)$, j = 1,...,q, the response that the DMO actually produces,
$C(y_d, y_a)$ the cost of the discrepancy between the ideal and the actual responses,

then a measure of Accuracy of the DMO is:

$$J = \sum_{i=1}^{n} p(x_i) \sum_{j=1}^{p} C(y_d(x_i), y_{aj}(x_i)) \cdot p(y_{aj}(x_i) \mid x_i). \qquad (2.1)$$

J is therefore the expected value of the Accuracy measure, and its expression as stated above will be retained in the sequel.

Timeliness, denoted as T, is the ability to respond to the input with a time delay T_d which is within the allotted time $[T_{min}, T_{max}]$, called the **window of opportunity**.

If we denote by

$T_d(x_i)$ the average processing delay of x_i,
1_Ω the characteristic function on the set Ω,

then a measure of Timeliness of the DMO is:

$$T = \sum_{i=1}^{n} p(x_i) \cdot 1_{[T_{min}, T_{max}]}(T_d(x_i)). \qquad (2.2)$$

The underlying assumption here is that the window of opportunity is independent of the input x_i. In fact, for practical reasons, the expected value of the processing delay will be chosen instead, as a measure of Timeliness, in the sequel:

$$T = \sum_{i=1}^{n} p(x_i) \cdot T_d(x_i). \qquad (2.3)$$

2.1.4. Measures of Effectiveness

The allowable values of the system parameters are defined as a set P in the parameter space Ω_P. The allowable organizational strategies are defined the same way as a set S in the strategy space Ω_{st}. Consequently, when $(\{p_i\}, D)$ varies over its allowable range, $x = (T, J)$ describes a locus in the MOP space Ω_{MOP} called the system locus L_s. Symmetrically, and independently, the mission requirements are translated in terms of requirements on the MOP's, generating the mission locus L_m. The comparison of L_s and L_m leads to **Measures of Effectiveness** (MOE's).

The parameters are usually held constant. The system locus is therefore parametrized by the strategies (Fig. 2.2). Since a given point in L_s can be reached for more than one organizational strategy, the values of the MOP's are not equally probable. A probability distribution f on L_s has to be defined in the MOP space, as

$$\begin{aligned} f:\ L_s &\rightarrow [0,1] \\ x &\rightarrow f(x), \quad \text{where } x = (T, J). \end{aligned}$$

We recall the following elementary notation:

$f: \Omega \rightarrow \Omega'$, an application from Ω to Ω'.
$\mathcal{L}(\Omega)$, the set of subsets of Ω.
$\forall A \in \mathcal{L}(\Omega),\ f(A) = \{y \in \Omega' \mid \forall x \in A, f(x) = y\}$.
$\forall B \in \mathcal{L}(\Omega'),\ f^{-1}(B) = \{x \in \Omega \mid f(x) \in B\}$.
$V(A)$, the volume of A subset of Ω: $V(A) = \int_\Omega 1_A \, d\tau$,
where $d\tau$ is the elementary volume in the set Ω.

Then the application $f: L_s \rightarrow [0,1]$ defined above is characterized by the following property:

$$\forall A \in \mathcal{L}(L_s),\ f(A) = \frac{V(f^{-1}(A))}{V(f^{-1}(L_s))} \tag{2.4}$$

The MOE which will be used throughout this Thesis is defined as follows:

$$E = \int_{L_s \cap L_m} f(x)\, dx. \tag{2.5}$$

In the example of Fig. 2.2, a given organizational strategy is described by a pair (u_1, u_2) in [0, 1] x [0, 1], which constitutes the strategy space Ω_{st}. The function

$$\mu: (u_1, u_2) \rightarrow x(u_1, u_2), \text{ where } x(u_1, u_2) = (T(u_1, u_2), J(u_1, u_2)),$$

realizes a mapping from the strategy space to the system locus in the MOP space. In this example, T_{min} has been set equal to 0, for convenience.

The methodology developed above can be applied then for the explicit computation of the MOE E (eqs. 2.4 and 2.5). For a given mission described in terms of a Timeliness requirement T° and an Accuracy requirement J°, E is computed by evaluating the volume of the strategy space which is mapped into the part of the system locus which meets these requirements, i.e., $L_s \cap L_m$.

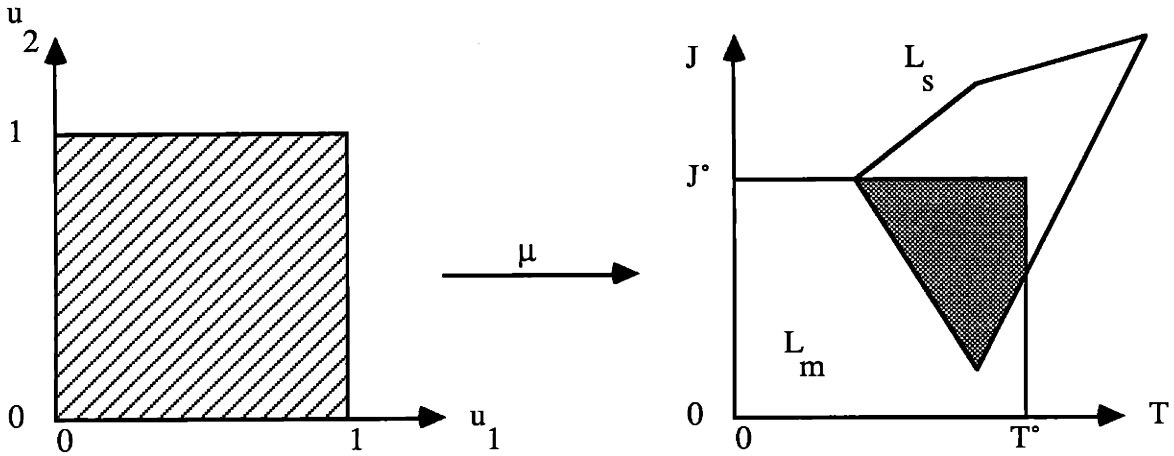
The mathematical expression yielding E is therefore:

$$E(J^\circ, T^\circ) = \int_{\Omega_{MOP}} f(x) \cdot 1_{[0,T^\circ] \times [0,J^\circ]}(x)\ dx \tag{2.6}$$

which gives the following when the integration is done in the strategy space:

$$E(J^\circ, T^\circ) = \frac{\int_{\Omega_{st}} 1_{(\mu^{-1}(L_s \cap L_m))}\, du}{\int_{\Omega_{st}} du} \quad . \tag{2.7}$$

E defines the degree of coverage of the mission requirements by the system capabilities. E does not discriminate between two system loci having the same intersection with L_m (and same probability distribution of this intersection), but different shapes outside L_m (Fig. 2.3). This inconvenience is overcome when the variations of E are investigated when the mission requirements ($T \leq T^\circ$, $J \leq J^\circ$) vary. What is obtained then is a three-dimensional locus (T°, J°, $E(T^\circ, J^\circ)$) called the **diagram of consistency** of the organization. This type of diagram serves as an ultimate tool to evaluate the different organizations which are considered.



Strategy Space

System and Mission Loci

Figure 2.2 System and Mission Loci (for $J_{min} = 0$ and $T_{min} = 0$).

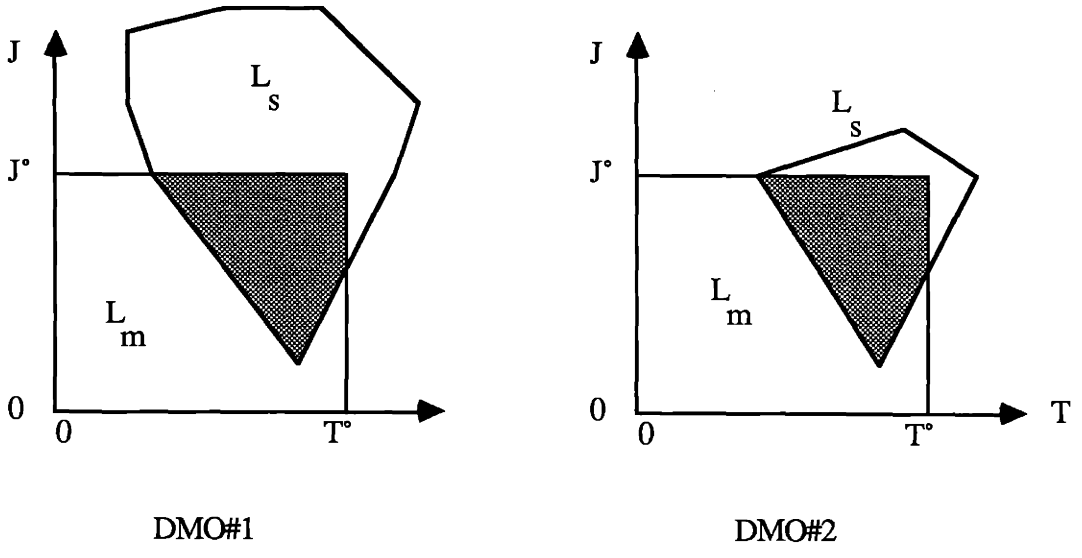


Figure 2.3 Two different System Loci but with identical $L_s \cap L_m$.

2.2 VARIABLE STRUCTURE ORGANIZATIONS

2.2.1 Definitions

A **variable structure decisionmaking organization** (VDMO) is a DMO for which the topology of interactions between the components can vary. Symmetrically, a DMO which has a constant pattern of interactions between its components, i.e., a fixed structure, is called a FDMO.

The relationships which tie the components together have been defined (in section 2.1.1) as being of three different levels: physical arrangements, links between components, and protocols ruling the arrangements of these links. The architecture of the organization allows simply the topology of interactions to vary. The way it does vary is implemented in the protocols themselves, but no matter what structure of interactions is chosen, it is still the same organization.

The rules setting the interactions can be of any kind. We distinguish three types of variability, each corresponding to characteristic properties that a VDMO may exhibit. These properties are dealt with separately in the present section, so that the concept appear clearly.

However, a VDMO may very well have these properties (to some extent) together and simultaneously.

* **Type 1 variability**: The VDMO adapts its structure of interactions to **the input it processes**. Admittedly, some patterns of interactions may be more suitable for the processing of a given input than others. A VDMO which sets its structure of interactions to the one which fits the best for each input is likely to achieve a higher performance.

* **Type 2 variability**: The VDMO adapts its structure of interactions to **the environment**. The MOP's and the locus which have been obtained depend strongly on the characteristics of the environment as perceived by the organization. In particular, for an air defense organization, the type of threats, and their probabilities of occurence have been set to specific values. Now, if the probability distribution of the occurence of the inputs is modified, then the MOP's and the system locus change, and the organization (with the interactions set as before the changes in the environment) may not meet the mission requirements anymore. Other types of interactions may fit better. A VDMO which can adapt to changes in the environment may then have better performance over the possible changes.

* **Type 3 variability**: The VDMO adapts its structure of interactions to **the system's parameters**. The performance of a system degrades strongly when individual components are affected by physical destruction of nodes or sensors, or electronic interference such as jamming of communications. For example, the removal of a link in a DMO with a fixed structure may very well mean that deadlock will occur in the flow of information within the DMO. The organization has ceased to function. A VDMO which is able to adopt a pattern of interactions between the components which remain, after changes in the system parameters, can still have a non-empty system locus. The possibility of adapting the interactions to these kinds of changes works both ways. In other words, the performance can also degrade when a resource or a link is added to the organization, for instance in leading to longer delays or decrease of Accuracy through inconsistency of information.

These three different types of variability can be related to the properties of **Flexibility** and **Reconfigurability**, and to **Survivability**. Survivability is a goal, or a requirement set by the designer of the organization. A DMO is survivable when it can still perform its

missions, under some wide range of changes either in the environment, or in the characteristics of the organization, or in the mission itself. The way to evaluate quantitatively Survivability is outside the scope of the present Thesis. Nevertheless, the requirement that a DMO be survivable can show in the extent to which it is flexible, and in the extent to which it is reconfigurable. Flexibility means that the DMO may adapt to the tasks it has to process, or to their relative frequency. Reconfigurability means that it can adapt to changes in its resources or in its mission(s). Both properties overlap, and their quantitative evaluation clearly falls outside the scope of this work. We will now adopt the framework of the three kinds of variability and investigate the ways to evaluate the extent to which they lead to an improvement of the effectiveness of the DMO, or the extent to which they do not.

2.2.2 DMO's with Type 1 Variability

Recall that a type 1 variable DMO is a VDMO which adapts its interactions, and hence its functionality, to the input it is processing. The set of inputs X is partitioned in classes $(X_i)_{i=1,...,r}$, each of which corresponds to a specific pattern of interactions, called Int#i. Int#i is associated with FDMO#i. The partition of X has the usual properties of a partition, which are stated as follows:

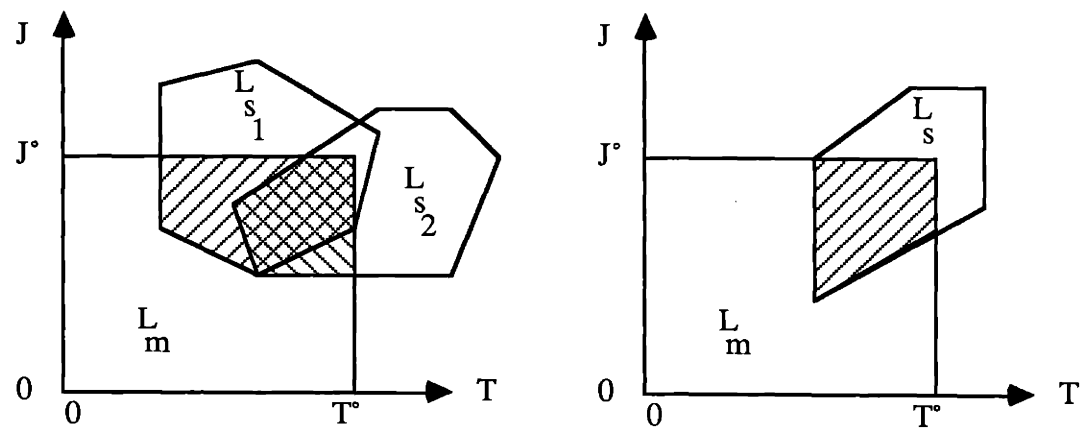
$$X = \bigcup_{i=1}^{r} X_i.$$

$$\forall (i, j) \in \{1,..., r\} \times \{1,..., r\}, \ (i \neq j) \Rightarrow (X_i \cap X_j = \emptyset).$$

Let us assume that the set of inputs that the organization has to process is already partitioned before its processing, for example by a preprocessor. Each input x_i has attached a parameter which indicates which pattern of interactions is required for its processing. x_i becomes $(x_i, Int(x_i))$, where $Int(x_i)$ is an integer in $\{1,..., r\}$. In that case, there is a one to one mapping between the set of classes of inputs $\Omega_{cl} = \{X_1,..., X_r\}$, and the set of possible interactions $\Omega_{int} = \{Int\#1,..., Int\#r\}$. In other words, since Int(x) has a constant value when x describes X_i, the function Int can be also considered as an application from Ω_{cl} to Ω_{int}. The assumption taken simply means that the function Int is **bijective**.

The FDMO#i's and the corresponding VDMO are all candidates to achieve a mission defined in the MOP space by its mission locus. Their system loci are first computed and

depicted in the MOP space (Fig. 2.4). Then the diagram of consistency for each design candidate is constructed. These diagrams allow to evaluate the extent to which each organization fulfills the requirements of its mission, over the possible ranges of these requirements. Finally, for each set of mission requirements, the candidate with the highest effectiveness is selected. Each candidate has therefore associated a range of mission requirements, i.e., a subset of the MOP space, in which it is the most effective design of all the considered candidates. The representation of this partition of the MOP space gives then a convenient tool to select the most effective organization for any set of mission requirements.



FDMO's (1 and 2) VDMO

Figure 2.4 Loci for FDMO's and VDMO.

2.2.3 DMO's with Type 2 Variability

As stated in section 2.2.1, a type 2 variable DMO is a VDMO which adapts its structure of interactions to the environment. The changes in the environment are limited here to changes in the probability distribution of occurence of the inputs (i.e., $p(x_i)$, for $x_i \in X$). We take as an assumption that the DMO has a way to perceive these changes in the environment, which happen on a much larger time scale than the inter-arrival times of the inputs.

The system locus and, as a result, the effectiveness of the DMO are function of this

probability distribution ($p(x_i)$). For a given mission, when ($p(x_i)$) varies over its possible range, the effectiveness E of the DMO changes. If we denote by Ω_{env} the set of possible modeled environments, i.e., the set of the n-dimensional vector of probabilities:

$$p = (p(x_i))_{i=1,...,n},$$

and Ω_{feas} the subset of Ω_{env} encompassing the environments which are actually considered, then the effectiveness E_c of a DMO in a changing environment is defined as the minimum effectiveness of the organization over the possible changes of environment. In formal language, it can be writen as:

$$E_c = \inf_{p \in \Omega_{feas}} E(p) \tag{2.8}$$

where E(p) is the effectiveness of the DMO for an environment characterized by its probability distribution p .

Consider now a set of r DMO's, labelled $(DMO\#i)_{i=1,...,r}$. We define the upper bound of the effectiveness $E^{sup}(p)$ as the maximum of the effectiveness $E_i(p)$ which can be realized by all of the r organizations in the accomplishment of the same mission, for a given p. In other words, this maximum effectiveness $E^{sup}(p)$ is the following:

$$E^{sup}(p) = \sup_{i = 1,...,r} E_i(p) \tag{2.9}$$

where $E_i(p)$ is the effectiveness of DMO#i in the environment p. Since the number r of DMO's is finite, for any environment p, $E^{sup}(p)$ is actually reached by at least one DMO. If it is reached by several DMO's from the set $(DMO\#i)_{i=1,...,r}$, then by convention only the DMO with the lowest index is retained. This index is noted i(p). We select therefore for every environment p the organization DMO#i(p) which has the highest effectiveness:

$$E^{sup}(p) = E_{i(p)}(p) \tag{2.10}$$

This definition partitions Ω_{env} in subsets $(\Omega_{env,i})_{i=1,...r}$, each corresponding to a specific

DMO. The intersection of any two $\Omega_{env,i}$ is the empty set, and the DMO which corresponds to $\Omega_{env,i}$ is DMO#i(p). In other words:

$$\forall\ (i, j) \in \{1,..., r\}x\{1,..., r\},\ (i \neq j) \Rightarrow (\Omega_{env,i} \cap \Omega_{env,j} = \emptyset).$$

$$\forall\ p \in \Omega_{env}, \forall\ i \in \{1,..., r\}, (\ p \in \Omega_{env,i}) \Leftrightarrow (i(p) = i).$$

Some subsets $\Omega_{env,i}$ may of course be empty.

Consider now a VDMO which would adopt the pattern of interaction of DMO#i whenever the environment p lies within $\Omega_{env,i}$. This VDMO would be of Type 2, and its global effectiveness in a changing environment could then be defined as the minimum of its local effectiveness (i.e., its effectiveness for each environment p) when p describes Ω_{env}:

$$E_c(VDMO) = \inf_{p \in \Omega_{env}} E^{sup}(p) = \inf_{p \in \Omega_{env}} \sup_{i=1,...,r} E_i(p) \qquad (2.11)$$

Clearly enough, no other DMO has a higher global effectiveness than the VDMO which has been so defined and constructed.

If the alphabet of inputs is too large, the selection process described above may very well be intractable. In that case, the alphabet of inputs X is partitioned in R classes $(X_i)_{i=1,...,R}$. The probability of occurence of a class X_i is simply the sum of the probability of occurence of the elements of X_i. This probability is denoted by $p(X_i)$. The approach described above is then carried out for changes in $(p(X_i))$, instead of in $(p(x_i))$. The results which it provides can be illustrated in a very convenient way for R = 2 and for R = 3.

Partitioning of X in two classes

When R = 2, only two classes of inputs make the partition of X, namely X_1 and X_2. Changes in the environment are then modeled as changes in (p_1, p_2), where $p_1+p_2 = 1$. The set of possible DMO's is, say, (DMO#1, DMO#2, DMO#3), generically called DMO#i. The System Locus, and as a result the Effectiveness of each of these organizations depend on the environment in which it functions (Fig. 2.5). The function $E(p_1)$ is then plotted (Fig. 2.6),

which allows to select the organization with the highest effectiveness for any value of p_1, i.e., for any value of the parameters (p_1, p_2) describing the environment.

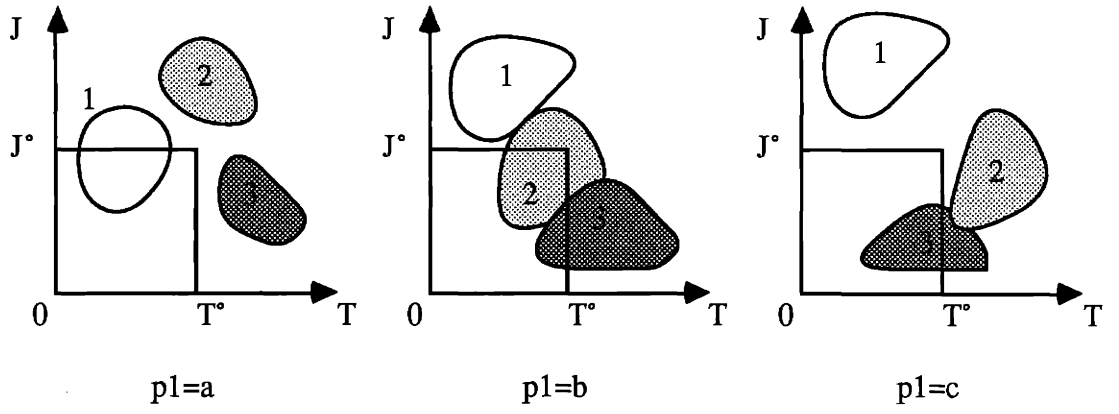


Figure 2.5 Loci for changing environments.

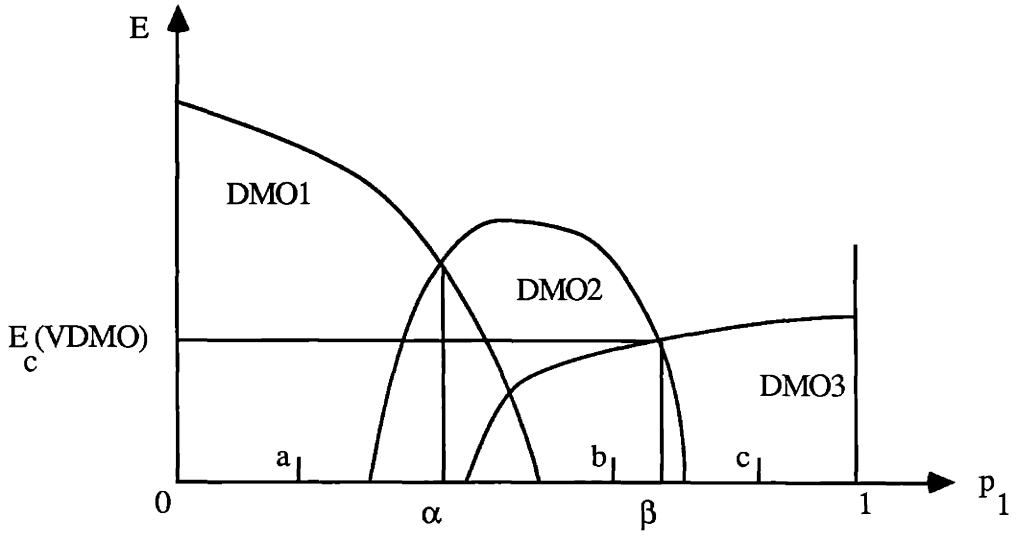


Figure 2.6 Comparative Effectiveness.

Then, assuming that it has a way to perceive these changes in the environment, a variable DMO, adapting to any environment p the pattern of interactions of the DMO#i which is the most effective, would be type 2 variable.

In this example, the most effective DMO's are the following:

$0 \leq p_1 \leq \alpha$: DMO#1. $\alpha < p_1 \leq \beta$: DMO#2. $\beta < p_1 \leq 1$: DMO#3.

Partitioning of X in three classes

When R = 3, i.e., when the set of inputs X has been partitioned in three classes X_1, X_2, and X_3, then the environment has its changes modeled by the values taken by the triplet $p = (p_1, p_2, p_3)$, with $p_1 + p_2 + p_3 = 1$.

The effectiveness E of the different DMO's is then evaluated, for every possible environment p. The design with the highest Effectiveness is then selected, as a function of p.

A convenient way to represent the results of this analysis is to use the barycentric coordinates of a point inside a triangle. A given environment p is represented in a 3-dimensional space by a point M of coordinates (p_1, p_2, p_3). Since $p_1 + p_2 + p_3 = 1$, the locus of M is a triangle (A, B, C) (Fig. 2.7). The triangle (A, B, C) can be represented in a 2-dimensional space, in which the environment p is represented by a point M of barycentric coordinates (p_1, p_2, p_3) inside the triangle.

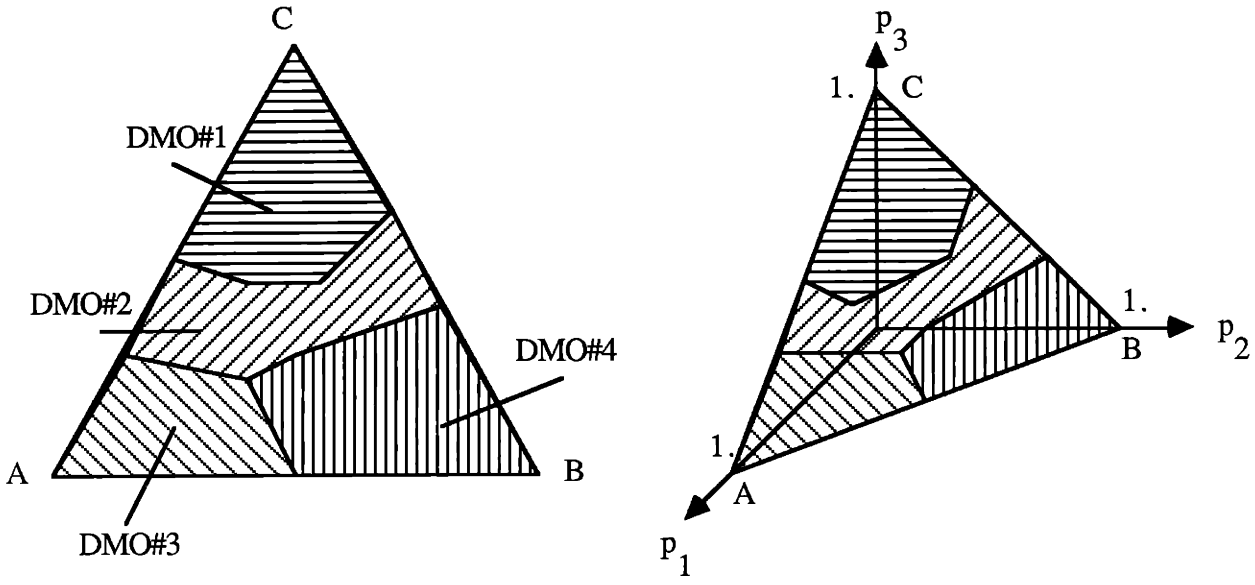


Figure 2.7 Comparative Effectiveness.

The triangle represents the set of possible environments Ω_{env}. The selection of the most effective design leads to the definition of the zones $\Omega_{env,i}$ within the triangle Ω_{env}. In other words, for any environment p in $\Omega_{env,i}$, the candidate DMO#i is the most effective among the candidates functioning in that environment.

Then a variable DMO, which for any environment $p = (p_1, p_2, p_3)$ adopts the pattern of interactions of DMO#i if $p \in \Omega_{env,i}$, would be type 2 variable.

2.2.4 DMO's with Type 3 Variability

Recall that a type 3 variable DMO is a VDMO which adapts the structure of its interactions to changes in the system parameters. Many of these parameters may be actually required to describe the decisionmaking process, and it seems unrealistic indeed to investigate the impact of each of them on the Effectiveness of a given DMO. There is a need therefore to identify some generic parameter types for which a tool for assessing the impact on the Effectiveness of the organization has to be developed.

Two main types of parameters are selected:

- those that describe the characteristics of the communications between the DM's, or between the DM's and the environment, or between the DM's and the decision-aids that they may use in their processing of the task

- and those that characteristize the components of the organization, such as the sensors and effectors, the decision-aids, the decisionmakers themselves, or any other resource used by the organization.

Characteristics of the communications

Two groups of characteristics (or attributes) can be defined, one addressing the time delay induced by the communications, and the others addressing the reliability of the information they carry (Bouthonnier, 1982).

The time delays of the communication links can be critical for the Timeliness of the DMO. They can be affected by the environment, or by the limited capacity of the links.

The delay to transmit a message can be modeled by associating a transmission time to any communication link and multiplying it by a factor k defined as the following:

$$k = 1 / (1 - \alpha)$$

where α is the degree of jamming (Andreadakis and Levis, 1987). The degree of jamming varies between 0 and 1. When α is equal to 0, there is no jamming in the communication links. When α is equal to 0.5, for example, there is a twofold increase in the communication delay. When α is equal to 1, no message can be transmitted through the corresponding link. When a particular link is too jammed, the organization may achieve higher Effectiveness by reconfiguring and performing with a pattern of interactions where the use of that link is minimized.

The same remarks apply to the reliability of the links. It can be affected by the environment (aging, weather) or by the enemy through his electronic warfare capability. If a given link gives unreliable information, the organization may have a higher Effectiveness by changing the pattern of its interactions.

Consider a set of DMO's $(DMO\#i)_{i=1,...,r}$, achieving the same mission. We select a set Ω_{com} of communication attributes which are considered to affect the Effectiveness of the organization the most, or the most likely to vary. The current element of Ω_{com} is denoted as q. As in section 2.2.3, the maximum effectiveness $E^{sup}(q)$ is defined as the upper bound of the Effectiveness which can be realized by all the r DMO's for a given set q of attributes of the communication links. Ω_{com} is partitioned in subsets $\Omega_{com,i}$ which correspond to ranges of the communication attributes for which DMO#i achieve the highest Effectiveness.

A variable DMO which adopts the pattern of interactions of DMO#i when q is in $\Omega_{com,i}$ is type 3 variable, and its global effectiveness can be evaluated as the minimum of $E^{sup}(q)$ when q describes the set Ω_{com} of allowable attributes.

Characteristics of the components

We consider in this section the attributes which describe the type of resources used by the organization, and in particular the decisionmakers themselves. A decisionmaker can be associated with a set of attributes that determine his identity as far as the model of the organization is concerned. In other words, two DM's with the same attributes could be

interchangable. The same argument holds for the characteristics of the resources used by the organization. The impact of variations of these attributes on the Effectiveness of an organization can be treated in the same manner as in the previous section.

Now the removal of a resource (or of a decisionmaker) from the organization is not gradual, as the variations of the attributes already described. The same methodology (partition of set of allowable values for the attributes) is not appropriate in that case, but these changes are still included in type 3 variability.

In the two last sections, a methodology for the evaluation of the effectiveness of variable DMO's has been addressed. The issue of when changes in the environment or changes in the system parameters occur, however, has not been addressed. We assume that the organization can determine such changes, and then trigger the reconfiguration of its interactions. What is of interest here is only the evaluation of such variable DMO's. However, before being evaluated, they need to be modeled; the mathematical tools used to model them are developed in the next two chapters.

CHAPTER III

PREDICATE TRANSITION NETS

Petri Nets are a very convenient tool to model and analyse concurrent and asynchronous processes. Since the information processing in a DMO exhibits these properties, Petri Nets have been used for their modeling (Tabak and Levis, 1985). They can show explicitly the structure of interactions between DM's, and allow their study at different levels of aggregation. This chapter reviews the basic definitions of the Petri Net formalism. More introductory material can be found in Peterson (1981), Brams (1983) and Reisig (1985). The Ordinary Petri Net formalism, however, is unable to treat large nets in a simple way, nor can it represent nets with changing structure. Its grammar has to be extended. One possible extension is the Predicate Transition Net formalism (PrTN). Since this formalism has been presented and developed in the literature under several different forms, all with the same basic ideas, we have chosen a particular approach called High Level Petri Nets (Genrich and Lautenbach, 1981), which seems the most intuitive. This chapter provides an introduction to these nets, but the motivation for their use in the modeling of variable DMO's (VDMO's) will appear in chapter V. Introductory material on the general theory of PrTN's can be found in Brams (1983).

3.1 PETRI NET REVIEW

3.1.1 Basic Definitions

Petri Net

A **Petri Net** is a bipartite directed graph represented by PN = (P, T, I, O), where:

$P = \{p_1, p_2, ..., p_n\}$ is a finite set of n **places**.

$T = \{t_1, t_2, ..., t_m\}$ is a finite set of m **transitions.**

I is a **mapping from PxT to {0, 1}**, corresponding to the set of directed arcs from places to transitions. I(p, t) = 1 means that the place p is connected to the transition t, in the sense that there exists a directed connector from p to t.

O is a **mapping from TxP to {0, 1}** corresponding to the set of directed arcs from transitions to places. O(t, p) = 1 means that place p is an output place of transition t.

An example of a Petri Net is shown in Fig. 3.1.

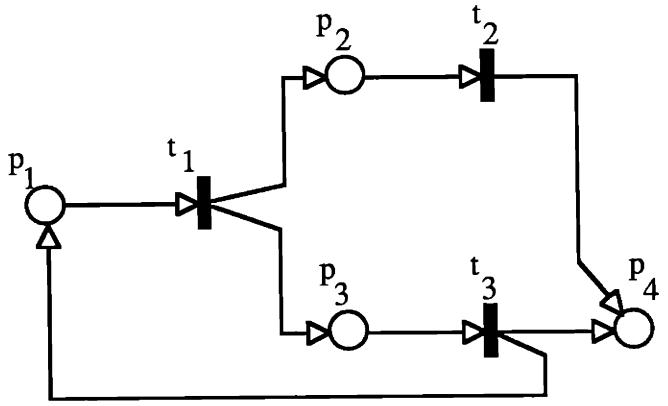


Figure 3.1 Petri Net PN1

In the case of PN1, we have:

$P = \{p_1, p_2, p_3, p_4\}$,
$T = \{t_1, t_2, t_3\}$,

and :

$I(p_1, t_1) = 1$	$I(p_2, t_1) = 0$	$I(p_3, t_1) = 0$	$I(p_4, t_1) = 0$
$I(p_1, t_2) = 0$	$I(p_2, t_2) = 1$	$I(p_3, t_2) = 0$	$I(p_4, t_2) = 0$
$I(p_1, t_3) = 0$	$I(p_2, t_3) = 0$	$I(p_3, t_3) = 1$	$I(p_4, t_3) = 0$

$O(t_1, p_1) = 0$	$O(t_2, p_1) = 0$	$O(t_3, p_1) = 1$
$O(t_1, p_2) = 1$	$O(t_2, p_2) = 0$	$O(t_3, p_2) = 0$
$O(t_1, p_3) = 1$	$O(t_2, p_3) = 0$	$O(t_3, p_3) = 0$
$O(t_1, p_4) = 0$	$O(t_2, p_4) = 1$	$O(t_3, p_4) = 1$

A **Node** is either a place or a transition.

A Petri Net is **Ordinary** when the mappings I and O take their values in {0, 1}. All the Petri Nets we consider in this Thesis are Ordinary.

A Petri Net is **Pure** if and only if it has no self loop, i.e., no place can be both an input and an output of the same transition. PN1 is pure.

We will denote throughout this Thesis the set of integers by $\mathbb{Z}$, and the set of non-negative integers by $\mathbb{N}$.

Marking

A **Marking** of a Petri Net PN is a mapping M from P to $\mathbb{N}$ which assigns a non-negative number of tokens to each place of the net. M is represented as a (nx1)-vector of non-negative integers.

In the example of Fig. 3.1, since none of the places of PN1 contains any token, the marking of PN1 is :

$$M^0 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Firing

A transition t of a Petri Net PN is **enabled** for a marking M if and only if for each place of the net, we have: $M(p) \geq I(p, t)$. In other words, each input place of t must contain at least one token.

When a transition t is enabled, it can **fire**, removing one token from each of its input places, and adding one in each of its output places. The new marking M' reached after the firing of t is defined as follows:

$$\forall p \in P,\ M'(p) = M(p) + O(t, p) - I(p, t) \qquad (3.1)$$

Reachability set

The sequential firing of transitions $t_{i1}, t_{i2}, ..., t_{is}$ is denoted:

$$\sigma_s = t_{is} \cdot t_{i(s-1)} \cdot \ldots \cdot t_{i1}.$$

The set of all the firing sequences which are feasible in PN, with M^0 as an original marking, is called $T^*(M^0)$. When a transition t is involved in a firing sequence σ, it is denoted as:

$$t \in \sigma.$$

The marking M of the net after the firing of a sequence σ from the original marking M^0 is equal to $M = \sigma(M^0)$. Then, given a Petri Net PN with an initial marking M^0, we call **reachability set of M^0** the set of all possible markings of PN reachable from M^0 by some sequence σ of allowable firings of transitions:

$$R(M^0) = \{M \mid \exists\ \sigma \in T^*(M^0), \sigma(M^0) = M\}. \qquad (3.2)$$

If the initial marking of PN1 is: $M^0 = (0, 1, 0, 0)^T$, then the reachability set of M^0 is:

$$R(M^0) = \{M^0, M^1\}, \text{ where } M^1 = (0, 0, 0, 1)^T.$$

3.1.2 Linear Algebra Results

Incidence matrix

The structure of a pure ordinary Petri Net can be represented by an integer matrix C, called the **incidence matrix**, whose elements C_{ij} are:

$$C_{ij} = O(t_j, p_i) - I(p_i, t_j), \text{ for } 1 \le i \le n,\ 1 \le j \le m. \qquad (3.3)$$

C_{ij} can therefore only take the values 0, 1, and -1.

We call C^+ and C^- the following integer matrices: $C^+ = (C^+_{ij}) = (O(t_j,p_i))$, and

$C^- = (C^-_{ij}) = (I(p_i,t_j))$.

The relation between C, C^+, and C^- is then obviously: $C = C^+ - C^-$.

The incidence matrix of PN1 is indicated as follows.

$$C(PN1) = \begin{bmatrix} -1 & 0 & 1 \\ 1 & -1 & 0 \\ 1 & 0 & -1 \\ 0 & 1 & 1 \end{bmatrix}.$$

Firing a transition

The new marking M' reached from M after the firing of a transition t_j enabled by M is:

$$M' = M + C\ N_j \qquad (3.4)$$

where N_j is the (mx1)-firing vector $(\delta_{jk})_{k=1,\ldots,m}$ and δ, the Kronecker symbol.

In the example of PN1, with $M^0 = (0, 1, 0, 0)^T$, if we consider the firing vector $N_1 = (0, 1, 0)^T$, then we have:

$$M^1 = M^0 + C(PN1)\ .\ N_1 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} -1 & 0 & 1 \\ 1 & -1 & 0 \\ 1 & 0 & -1 \\ 0 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

S- invariants

A **S-invariant** is a n-dimensional non-negative integer vector X of the null-space of C^T, i.e., such that :

$C^T X=0$.

Support

The set of places whose corresponding components in X are strictly positive is the **support** of the invariant, noted <X>. The support of an invariant is **minimal** if it does not contain the support of any other invariant but itself and the empty set.

S- components

The **S-component** associated with an S-invariant X is the subnet of PN whose places are the places of X and whose transitions are the input and output transitions of the places of X.

Theorem

X is a S-invariant of PN iff for any initial marking M^0 and for any marking M reachable from M^0, we have:

$$X' M = X' M^0. \tag{3.5}$$

The proof is straightforward, when Eq. (3.4) is invoked.

3.1.3 Properties of Petri Nets

Two properties that Petri Nets can have and which will be relevant in the further development of this Thesis are **Boundedness** and **Liveness.**

Boundedness

A marking M^0 is **bounded** if and only if there exists an integer k which bounds the number of tokens of any place of the net for any marking in the reachability set of M^0, or, in other words, if:

$$\exists\, k \in N, \forall\, M \in R(M^0), \forall\, p \in P, M(p) \leq k. \tag{3.6}$$

A Petri Net is **structurally bounded** iff it is bounded for any initial marking M^0.

PN1 is not bounded since for: $M^0 = (1, 0, 0, 0)^T$, the number of tokens in p_4 can be arbitrarily high.

Liveness

A marking M^0 is **live** (or deadlock free) if and only if for any marking M in the reachability set of M^0, there is at least one transition t which is enabled:

$$\forall\, M \in R(M^0), \exists\, t \in T, \exists\, \sigma \in T^*(M),\ t \in \sigma. \qquad (3.7)$$

A Petri Net is **structurally live** iff it is live for any initial marking M^0.

PN1 is live for $M^0 = (1, 0, 0, 0)^T$, but is not live for $M^0 = (0, 1, 0, 0)^T$.

3.1.4 Petri Nets with Switches

The grammar of ordinary Petri Nets has been extended to take into account the possibility of alternatives in the firing of a transition (Levis, 1984). A new kind of transition is defined, called a **switch**. A switch is a node of the Petri Net which can only be connected to places, which is enabled whenever there is at least a token is each of its input places, and when it fires, puts a token in only one of its output places. The output places of the switch are called the **branches** of the switch.

For example, the switch s_1 in the Petri Net PN2 (Fig. 3.2) has two branches p_1 and p_2.

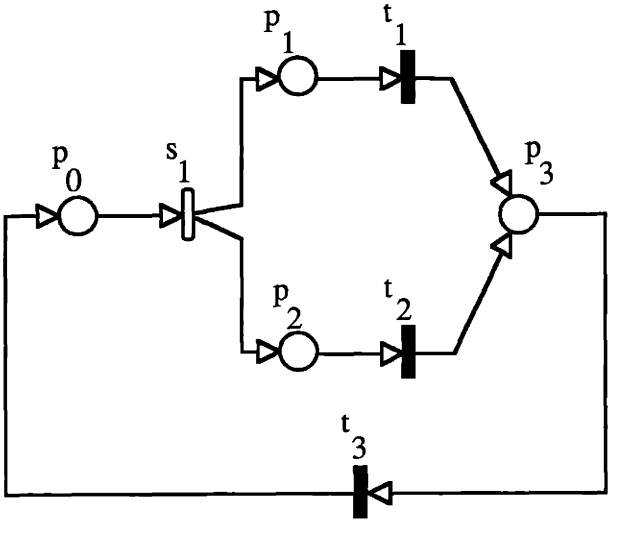


Figure 3.2 Petri Net with switch, PN2.

The decision rules which determine the branch of the switch which is activated in the firing process can be virtually anything. They can be deterministic functions of the input of the switch, or stochastic, i.e., defined as a probability distribution over the set of branches of the switch. They can also be dependent on the state of the entire net.

Petri Nets with switches are represented analytically by an incidence matrix which is (nx(m+s)) dimensional, where n is the number of places in the net, m, the number of transitions, and s, the number of switches.

The Petri Net with switch PN2 (Fig. 3.2) has then the following incidence matrix:

$$C(PN2) = \begin{bmatrix} 0 & 0 & 1 & -1 \\ -1 & 0 & 0 & 1 \\ 0 & -1 & 0 & 1 \\ 1 & 1 & -1 & 0 \end{bmatrix} \begin{matrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{matrix}$$

$$\begin{matrix} t_1 & t_2 & t_3 & s_1 \end{matrix}$$

The incidence matrix accounts only for the topological structure of the net. It does not tell anything about the nature of the decision rules of the switches.

3.2 PREDICATE TRANSITION NETS: PRIMITIVES

As stated in the introduction of this chapter, we shall call **Predicate Transition Nets** (PrTN's) the formalism introduced by its original authors (Genrich and Lautenbach, 1981) under the name of **High Level Petri Nets**. The formalism developed in the present Thesis is slightly different from the one of High Level Petri Nets, as described in the literature. Some specific features, such as the meaning of the connectors, have been added to suit better the systems these nets will be used to model.

In PrTN's, the tokens have an identity: they are objects of more generic classes called variables. They are thus individual tokens, and as such, are the arguments of Predicates, which are associated with places, and of the formulas, which are associated with transitions. The transitions fire according to their built-in formulas.

These nets have a fixed part and a variable part, which are represented separately. The fixed part is an ordinary relational structure, comparable to that of Ordinary Petri Nets. It is called the **support** of the net. The variable part consists of the annotations of the net. The variable relations between individual tokens appear at the places of the net: they are Predicates, or relation symbols. The variable functions, according to which the individual tokens operate, appear at the transitions of the net: the transitions have attached an operator (or formula), or function symbol. Relation and function symbols are formally described in terms of the language of first order predicate logic. This section, however, will not enter in much detail into the formalism, but will describe the relevant concepts at their intuitive level.

PrTN's are then very convenient for the treatment of processes which involve tokens with an identity, distinguish among them, and establish variable relations between them.

3.2.1 Tokens

Definition

Each token may have an **identity**. If it is the case, it is called **individual token**. The set of individual tokens of the net is partitioned in classes called **variables**. A given variable can also receive different names.

Example

For example, we define the variable x as a variable which can take the identities a, b, or c. In other words, the allowable alphabet of the variable x is the set $\underline{x}$:

$$\underline{x} = \{a, b, c\}.$$

The variable x can only take one of these three distinct identities. If a, b, and c are themselves variables, then there would exist some instances in which they could have the same identities. However, as far as x is concerned, they are distinct individual tokens, and are treated that way. Though x has been defined as being allowed to take its identities only from the set {a, b, c}, more than one instance of a given individual token can coexist in the net, or even at the same place in the net.

The variable x can also be called y, or z:

$\underline{x} = \underline{y} = \underline{z} = \{a, b, c\}$.

Unary and n-ary variables

Variables like x are called **unary**. They are variables which consist of only one component. Unary variables can be aggregated in **n-ary variables** which are then represented as follows:

$\underline{x} = \underline{y} = \underline{z} = \{a, b, c\}$.
$\underline{u} = \{\alpha, \beta, \gamma, \delta\}$.
$\langle \underline{x,u} \rangle = \{\langle a,\alpha \rangle, \langle a,\beta \rangle, \langle b,\gamma \rangle, \langle c,\delta \rangle\}$.
$\langle \underline{x,y} \rangle = \{\langle a,b \rangle, \langle a,a \rangle, \langle a,c \rangle\}$.

<x,u> is a 2-ary (or binary) variable made up with the unary variables x and u. Although x and u have had their respective identities defined as elements of sets, the identities that <x,u> can actually take are restricted to the indicated set. Finally, x can be aggregated with itself, in which case two different names for x have to be used.

Note that had the alphabet of <x,u> been restricted to {<a,a>, <b,b>, <c,c>}, <x,x> would have been an acceptable notation. Note also that $\langle x,u \rangle \neq \langle u,x \rangle$.

The reason why tokens can be aggregated together according to some predefined rules is that a means is provided for relating individual tokens and for making them move together within the net in the firing process. Then tokens of different kinds can stay in the same place, and be the arguments of the same predicate. (see section 3.2.2, Places).

The tokens which have no identity, i.e., are indistinguishable, are labeled ¢ (for "no color"). Such tokens are considered as elements of a **0-ary variable**.

3.2.2 Places

Definition

A place p of a PrTN is associated with a n-ary variable x. It may also be associated with a **Predicate** H, which in that case is n-ary as well. The predicate H(x) is a proposition with changing truth value whose **arguments** are the components of the variable x. A place p can only host individual tokens of the same variable x attached to p.

For example, a place p allowing tokens of variable <x,u> is associated with a predicate H(x, u). This Predicate is a relation symbol, which states a property that individual tokens of variables x and u have when they stand together at that particular place. If p contains only one individual token <a, α>, then H(a, α) is true, but H(x, u) is false for any <x,u> such that ($x \neq a$) and ($t \neq \alpha$). If p contains two tokens, namely <a,α> and <a,β>, then H(x, u) is true for each of these two identities of <x,u>, and only for them.

Example

An example of what the predicates can be is the following. Assume that in the formalism of modeling the decision process in a DMO with a Petri Net, the tokens have an identity: they are elements of the alphabet $\underline{x}$. If the information process is modeled by a Petri Net (Fig. 3.3) whose two transitions are respectively labeled as the Situation Assessment stage and the Response Selection stage, then its three places, which can all carry tokens of variable x, are associated with the following predicates:

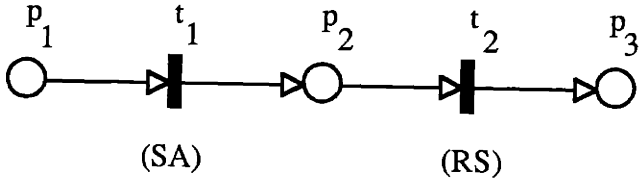


Figure 3.3 Predicates in PrTN

p_1 : $H_1(x)$ = 'The input x is ready to be processed in the SA stage'.

p_2 : $H_2(x)$ = 'The input x is ready to be processed in the RS stage'.

p_3 : $H_3(x)$ = 'The input x has been processed completely'.

If p_1 contains a token a, then $H_1(a)$ is true: the input represented by the token a is ready to be processed by an algorithm in the SA stage.

Places supporting indistinguishable tokens ¢ are just like usual places in an ordinary Petri Net (Fig. 3.4). For completeness, these places are associated with 0-ary predicate; a 0-ary predicate H in a place p has no argument like a variable. H is true whenever there is at least

one token ¢ in p.

Places need not be associated with any predicate at all. The predicates translate the relations between tokens in terms of what these tokens represent and of what the net is modeling. When the structure and behavior of the net are the only concerns, they can be left aside in the analysis of the net.

Marking

The **marking** of a PrTN with n places and m transitions is a n-dimensional vector whose components are the formal sum of the individual tokens present in the places.

For example, a place p of the net containing the individual tokens a and b would have the following marking:

$$M(p) = a + b.$$

Formally, the **Marking M(p) of a place** is an application from $\underline{x}$ (i.e., the alphabet of the variable x) to $\mathcal{N}$, which associates to each element of the alphabet (i.e., each individual token of x) a positive integer.

If $\underline{x} = \{a, b, c, d\}$, then the application M(p) is defined as follows:

$$M(p) : \underline{x} \rightarrow \mathcal{N}, \text{ such that : } a \rightarrow 1, b \rightarrow 1, c \rightarrow 0, d \rightarrow 0,$$

but M(p) is denoted, for convenience, as the symbolic sum M(p) = a + b. For completeness, the marking of a place p which does not contain any token is denoted as being 0:

$$M(p) = 0.$$

The **Marking of the net** is then a n-dimensional vector whose components are the M(p)'s:

$$M(PN) = (M(p_i))^{T}_{i=1,\ldots,n}$$

The Marking of a PrTN is therefore an application M from $(\underline{x}_i)_{i=1,...,n}$ to $\mathbb{N}$, such that:

$$\forall\, i \in \{1,..., n\},\ \forall\, x_i \in \underline{x}_i,\ M: x_i \rightarrow M(p_i)(x_i).$$

M(p) = 2 M(p) = a M(p) = 2a + b M(p) = <a,b>

Figure 3.4 Places and Marking of Places.

3.2.3 Connectors

Definition

A connector is labeled with a formal sum of variables, which indicates the kind of tokens it can carry. It can only support individual tokens of the same given variable.

With the examples of variables and individual tokens which have been defined in section 3.2.1, we have the following possibilities for a label:

"a" The connector carries only individual tokens a, and one at a time.

"x" The connector can carry any individual token of variable x, one at a time.

"2x" The connector can carry two instances of variable x, i.e., two individual tokens at a time provided that they have the same identity. For example, {a, a}, or {b, b} are acceptable. It cannot carry only one individual token.

"x+a" The connector can carry two individual tokens of x, one of which has to be a. It cannot carry a alone.

"x+y" The connector can carry two individual tokens of x without restriction on their

identity whatsoever. For example, {a, a} or {a, b}.

"x+2y" The same as above, with three tokens, two being identical. For example, {a, b, b}.

"¢" Carries tokens without identity.

"x+¢", and "x+u", are not valid expressions.

The labels of the connectors have a formal definition. In order to state it, let us first introduce some notation:

$\underline{x}$: the alphabet of the variable x; $\underline{x}$ is assumed to be finite.
λ: an application from $\underline{x}$ to $\mathbb{Z}$.
$\lambda(\underline{x}) = \{n \in \mathbb{N} \mid \exists\, x_i \in \underline{x},\ n = \lambda(x_i)\}$.
$L^+(x)$: the set of all the applications λ from $\underline{x}$ to $\mathbb{N}$.
conn: generic name for a connector from a node to another node.

An element λ of $L^+(x)$ can also be represented with a symbolic sum (as for a Marking) where the non-negative weighting coefficients are the $\lambda(x)$'s (Eq. 3.8):

$$\lambda = \sum_{x_i \in \underline{x}} \lambda(x_i) \,.\, x_i \qquad (3.8)$$

The **label of a connector** *conn* is a set L_{conn} of elements λ of $L^+(x)$ such that any λ in L_{conn} uses in its symbolic sum representation the same set of weighting coefficients.

The **support** of an element λ of L_{conn}, noted supp(λ), is the set of individual tokens to which it corresponds. Throughout the Thesis, the term *label* is either designating L_{conn} or its aggregated symbolic sum representation (for instance x+y), if it exists.

Examples

A possible label for a connector is the following, where $\underline{x}$={a, b, c, d}:

$\lambda : \underline{x} \rightarrow \mathbf{N}, a \rightarrow 1, b \rightarrow 2, c \rightarrow 0, d \rightarrow 0.$

λ can be expressed as the symbolic sum: $\lambda = a + 2b$. It carries two instances of the individual tokens b together with the individual token a:

$$\xrightarrow{a+2b}$$

The support of λ is then {a, b, b}.

Let us consider the following connectors: $\lambda = a + 2b$, $\lambda' = 2a + b$, and $\lambda'' = a + b$. Since λ and λ' use the same set of weighting coefficients {1, 2} in their symbolic representation, they can be elements of the same set L_{conn}. However λ'' uses a different set of coefficients, {1, 1}, and can not belong to the same L_{conn} as λ or λ'.

Now if $\lambda_a, \lambda_b, \lambda_c, \lambda_d$ are the following applications:

λ_a: $\underline{x} \rightarrow \mathbf{N}, a \rightarrow 1, b \rightarrow 0, c \rightarrow 0, d \rightarrow 0.$
λ_b: $\underline{x} \rightarrow \mathbf{N}, a \rightarrow 0, b \rightarrow 1, c \rightarrow 0, d \rightarrow 0.$
λ_c: $\underline{x} \rightarrow \mathbf{N}, a \rightarrow 0, b \rightarrow 0, c \rightarrow 1, d \rightarrow 0.$
λ_d: $\underline{x} \rightarrow \mathbf{N}, a \rightarrow 0, b \rightarrow 0, c \rightarrow 0, d \rightarrow 1.$

then the set $\{\lambda_a, \lambda_b, \lambda_c, \lambda_d\}$ is the label of a connector which has an aggregated representation, called x:

$$\xrightarrow{x}$$

For completeness, the label of uncolored connectors is also defined within the same formalism: $L^+(\not{c}) = \mathbf{N}$. The label of such a connector is an element m of $\mathbf{N}$ ($\lambda = m$ or, equivalently, $\lambda = m\ \not{c}$).

A place p associated with a predicate whose n arguments are the components of a n-ary variable x is related to its input and output transitions with connectors labeled by subsets of $L^+(x)$. In the example (a) of Fig. 3.5, the place is of an ordinary Petri Net: the connectors

are labeled ¢, and can only carry one uncolored token at a time. The label is an element of $\mathbb{N}$, namely 1, since ¢ has been matched to 1.

In example (b), the place is associated with a predicate whose argument is the variable x. The input connector is labeled x: it can carry any individual token of variable x. If $\underline{x}$ = {a, b, c, d}, as in the example above, then the input connector is labeled by the subset $\{\lambda_a, \lambda_b, \lambda_c, \lambda_d\}$ of $L^+(x)$. However, the output connector is labeled by the subset $\{\lambda_a\}$ of $L^+(x)$, since only the individual token a can leave the place through this connector.

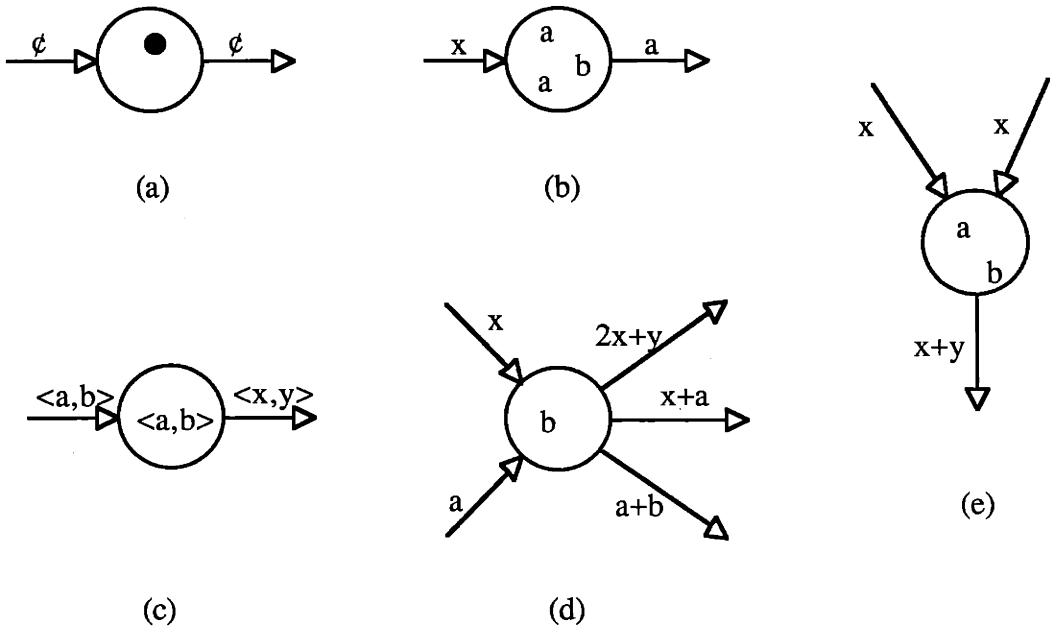


Figure 3.5 Places and Connectors.

In example (c), the place is associated with a binary variable <x,y>. If a predicate is also attached to the place, it is binary as well and its arguments are the components x and y of the binary variable. The marking of the place is <a,b>; as depicted in Fig. 3.5, the input connector can only carry individual tokens <a,b>. However, any kind of individual token of the variable <x,y> can leave the place.

In example (d), the variable associated with the place is x. One input connector can only carry individual tokens a, whereas the other labeled x can carry any individual token of

variable x. The output connector labeled a+b carries only couples of individual tokens equal to {a, b}. The connector labeled x+a supports couples of individual tokens of variable x, one which has to be a. Finally, the connector labeled 2x+y carries 3-uples of individual tokens, two of which are the same, for instance {a, a, b}. If the marking of the place is {a, b}, these two tokens can leave the place either through the connector a+b or the one labeled x+a.

Finally, in example (e), the place is associated with the variable x, and has two input connectors labeled x. For instance, an individual token a can be added to the place through one of them, and another token b can be added to the place through the other. These two individual tokens can leave the place together through the output connector x+y, which carries any couple of individual tokens of variable x.

3.2.4 Transitions

Definition

The transitions may have attached a **logical formula**, built from operations or relations on the components of the variables and on the identities of the components of the individual tokens which are involved in the labels of the input connectors. The formula has a **truth value** which depends on the tokens present in the input places of the transition. When the truth value is True, the transition is **enabled.**

Example

The transition t_1 shown in Fig. 3.6 has only one input connector, which is labeled x. The logical formula attached to t_1 has therefore x and the identities that x can have as its arguments. In the case of Fig. 3.6, the formula is the following:

t_1: ($\exists$ x $\in$ p_1, x $\neq$ a).

If there is no token in p_1, or if there is a token a in p_1, then

t_1 = False.

If there is one token in p_1 different from a, then

t_1 = True,

and the transition t_1 is enabled.

If there are several tokens in p_1 which are different from a, then there is a conflict in knowing for which token the transition t_1 is enabled. This problem will be deferred until section 4.2 on Conflict Resolution.

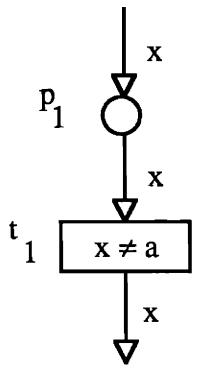


Figure 3.6 Operator associated with a transition.

Operator

The formula indicated in the transition is only a part $\mathcal{P}$ of the actual operator, the part which ignores the quantifiers. It has the form of $\mathcal{P}((\lambda_i)_i)$, where the expression $\mathcal{P}((\lambda_i)_i)$ is a logical statement whose arguments are the terms of the symbolic sum indicated in λ_i, element of $L^+(x_i)$. Its truth value depends on the λ_i's. The whole logical formula is then restated with the knowledge of the allowed variables and of the label of the input places of the transition. Before developing the way this is done, a further definition is needed:

Let λ and λ' be two elements of $L^+(\Omega)$, which is the set of applications from a finite set Ω to $\mathcal{N}$. Then $\lambda \leq \lambda'$ (resp. $\lambda < \lambda'$) if and only if we have:

$$\forall\ \omega \in \Omega,\ \lambda(\omega) \leq \lambda'(\omega).\ (\text{resp. } \lambda(\omega) < \lambda'(\omega)).$$

We denote by:

t: a transition of the net.

$\mathcal{P}$: the part of the operator associated with t which is indicated in it.

I(t): the number of input places of t.

p_i: the input places of t, for i=1,..., I(t).

x_i: the variables (argument) of the predicates $H_i(x_i)$ associated with p_i.

conn(i): the connector from p_i to t.

$\wedge$: operator AND.

$\vee$: operator OR.

Then the complete logical formula is the following:

$$\left(\bigwedge_{i=1}^{I(t)} (\exists\, \lambda_i \in L_{conn(i)},\ \lambda_i \leq M(p_i)) \right) \wedge (\mathcal{P}(\lambda_1,...,\lambda_{I(t)}). \qquad (3.9)$$

which says that each input place p_i of t has a set $supp(\lambda_i)$ of individual tokens, such that the formula $\mathcal{P}$ be true.

This chapter has presented a review of the fundamental concepts of the Petri Net formalism. Predicate Transition Nets have been introduced, and their primitives have been described. This modeling tool was adapted from those found in the literature to suit the problem of this Thesis, which is the modeling of VDMO's. The tokens of a PrTN can be distinguished, and are the arguments of predicates associated with the places which host them. These predicates have their meaning derived from the real system that the net models. The transitions have attached a formula which allows the processes of the individual tokens in the net to be different, depending on their identity. The next chapter will deal with more advanced topics, and in particular with the way the transitions fire, and how the conflicts which may arise during that firing process are solved.

CHAPTER IV

PREDICATE TRANSITION NETS: ADVANCED TOPICS

The development in the preceding section has introduced the primitives of Predicate Transition Nets, namely the concepts of individual tokens and variables, places and Predicates, connectors and Labels, transitions and Operators. In this chapter, these concepts are extended to the study of the firing process: the way the individual tokens are removed from or added to the places is developed; the conditions of enablement of a transition which have already been established in section 3.2.4 are recalled; the rules which are used to solve the conflicts during the firing process are then described. The representation of both the structure and the behavior of the net using the linear algebra formalism is proposed. These ideas can be applied to a very convenient modeling of the switches, as introduced in section 3.1.4, and to the folding of nets, which is a methodology leading to an aggregated representation of large nets exhibiting some symmetry. These two applications will be extensively used in the development of the Thesis. Finally, the concept of time in PrTN's is introduced.

4.1 FIRING PROCESS

4.1.1 Definition

In PrTN's, transitions behave like rules of an inference net, i.e., their generic form is the following:

if (Statement 1)
then
do (statement 2)

(Statement1) determines the conditions of enablement of the transition and has been investigated in the previous chapter: it is the complete logical formula (3.9) associated with the transition.

Now (Statement 2) can be anything indeed: it can be just a simple removal and addition of tokens, taking place in the input and output places of the transition, with no change of identity. It can also do so while changing the identity of the tokens involved in that firing process according to some arbitrary rules. At this point, it seems important to narrow the scope and to consider only cases for which changes of identity of tokens are dictated by the labels of the output connectors. In other words, the only statement which appears in a transition is the part of the operator which corresponds to the left-hand side of the rule described above (i.e., statement 1), without quantifiers (these latter can be deduced from the labels of the input connectors). The right-hand side of the rule embodied in the transition is not shown either (i.e., statement 2), since we make the assumption that it can be inferred from the labels of the output connectors in the course of a matching process which is described in this section.

Let us consider a transition t with a logical formula attached to it, with I(t) input places p_i, and O(t) output places p_j. If the logical formula (i.e., the left-hand side of the rule) has a truth value equal to True for some combination of λ_i, $i \in \{1,..., I(t)\}$, then t is enabled. We suppose that all conflicts have been solved in this selection process (see section 4.2), and that only one λ_i is selected for each p_i. The extension of the methodology to the case of several connectors λ_i (i.e., several combinations of individual tokens) being selected simultaneously, does not require new concepts or tools, only cumbersome notation. Then the individual tokens corresponding to the support of the λ_i's are removed from the input places whereas other individual tokens are added to the output places of t, according to the labels of its output connectors.

Let λ and λ' be two applications from Ω to $\mathbb{N}$, i.e., two elements of $L^+(\Omega)$. We define the following operations on $L^+(\Omega)$:

$$\forall\ (\lambda,\lambda') \in L(\Omega)^2, \forall\ \omega \in \Omega, \quad (\lambda + \lambda')(\omega) = \lambda(\omega) + \lambda'(\omega).$$
$$\forall\ z \in \mathbb{N}, \quad (z.\lambda)(\omega) = z.\lambda(\omega).$$

If any two elements λ and λ' in $L^+(x)$ are such that: $\lambda \leq \lambda'$ (see section 3.2.4), then the operation subtraction of λ from λ' can be defined as:

$$\forall\ \omega \in \Omega, \quad (\lambda' - \lambda)(\omega) = \lambda'(\omega) - \lambda(\omega).$$

The application $\lambda'' = \lambda' - \lambda$ is also an element of $L^+(x)$.

When the transition t fires, the marking M of the net becomes M'. M'(p) is equal to M(p) if and only if p is neither an input place, nor an output place of t. If p is an input place of t, M'(p) is deduced from M(p) by the removal of the support of the connector (see section 3.2.3) which has been selected. For:

$t, I(t), p_i, x_i, conn(i), \lambda_i$

as previously defined, we have:

$$\forall\, i \in \{1,..., I(t)\},\ M'(p_i) = M(p_i) - \lambda_i. \qquad (4.1)$$

We denote then by:

O(t) the number of output places of t.
p_j the output places of t, for j = 1,..., O(t).
x_j the variables (arguments) of the predicates H_j associated with p_j, i.e., $H_j(x_j)$.
conn(j) the connectors from t to p_j.

The components of the variables involved in the labels of the input connectors are then matched to those of the output connectors conn(j). Their identities, which have been selected in the enablement process, are transferred to the components of the output variables.

The information carried by the components of the individual tokens which are not matched, and therefore not transferred, is lost. The transition t acts for them as a sink. Conversely, the terms of the output variables which are not matched have their identities generated as indicated in the label of the connector. In that case, t behaves like a source of information.

The matching process allows to select an element λ_j in $L_{conn(j)}$, for any j = 1,..., O(t). Then the corresponding individual tokens in $supp(\lambda_j)$ are added to the output place p_j of t according to the following relation:

$$\forall\, j \in \{1,..., O(t)\},\ M'(p_i) = M(p_i) + \lambda_j. \qquad (4.2)$$

4.1.2 Examples

Some examples of transitions, places, and connectors are shown in Fig. 4.1.

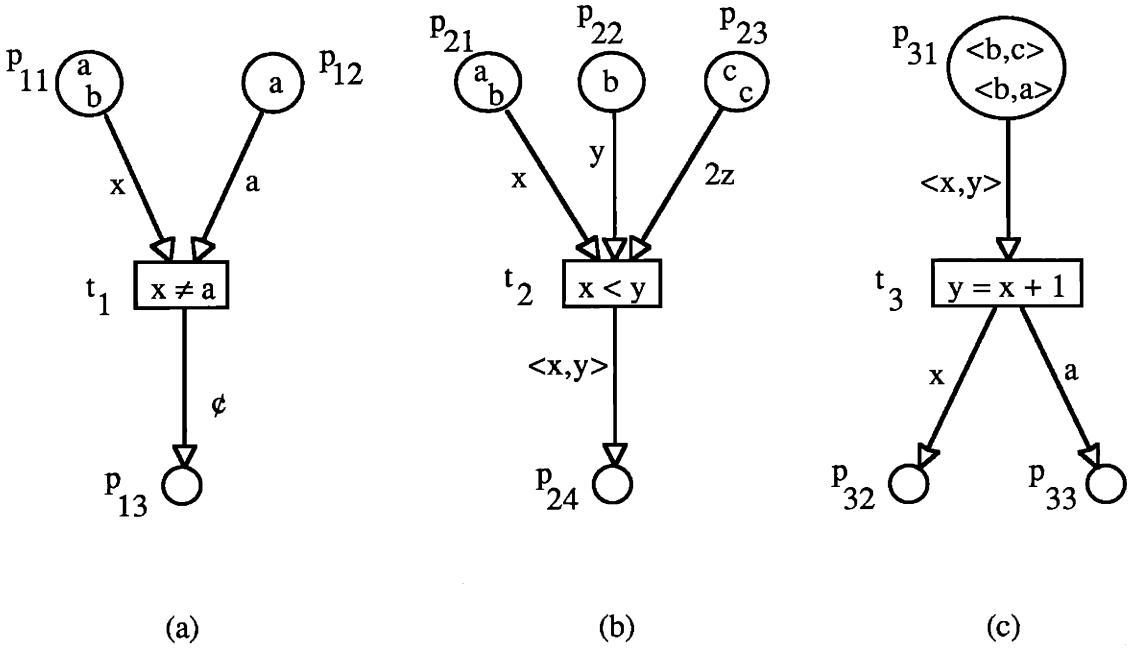


Figure 4.1 Firing Process: examples.

Example (a)

In example (a), the complete logical formula associated with t_1 is displayed as follows:

t_1: $(\exists\, x \in p_{11})\,(\exists\, y \in p_{12})\,((y = a) \wedge (x \neq a))$.

The initial marking M^0 of the net is such that: $M^0(p_{11}) = a + b$, $M^0(p_{12}) = a$, and $M^0(p_{13}) = 0$. We denote also by:

conn(1), the connector from p_{11} to t_1,
conn(2), the connector from p_{12} to t_1,
conn(3), the connector from t_1 to p_{13},
and $\underline{x} = \{a, b, c, d\}$.

The label of the connector conn(1) is the set $\{\lambda_a, \lambda_b, \lambda_c, \lambda_d\}$, whose elements are the following:

λ_a: $\underline{x} \rightarrow \mathbb{N}$, a$\rightarrow$1, b$\rightarrow$0, c$\rightarrow$0, d$\rightarrow$0.
λ_b: $\underline{x} \rightarrow \mathbb{N}$, a$\rightarrow$0, b$\rightarrow$1, c$\rightarrow$0, d$\rightarrow$0.
λ_c: $\underline{x} \rightarrow \mathbb{N}$, a$\rightarrow$0, b$\rightarrow$0, c$\rightarrow$1, d$\rightarrow$0.
λ_d: $\underline{x} \rightarrow \mathbb{N}$, a$\rightarrow$0, b$\rightarrow$0, c$\rightarrow$0, d$\rightarrow$1.

The label of the connector conn(2) is the set $\{\lambda_a\}$.

$\mathcal{P}$ is defined as the following boolean operation:

$$\forall\, \lambda \in L_{conn(1)},\ \forall\, \lambda' \in L_{conn(2)},\ \mathcal{P}(\lambda, \lambda') = (\lambda \neq \lambda_a).$$

In other words, $\mathcal{P}(\lambda, \lambda')$ is True whenever λ is not equal to λ_a, regardless of what the actual value of λ' is.

The complete logical formula attached to t_1 is given by Eq. (3.9), and if we denote by M a generic marking of the net, it is the following:

$$(\exists\, \lambda_1 \in L_{conn(1)},\ \lambda_1 \leq M(p_{11})) \wedge (\exists\, \lambda_2 \in L_{conn(2)},\ \lambda_2 \leq M(p_{12})) \wedge (\mathcal{P}(\lambda_1, \lambda_2)).$$

For the marking M^0, the formula displayed above is True since we have:

$\lambda_1 = \lambda_b$, $\lambda_b \in L_{conn(1)}$, and $\lambda_b = b \leq M^0(p_{11}) = a + b$.
$\lambda_2 = \lambda_a$, $\lambda_a \in L_{conn(2)}$, and $\lambda_a = a \leq M^0(p_{12}) = a$.
$\mathcal{P}(\lambda_1, \lambda_2) = P(\lambda_b, \lambda_a)$ = True, since $b \neq a$.

t_1 is therefore enabled, and can fire. The new marking M' of the input places p_{11} and p_{12} of t_1 is given by:

$$\begin{aligned} M'(p_{11}) &= M(p_{11}) - \lambda_1 \\ &= (a + b) - (b) \\ &= a. \end{aligned}$$

and

$$M'(p_{12}) = M(p_{12}) - \lambda_2$$
$$= a - a$$
$$= 0.$$

Now since the output connector conn(3) is labeled $\not\in$, all the information carried by conn(1) and conn(2) is lost. No matching and transferring of terms and variables is achieved in that case. A token is generated by t_1 according to the label of conn(3), which is $\not\in$. With the notation which has been adopted so far, the label λ_3 of the connector conn(3) is $\lambda_3 = 1$. Then the new marking of p_{13} is:

$$M'(p_{13}) = M(p_{13}) + \lambda_3$$
$$= 0 + 1$$
$$= 1, \quad \text{since } M(p_{13}) = 0.$$

In summary, the transition t_1 is enabled for the marking M and the firing process changes M into a new marking M' as follows:

$$M(PN) = (a + b, a, 0)^T \quad \rightarrow \quad M'(PN) = (b, 0, 1)^T.$$

Example (b)

In example (b), a partial order relation (R) has to be defined on the set $\underline{x}$. Recall that: $\underline{x} = \{a, b, c, d\}$. (R) is simply defined as being the lexicographic order. In other words, (R) is the only partial order relation on $\underline{x}$ which satisfies the following:

$$a < b, \quad b < c, \quad c < d.$$

The complete logical formula associated with t_2 is then:

$$t_2: (\exists\, x \in p_{21})\, (\exists\, y \in p_{22})\, (\exists\, z \in p_{23})\, (x < y).$$

The logical statement which appears in t_2 involves two terms, x and y, and the partial order relation (R). In the enabling process and for the initial marking considered in Fig. 4.1, x is matched successively to a, and b, whereas y is matched to b, and z to c. The only combination (x, y, z) for which the logical formula is True is: x = a in p_{21}, y = b in p_{22}, and

$z = c$ in p_{23}. Although the variable z does not appear in the logical statement without quantifiers, it does appear in the formula as a whole, precisely through these quantifiers: the label of the input connector from p_{23} to t_2 is precisely 2z. t_2 is then enabled.

When t_2 fires, a is removed from p_{21}, b is removed from p_{22}, and two instances of c, from p_{23}. x and y, with their respective values, are matched into the terms of <x,y>, the label of the output connector. z is lost in the matching process. Finally, an individual token <a,b> is added to p_{24}. Some information has been lost in that process, and the variables x and y have been aggregated.

Example (c)

In example (c), an operation needs to be defined on $\underline{x}$: $x \rightarrow y = x + 1$. In that particular case, x + 1 is defined as follows:

$b = a + 1.$
$c = b + 1.$
$d = c + 1.$
$a = d + 1.$

The complete logical formula associated with t_3 is then:

t_3: $(\exists \langle x,y \rangle \in p_{31})\ (y = x + 1)$

The transition t_3 is then clearly enabled for <x,y> = <b,c>. Then the term x of <x,y> is matched into an unary variable, and an individual token of variable x, namely b, is put in p_{32}. The term y does not appear in the labels of the output connectors, thus the information it contains is lost. Whenever the transition fires, since the output connector allows only individual tokens a, a token a is generated and added to p_{33}. In that case, some information has been lost, some has been generated, and the components of a binary variable <x, y> have been taken apart.

4.2 CONFLICT RESOLUTION

The development of the preceding section and of the preceding chapter left aside the issue of the resolution of conflicts, which is addressed here. This issue arises after the enablement

process, and its aim is to derive from the set of possible combinations of tokens which enable the transition, the subset of these combinations which will actually participate in the firing process and will be removed from the input places.

Conflicts in Ordinary Petri Nets only happen when a place has two (or more) output transitions. In that case, a token in that place enables each of these transitions, but only one can fire. In PrTN's, however, conflicts can be found in two different areas:

- In the selection in a given input place of the set of tokens which will participate in the firing process. Indeed, since the tokens are not indistinguishable any more, it matters which individual token will actually participate in the firing process, when the same transition is enabled for more than one possible token.

- In the selection of the transitions which will actually fire, if several of them are output nodes of the same place, and enabled for the marking of their common input place. This is the kind of conflict occuring in Ordinary Petri Nets.

The first type of conflicts arises in every firing of transitions of a PrTN, provided that more than one token is present in their input places. The second type happens only when one place has more than one output transition. In that case, however, the two types of conflicts may arise simultaneously.

4.2.1 Token Selection

One input place

Let us first consider a transition t with only one input place p_1 (Fig. 4.2). The variable x is the argument of the predicate associated with p_1, and has still its set of values equal to {a, b, c, d}. The operator in the transition t is named Op(x), without further explanation. Assume that the marking M of the net is such that:

$$M(p_1) = a + b + c + d.$$

In that case, the enablement process first scans the set of possible combinations of tokens in p_1 (combination according to the label of the connector), and then determines a subset

$\Omega_{enab}(M, p_1)$ of those for which t is enabled (i.e., Op(x) is true).

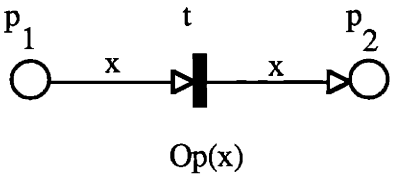


Figure 4.2 Token selection: one input place.

The selection within that subset of the combination of tokens which will actually participate in the firing process is then carried out by a rule $\mathfrak{R}$ of conflict resolution, determined by the designer. This rule $\mathfrak{R}$ can be anything, and the designer of the net has full freedom to implement any rule he can think of. Some examples follow:

* $\mathfrak{R}$ = random: an individual token is selected at random in $\Omega_{enab}(M, p_1)$. It, and only it, is removed in the firing of t.

* $\mathfrak{R}$ = Sup: an order relation is defined on the set $\underline{x}$ and can then be extended to the set of linear combinations of values of x. The rule $\mathfrak{R}$ is then to select the maximal element of $\Omega_{enab}(M, p_1)$.

* $\mathfrak{R}$ = Sup_n: this is a variant of the preceding relation, and selects at most the n first elements of $\Omega_{enab}(M, p_1)$ as ordered by the order relation previously defined.

* $\mathfrak{R}$ = Inf: $\mathfrak{R}$ selects the minimal element of $\Omega_{enab}(M, p_1)$.

The set of combinations of individual tokens which are selected by $\mathfrak{R}$ is then given by the transformation of $\Omega_{enab}(M, p_1)$ under $\mathfrak{R}$, leading to the set $\Omega_{fire}(M, p_1)$ of combinations of tokens which will actually be removed from the input place p_1 of t:

$$\Omega_{fire}(M, p_1) = \mathfrak{R}(\Omega_{enab}(M, p_1)). \qquad (4.3)$$

Several input places

The transition t has I(t) input places p_i, each associated to a Predicate $H_i(x_i)$ (Fig. 4.3). The connector from p_i to t is labeled conn(i), as a subset of $L^+(x_i)$. The transition t has

associated with it an operator Op whose arguments are the terms involved in conn(i). The application of the operator Op to the markings $M(p_i)$ determines a subset of combination of individual tokens. Each combination in that subset enables t. This subset is called Ω_{enab}. It is a I(t)-dimensional vector, whose components are the sets Ω^i_{enab} of individual tokens enabling t in place p_i. This is denoted as:

$$\Omega_{enab} = \left[\Omega^i_{enab}(M, p_i) \right]_{i=1. \ ..., \ I(t)} \tag{4.4}$$

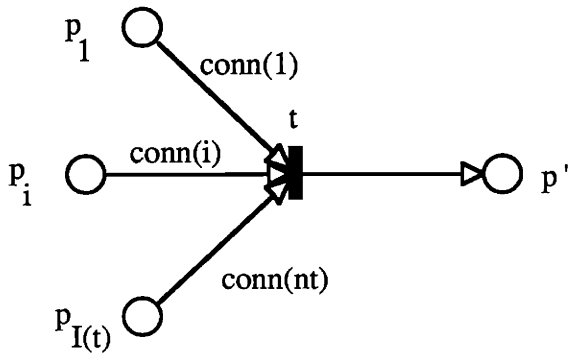


Figure 4.3 Token selection: several input places.

Now the application on this set of the conflict resolution rule $\mathfrak{R}$ provides the set of tokens ready to be removed from the input places during the firing of t. As in the case of a single input place, the rule $\mathfrak{R}$ can be of different types, but here it has to be multi-dimensional: it has to take the markings of p_1 through $p_{I(t)}$ into account. The tokens which will eventually leave the p_i's actually correspond to the set Ω_{fire} deduced from Ω_{enab} by the application of the rule $\mathfrak{R}$:

$$\Omega_{fire} = \mathfrak{R} \ (\Omega_{enab}),$$

i.e.,

$$\Omega_{fire} = \mathfrak{R} \left(\left[\Omega^i_{enab}(M, p_i) \right]_{i = 1, \ ..., \ I(t)} \right) = \left[\Omega^i_{fire}(M, p_i) \right]_{i = 1, \ ..., \ I(t)} \tag{4.5}$$

4.2.2 Transition Selection

We only consider the case of a place p being input place to two transitions t_1 and t_2 (Fig. 4.4). The same methodology addresses the case of a place being input to more than two different transitions.

The place p is associated with a predicate H(x). The connector between p and t_1 (resp. between p and t_2) is conn(1) (resp. conn(2)). For a given marking M, we call $\Omega_{enab}(t_1)$ (M, p) and $\Omega_{enab}(t_2)$ (M, p) the sets of combinations of individual tokens in p for which t_1 or t_2 is enabled. These sets are actually subsets of conn(1) and conn(2).

If these two sets of combinations of tokens are not disjoint, then an individual token belonging to the intersection of these sets can be fired either by t_1 or by t_2, but not by both. This conflict has to be resolved. Just as in the previous section, there are many ways to resolve it: we present three of them, only the last of which will be used in this Thesis.

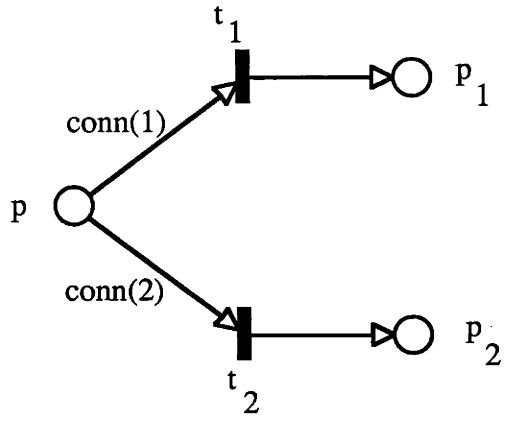


Figure 4.4 Transition selection: one input place.

We call $\mathfrak{R}_1$ (resp. $\mathfrak{R}_2$) the rule which resolves the conflict among the possible tokens in p which enable t_1 (resp. t_2); we call $\mathfrak{R}'$ the rule which solves the conflict among the tokens in p which enable both t_1 and t_2. $\mathfrak{R}'$ partitions by some means this intersection set in two subsets, one being directed to t_1, Ω^1_{fire} and the other to t_2, Ω^2_{fire}. The conflict can be solved by applying $\mathfrak{R}_1$ and $\mathfrak{R}_2$, and then $\mathfrak{R}'$. The final sets of tokens ready to go is then:

$$\left(\Omega^1_{fire}(M, p), \Omega^2_{fire}(M, p)\right) = \mathfrak{R}'\left[\mathfrak{R}_1\left(\Omega^1_{enab}(M, p)\right), \mathfrak{R}_2\left(\Omega^2_{enab}(M, p)\right)\right] \qquad (4.6)$$

The net is guaranteed never to reach a deadlock, as far as the place p is concerned, if and only if the following property ($\mathfrak{P}$) is true (we call M^0 the initial marking):

$$(\mathfrak{P}): \quad \forall\, M \in R(M^0),\ \Omega^1{}_{fire}(M, p) \cap \Omega^2{}_{fire}(M, p) = \emptyset. \qquad (4.7)$$

This property can be very difficult to prove. We can even say that in the general case, it is unprovable, since it would require the knowledge of the entire reachability set of the marking M^0. And the Predicate Transition Nets have been proven to be undecidable (Brahms, 1983).

The underlying idea of the two other ways to automate the resolution of this conflict is then to weaken the property ($\mathfrak{P}$) and to give only a sufficient condition for the non occurence of deadlocks.

The second way then is to have ($\mathfrak{P}$) true without the resolution rules $\mathfrak{R}_1$ and $\mathfrak{R}_2$. Only $\mathfrak{R}'$ applied on the total marking of p will guarantee that there will be no conflict.

$$(\Omega^1_{fire}(M, p), \Omega^2_{fire}(M, p)) = \mathfrak{R}'\left[\ \Omega^1_{enab}(M, p), \Omega^2_{enab}(M, p)\ \right] \qquad (4.8)$$

In that case, the property ($\mathfrak{P}$) becomes ($\mathfrak{P}'$) as follows, where the sets $\Omega^1{}_{fire}$ and $\Omega^2{}_{fire}$ are deduced from $\Omega^1{}_{enab}$ and $\Omega^2{}_{enab}$ by the application of the rule $\mathfrak{R}'$:

$$(\mathfrak{P}'): \quad \forall\, M \in R(M^0),\ \Omega^1{}_{fire}(M, p) \cap \Omega^2{}_{fire}(M, p) = \emptyset. \qquad (4.9)$$

However, the property ($\mathfrak{P}'$) has the same complexity as ($\mathfrak{P}$), since the knowledge of the reachability set of M^0 is still required.

The third way to solve the conflict is to turn ($\mathfrak{P}$) into a sufficient only condition by making it independent of the initial marking, and of the rules $\mathfrak{R}_1$, $\mathfrak{R}_2$, and $\mathfrak{R}'$:

For any marking M of the net, we have:

$$\begin{cases} \Omega^1_{fire}(M, p) = \Omega^1_{enab}(M, p) \\ \Omega^2_{fire}(M, p) = \Omega^2_{enab}(M, p) \end{cases} \quad (4.10)$$

The property ($\mathcal{P}$) becomes therefore ($\mathcal{P}$"):

$$(\mathcal{P}''): \quad \forall\, M(p) \in L^+(x),\ \Omega^1_{fire}(M, p) \cap \Omega^2_{fire}(M, p) = \emptyset. \quad (4.11)$$

In other words, the operators of t_1 and t_2 are such that if t_1 is enabled by a combination of tokens in p_1 then t_2 is not enabled for that combination.

This last property ($\mathcal{P}$") is a lot easier to guarantee than ($\mathcal{P}$) or ($\mathcal{P}$'), and the nets which are considered in this Thesis are all such that ($\mathcal{P}$") be verified. This approach for conflict resolution can be easily extended to the case of several places having in common several output transitions. We will not proceed any further in that area, since the concepts are the same, only the notation becomes more cumbersome. An example of application of the property ($\mathcal{P}$") as a conflict resolution rule is presented in section 4.4.1 for the representation of switches.

4.3 LINEAR ALGEBRA

4.3.1 Definition

Linear algebra plays an important role in net theory because the structure of the net can be easily represented by a matrix, and because the firing process of a transition has a linear representation. The representation with matrices is a convenient tool to investigate the structural properties of the net (e.g., connectivity) as well as the topological structure of the set of nets which are considered (e.g., lattices, meet and join operations) (Remy and Levis, 1987). An incidence matrix for Predicate Transition Nets could be easily formulated. Unfortunately, a set of properties comparable with the structural and topological ones for Ordinary Petri Nets can not be derived from the manipulation of such incidence matrices, since they do not show the notation of the net, i.e., neither the Predicates nor the Operators which actually determine the behavior of the net. These matrices take only into account the

fixed part of PrTN's (see section 3.2).

A totally different area of investigation lies in the dynamics of the net, i.e., the study of the firing processes which can occur in the net and of the evolution of the marking of the net. The study of the behavior of Ordinary Petri Nets provides some results concerning the reachability set of a marking, its liveness, its boundedness, etc. This investigation is greatly facilitated by the existence of a linear representation of the firing process of a transition, as described in section 3.1.2. The presentation of a similar linear representation of the static structure and of the firing process for Predicate Transition Nets is the purpose of this section.

We assume that all the Petri Nets considered in this Thesis are **Pure**, i.e., a place cannot be input and output of the same transition at the same time.

The **incidence matrix** of a Pure Predicate Transition Net with n places and m transitions is a (nxm) dimensional matrix (C) whose components $C_{i,j}$, i = 1,..., n, and j = 1,..., m, are the following:

$$C_{i,j} = \begin{cases} -L, & \text{where L is the label of the connector from } p_i \text{ to } t_j \\ L, & \text{where L is the label of the connector from } t_j \text{ to } p_i \\ 0, & \text{otherwise} \end{cases}$$

The knowledge of C allows then to construct the structure of the net. The incidence matrix is particularly easy to handle and manipulate when the labels of the connectors can be represented by linear combinations of terms instead of simple sets L_{conn}. When this is the case, and when the operators associated with the transitions consist only of the sequence of quantifiers (i.e., $\mathcal{P}$ = True in Equation 3.9), then the structure and the behavior of the net are completely described by its incidence matrix. The PrTN has then the property of **transparency**.

4.3.2 Example

Let us consider the example of the PrTN of Fig. 4.5. In that net, the variable x still denotes the set of values {a, b, c, d}.

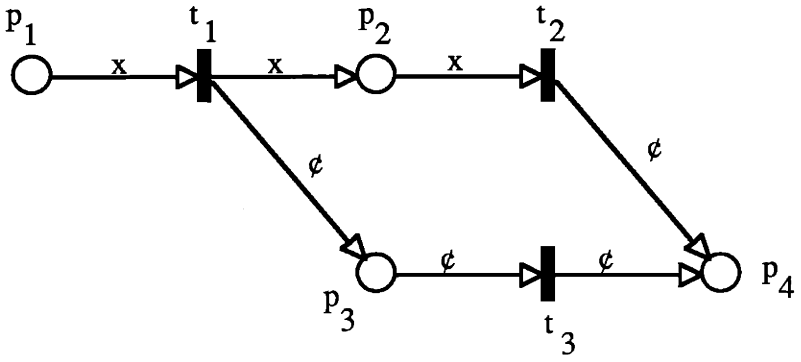


Figure 4.5 Predicate Transition Net PrTN1.

The incidence matrix of the net PrTN1 is the following:

$$C(PrTN1) = \begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \\ p_4 \end{array} \begin{array}{c} \begin{array}{ccc} t_1 & t_2 & t_3 \end{array} \\ \begin{bmatrix} -x & 0 & 0 \\ x & -x & 0 \\ ¢ & 0 & -¢ \\ 0 & ¢ & ¢ \end{bmatrix} \end{array}$$

A connector carrying uncolored tokens can be denoted as either ¢ or 1.

Let us consider now the examples of nets shown in Fig. 4.1. Their respective incidence matrices C_a, C_b, and C_c are the following (with the underlying natural ordering of the nodes):

$$C_a = \begin{bmatrix} -x \\ -a \\ ¢ \end{bmatrix}, \qquad C_b = \begin{bmatrix} -x \\ -y \\ -2z \\ <x,y> \end{bmatrix}, \qquad C_c = \begin{bmatrix} -<x,y> \\ x \\ a \end{bmatrix}.$$

4.3.3 Firing Process

We introduce the linear algebraic formulation of the firing process through the simple example of PrTN1 in Fig. 4.5. The original marking of PrTN1 is M^0 where

$$M^0 = (a + b , 0 , 0 , 0)^T.$$

The transition t_1 is enabled for both tokens a and b, and no other transition is enabled in PrTN1. A conflict in the selection of tokens in p_1 has to be solved. If for instance the conflict resolution rule $\mathfrak{R}$ associated with the place p_1 is $\mathfrak{R}$ = Random, and if the token selected by $\mathfrak{R}$ is a, then when t_1 fires it removes the individual token a from p_1. The token b is not affected in that process. The new marking M^1, reached when t_1 has fired, is then the following:

$$M^1 = (b , a , \not{c} , 0)^T.$$

The relation between M^1 and M^0 is then the following:

$$M^1 = M^0 + \begin{bmatrix} -a \\ a \\ 1 \\ 0 \end{bmatrix}$$

We define then the vector Δ as follows:

$$\Delta = \begin{bmatrix} -x \\ x \\ 1 \\ 0 \end{bmatrix}$$

The value of the vector Δ for x = a is denoted $\Delta:S_a$, where $S_a = \{a\}$. More generally, if S is a set of individual tokens, $\Delta:S$ is the set of values for Δ when the variables involved in Δ

take the values of the individual tokens elements of S. Then we get

$$M^1 = M^0 + \Delta : S_a .$$

On the other hand, we have:

$$\Delta = \begin{bmatrix} -x \\ x \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} -x & 0 & 0 \\ x & -x & 0 \\ 1 & 0 & -1 \\ 0 & 1 & 1 \end{bmatrix} . \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = C(PrTN1) . \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = C(PrTN1) . F$$

where we denote by F the firing vector which corresponds to the firing of t_1. Therefore

$$\Delta : S_a = C(PrTN1) . F : S_a .$$

Then, we have

$$M^1 = M^0 + \left[C(PrTN1) . \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \right] : S_a$$

$$= M^0 + C(PrTN1) . \left(\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} : S_a \right)$$

This leads then to:

$$M^1 = M^0 + C(PrTN1) . F_a , \qquad \text{where} \quad F_a = F : S_a ,$$

which is a similar relation to the one obtained in the case of Ordinary Petri Nets. The difference lies in the firing vector F_a which is considered as a function of the variables x_i and which carries the information about what individual tokens are selected. It outlines the fact

that the transition has different colors, in the sense that it behaves like a set of transitions of an Ordinary Petri Net, from which set only one has been activated and fired by the firing vector F_a. This need not be the case in general, and depending on the rules chosen for conflict resolution, more than one Ordinary transition in a colored transition can be activated by the same firing vector, in which case several individual tokens would be removed simultaneously.

4.4 APPLICATIONS

In this section, two applications of the formalism which has been developed so far are presented. They will be used extensively in this Thesis. One deals with the representation of switches, as introduced in the Petri Net formalism in section 3.1.4, with Predicate Transition Nets. The other treats the folding of nets, which is a powerful tool for obtaining an aggregated version of large nets.

4.4.1 Switch Representation

Let us consider an ordinary Petri Net PN with a switch s with two branches and one input place. The decision rule of the switch s is given by a strategy u which can take the values 0 or 1 (Fig. 4.6). If a token is present in the place p_1 then s is enabled and can fire. If $u = 0$, a token is added to p_{11}. If $u = 1$, it is added to p_{21}.

The subnet which consists of the nodes $\{t_0, t_1, s, p_{11}, p_{21}\}$ together with their connectors is then replaced by a Predicate Transition Net (Fig. 4.7). The transition t_0 receives an uncolored token ¢ but when it fires, it puts an individual token from variable u in p_1. The variable u is defined as the following set: $u = \{0, 1\}$. The way t_0 assigns a color to the incoming token is done in accordance with some rule implemented in it. It can do this assigning at random, or as a function of the attributes of the uncolored token which has been removed from p_0.

The conflict of selection of a transition (t_{10} or t_{20}) does not occur in p_1, since the operators associated with these two transitions are mutually exclusive: either $u = 0$, or $u = 1$. For the variable u, either one output transition of p_1 is enabled, or the other, but never both, nor none. For example, if t_0 puts an individual token 0 in p_1, then t_{10} is enabled and fires, adding an uncolored token to p_{11}. The transition t_{10} (and t_{20} as well) acts like a

sink for the variable u. The flow of tokens after p_{11} and p_{21} is identical to that of the original Ordinary Petri Net.

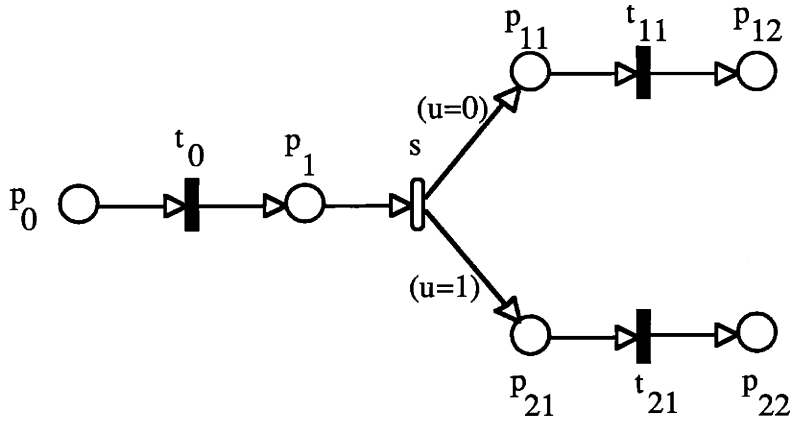


Figure 4.6 Petri Net with switch PN.

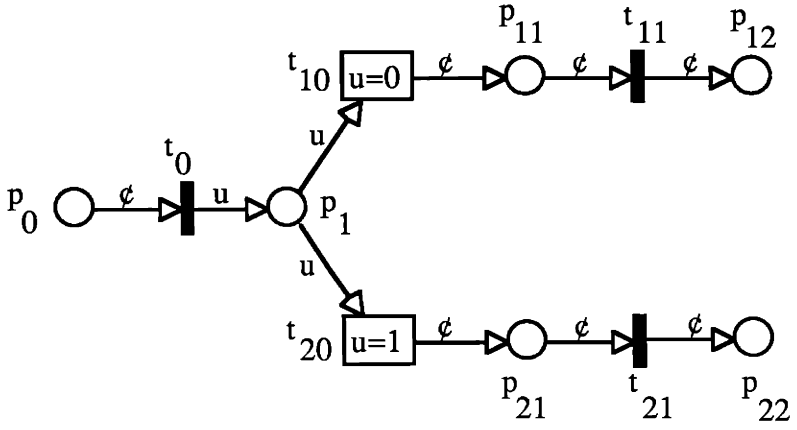


Figure 4.7 Predicate Transition Net equivalent to PN.

The subnet $\{t_0, p_1, s, p_{11}, p_{21}\}$ has therefore been removed, and replaced by another subnet $\{t_0, p_1, t_{10}, t_{20}, p_{11}, p_{21}\}$ without any other change in the rest of the net. The two nets keep the same behavior. Petri Nets with switches are thus subsumed in PrTN's. This facilitates the study of the effects of the decision rules of the switches by re-expressing the problem in terms of conflict resolution and truth tables of the logical operators associated

with the colored transitions.

4.4.2 Folding of Nets

Predicate Transition Nets can also be used to produce simpler, more aggregated, and still workable representations of Ordinary Petri Nets which exhibit some properties of symmetry. A very simple example of that is an Ordinary Petri Net PN which is made of n identical subnets PN0, each of which is connected the same way to the same input place p_0 and to the same output place p_1 (Fig. 4.8).

We define then a variable x as being the set: $\underline{x} = \{1,..., n\}$. Each subnet of type PN0 is colored with a specific color taken from the set $\underline{x}$. A Predicate Transition Net PN1 is defined as the net which has the same structure as PN0 but where the connectors are all labeled x. p_0 and p_1 are connected to PN1 with connectors labeled x as well. Then the obtained net, PrTN, describes the same system, and has the same behavior as PN (Fig. 4.9).

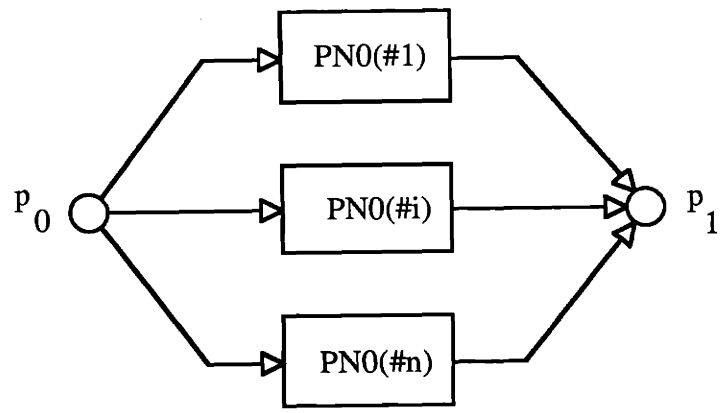


Figure 4.8 A symmetrical Petri Net PN.

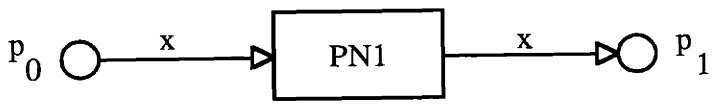


Figure 4.9 PrTN equivalent of PN.

Petri Nets may have other properties of symmetry than the one of being made of identical subnets. A famous example of other possible aggregations of a net is described in the literature as the Philosophers' problem (Brams, 1983). We consider here a simple example which is derived from that and which contains the same ideas in the aggregation and folding process.

The Petri Net of Fig. 4.10 is symmetrical (subnets of nodes indexed by a and by b). Furthermore, it embodies a potential conflict in p_1, which has two output places t_a and t_b. The net is folded through the following mechanism:

-differenciation of the tokens of the original marking.

-labeling of the corresponding connectors.

-folding of each pair of symmetrical nodes into a single one.

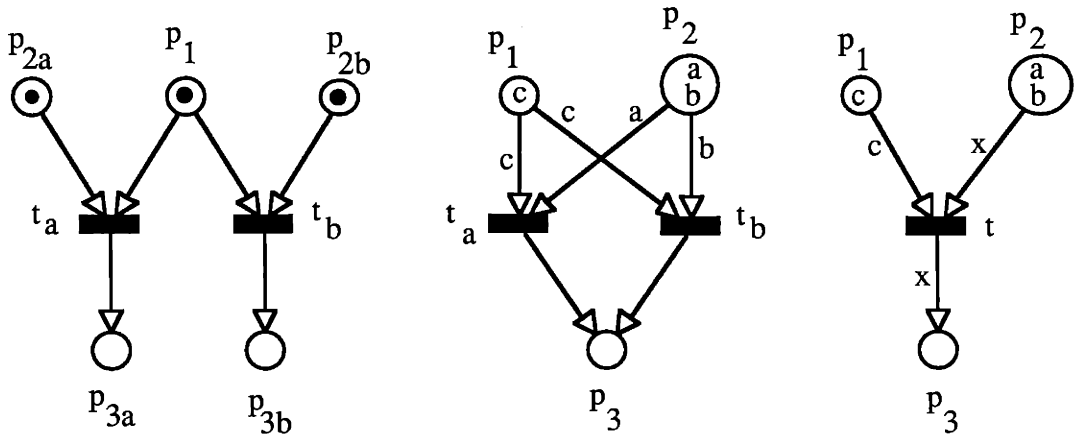


Figure 4.10 Folding of a net.

In this example, the tokens in places p_1 (resp. p_{2a}, p_{2b}) are labeled as individual tokens c, (resp. a, b) of a variable x defined as the following set: $\underline{x} = \{a, b, c\}$. In the firing of t_a or t_b the identity of the token which will be removed from p_2 is not important. For that reason, the connector from p_2 to t is labeled x. On the other hand, since p_1 means that a conflict is present in the firing of t_a and t_b, the identity of the individual token c has to appear in the label

of the connector from p_1 to t. The output connector of t is labeled x for the same reason as for p_2.

The two incidence matrices of the original net (C_1) and of the final Predicate Transition Net (C_2) are the following:

$$C_1 = \begin{array}{c} p_1 \\ p_{2a} \\ p_{2b} \\ p_{3a} \\ p_{3b} \end{array} \begin{array}{c} \left[\begin{array}{cc} -1 & -1 \\ -1 & 0 \\ 0 & -1 \\ 1 & 0 \\ 0 & 1 \end{array}\right] \\ \begin{array}{cc} t_a & t_b \end{array} \end{array}; \qquad C_2 = \begin{array}{c} p_1 \\ p_2 \\ p_3 \end{array} \begin{array}{c} \left[\begin{array}{c} -c \\ -x \\ x \end{array}\right] \\ t \end{array}.$$

4.5 TIME IN PREDICATE TRANSITION NETS

The introduction of time in the Petri Net formalism allows to model and analyse quantitatively real time processes (Hillion, 1986). In that formalism, a processing time is associated either with the places, or with the transitions. Since it has been proven that the two are equivalent, we choose to associate the processing time with the transitions.

A Timed Petri Net is therefore the pair (PN, μ) where PN is an Ordinary Petri Net, with n places and m transitions, and μ a real function from the set of transitions to $\mathbb{R}^+$, the set of non-negative real numbers:

$$\mu : \{t_1, \ldots, t_m\} \rightarrow \mathbb{R}^+, \; t_i \rightarrow \mu(t_i).$$

The Petri Net is assumed to have a clock indicating the current time τ.

The transition t_i takes $\mu(t_i)$ units of time to fire. It means that when t_i is enabled, the relevant tokens are removed from its input places at the current time τ, and will be added to

the output places at time $\tau + \mu(t_i)$. Between τ and $\tau + \mu(t_i)$, the transition t_i can be enabled again, but cannot fire. It has to wait at least until $\tau + \mu(t_i)$ to start removing another combination of tokens from its input places.

The function μ can have its value in a set of random variables, instead of $\Re^+$. In that case, the transitions have associated a stochastic processing time. The macro-transition, consisting of a switch together with its branches whose transitions have constant real processing times has attached a stochastic processing time: the probability of occurence of a value within the set of possible processing times match exactly the probability that a branch is activated. A switch itself can have a non zero processing time, depending on the modeling assumptions which have been made.

The extension of the concept of time to Predicate Transition Nets is straightforward: each transition may have attached a processing time which is a function of the particular combination of individual tokens which have been selected in the enablement and firing processes.

CHAPTER V

MODELING METHODOLOGY FOR VARIABLE DMO'S

In chapter II, the definition of three types of variable DMO's was presented, and for each of them a methodology for their evaluation was developed. What was needed at that point was a modeling methodology for these organizations, regardless of the type of variability they exhibit: this is the goal of the present chapter. In that perspective, this chapter introduces first the Petri Net model of the internal processing of information of a decisionmaker (Tabak and Levis, 1985). It then outlines the disadvantages and the complexity of a representation of variability in DMO's by Petri Nets with Switches. The Predicate Transition Nets, which have been presented in chapter III and chapter IV, are shown to be a much better tool to account for the concept of variability. A step-by-step procedure for the modeling of VDMO's is then developed. An example of a three member organization with type 1 variability illustrates that methodology. Some more examples of decisionmaking organizations with type 2 and type 3 variability are provided in chapter VI.

5.1 THE DECISIONMAKER MODEL

5.1.1 The Four Stage Model

The Petri Net formalism has been found to be very convenient for describing the concurrent and asynchronous characteristics of the processing of information in a decisionmaking organization. The internal information processing which takes place in any decisionmaker has been modeled by a subnet with four transitions and three internal places. A simplified version of this so-called **four stage model** is shown in Fig. 5.1.

This model allows to differentiate among the outputs and the inputs of the decision maker, and to describe the types of interactions which can exist between two decisionmakers.

The decisionmaker receives an input signal x from the environment, from a preprocessor, from a decision-aid, or from the rest of the organization. He can receive one input to the **Situation Assessment** stage (or SA) at any time. He then processes this input x with a specific algorithm which matches x to a situation the decisionmaker already knows. He

obtains an assessed situation z which he may share with other DM's. Symmetrically, he may also receive at this point other signals from the rest of the organization. He combines the information with his own assessment in the **Information Fusion** stage (IF), which is an algorithm which provides him with his final assessment of the situation, labeled z'. The next step is the possible consideration of commands from other DM's which would result in a restriction of his set of alternatives for generating the response to the input. This is the **Command Interpretation** stage, or CI. The outcome of the CI stage is a command v which is used in the **Response Selection** stage (RS) to produce the output y. y is the response of the decisionmaker, and he sends it to the actuators of the organization (see section 2.1.1), or to other DM's within the DMO.

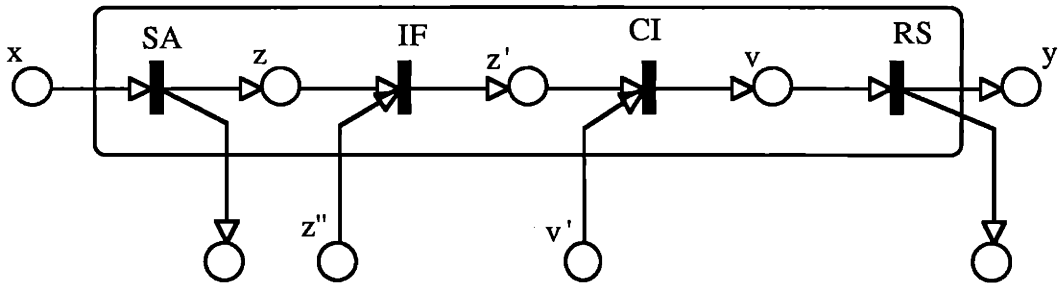


Figure 5.1 Four stage model of a DM.

In the Petri Net representation of this processing, the transitions stand for the algorithms, the connectors for the precedence relations between these algorithms, and the tokens for their input and output. The places act like buffers, hosting the tokens until all the input places of a transition t are non-empty, in which case the algorithm embodied in t can run and remove the tokens. The time taken by the algorithm to run is the transition processing time $\mu(t)$. The tokens in this model are all indistinguishable. A token in a place p means simply that a piece of information is available there for the output transition(s) of p. This information can be formatted, as part of a well-known set of identities, or alphabet, if p is an output place of a defined algorithm. This may also be the case for the input x, if the corresponding decisionmaker receives this input from another decisionmaker, of from the environment through a preprocessor. However, if the DM receives his information directly from a sensor, the question of what the meaning of x is arises. Ideally, the organization is submitted to a

continuous flow of information. However, the decision process is more discrete-like, and the approximation of an organization having to respond to discrete events is legitimate. Thus, the tokens x are unformatted information. The decisionmaker has to make an assessment of what x stands for in the first stage of his processing.

We call **attributes** the parameters describing completely what the tokens represent. For instance, if the decisionmaker has to identify a threat coming in a given area of the sky and to give a response to it, then a token on the input place of his SA stage may be just a blip on the DM's radar screen. The token that the SA algorithm produces is in turn formatted information which includes the DM's measurement, or assessment, of the position, speed, nature, behavior, or size of the threat. The DM can receive from elsewhere in the organization other formatted information, not necessarily of the same format, provided that it matches what his IF algorithm expects as inputs formats. The different tokens in the different places have then different formats, and different attributes. But as long as the protocols ruling their processing do not vary from one set of attributes to the other, they are indistinguishable tokens.

5.1.2 The Four-Stage Model with Switches

A decisionmaker may have at a particular stage of his processing, a set of different algorithms processing in different ways the same input to produce the same format of output, instead of just one algorithm. In this model, it has been assumed that only the SA and RS stages of the decisionmaking process consist of a set of U and V algorithms respectively. The SA or RS stage is represented as a **macro-transition** standing for an aggregated subnet of a Petri Net with switches (Fig. 5.2). A particular stage consists of a switch s together with a number of branches, in each of which the transitions models a specific algorithm. The switch indicates that a choice has to be made among the possible algorithms.

In the SA stage, this choice is denoted by the variable u, taking its values in, say, $\{1, 2,..., U\}$. The probability distribution of u, $(p(u = i)_{i=1,...,U})$ is called the **decision strategy** of the decisionmaker for the particular stage. If one branch of the switch is always chosen, i.e., if there is an i in $\{1,..., U\}$ such as $p(u = i) = 1$, then the strategy is **pure**. Otherwise, it is **mixed**.

The strategy that a decisionmaker uses at a particular stage may well depend on the input

of that stage. In that case, the probabilities p(u = i) are conditional probabilities. It does not make sense to condition these probabilities at the SA stage, since x, the input, is totally unformatted, and not assessed.

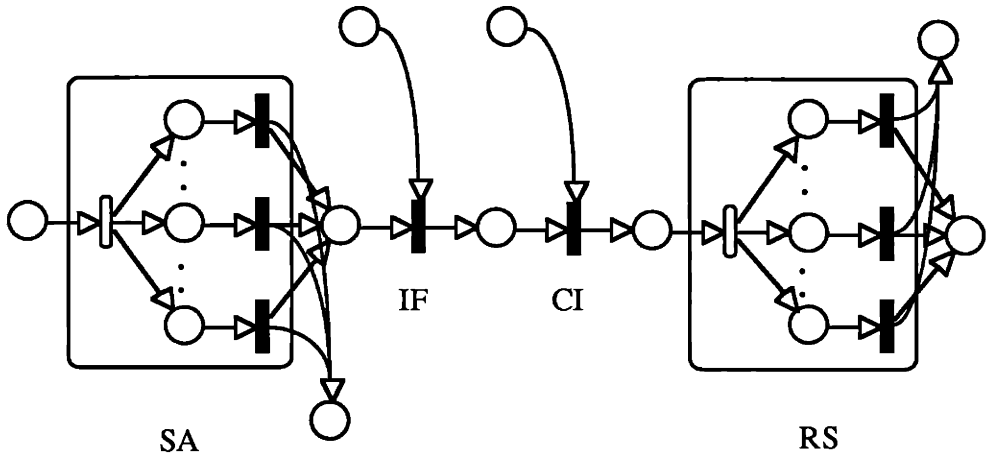


Figure 5.2 Four stage model with switches.

A **pure organizational strategy** is a strategy in which the DM's choose always the same algorithm in their Situation Assessment stage, and for each possible value of the input of their Response Selection stage, the same algorithm to produce a response. If there are n(s) switches s_i in the entire organization, if the alphabet of inputs of the switch s_i has n_i terms, and if s_i has U_i branches, then the maximum number n(pure) of pure organizational strategies is the following:

$$n(\text{pure}) = \prod_{i=1}^{n(s)} \left[U_i \right]^{n_i} \qquad (5.1)$$

with the assumption that n_i is equal to 1 whenever the probabilities of the corresponding switch are not conditioned, as in the case for the SA stage. This number is the **maximum** number of possible pure strategies, because the probabilities are conditioned on the inputs z_j of the switch. The z_j are themselves outputs of other algorithms and may not describe their whole alphabet. For instance, if the value $z_j = z_{j^\circ}$ is never reached, the pure organizational

strategies for which we have:

$$p(u = i \mid z_j = z_{j^*}) = 1$$

are never used. However, these pure organizational strategies are included in n(pure).

The structure of the interactions between decisionmakers does not depend on the setting of the switches. In order to see that, the switches and their branches are aggregated in macro-transitions, which are indicated in Fig. 5.2 by the boxes with rounded corners. In other words, the subnet constituted by the switch, its branches (i.e., its output places), and the transitions which model the alternative algorithms is aggregated in a super-node (Kyratzoglou, 1987) or macro-transition. When the processing of any given input of a switch has been completed, a token is added in each of the output places of the macro-transition, no matter what algorithm has been activated. The alternative algorithms of the switch should of course produce the same formats of output from the same formats of inputs, and should deliver them to the same places.

The macro-transitions behave then as ordinary transitions. They are enabled whenever there is a token in all of their input places, and when they fire, they add a token in each of their output places. No matter what the setting of the switch was, the tokens which are produced in a given output place have the same format, with the same set of attributes. The values of their attributes depend of course on the chosen algorithm (Fig. 5.3).

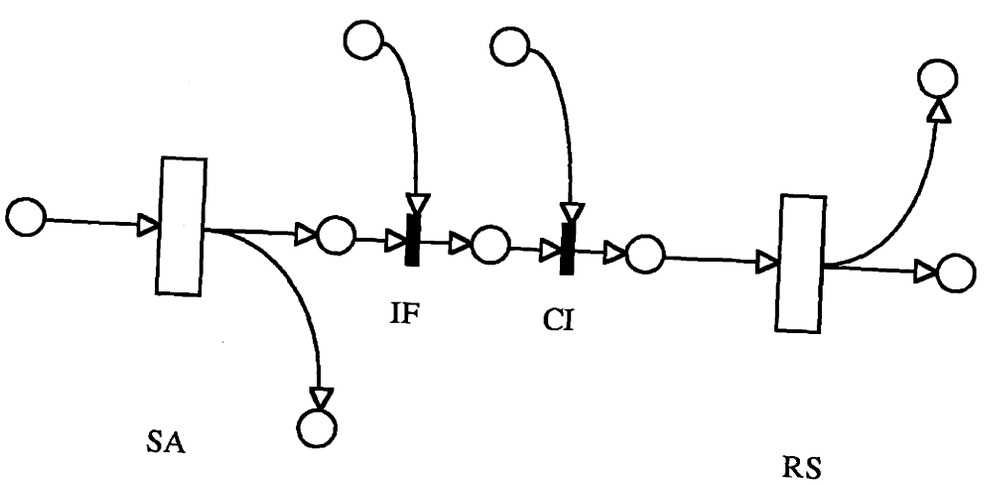


Figure 5.3 Four stage model of Fig. 5.2 with aggregated switches.

5.1.3 Interactions between DM's

As shown in Fig. 5.1, the decisionmakers can only receive inputs at the SA, IF, and CI stages, and send outputs at the SA and RS stages (Remy and Levis, 1987). The interactions which are the most significant are shown in Fig. 5.4. For the sake of clarity, however, this figure only accounts for the interactions as oriented links from DM_i to DM_j. Symmetrical links from DM_j to DM_i may of course exist as well.

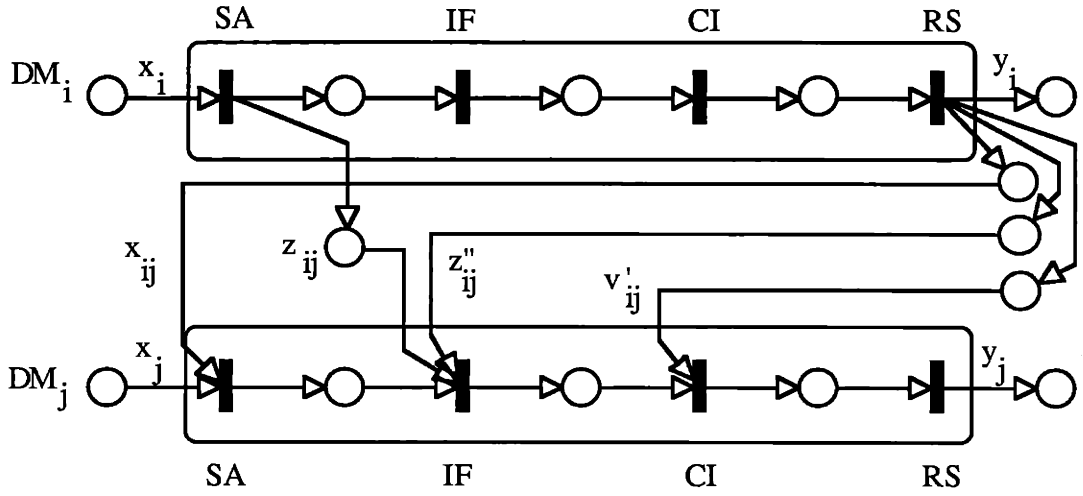


Figure 5.4 Allowable interactions between DM's.

x_i and y_i are respectively the input and output that DM_i receives from and sends to the environment. z_{ij} is an information sharing link. x_{ij} refers to the case of a serial arrangement of the decision process: DM_i sends some information to DMj who assesses the situation from it. In that case, the decisionmaker does not know in advance the format of the information he receives, nor where in that message the information he needs could be. An example of this can be that DM_j receives a large number of signals on a screen. An assessment of the situation is required.

z''_{ij} is a result sharing link. DM_i sends information to DM_j in a form already recognizable by him. v'_{ij} is a command from DM_i to DM_j and introduces explicitly the notion of a hierarchy between the two DM's.

Two kinds of places can be distinguished: the **internal places**, or **memory places,** which are the places where the decision maker stores his own information: between SA and IF, IF and CI, or CI and RS. The places between the DM's and the sensors, the preprocessors, or the actuators, as well as those between two DM's are called **interactional places**. The knowledge of the set of interactional places is equivalent to that of the whole structure of the net.

A decisionmaker may not have all his four stages present. Depending on the interactions he has with the rest of the organization and with the environment he may exhibit different internal structures:

- SA alone.
- SA, IF, CI and RS (IF and CI can be simple algorithms that copy the signal).
- IF, CI, and RS.

Depending on what the designer of the organization requires, different constraints on the allowable interactions can be expressed, which limit or expand the set of possible organizations.

5.2 VARIABLE DMO'S AS PETRI NETS WITH SWITCHES

5.2.1 Variable Interactions and Petri Nets with Switches

In the Petri Net representation of the internal processing of a decisionmaker introduced in section 5.1.2, the interactions that the DM had at a particular stage did not depend on which particular algorithm had been chosen. The decisionmaking organization had a fixed structure. In variable structure organizations, however, the structure of the interactions between DM's depends on the settings of the switches.

An example of a decisionmaker in a variable structure organization is presented in Fig. 5.5. The DM has three algorithms in his SA stage, and three algorithms in his RS stage. The transitions which model these algorithms may not have the same output places, since the interactions between the DM and the rest of the organization are algorithm-dependent. If the switch, its output places, and the transitions which model the alternative algorithms are aggregated in a super-node (as in section 5.1.2), the macro transitions which are obtained do

not act like ordinary transitions: although they have the same rule of enablement than the ordinary transition, they add, when they fire, tokens in only some of their output places, depending on what branch has been activated (Fig. 5.6).

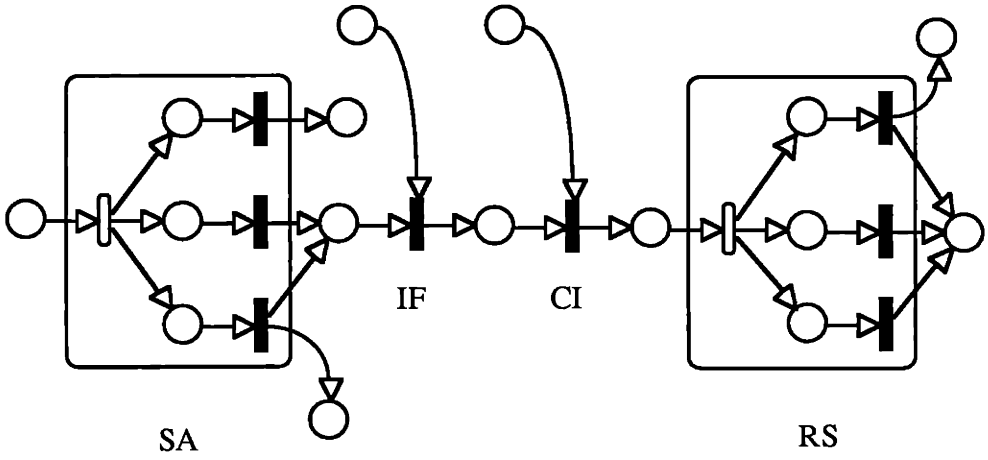


Figure 5.5 DM in a Variable DMO.

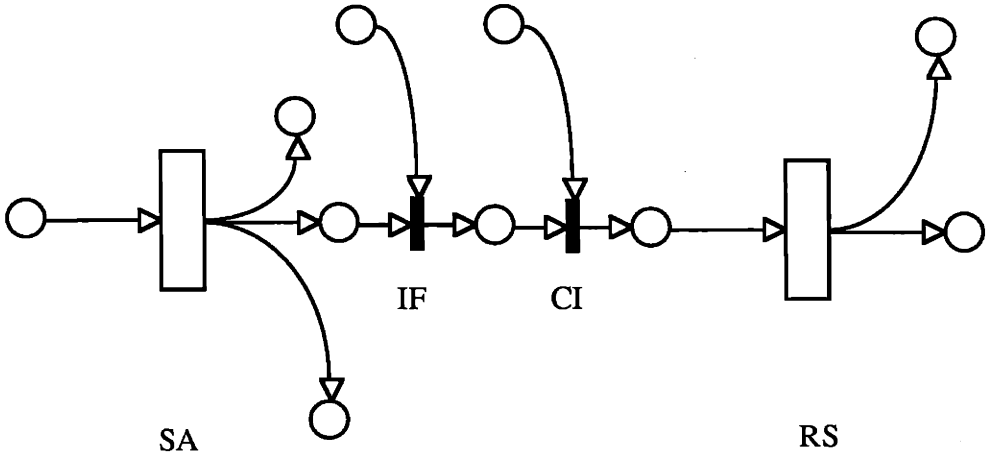


Figure 5.6 Aggregated model of a DM in a Variable DMO.

The switches in the Petri Net formalism were convenient for the modeling of alternatives in a particular stage of the decisionmaking process of a DM. It seems legitimate at this point to extend that grammar to variable organizations. We will see with an example that this idea involves a great deal of complexity.

5.2.2 An Example

We consider a variable organization (VDMO#1) of two decisionmakers DM1 and DM2 (Fig. 5.7), in which two patterns of interactions are allowed:

Setting 1 DM1 receives the situation as assessed by DM2 in his IF stage; he issues a command to DM2 and gives his own response in his RS stage. The branch (2) of switch s_1 and the branch (1) of s_2 are always chosen.

Setting 2 DM1 and DM2 have completely parallel activities in their treatment of the input, i.e., the branches (1) of s_1 and (2) of s_2 are always chosen. DM1 and DM2 never interact.

The type of variability that the organization exhibits, however, is not the immediate concern. The issue of what can trigger changes in the settings of the interactions is not addressed in this section.

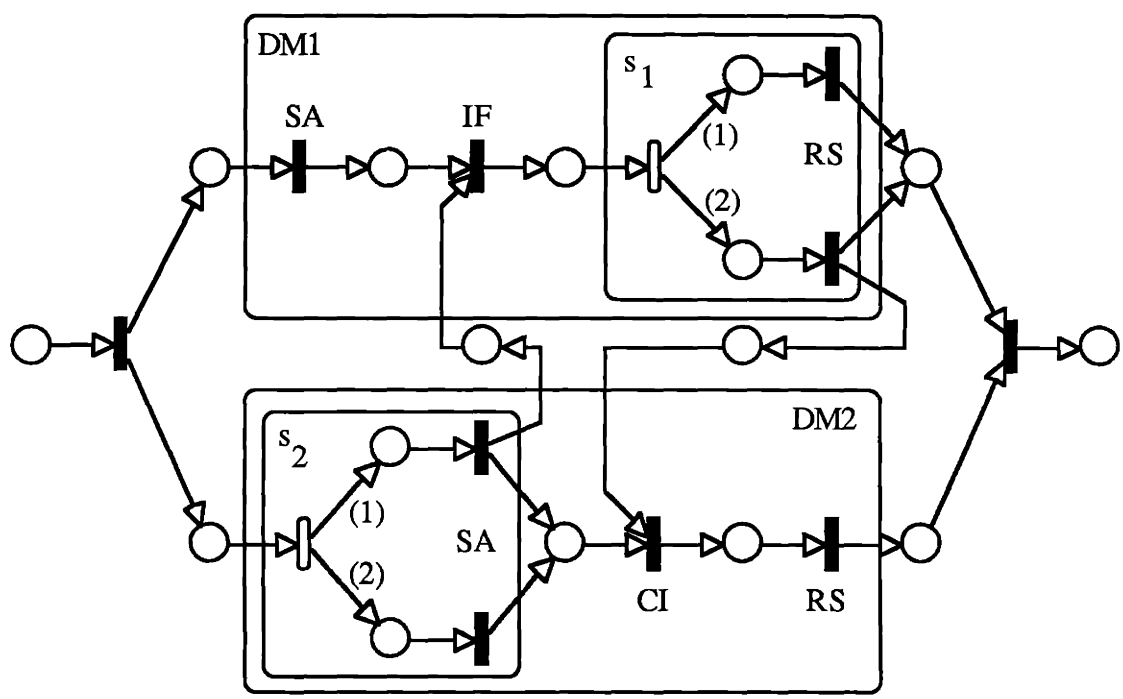


Figure 5.7 An example: VDMO#1.

The representation of VDMO#1 with Petri Nets with switches is incomplete for two reasons:

- the decision rules of the switches have to be correlated, since only two out of the four combinations of active branches are allowed for any incoming input: if we represent by (u, v) the numbers of the branches of the switches (s_1, s_2) which are activated at each time, then branches (2, 1) for setting#1, and (1, 2) for setting#2 only are valid.

- the Information Fusion stage of DM1 or the Command Interpretation stage of DM2 have different rules of enablement depending on the setting of the interactions which has been chosen. In the representation of Fig. 5.7, the transition modeling the IF stage of DM1 is enabled when there is a token in each of its two input places. As stated above, DM1 has no way of knowing when he will receive some information from DM2. If DM2 sends to DM1 either some information, or a null message to tell DM1 to continue his processing, the deadlock may be overcome: but all the advantage of variability has been totally lost, since DM1 has to wait before continuing the processing.

The representation of variable organizations must therefore take into account these two requirements of *correlation of rules* and *deadlock avoidance.*

5.2.3 Correlation of Rules

The solution is to represent the IF and CI stages with switches (Fig. 5.8), and to associate with the Petri Net depicting the organization (where the branches of the switches have been labeled by integers), a table showing the intercorrelation of the rules of the switches (Table 5.1).

The transitions in a particular stage now stand for different algorithms which have little in common. In particular, they do not have the same inputs, nor the same format in their inputs. The tokens they produce are not the same across the set of algorithms. However, when these tokens are directed to an identical place, they have the same format. And the tokens are still indistinguishable.

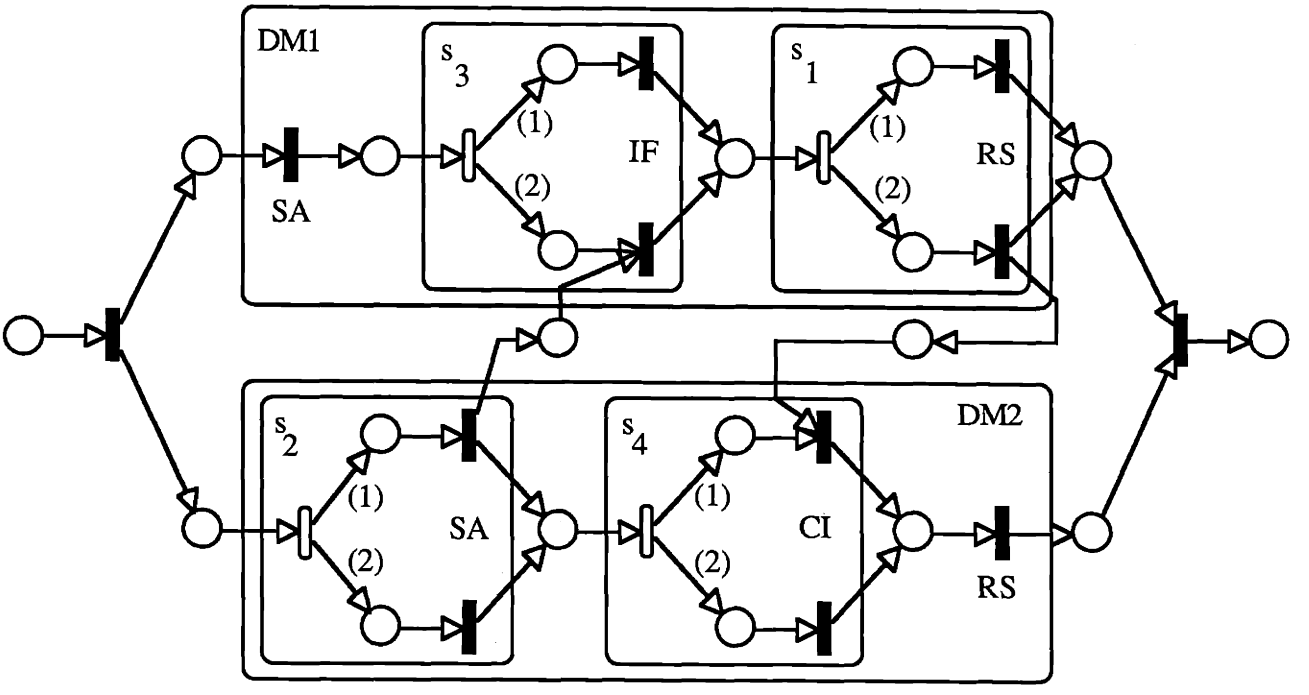


Figure 5.8 An example: VDMO#1 (bis).

TABLE 5.1 Settings of the switches (VDMO#1).

		Switches			
		s1	s2	s3	s4
Settings	1	2	1	2	1
	2	1	2	1	2

We consider now another variable organization VDMO#2, which consists of the decisionmakers DM1 and DM2, and which allows the following settings:

Setting 1 As in VDMO#1.

Setting 2 As in VDMO#1.

Setting 3 DM1 receives the situation as assessed by DM2. The further processing of the two DM's is independent.

Setting 4 The only interaction between the DM's is a command from DM1 to DM2.

The Petri Net with switches which represent the VDMO#2 is the same as that of VDMO#1 (Fig. 5.8). The table of correlation of the rules of the switches, however, has to be modified (Table 5.2), since more settings are allowed.

TABLE 5.2 Settings of switches (VDMO#2).

		Switches			
		s1	s2	s3	s4
Settings	1	2	1	2	1
	2	1	2	1	2
	3	1	2	2	2
	4	2	2	1	1

Therefore, the Petri Net model of a variable structure decisionmaking organization requires a table of correlated switch settings. Although the model which is obtained at this point is guaranteed never to deadlock (e.g., a decisionmaker does not wait for information which is not sent), it still leaves aside a whole set of issues, such as knowing how effectively the DM's communicate their strategy to each other, or who will decide the strategy for setting the switches of the other DM's.

5.2.4 Motivation for Predicate Transition Nets

The Petri Nets with switches are not a convenient formalism for the representation of

variable DMO's. As illustrated in section 5.2.3, they introduce a set of problems which can be listed as follows:

- other switches are needed in addition to the ones in the SA and RS stages.

- the intercorrelation between the switches is not indicated on the net. A table has to be attached to it. The actual way the DM's communicate their choice of algorithms is not modeled explicitly.

- the relation between the inputs and the patterns of interactions is not shown explicitly. The very high illustrative power of Petri Nets is lost since the behavior of the net can not be deduced from its representation.

- the representation becomes quite complex even for simple organizations. This modeling tool may become unworkable when applied to organizations with more decisionmakers, and more possible interactions between them.

- the addition of decisionmakers, of possible links, or their removal, obliges the designer to redesign the net and the attached table totally.

Some other tool has then to be applied for such modeling. The first approach is to attach to the tokens the information of the setting of the switches. If a decisionmaker has a deterministic way to dispatch his outputs as a function of the attributes of the token which he possesses, then he has a way to know what would be the strategies of the other DM's, as far as their interactions are concerned. Another approach is to represent separately the different patterns of interactions which are allowed in the DMO, and to make the DM's move from one pattern to the other depending on the information they receive.

What is needed then is a tool which would allow to distinguish among the tokens, and which would have the capability to implement logic able to determine explicitly what interaction and what DM's have to be active for the processing of a given input. Individual tokens, Predicates, and Operators can meet these requirements. The application of the Predicate Transition Nets to that purpose is developed in the next section.

5.3 MODELING METHODOLOGY

5.3.1 A Modular Architecture

The purpose of this section is to present a methodology for the modeling of variable structure organizations (VDMO) using Predicate Transition Nets. The methodology has a modular architecture (Fig. 5.9), and consists of five modules:

1- Interface with the environment.
2- Scarce resources.
3- Interactions.
4- Switching module.
5- Algorithm implementation.

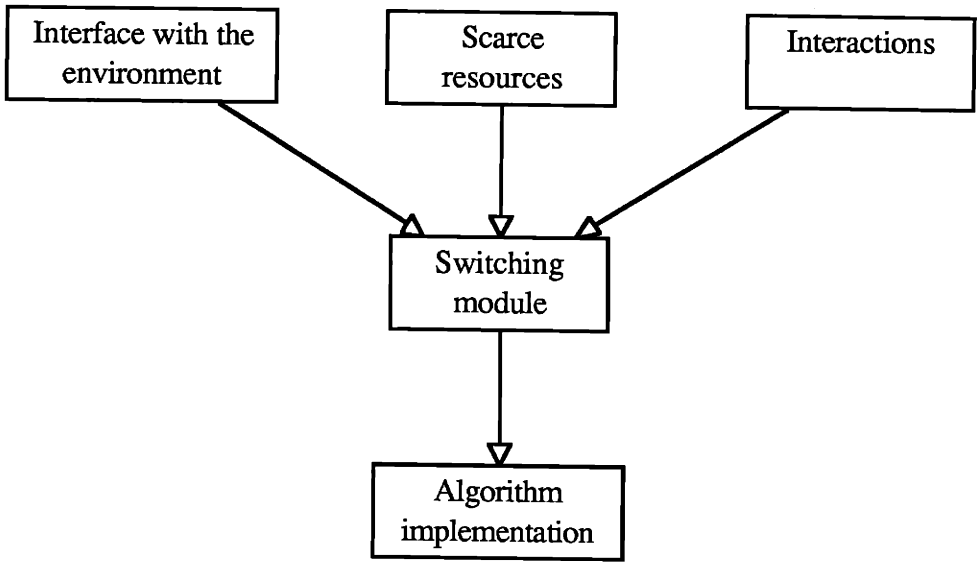


Figure 5.9 Architecture of the modeling methodology.

Each of the first three tasks can be done independently and in arbitrary order. The three tasks address the sub-problems (a) of modeling of the inputs that the DMO receives and the responses that it gives, (b) of the representating the scarce resources that the DMO needs in that processing, and (c) of modeling of the possible interactions which can exist between the components.

When the first three tasks have been completed, the switching module is designed. The switching module is the part of the model where the logic, which rules the variability feature of the organization, is implemented. For each incoming input, this is the part of the model which decides what particular resources, and what particular setting of interactions will be adopted. The way this choice is made will determine what type of variability the VDMO exhibits.

What is obtained at this point is a Predicate Transition Net where only the non trivial operators, i.e., the operators which consist of more than a sequence of quantifiers (see section 3.2.4) are indicated in the corresponding transitions. The fifth and last part of the methodology consists of the rigorous labeling of the nodes, connectors, and tokens of the net. It also aims to give a precise meaning to what the individual tokens stand for (i.e., the list of their attributes), depending on the place which hosts them, and on what algorithm, or what set of algorithms, a particular transition models. The processing time of the different algorithm is defined in that part of the methodology.

The steps of that methodology are independent enough to allow easy changes in any subproblem, without threatening the functioning of the whole model. The modular architecture is also very convenient for the implementation of extensions of the model, which simply become new modules, or new well-defined subproblems. Examples of extensions are given at the end of the present chapter.

The present section focuses on the modeling of type 1 variable DMO's. An example of a three member organization with type 1 variability serves to illustrate the methodology. Examples of VDMO's exhibiting type 2 or type 3 variability are included in chapter VI. These models will be produced by the same type of modular methodology.

5.3.2 Interface with the Environment

The goal of this sub-problem is to achieve a representation of the input and output alphabets. In the modeling of decisionmaking organizations (see section 5.1) we have defined the discrete representation of information sets as lists of attributes, an instance of which is called a token. In the ordinary Petri Net representation of a DMO, the values of the attributes were of no importance as far as the Petri Net was concerned; no matter what these values were, the treatment of the token was the same: the interactions between the

components were the same, the algorithms, and their precedence relations were the same. The only difference which appeared was the strategy used in a switch ruled by conditional probabilities. The interactions, however, between the DM's were not affected by these strategies and remained the same across the alphabet of inputs.

In the case of type 1 variable DMO's, the inputs are separated in classes, each of which needs different resources, and different interactions between these resources. The inputs are still lists of attributes, but they have associated an additional attribute which indicates the interaction and the resources required for its processing. Since the other attributes are not necessary to determine that information, they are not explicitly mentioned.

The alphabet X of inputs is therefore partitioned in r classes, namely X_i, for i = 1,..., r. All inputs x belonging to the same class are processed with the same resources used with the same pattern of interactions. A given input x cannot belong to more than one class, which is to say that it can only be processed with one specific set of resources, and one specific kind of interactions.

The inputs are then represented by individual tokens; the identity of a token is the class X_i to which it belongs, i.e., its number #i. The variable "class of inputs" is denoted by x and has the following set of allowable identities:

$$\underline{x} = \{1,..., r\}.$$

We suppose that the processing of the response does not depend on its attributes. Then the tokens which model the outputs of the organization do not have an identity. They are instances of the 0-ary variable $\not\epsilon$.

Example: Step 1

The example used in this section is a three member organization with four possible interactions between the decisionmakers. The DMO consists of two field units, FU1 and FU2, and one headquarters, HQ. The possible interactions are the following:

Int#1- FU1 and HQ (HQ fuses its assessment with FU1's, and issues a command to him).

Int#2- FU2 and HQ (HQ fuses its assessment with FU2's, and issues a command to him).

Int#3- FU1 alone (SA and RS stages).

Int#4- FU2 alone (SA and RS stages).

The alphabet of inputs X is therefore partitioned in four classes X_i, i = 1,..., 4. The variable x representing the class of the inputs has a set of identities {1, 2, 3, 4}. The outputs are not partitioned. The model of the organization which is obtained at this point is shown in Fig. 5.10.

Figure 5.10 Example - Step 1.

5.3.3 Scarce Resources

A resource is a generic name which designates something which is needed for the processing of an input to be accomplished. A resource is scarce when it cannot be allocated freely to the processing of any incoming input because of insufficient or limited supply. The resource has to be shared. For example, two consecutive inputs which need the same resource have to be processed successively. The scarcity of resources bounds from above the performance of the organization.

Scarce resources are modeled in a convenient way in the Petri Net formalism. They are represented by places with multi-input transitions and multi-output transitions, and non-zero original marking. In the example depicted in Fig. 5.11, the resource place R means that either the transition t_1 or t'_1 fires, but they cannot fire simultaneously, if the initial marking of the place R is 1.

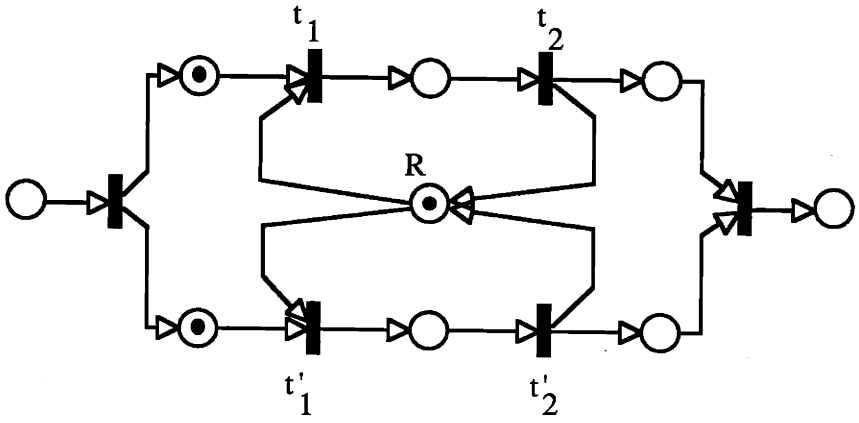


Figure 5.11 Scarce resource.

Examples of scarce resources can be common databases with limited access, communication links with limited capacity, mainframes with shared processing time, or weapons platforms capable of handling a limited number of threats at a time.

In the modeling methodology, the decisionmakers are treated at the same level as these scarce resources. Actually, in the decisionmaking process, they act exactly like resources: they are assigned to an incoming input; once they have been assigned to a certain number of inputs, the other inputs have to wait in line to be processed; they can process different kinds of inputs, and have different kind of interactions, just like the token in place R in Fig. 5.11 can be fired by either t_1 or t'_1.

The pool of decisionmakers which implements the organization is partitioned in classes of DM's who have the same function within the organization, i.e., who possess the same kind of algorithms. Two decisionmakers who belong to the same class are then interchangeable. The DM's of a class are represented by individual tokens of a variable, and placed in the corresponding resource place. If there is only one class of DM's, then the DM's are represented by uncolored tokens. The other resources that the organization may need are partitioned and associated with variables and places the same way.

The input and output connectors of a given resource place R where the corresponding variable is x are labeled by elements of $L^+(x)$ (see section 3.2.3).

Example: Step 2

In the example, two DM's are interchangeable as far as their interactions with the rest of the organization are concerned: these are the field units FU1 and FU2. HQ has a specific function in the DMO, and is the only one in that case. The three DM's are then represented by the following variables:

- Resource place FU: associated with the variable $\underline{s} = \{1, 2\}$. The individual token 1 models the decisionmaker FU1. The token 2 stands for FU2.

- Resource place HQ: since there is only one HQ, the place carries an indistinguishable token ¢, shown as a dot in the place HQ.

The modeling of the DMO obtained at this point is shown in Fig. 5.12.

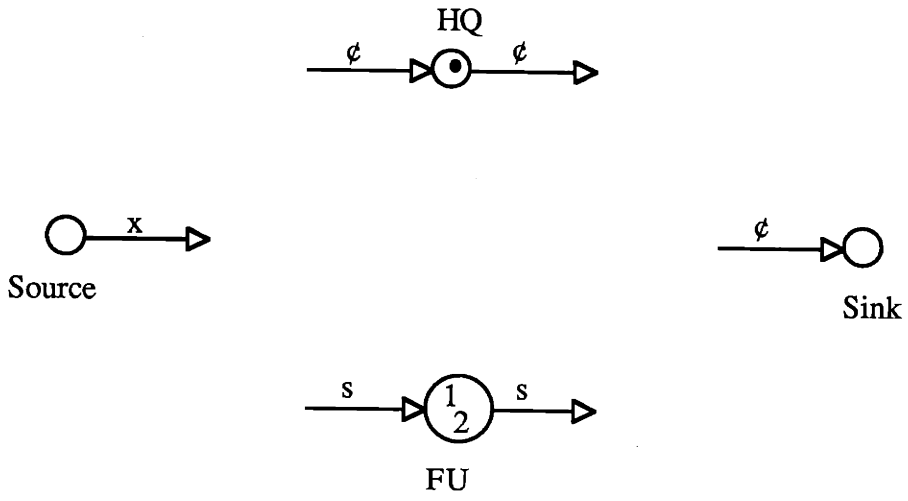


Figure 5.12 Example - Step 2.

5.3.4 Interactions

The interactions between components which are allowed in the organization are represented without paying attention to the identity of the resources they involve. What is of interest, at this point in the modeling, is only the topology of interactions that can be found in the DMO. The way one interaction is chosen instead of another is not of immediate concern.

The typical model obtained at this point is shown in Fig. 5.13: it is a list of the possible patterns of interactions depicted in their most aggregated form. The input and output places of these possible interactions would have been the source and sink places, had these interactions been considered alone as DMO's with fixed structure.

The possible interactions can be partitioned in four generic types, as illustrated in Fig. 5.13:

- Type (a): the pattern of interactions is that of an organization with a fixed structure which processes the inputs without resources. It is represented by an ordinary Petri Net which can be aggregated in a super-node Int#1.

- Type (b): the pattern of interactions has the same characteristics as in type (a), but the net which models that pattern exhibits some properties of symmetry. A more convenient representation is obtained by folding the net, as explained in section 4.4.2. The Predicate Transition Net which is obtained is similar as that of Fig. 4.9, and is aggregated in turn in a super-node Int#2.

- Type (c): the pattern of interactions is the same as type (a), but the DMO with that pattern requires a resource R_1 for the processing of the inputs. This resource is used from the beginning of the processing until its completion. The ordinary Petri Net which models that pattern is therefore aggregated in a super-node and the resource place R_1 is both an input and an output places of that macro-transition Int#3 (the underlying Petri Net is still pure, however).

- Type (d): the pattern of interactions is similar as in type (c), except that the resource R_2 is not used during the whole processing of the inputs. In the particular case of Fig. 5.13(d), it is only needed at its beginning. The ordinary Petri Net modeling that pattern is then aggregated in two super-nodes, (Int#4,1) which stands for the part of the processing of the task using the resource R_2, and (Int#4,2) which accounts for the remaining of the processing.

Any other combination of type (a), (b), (c), or (d) can be encountered as well. In particular, the number and diversity of resources required and the lack of symmetry of the pattern of interactions may lead to the irrelevance of the aggregation in super-nodes. In that

case, the net which would appear in Fig. 5.13 would show in detail all the stages of the decisionmaking process.

No matter where the resource places are connected, the subnet which is subsumed in a macro-transition represents a decisionmaking organization where the internal processing of the input is modeled by the four stage representation that was described in section 5.1. That net stands, therefore, for an organization with fixed structure, which is to say, that it may contain some switches, but the setting of these switches does not affect the structure of the interactions between the decisionmakers (whose identity is not defined) (see section 5.2.1, Variable DMO's with Petri Nets with Switches). If each switch is aggregated in a macro transition, then the ordinary Petri Nets which are obtained are all **Event Graphs**, i.e., a place can have only one input transition, and only one output transition.

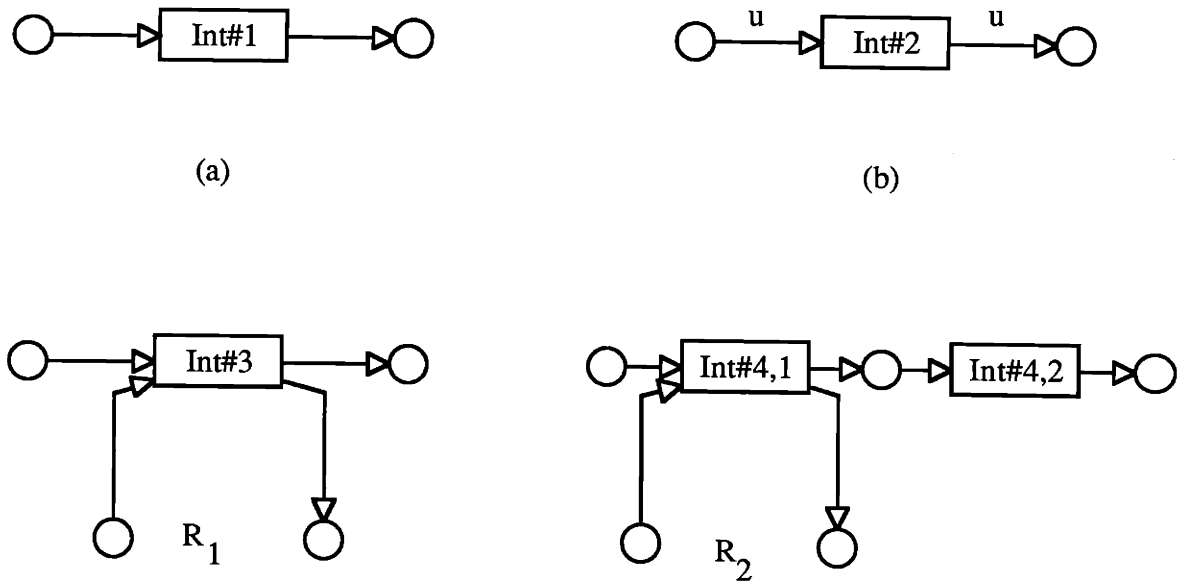


Figure 5.13 Allowable interactions.

Example: Step 3

In the example, although four kinds of organizations have been allowed, only two

patterns of interactions are actually distinct: one where the HQ interacts with a FU, and one where the FU processes the task alone. The first part of the modeling consist of representing these patterns in detail (Fig. 5.14). Then an aggregated model comparable to Fig. 5.13 can eventually be produced.

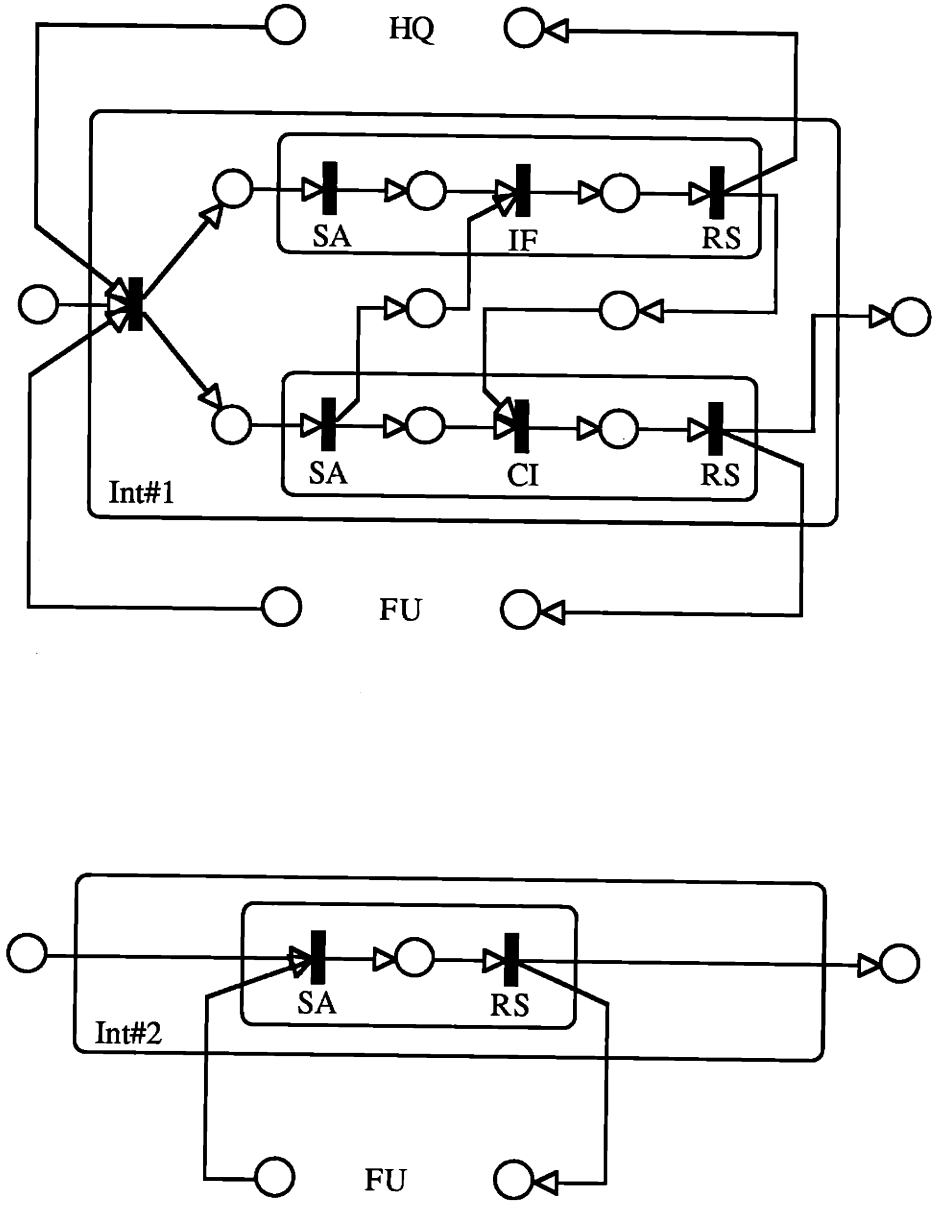


Figure 5.14 Example - Step3.

For the first pattern of interactions, two resources are required, namely HQ and FU. The

resource HQ is not used in the decision process until a response is chosen, and can be free before that. The resource FU, however, is needed from the beginning of the processing to the end. Finally, this pattern of interactions is such that no aggregation in super-nodes is possible.

For the second pattern, the only resource used is FU, and it is needed during the whole processing of the input. An aggregated version of Int#2 would then be similar to Fig. 5.13b.

5.3.5 Switching Module

The objective of this part is the representation of the decision rule which determines for any incoming input what the actual configuration of the organization will be. Again, the switching module is the part of the modeling where the type of variability of the organization will be modeled. It supposes that the first three sub-problems have been already completed.

A switch is implemented as an output node of the source and the resource places. This switch is modeled as indicated in section 4.4.2, and as such, consists of a set of transitions with operators, whose arguments are the individual tokens in the source and resource places. Recall that the focus of this section is a DMO with type 1 variability, and that it has been assumed that each class of inputs has associated only one possible pattern of interactions. Thus, if the number of classes of inputs is r, there are at most r branches in that switch.

A decisionmaking organization needs an interaction and some resources to process an incoming input. The type 1 variable DMO which has been considered so far adopts for each class of inputs, a specific interaction and set of resources. The formal notation for the inputs, resources, interactions, and their relations is the following:

Inputs

- An input is an individual token of variable x.
- The source place SO is associated with variable x.
- The set of allowable identities for x is $\underline{x} = \{1, ..., r\}$.
- An input of variable x belongs to the class X_i, where $x = i$.

Resources

- The resources places are R_k for $k = 1, ..., K$.

- The resource place R_k is associated with the variable s_k.
- The set of allowable identities for s_k is $\underline{s}_k = \{1, ..., S_k\}$

Interactions

- The patterns of interactions are Int#(γ), for $\gamma = 1, ..., \Gamma$.
- There are J transitions t_j in the switch.
- t_j is associated with the Operator Op_j.
- t_j is associated with the pattern of interactions #ϕ(j), i.e., Int#(ϕ(j)).

Relations

- The input x requires a pattern of interactions #γ(x), i.e., Int#(γ(x)).
- The input x requires some resources from R_k, which are:
  $res(k, x) = \{s_{k,n}(x) \mid n = 1, ..., N(x)\}$.
- γ(x) and res(k, x) for any k are functions of x.
- ϕ(j) is a function of j; ϕ is attached to the switch.

An incoming input, modeled as an instance of an individual token x, belongs to the class X_i. The organization is type 1 variable, and it adapts the pattern of its interactions to the class of the incoming input. The processing of the input x requires a precise pattern of interactions, namely Int#(γ(x)). Since the same interactions can be adopted for different classes of inputs, the function γ is not bijective, and the number Γ of interactions is necessary smaller than the number r of classes of inputs. The processing of this individual token x also needs some resources of type R_k, given by the set of individual tokens res(k, x). The transition of the switch which corresponds to the pattern of interactions Int#(γ(x)) is the transition t_j such that $\phi(j) = \gamma(x)$; there is only one j such that this relation is verified, which is denoted as $\phi^{-1}(\gamma(x))$.

If all the conditions stated above are fulfilled, then the input x is processed. In other words, for $\phi(j) = \gamma(x)$, the transition t_j is enabled and fires. The operator Op_j associated with t_j expresses in logical terms the above conditions, and can be written as follows:

$$(\exists\, x \in SO) \wedge (\gamma(x) = \phi(j)) \wedge (\exists\, res(k, x), R_k \supseteq res(k, x)) \qquad (5.2)$$

Since the transition t_j corresponds to Int#(ϕ(j)), and since this pattern of interactions may be needed for more than one class of input, the actual operator associated with t_j is the logical

OR ($\vee$) of the operators (5.2) for the inputs x such that $\gamma(x) = \phi(j)$, i.e., for all the inputs x in the set $\gamma^{-1}(\phi(j)) = \{x \mid \gamma(x) = \phi(j)\}$. The operator Op_j associated with t_j is finally the following:

$$Op_j = \bigvee_{x \in \gamma^{-1}(\phi(j))} [(\exists\, x \in SO) \wedge (\exists\, res(k, x),\ R_k \supseteq res(k, x))] \qquad (5.3)$$

The operators $(Op_j)_{j = 1,...,J}$ which are attached to the transitions t_j which constitute the branches of the switch are such that the property "P" of section 4.2.2 of conflict resolution is verified: for any input x in the place SO, there is one and at most one transition is the set (t_j) which is enabled, the one with the number $j = \phi^{-1}(\gamma(x))$. There is therefore no conflict and as soon as the required resources res(k, x) are available, t_j can fire.

The connectors from the place R_k transition t_j are labeled by the set $L_{conn}(R_k, t_j)$ (see section 3.2.3, Connectors) whose elements are the symbolic sums of the individual tokens in res(k, x). If the set res(k, x) is non empty, the connector from R_k to t_j has the following label:

$$L_{conn}(R_k, t_j) = \left\{ \lambda \in L^+(s_k) \mid \lambda = \sum_{n=1}^{N(x)} s_{k,n}(x) \text{ and } \gamma(x) = \phi(j) \right\} \qquad (5.4)$$

Example: Step 4

In the example, the switching module is composed of two transitions t_1 and t_2. Using similar notation as above, we get:

Inputs: $\underline{x} = \{1, 2, 3, 4\}$.

Resources: R_1 = HQ, associated to the 0-ary variable ¢.
R_2 = FU, associated to the variable s, with $\underline{s} = \{1, 2\}$.

Interactions: Int#1, corresponding to transition t_1.
Int#2, corresponding to transition t_2.

Relations: For any input x, the pattern of interactions Int#(γ(x)) is:

$\gamma(1) = 1$
$\gamma(2) = 1$
$\gamma(3) = 2$
$\gamma(4) = 2$

For any input x, the required resources are:
res(1, 1) = res(1, 2) = {1¢}
res(1, 3) = res(1, 4) = $\emptyset$

res(2, 1) = {1}
res(2, 2) = {2}
res(2, 3) = {1}
res(2, 4) = {2}

The operators $Op_1$and Op_2 can then be written (without mentioning the quantifiers) as follows:

Op_1: $[(x = 1) \wedge (s = 1)] \vee [(x = 2) \wedge (s = 2)]$.
Op_2: $[(x = 3) \wedge (s = 1)] \vee [(x = 4) \wedge (s = 2)]$.

The operators can actually be aggregated into a more convenient form:

Op_1: $[[(x = 1) \vee (x = 2)] \wedge (s = x)]$.
Op_2: $[[(x = 3) \vee (x = 4)] \wedge (s = x\text{-}2)]$.

The net obtained up to this point is a net where the patterns of interactions, the resources, the source, the sink, and the transitions of the switch are connected together, and where these transitions show the operators which are assigned to them. The patterns of interactions, however, are still in their most aggregated form, and the connectors are not all labeled (Fig. 5.15). This net is **temporary**, as were all the previous nets (steps 1 through 3). The purpose of the next module will be precisely to make this net functional by completing its annotation.

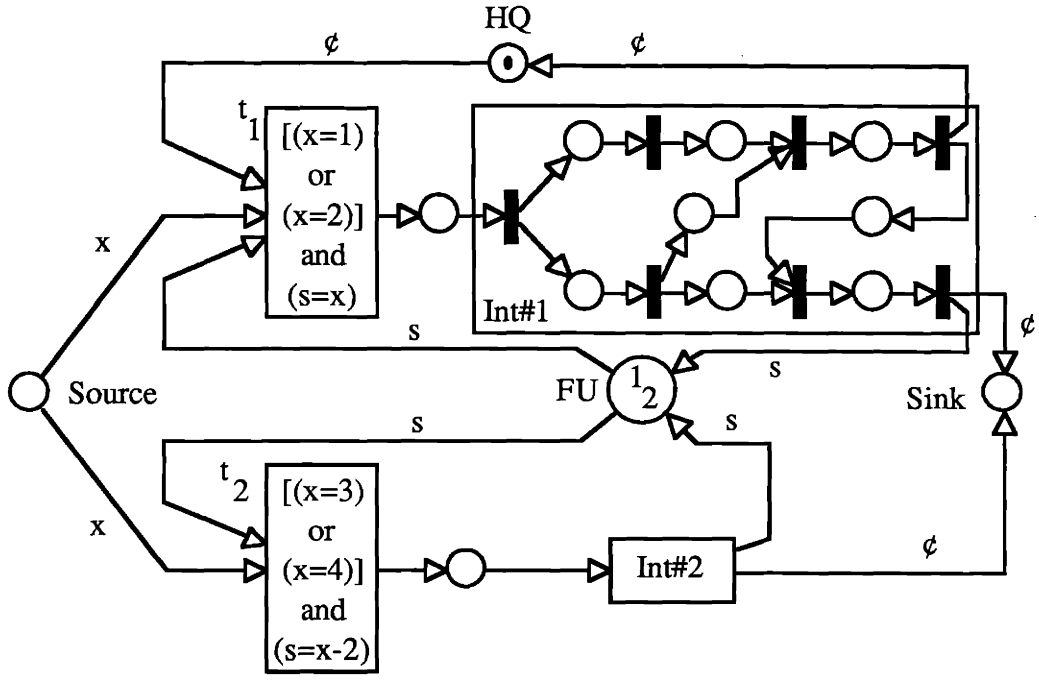


Figure 5.15 Example - Step 4.

5.3.6 Algorithm Implementation

This final part of the methodology deals with the labeling of the connectors, with the definition of the attributes of the tokens which can be found at different places, and with the algorithm that the various transitions represent. The rules of firing are also established in the present section.

Labeling of connectors: The connectors from the source to the transitions of the switch are labeled x, i.e., with the variable designating the class of the inputs. Those from the input nodes of the sink to the sink itself are labeled ¢. The labels of the output connectors of the resource place R_k have already been given in section 5.3.5 (Eq. 5.4). The input connectors of R_k are labeled accordingly.

Each pattern of interactions Int#(γ) is adopted whenever the incoming class of input x is such that $\gamma(x) = \gamma$. When x describes the set of classes of inputs $\underline{x}$, the number of times Int#(γ) is activated is equal to the number of times $\gamma(x) = \gamma$, i.e., to the cardinal γ° of the set $\{x \mid \gamma(x) = \gamma\}$. The connectors which are involved in the representation of the organization

with a pattern of interaction Int#(γ) can then be labeled with a variable ρ_j whose set of allowable identities is the following:

$$\underline{r}_j = \{1, 2, ..., \gamma^\circ\}.$$

These labeling rules are the most general that can be presented, and can be applied to any case. However, some variable organization can be such that another labeling may be more intuitive, or more convenient, in which case that latter labeling would be preferred to the more generic one developed in this section.

Firing rules: The firing rules are actually problem dependent, and can be revised at any time. However, they are generally the following:

- the transitions which constitute the switch are enabled and fire consecutively, i.e., with one input at a time.

- the transitions which are part of the subnets representing the possible interactions with Ordinary Petri Nets are enabled and fire in the same consecutive manner. In other words, if a given place in one of these subnets contains more than one token, its only output transition ("only" because the subnet is an event graph) is enabled by more than one token. But it will fire them only one by one.

- the transitions which are part of the subnets representing the possible interactions with Predicate Transition Nets, i.e., when the original Petri Net has been folded (sections 4.4.2 and 5.3.4), can allow simultaneous firing; depending on the circumstances, two tokens in the same place of a given net of that kind can enable the same transition at the same time and leave simultaneously the same place.

A given transition which is not part of the switching module, but of the subnet Int#(γ) models a set of at most γ° algorithms, or a set of γ° sets of algorithms, i.e., switches. Depending on the identity of the individual token of variable ρ_j which enables it, a particular algorithm, or a particular switch, is activated, and processes the input that the token represents. Depending also on the organization that the net models, this transition can very well consist of only one algorithm, which is always activated and executed when the transition is enabled and fires, regardless of the identity of the individual token which has

triggered that process. The rule which selects the algorithm which will process the token which enabled the transition is problem dependent, and as such, defined for each particular case.

The individual tokens represent different information sets depending on the variables they belong to, and depending on the places which host them, and on the algorithm which they activate when they are removed from an input place of a transition. However, the only attributes which are represented are those which are relevant as far as the interactions between components are concerned. Two individual tokens which occupy the same place are necessarily from the same variable, but their other attributes can very well differ in terms of their actual identities, or in terms of their formats. The rules which associate a given individual token with a given algorithm within the set that a given transition represents can account for these attributes. Once again, since such rules do not determine the interactions between the components, their definition will not be pursued in this Thesis.

Example: Step 5

The final representation of the example is given in Fig. 5.16. In this example, since the organization is fairly simple, a simplified and self-explanatory labeling has been adopted.

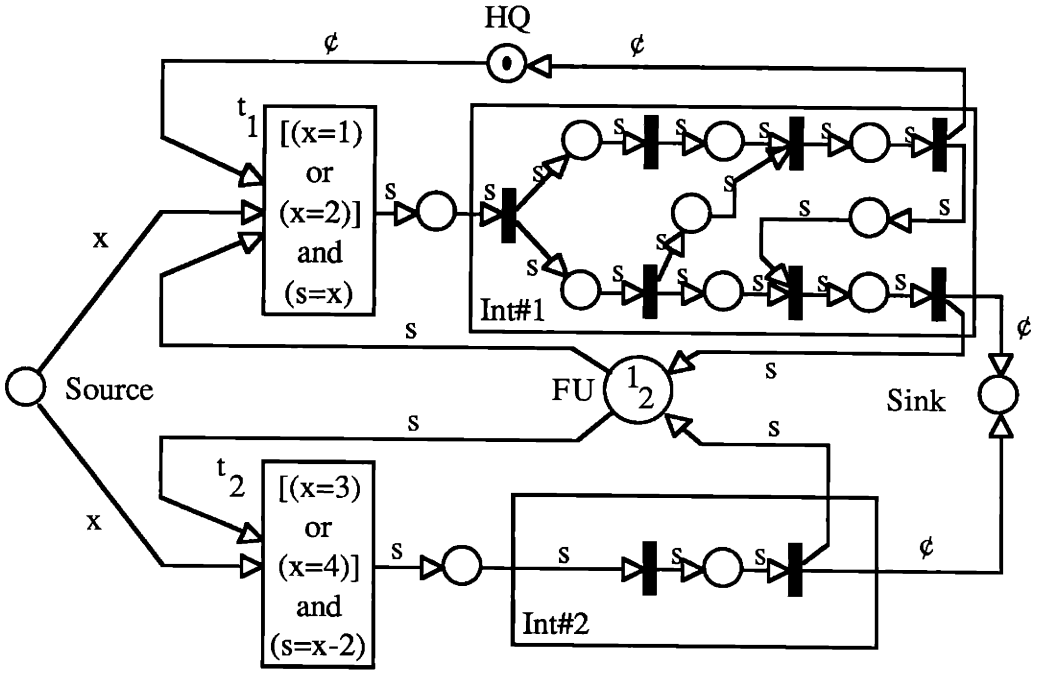


Figure 5.16 Example - Step5.

5.3.7 Extensions of the Methodology

As pointed out in section 5.3.1, the architecture of the methodology which has been presented is highly modular. It allows for easy modification of the present organization being modeled, and for possible extensions or enhancements which would take into account other features of the actual organization.

Modifications

Modifications to the model of the three member organization which has been obtained at the end of section 5.3.6 could include:

- **other classes**: the partitioning of the alphabet of inputs X can be modified, and the operators which make up the decision rules of the switch changed accordingly.

- **other interactions**: other possible interactions can be added to the set of allowable ones described in section 5.3.4, whereas some others can be removed from that set. It it then necessary to modify the partitioning of the alphabet of inputs, and the operators of the switch as well.

- **more or fewer resources**: another field unit, say DM3 can be added to the pool of decisionmakers of the organization, or some others can be removed. Other types of resources can be added: for instance, a weapon system (the actuator) which is necessary to the organization no matter what configuration has been chosen, but which can only be assigned to one target at any time. This weapon system could be represented as a scarce resource R_W containing one token $\not\subset$ and with the same connections with the transitions of the switch and with the subnets Int#(γ) as the resource FU. In that case, corresponding modifications have to be made in the algorithms associated with the transitions of the subnets, and in the switching module.

- **Different switching module**: a change in the switching module can be made, everything else being equal, assigning, for instance, other sets of resources to the same input.

Extensions

Different meanings can be given to the components which make up the model of VDMO's. Some new modules can also be added to the current architecture of the methodology. We present in this section two possible extensions, which would allow the modeling of DMO's with type 2 or with type 3 variability.

- **Perception of the environment**: a new module could consist of assessing the changes in the environment by keeping track of the last N tasks that the organization has been processing, and evaluating the probability distribution of the classes of inputs on the basis of this sample of size N. When a modification in that distribution justifies the reconfiguration of the organization, the DMO changes the pattern of its interactions and the resources that it uses in its function. Then it is type 2 variable.

- **Multi-matching**: a class of incoming inputs can be treated in different ways; instead of having only one possible pattern of interactions and one possible combination of resources, different combinations of resources and interactions can be used for their processing, with of course different performance. A model which would account for such property could be the extension of the meaning of the switching module to a multi-level switching module. This switch would propose successively these alternatives to the incoming input, following in an order corresponding to a degraded performance. Then, if a component or a resource of the organization is removed, the combination of resources and interactions would differ from the one which would have been adopted otherwise. The organization is type 3 variable.

The following chapter will develop these two extensions using two different examples to illustrate them.

CHAPTER VI

MODELING METHODOLOGY: APPLICATIONS

This chapter provides some applications of the methodology developed in chapter V. The focus of chapter was the modeling of organizations with type 1 variability. The DMO adapted the pattern of its interactions to the class of the incoming input. Another meaning can be given to the identity of the inputs of the DMO, allowing to account for type 2 variability: the first organization considered in the present chapter is a two-member DMO where each decisionmaker can serve as a Head Quarter or as a Field Unit, depending on the environment in which it functions. Different combinations of resources and interactions can also be used for the processing of a given task, leading to type 3 variability; the second organization considered in this chapter allows this multi-matching, and is modeled using the same methodology as in chapter V, with a different switching module. Finally, the third organization has a fixed structure, but exhibits variability in the interactions between its decisionmakers and a decision aid (Grevet, 1987). This DMO is modeled with the same methodology as in chapter V, and is shown to be type 1 variable.

6.1 A SYMMETRIC ORGANIZATION WITH TYPE 2 VARIABILITY

6.1.1 The Organization

We consider a decisionmaking organization of two DM's, namely DM1 and DM2. Only one kind of interaction is allowed between these two DM's: after having both assessed the situation, one of them sends his information to the other, who fuses it with his own. The latter DM issues then a command to the former, who produces the final response of the organization.

These two DM's however are not totally interchangeable. They do not possess the same set of algorithms. In other words, they are experts in different fields; and each is a novice in the field in which the other is an expert. Since it makes sense that the expert issue a command to the novice instead of the contrary, considering an organization with variable interactions between its DM's is appropriate.

An example of such complementarity of the domains of expertise can be found in an organization in charge of the air defense of a particular sector, where the threats are of two different kinds, aircraft or missiles. One DM (namely DM1) would be a specialist in the identification of the aircraft (type, function, and armament), whereas the other (namely DM2) would be one in the tracking of a missile threat.

In an environment where the tempo of operations is fast, a type 1 variable DMO which would adapt the pattern of its interactions to the identity of the inputs (e.g., aircraft or missiles) would probably not be workable, since it would require a lot of switchings between the two possible interactions. Instead of switching for different input classes, a type 2 variable organization would adopt each pattern for a given range of environments modeled by the probabilties $p = (p_1, p_2)$, where p_1 is the probability of occurence of an aircraft, and p_2 the probability of occurence of a missile, with $p_1 + p_2 = 1$. For instance, in an environment in which the threat is much more likely to be an aircraft than a missile (i.e., p_1 much higher than p_2) the decisionmaker who is an expert in aircraft will serve as Headquarters unit.

6.1.2 The Model

The methodology for the modeling of this type 2 variable DMO is similar to the one developed in chapter V, except that the module "interface with the environment" is adapted to this type of variability.

The alphabet X of the inputs consists of occurences of threat which can be either aircraft or missiles. The inputs are uncolored, i.e., modeled by indistinguishable tokens. They are generated in a source place SO and, when their processing has been completed, they are added to the sink SI. A preprocessor (PP) is inserted between the source and the organization (i.e., between the places SO and SO2), whose mission is to associate with every input a color determining the type of environment (and as a result the type of interactions) in which the organization functions. This coloring of tokens can be done by memorizing the N last occurences of the threat, and computing the probability distribution of their identity. In the present case, this distribution is simply given by the probabilities $p = (p_1, p_2)$. If there are N_1 occurences of aircraft among the N threats, then p_1 is simply given by:

$$p_1 = N_1 / N.$$

For any new occurence, p_1 is evaluated, and if it is higher than a certain threshold determined by the designer, the color 1 is associated with the token which models that occurence. Accordingly, if p_1 is lower than the threshold, the token is colored by 2. At the end of the preprocessing, the occurences are therefore modeled as individual tokens of variable s, whose set of identities is $\underline{s} = \{1, 2\}$. The way the color of the incoming token relates to the interaction and the resources used will be described in the switching module part of the methodology.

The decisionmakers are represented by individual tokens of a variable x, which can also be denoted by y. DM1 is then modeled by a token labeled 1 and DM2 by a token labeled 2:

$$\underline{x} = \underline{y} = \{1, 2\}.$$

These two decisionmakers are the only resources of the organization.

The switching module consists of only one transition t, which assigns the role of the expert to either DM1 or DM2 depending on the class of the incoming token; its input places are the output place SO2 of the preprocessor PP and the resource place DM which contains the decisionmakers. The matching process (as described in section 4.3) which occurs in t is the following (see Fig. 6.1):

s = 1: DM1 is an expert and the Headquarters, and DM2 is the novice and the Field Unit. Then the inputs 1 and 2 of variable x are removed from the place DM. x is matched to 1, and y to 2.

s=2: DM2 is an expert and the Headquarters, and DM1 is the novice and the Field Unit. Then the inputs 1 and 2 of variable x are removed from the place DM. x is matched to 2, and y to 1.

This can be summarized in the following operator:

$$(\exists\, s \in SO2) \wedge (\exists\, x \in DM) \wedge (\exists\, y \in DM) \wedge (x = s) \wedge (y = 3 - s) \qquad (6.1)$$

The final representation is shown in Fig. 6.1.

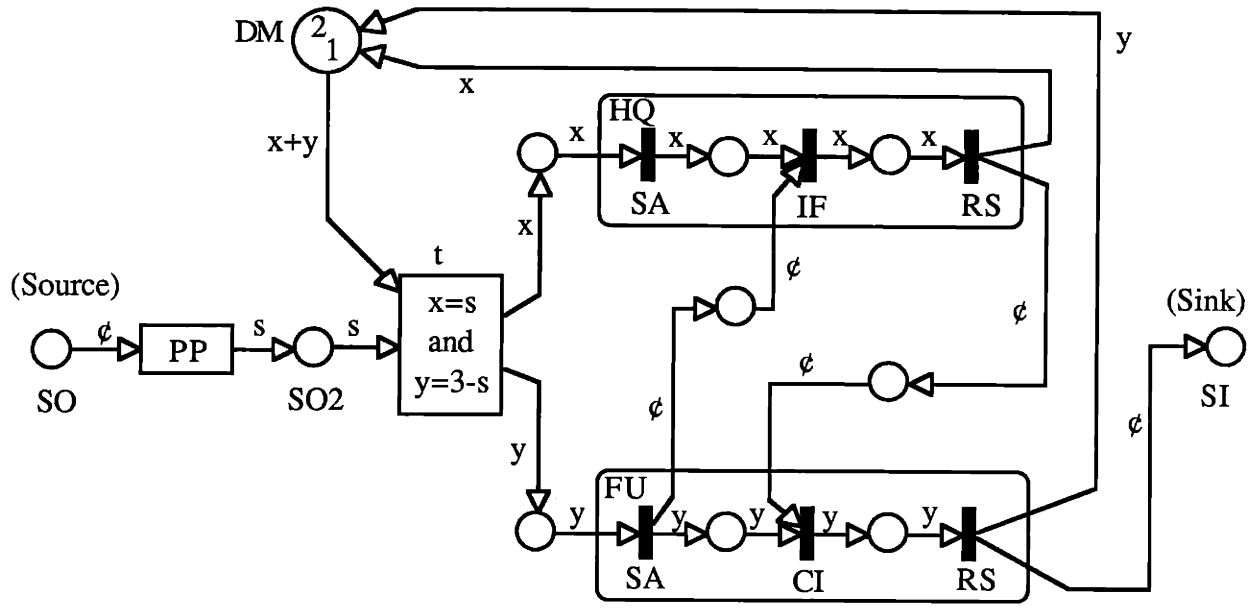


Figure 6.1 A symmetrical organization.

6.2 A DMO WITH INTERCHANGEABLE PARTS AND TYPE 3 VARIABILITY

6.2.1 The Organization

The variable organizations modeled in chapter V associate with each class of inputs, a specific combination of resources together with a specific pattern of interactions. The present section develops the modeling of a DMO which can process the same input in several different ways. The evaluation of the performance of the organization is clearly dependent on the combination chosen, but will not be addressed here.

The organization adapts the pattern of its interactions to the class of the incoming input. As such, it is type 1 variable. If a component (decisionmaker or any other resource) is removed from the organization, then the DMO could use a different combination of resources to still perform its mission. The DMO would then also be type 3 variable.

We consider the same three member organization as the example in chapter V. The inputs belong to the same alphabet X. X is partitioned in four classes, X_1 through X_4. The decisionmakers are the same, namely FU1, FU2 and HQ, with the same possible interactions

between them.

An input of a given class is preferentially processed with the resources R and the interactions Int#(γ) as stated in chapter V:

input of class 1: R = {FU1, HQ}; Int = Int#1.

input of class 2: R = {FU2, HQ}; Int = Int#1.

input of class 3: R = {FU1}; Int = Int#2.

input of class 4: R = {FU2}; Int = Int#2.

The input can also be processed using a different combination of resources and interactions. Each of these possible combinations would lead to a different level of performance for the organization. For example, an input of class 1 can be treated by FU1 and HQ adopting Int#1 as a pattern of interactions, but also by FU2 and HQ with the same pattern, and even by FU1 alone (i.e., with Int#2). The performance of the organization (namely Timeliness and Accuracy) may be very different for these alternatives, but presumably, the first combination would be preferred to the other two.

We assume at this point that for any class of inputs, there is a set of possible combinations of resources and interactions which can process it. These combinations are ordered by some criterion. The way this order relation is established is not defined. Nevertheless, it makes sense to assume that such an order exists: in the example mentioned above, FU1 and FU2 may be interchangeable as far as their interactions with the Headquarters are concerned, but they may also not possess the same set of algorithms to process an input of class 1. FU1 could be a lot more efficient in that respect than FU2. Furthermore, FU1 alone could still perform that task, with a reduced value for an index of performance.

6.2.2 The Model

The representation of a variable organization which can process the same input with different combinations of resources and interactions differs from that developed in chapter V

in the definition of its switching module.

The switching module of the model of the DMO is composed of a set of two transitions t_1 and t_2 which are output transitions of the same place SO (the source). These transitions have attached the operators Op_1 and Op_2 respectively. In the case of a type 1 variable DMO in section 5.3.5, the operators associated with the transitions of the switching module were given by Equation (5.2) (or Eq. 5.3); they consisted of the union (expressed by the logical AND) of conditions to be satisfied so that the operators have a value equal to True.

In the present case, the operators Op_1 and Op_2 are an ordered sequence of operators $(Op_{1,m})$ with m = 1, 2 for Op_1, and $(Op_{2,m'})$ with m' = 1,..., 4 for Op_2. Each of these operators $Op_{1,m}$ and $Op_{2,m'}$ has a formulation similar to Eq. (5.3). Each is True when a certain combination of resources needed for the processing of an input is available:

$$Op_1 = (Op_{1,m})_{m=1,2} \quad \text{and} \quad Op_2 = (Op_{2,m'})_{m'=1,...,4}.$$

Consider an incoming input of class x in the place SO (source). The operators $Op_{1,1}$ in t_1 and $Op_{2,1}$ in t_2 are first examined. They correspond to the combination of resources and interaction between decisionmakers which is the best suited for the processing of the input. Only one of them can have the value True. If this is the case, say, for $Op_{1,1}$, then the transition t_1 is enabled and fires. If none of these two operators has a value equal to True, then the operators $Op_{1,2}$ and $Op_{2,2}$ are examined. They also correspond to a combination of resources and interactions, but coming second (according to some criterion) for the processing of the input of class x. Again, only one of them can have the value True. If it is the case for one of them, the corresponding transition fires. If it is not, the operator $Op_{2,3}$ is examined. If the value of $Op_{2,3}$ is true, t_2 fires and the corresponding resources are removed from their resource places; if it is not, $Op_{2,4}$ is examined. Again, if it is true, t_2 fires, removing the corresponding resources from their resource places. These resources are of course different from the ones involved in $Op_{2,3}$; if they are not, then there is no possible combination of resources available in the organization at this time which would allow the processing of the input of class x; however, as soon as an event modifies the availability of the resources in the organization (for instance when a decisionmaker has just finished processing a task and when the individual token which models him is put back in the corresponding resource place), the whole process starts again and the operators $Op_{1,1}$ and $Op_{1,2}$ are examined again.

The different combinations of resources and interactions which are possible for the processing of an input of class x are given in Table 6.1. For instance, the best suited combination for an input of class 1 is the Field Unit FU1 interacting with the Headquarters HQ with a pattern of interactions Int#1 (combin.#1 in the Table 6.1). The combination of resources which comes second (according to some criterion) for the processing of the input of class 1 is the Field Unit FU2 interacting with the Headquarters with a pattern of interactions Int#1 (combin.#2 in the Table 6.1). This means that FU1 and FU2 are interchangeable to the extent that they can both process an input of class 1 by interacting with the Headquarters. However, the performance of the organization is better ("better" in the sense that it is preferred according to that criterion) if FU1 processes the input than if FU2 does it.

TABLE 6.1 Possible combinations of resources.

		Combinations			
		Combin.#1	Combin.#2	Combin.#3	Combin.#4
Classes of inputs	x=1	Int#1 FU1 + HQ	Int#1 FU2 + HQ	Int#2 FU1	Int#2 FU2
	x=2	Int#1 FU2 + HQ	Int#1 FU1 +HQ	Int#2 FU2	Int#2 FU1
	x=3	Int#2 FU1	Int#2 FU2		
	x=4	Int#2 FU2	Int#2 FU1		

The operator $Op_{1,1}$, for example, expresses in formal language the existence of a token in the place HQ and of an individual token 1 (i.e., FU1) in the place FU if the token in the source place SO is of class 1, or of a token in the place HQ and of an individual token 2 (i.e., FU2) in the place FU if the token in the source place SO is of class 2. $Op_{2,1}$ is deduced from Table 6.1 in a similar way:

$$Op_{1,1} = [(\exists\, x \in SO, x = 1) \wedge (\exists\, s \in FU, s = 1) \wedge (\exists\, \not\in \in HQ)] \vee [(\exists\, x \in SO, x = 2) \wedge (\exists\, s \in FU, s = 1) \wedge (\exists\, \not\in \in HQ)]$$

$$Op_{2,1} = [(\exists\, x \in SO, x = 3) \wedge (\exists\, s \in FU, s = 1)] \vee [(\exists\, x \in SO, x = 4) \wedge (\exists\, s \in FU, s = 2)]$$

The formal expressions of the other operators have been deduced from Table 6.1.

The output connectors of the transitions t_1 and t_2 as well as those which are involved in the subnets Int#1 and Int#2 are conveniently labeled by <x,s> in the case of the present organization; for instance, an individual token <1,1> of the binary variable <x,s>, occupying a place of the subnet Int#2 is an association of an individual token 1 of variable s, representing the decisionmaker FU1, and of an individual token 1 of variable x, representing an input of class 1. This particular input has been associated with FU1 who processes it alone, i.e., without interacting with the Headquarters. The representation of the organization is shown in Fig. 6.2.

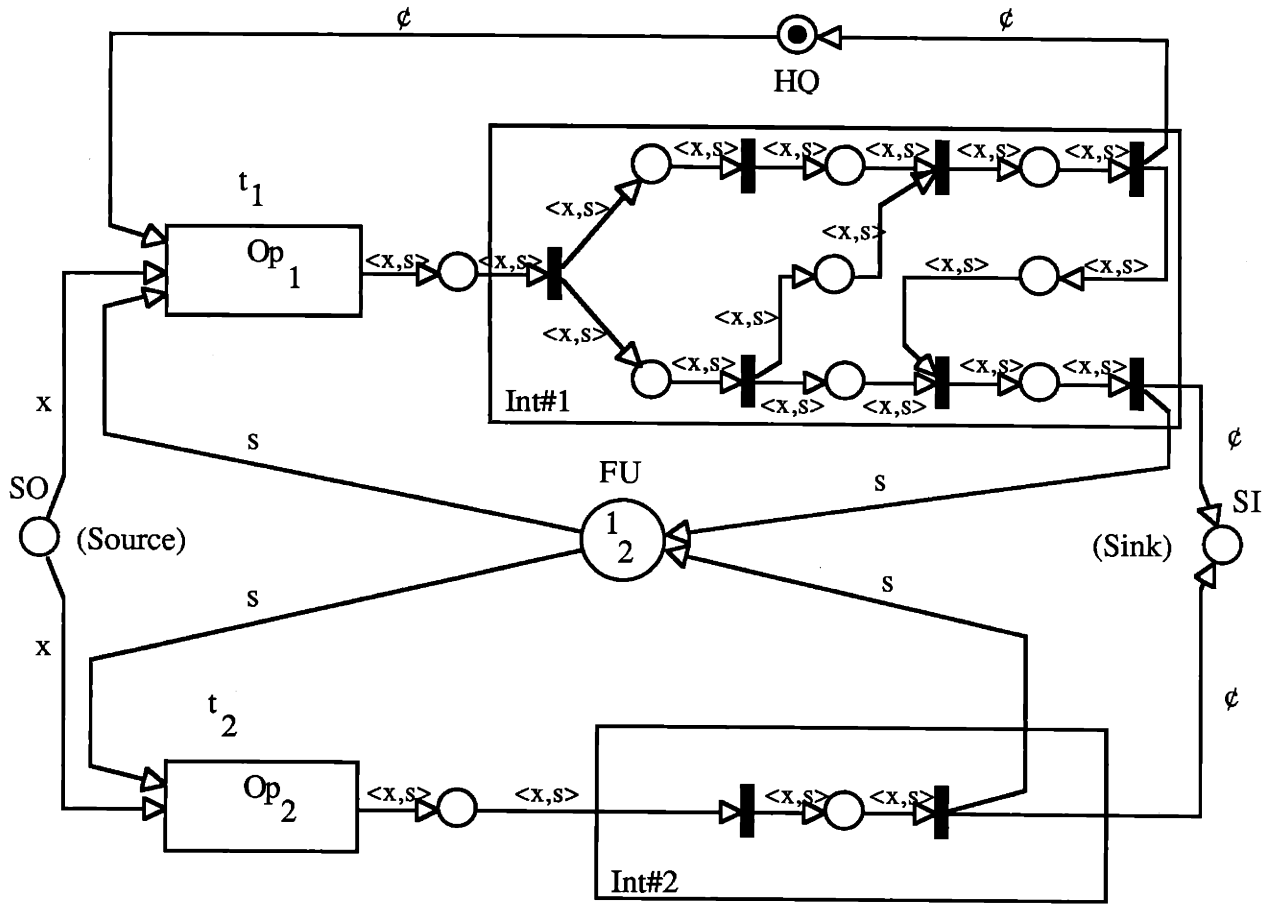


Figure 6.2 Organization with interchangeable parts.

This organization is type 3 variable because it can still perform its mission when changes happen in its resources. Consider for instance the removal from the organization of the Field Unit FU1. The number of possible combinations of resources and interactions is then reduced. An incoming input of class 1 is preferentially processed with the Field Unit FU2 interacting with the Headquarters with the pattern of interaction Int#1. If these resources are not available, then this input can be processed by FU2 alone.

6.3. DMO WITH DECISION SUPPORT SYSTEM AND TYPE 1 VARIABILITY

6.3.1 Organizations with Decision-aids

The variable decisionmaking organizations considered so far have been DMO's where the interactions between decisionmakers can vary. Similar modeling methodologies apply for each type of variability that these VDMO's exhibit. A closely related set of variable decisionmaking organizations can be DMO's in which the interactions between the DM's and the non-human components of the organization (such as decision-aids, see section 2.1) can vary.

When a particular decision-aids can be consulted by only one decisionmaker at a particular stage of its decisionmaking process, this stage can be modeled by a switch whose branches include the algorithms with and without the decision-aid (Weingaertner, 1986). The switch and its branches can then be aggregated in a super-node, and the aggregated net which is obtained is the model of a DMO with a fixed structure.

However, when the same decision-aid is used by several decisionmakers, the model with Petri Net with switches does not work, for identical reasons as described in section 5.2. In that case, the formalism of variable structure organizations can be applied. The methodology developed in chapter V produces convenient models of such organizations.

6.3.2 The Organization

An example of a decision-aid of the type mentioned in section 6.3.1 is the decision support system (DSS) described by Grevet (1987).

A DSS is a local area network composed of intelligent terminals, transmission modules

and a central computer. In an organization supported by a DSS, each DM may have at hand an intelligent terminal. The DM's can use the mainframe computer to benefit from its large computational power or to have access to a common data base. The consultation of the mainframe computer is done through the intelligent terminal and a transmission module.

We consider an organization composed of two decisionmakers, HQ and FU, who interact during the processing of a given input. They each assess the task simultaneously. HQ then fuses the information that FU sends to him with his own, and issues a command to FU, who in turn produces the final response of the organization.

HQ and FU can use the DSS in their Situation Assessment stage. As a result, they have several strategies in their SA stage, which are the same for HQ and FU. HQ, for instance, can do the following:

- assess the situation without using the decision-aids.

- assess the situation with the aid of the intelligent terminal: HQ assesses the situation with an algorithm of his own and consults the intelligent terminal. He then compares the two assessments and produces the final one.

- assess the situation with the aid of the intelligent terminal: HQ takes the assessement of the intelligent terminal. He does not make an independent assessment.

- assess the situation with the aid of the mainframe: HQ assesses the situation with an algorithm of his own and consults the mainframe. He then compares the two assessments and produces the final one.

- assess the situation with the aid of the mainframe: HQ takes the assessment of the mainframe. He does not make an independent assessment.

Again, FU possesses the same set of alternatives in his SA stage.

The ordinary Petri Net model of this organization is depicted in Fig. 6.3, due to Grevet (1987).

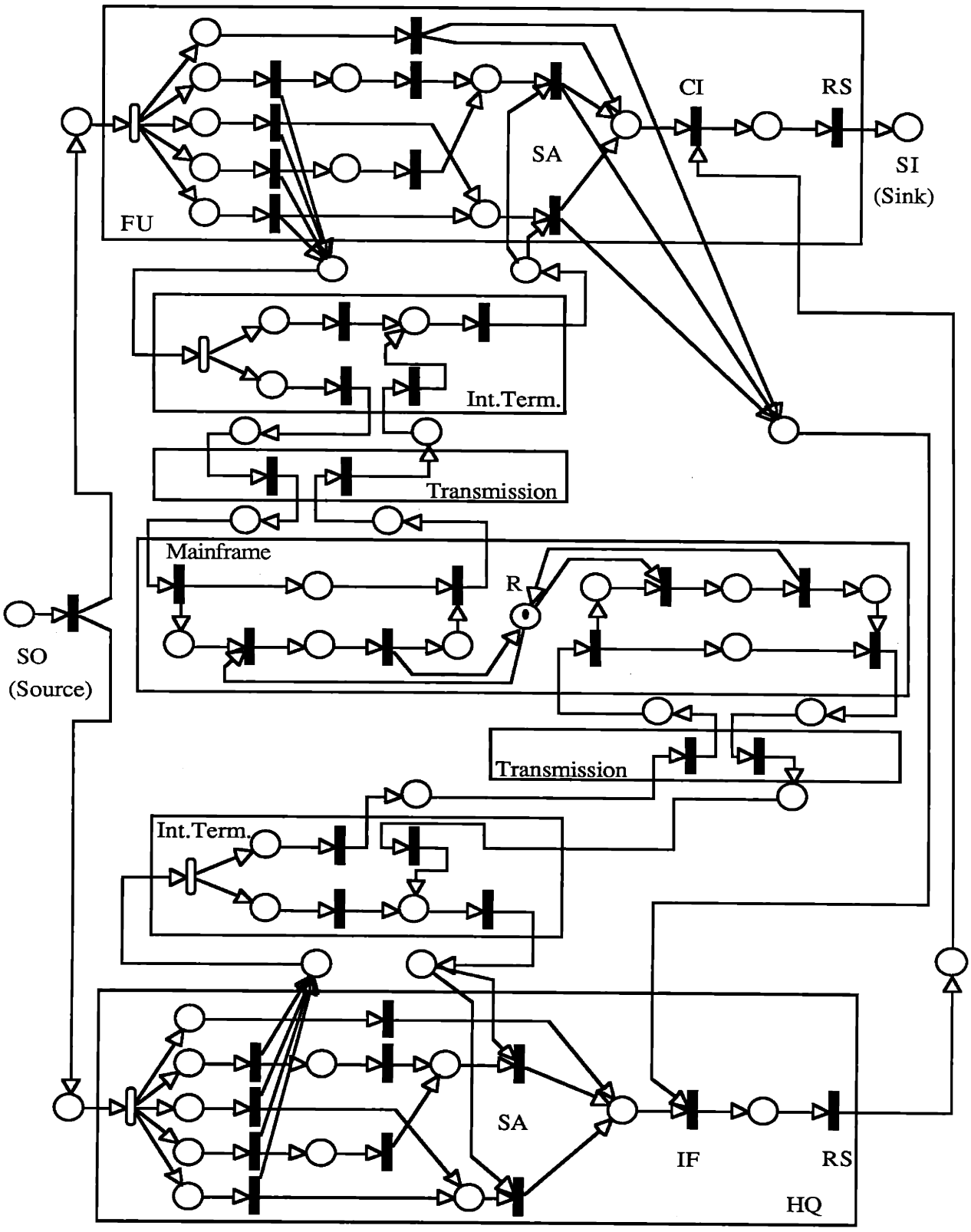


Figure 6.3 Petri Net model of a DMO with a DSS.

The intelligent terminal, the transmission modules and the mainframe are modeled in distinct boxes. The access to the mainframe is limited to one decisionmaker at a time, which is modeled in the Petri Net formalism as a scarce resource with initial marking 1 (place R). This scarce resource makes the two DM's interdependent through the use of the mainframe.

6.3.3 VDMO Model of the DMO with DSS

The organization consists of the two decisionmakers and the DSS. In that sense, it is a VDMO and the methodology developed in chapter V can be applied, or at least modified to account for the variability of this organization. The idea is to represent separately the possible interactions that the DM's can have with their terminals, those that they have with their mainframe, and those between the DM's themselves. The request of a DM to consult the terminal or the mainframe is modeled by an individual token, and so is the decision to execute his own Situation Assessment algorithm (Fig. 6.4).

In that representation, the interactions between DM's are represented with a Petri Net which is Ordinary, i.e., uncolored, except at the SA stages. These stages are modeled by a subnet, with one input transition SA1 and one output transition SA2. The former acts like a source of information (see section 4.4.1), whereas the latter behaves like a sink. For instance, the decisionmaker HQ receives at his Situation Assessment stage an input modeled by an uncolored token ¢. Depending on the strategy he intends to choose, he assigns the attributes s and s' to this incoming token (SA1 stage):

The attribute s accounts for the decision aid which will be consulted:

s = 0: none.
s = 1: intelligent terminal (IT).
s = 2: intelligent terminal (IT) + transmission module (TM) + mainframe (MF).

The attribute s' accounts for the fact that the DM will also execute his SA algorithm:

s' = 0: no personal processing.
s' = 1: personal processing.

We have therefore $\underline{s} = \{0, 1, 2\}$ and $\underline{s}' = \{0, 1\}$.

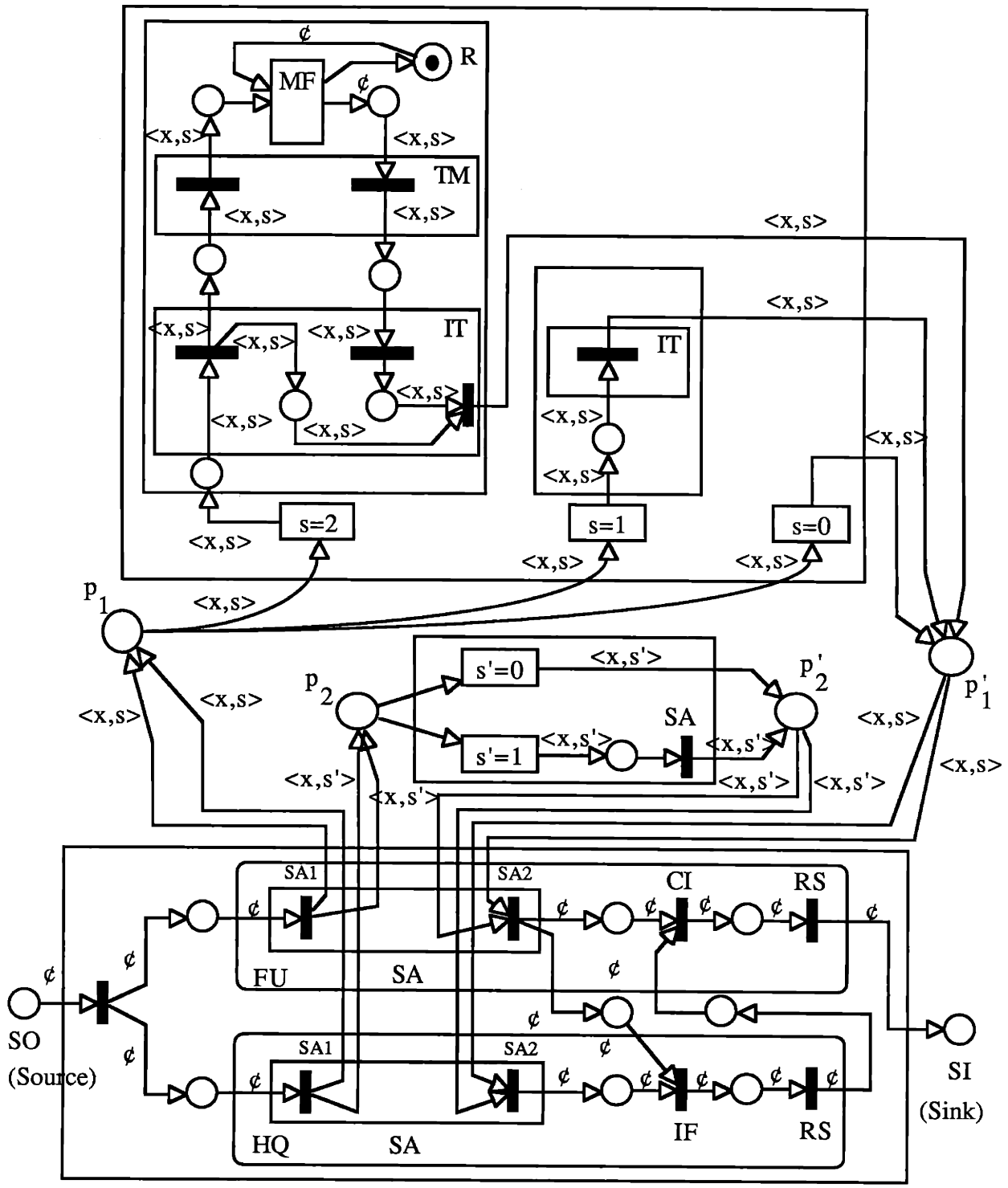


Figure 6.4 PrTN model of the DMO with DSS.

Recall that a decisionmaker is an individual token of variable x, whose set of values $\underline{x}$ is {0, 1}.

Depending then on the strategy chosen, the corresponding individual tokens of variable <x,s> and <x,s'> are created and added respectively to the marking of p_1 and p_2. For instance, if DM1 chooses to consult his intelligent terminal and not to process his own algorithm, in other words, if he chooses to take for granted the assessed situation that the terminal will produce, then a token <1,1> is added to p_1, and a token <1,0> to p_2. These individual tokens will eventually appear in p'_1 and p'_2 after having been through their corresponding subnets. The transition SA2 consists of a set of algorithms, one of which is activated depending on the values of s and s' of the incoming tokens. In this example, the algorithm activated in SA2 in a simple procedure "copy". If the attributes s and s' are respectively equal to 1 and 1, then the algorithm which is activated would be one of information fusion of the two assessements, the DM's and the intelligent terminal's.

This representation of the organization with Predicate Transition Nets is convenient since it avoids the redundancy which occurs when the same kind of interaction is depicted more than once (here twice, once for HQ and once for FU). If another decisionmaker is added to the organization, or if the DM's want to use the DSS in some other stage of their decisionmaking process, then the modification in the net of Fig. 6.4 would only consist of the addition of a set of four connectors between the aided stage and (respectively) the places p_1, p_2, p'_1 and p'_2.

CHAPTER VII

APPLICATION: EFFECTIVENESS OF A TYPE 1 VARIABLE DMO

In chapter 2, the general framework was provided for the evaluation of the effectiveness of a decision making organization, variable or not. In chapter 5, a methodology for the modeling of VDMO's was presented, and it was assumed that the inputs were partitioned in classes, corresponding to specific patterns of interactions, before being processed by the organization. An example of a three member variable organization for which this assumption is relaxed is considered in the present chapter. The organization is first described and modeled with an Admissible Net. The organization is type-1 variable. The Effectiveness of the type 1 variable DMO and the Effectiveness of comparable organizations with fixed structure are then compared. Each organization is associated with a range of mission requirements for which it is the most effective.

7.1 THE ORGANIZATION AND ITS MODEL

7.1.1 The Organization

We consider an organization composed of three decisionmaking units, the Headquarters (HQ) and two Field Units (FU1 and FU2). Its mission is the defense of a given area against aerial threats, aircraft or missiles. Each incoming threat is identified by HQ, and its location determined by both Field Units. HQ communicates then the identity of the threat to the FU's who decide to fire or not to fire, depending on that information.

DMO's with a fixed structure: FDMO1 and FDMO2

Different settings for the interactions between the DM's are possible. In the first case (FDMO1), the HQ and the FU's receive simultaneously the input and HQ sends its information on the identity of the threat to each of the FU's at the same time. They each fuse their assessement of the situation with that information, and give a response to the threat in a simultaneous way. In the second case (FDMO2), only FU1 receives information from HQ, which he fuses with his own assessment of the situation and sends to FU2. FU2 fuses in turn this information with his own assessment and produces the final response of the organization (Figs. 7.1 and 7.2).

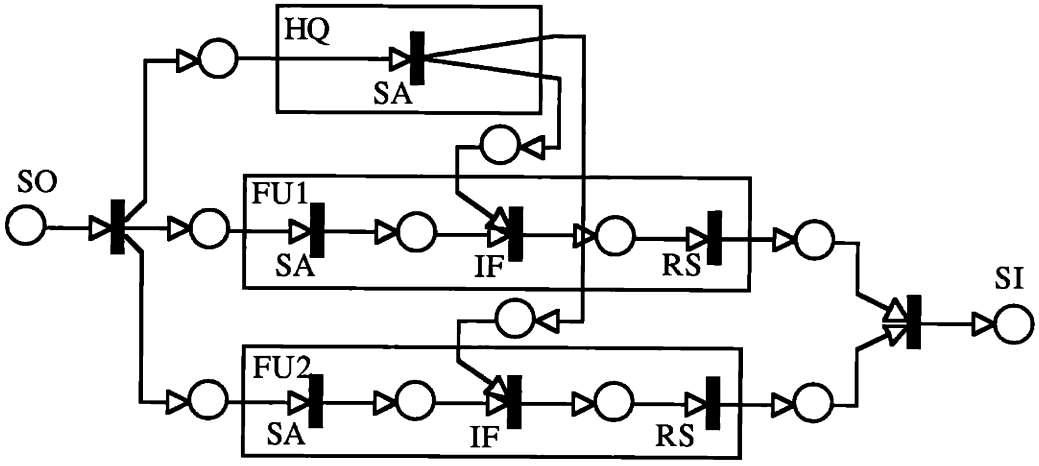


Figure 7.1 Candidate #1: FDMO1.

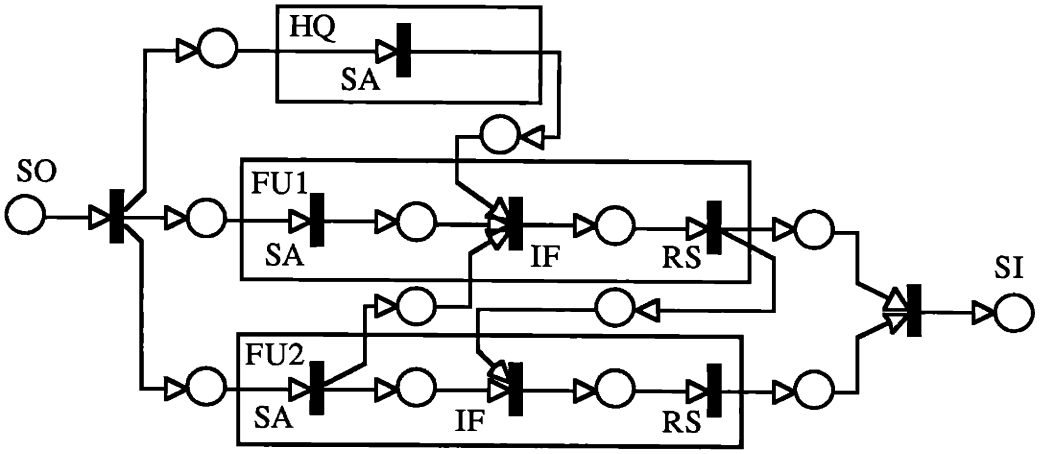


Figure 7.2 Candidate #2: FDMO2.

Type 1 variable DMO: VDMO

In general terms, it is legitimate to suspect that FDMO1 would take less time to respond than FDMO2, since the two Field Units have parallel activities in the first case, but have to interact in the second. However, the same reason may result in the response of FDMO2 being more accurate than the one of FDMO1.

An organization in which the three decisionmakers would concurrently and simultaneously assess the situation, and in which Headquarters would decide the type of interactions to be adopted between the FU's for their final processing, is likely to perform

better; by better performance, we mean lower processing delay and higher accuracy of the response. The organization which would be obtained that way would be **type-1 variable**, and the Headquarters in that case would be the equivalent of the preprocessor mentioned in the methodology developed in chapter 5. The inputs arrive and are indistinguishable; then the HQ attaches to each of them an attribute, or class, which determines the type of interactions that are best suited for their processing. There are, therefore, three candidates for that air defense mission, two organizations with a fixed structure (FDMO1 and FDMO2), and a variable structure organization (VDMO).

PrTN model of the VDMO

The variable organization is modeled with a Predicate Transition Net using the methodology developed in chapter 5. The Situation Assessment stage of the HQ acts as a source of information and associates an attribute u to the incoming token.

What is obtained then is an hybrid representation, using the formalisms of both Ordinary Petri Nets and Predicate Transition Nets. The VDMO is shown in Fig. 7.3. The variable controling the variability is called u, whose set of allowable values has been set to {0,1}. The Situation Assessment stages ogf the Field Units are modeled with the conventional representation. After an input has been processed in these stages, the FU's are modeled with individual tokens of a variable x. The set of allowable values for x is {1, 2}, an individual token 1 (resp. 2) standing for FU1 (resp. FU2).

7.1.2 The Inputs

The three decisionmakers are geographically dispersed. They communicate with the help of wired links or radio. The threats are characterized by their radial distance, i.e., they are modeled as occurences on a line. Their position on this line is measured by a variable x, $x \in [0, 3]$. They appear one at a time and they are independent. The line is divided in three sectors (Fig. 7.4), namely [0,1],]1,2[, [2,3]. Since the Field Units are placed close to the extreme sectors, they perform the same algorithm which determines the position of the target on the line but with different accuracy, depending on the sector in which the target appears. For instance, FU1 is accurate when a threat appears in [0,1], less accurate when it appears in]1,2[, and even less when in [2,3]. The accuracy of FU2 is symmetrical to FU1's.

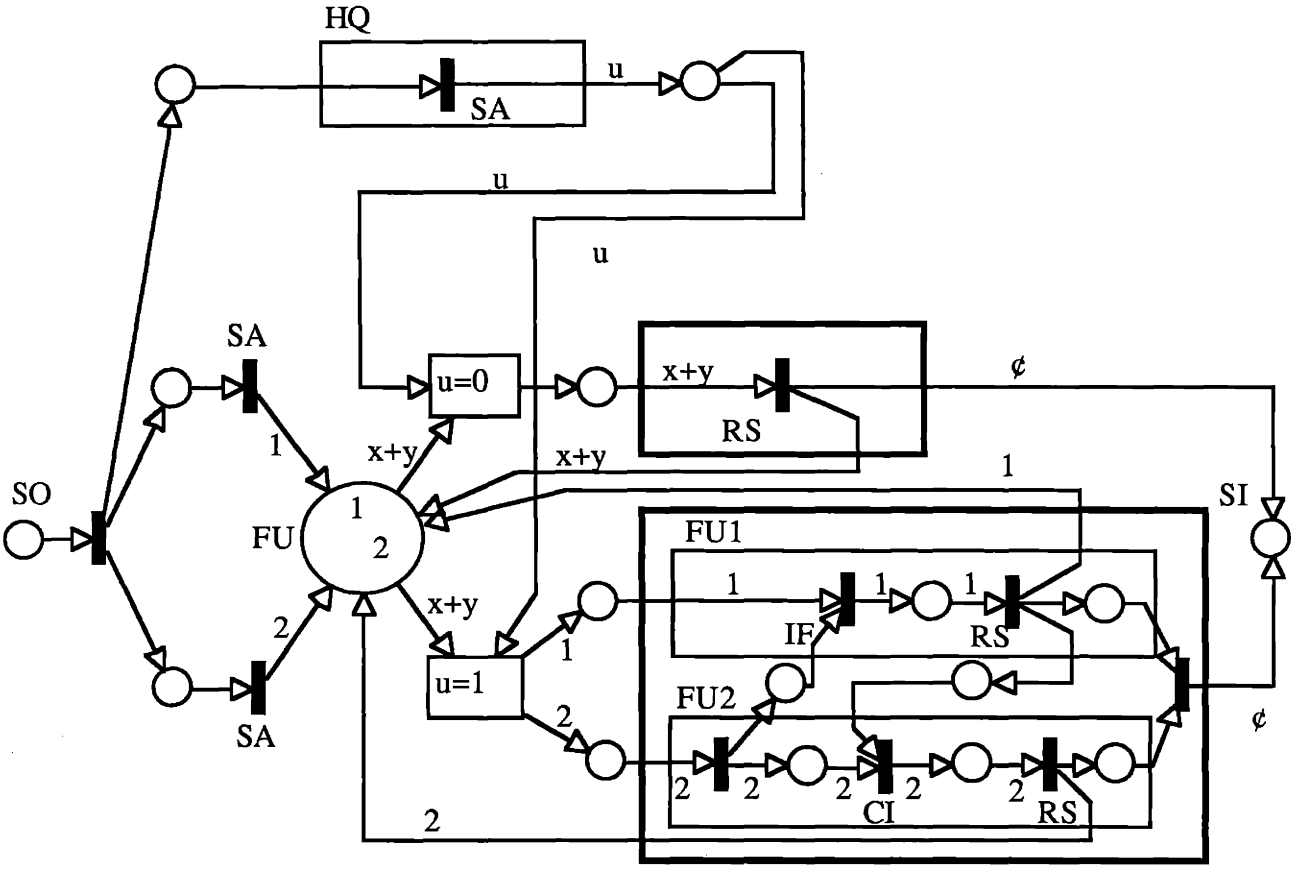


Figure 7.3 Candidate #3 VDMO.

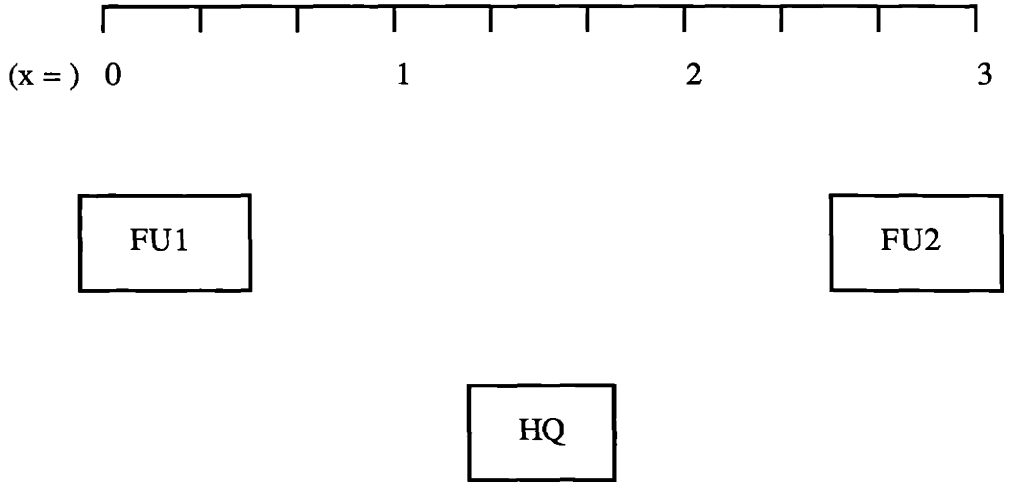


Figure 7.4 Topology of the organization.

The inputs are instances of elements x of an alphabet X. A given instance is modeled by the pair x = (z, Name), where z is a real in [0,3] and Name is a string in {00, 10, 01, 11}.

The name of the input represents the identity of the threats. They can be thought as being types of aircraft, or types of behavior. The threats whose Name is 00, 01, or 10 represent Foes, and have to be destroyed. Only 11 is Friend.

The position of the threat on the line is denoted by z. This is the actual position, but the Field Units, who are in charge of determining it, only achieve their own measure [z] of z. In other words, each of them has an interval (of uncertainty) for the value of z. The accuracy of their measure decreases with the remoteness, and so does the length of the interval. In order to keep the computations simple, the position z in [0, 3] is discretized such that only 30 different positions are allowed, namely 1, 2, ..., 30. Any input which appears actually in [0.1*(i - 1), 0.1*(i)[is called z_i, where i is an integer between 1 and 30. For completeness, the last interval is [2.9, 3.0].

Consequently, the alphabet X consists of elements $x = (z_i, Name_j)$, with:

$z_i \in \{1, 2, ..., 30\}$
$Name_j \in \{00, 01, 10, 11\}$, for $j = 1, ..., 4$.

7.1.3 Strategies of the DM's and Cost Matrix

For any incoming input x_i, the Field Units determine a measure of the position of the threat, and the Headquarters identifies its Name.

Situation Assessement of the Field Units

Each FU's has the same set of two algorithms in the SA stage, called SA1(FU) and SA2(FU). SA1(FU) is more accurate than SA2(FU), and, as a result, takes more time to produce a response. Each algorithm yields a measure of the position of an input x_j with a precision δ represented by an integer. A precision of 1 means that there is no uncertainty in the knowledge of z_i, and that the measure of its position $[z_i]$ is equal to z_i. The interval of uncertainty is reduced to $\{z_i\}$. A precision of 3 means that the measure $[z_i]$ can be at tany one of hree different positions: $\{z_i - 1, z_i, z_i + 1\}$.

The algorithms used in the Situation Assessment of the Field Units are characterized by the precision of the measure they can achieve. In this model, the precision of the measure is taken as a function of the sector to which the threat belongs: the precision δ is supposed to be a linear function of the remoteness, at least in this range of positions of the threat. The precision of these algorithms is as follows:

-Algorithm SA1(FU), (for FU1):

$$1 \leq i \leq 10 \quad \Rightarrow \quad \delta = 1$$
$$11 \leq i \leq 20 \quad \Rightarrow \quad \delta = 3$$
$$21 \leq i \leq 30 \quad \Rightarrow \quad \delta = 5$$

-Algorithm SA2(FU), (for FU1):

$$1 \leq i \leq 10 \quad \Rightarrow \quad \delta = 3$$
$$11 \leq i \leq 20 \quad \Rightarrow \quad \delta = 5$$
$$21 \leq i \leq 30 \quad \Rightarrow \quad \delta = 10$$

The precision of measurements for FU2 are deduced from the above by symmetry (i.e., $i' \rightarrow (30 - i)$).

The values of the precision δ are quantized so that they are the same wherever the threat appears in a given sector. Their dependence on the distance has been set to account for a rapid decrease in accuracy when the distance increases. The delay of the second algorithm has been set arbitrarily at one unit of time. At this point, we assume that if one obtains a measurement with precision δ but spends T units of time in that operation, then one will require more than 2T units of time to obtain a precision $\delta/2$. Since the first algorithm is twice as accurate as the second one, the processing delay of the first one is set to three units of time.

Situation Assessement of the Headquarters

The Headquarters possesses a set of two algorithms in its SA stage. The first one, SA1(HQ), identifies the name of the threat by reading the two characters of the string. In that case, the threat is completely identified. The second algorithm, SA2(HQ), only reads the first character of the string and is less accurate than the first one. The same argument as

above leads to a processing delay of two units of time for SA2(HQ) and four units of time for SA1(HQ).

Internal Strategies

The set of alternative algorithms that the decisionmakers possess leads to the definition of their **internal strategies**. The variables u_1, u_2, and u_3 are first defined to have their set of values equal to {1, 2}, and to correspond to the settings of the switch of the situation assessement stage of FU1, FU2, and HQ, respectively. The variable u_1 for instance is set to:

$u_1 = 1$ if FU1 processes his input with the algorithm SA1.
$u_1 = 2$ if FU1 processes his input with the algorithm SA2.

u_2 and u_3 are determined accordingly. Now the internal strategy of, say, FU1 is the probability distribution of the variable u_1, as indicated in the following:

$D(FU1) = \{p(u_1 = 1), p(u_1 = 2)\}$.
$D(FU2) = \{p(u_2 = 1), p(u_2 = 2)\}$.
$D(HQ) = \{p(u_3 = 1), p(u_3 = 2)\}$.

A decisionmaker uses a **Pure Strategy** when he always processes the incoming input with the same algorithm. Otherwise, he uses a **Mixed Strategy**. In the present case, each DM possesses two pure internal strategies.

Information Fusion stages

The time delay of the Information Fusion stages is logically a function of the number of the number of inputs to be fused. If two inputs have to be fused, the processing delay is one unit of time. If three inputs have to be fused, the delay will be two units of time. All other algorithms have associated a delay of one.

When the two Field Units fuse their measurements of the position of the threats, the precision is increased, if these measurements are consistent, which in this Thesis is supposed to be the case. If two measurements of a same input with precision respectively δ_1 and δ_2 are fused into a measurement with precision $\delta = Fus(\delta_1, \delta_2)$, then the results are as follows:

TABLE 7.1 Precision of Fused Information.

Fus(1, -)	= 1	Fus(5, 5)	= 3
Fus(3, 5)	= 2	Fus(5, 10)	= 5
Fus(3, 5)	= 2	Fus(10, 10)	= 10
Fus(3, 10)	= 3		

Response Selection Stages and Cost Matrix

The decisionmaker in each Field Unit can either send a missile to the target, or do nothing. If he sends a missile to the place where he has measured the threat to be located, then he can either hit the target or miss it, depending on the accuracy of his measure. The FU's response is denoted as y, the place where the missile is targetted: y can take the values x, if the missile is sent exactly where the target is, ⌉ x, if a missile is sent to a wrong position, and † if no missile is sent.

The ideal response for a Friend input (Name 11) is of course to do nothing, whereas the ideal one for a Foe input is to destroy it. There is, furthermore, a penalty for an over-consumption of missiles. The cost associated to any discrepancy between the ideal and the actual responses is indicated in the following cost matrix:

TABLE 7.2 Cost Matrix.

x_1	x			⌉x			†		
x_2	x	⌉x	†	x	⌉x	†	x	⌉x	†
Foe: Y = D	1	1	0	0	6	6	0	6	6
Friend Y=ND	3	3	1	3	2	1	3	1	0

In that matrix, the left column corresponds to the ideal response of the organization. The top row labeled x_1 indicates the response of FU1, whereas the one labeled x_2 represents the response of FU2. The costs are adjusted to reflect subjectively the ranking of the seriousness of the actual responses of the organization. For example, the ideal response for a Friend input is for the Field Units to send nothing, i.e., x_1 and x_2 to be inactive (†). If one missile is sent to a wrong position, in other words if x_1 = †, and x_2 = ⌉ x, (or the reverse), then the cost of wasting one missile is estimated to be one. The cost of sending one missile and hiting the friendly target is set to be three. These values can be modified to account for any other set of beliefs.

Probability distribution of the inputs

The probability distribution of the occurences of the inputs is assumed to be uniform, unless otherwise specified. The probability for the input x of the alphabet X of having its Name equal to a given $Name_j$ is then 1/4, whereas the probability that this input has a position equal to a specific z_i is 1/30. We have then:

$$p(x = (z_i, Name_j)) = 1/120, \text{ for all } z_i \text{ in } \{1,..., 30\} \text{ and all } Name_j \text{ in } \{00, 01, 10, 11\}.$$

7.2 MEASURES OF PERFORMANCE

7.2.1 Accuracy and Timeliness for Pure Strategies

The previous section has described the parameters which specify the organizations FDMO1 and FDMO2, as well as the internal strategies of the decisionmakers. The performance of the organization is a function of the strategy of the organization as a whole, or **organizational strategy**. It is given by the triplet:

$$S = \{D(FU1), D(FU2), D(HQ)\}.$$

Since the three switches which are present in the organization are in the Situation Assessement stages, the internal strategies are not formulated with probabilities conditioned by the inputs. There are, therefore, eight Pure Organizational Strategies, which are the triplets of the pure internal strategies. These Pure Strategies S_i, i = 1,..., 8, can be defined by the algorithms the DM's are using, as follows (the order is FU1, FU2, HQ):

S_1 = (SA1, SA1, SA1)
S_2 = (SA1, SA2, SA1)
S_3 = (SA2, SA1, SA1)
S_4 = (SA2, SA2, SA1)
S_5 = (SA1, SA1, SA2)
S_6 = (SA1, SA2, SA2)
S_7 = (SA2, SA1, SA2)
S_8 = (SA2, SA2, SA2)

The application of the formulas (2.1) and (2.2) of chapter 2 gives immediately the values for Timeliness T and Accuracy J for FDMO1 and FDMO2, for each Pure Strategy S_i. The results are shown in Table 7.3, with T in units of time.

The type 1 variable organization VDMO which has been considered in section 7.1 adapts the interactions between the Field Units to the inputs that they have to process. In order to do that, Headquarters, in its Situation Assessment stage, associates a variable u with the information that it sends to the Field Units. This variable u will determine the interactions between FU1 and FU2. HQ has indeed many ways to partition the alphabet of inputs in classes. We consider here the case where the inputs are distinguished **on the basis of the sectors in which they have appeared.** HQ is assumed to be able to determine the sectors of occurence of the threat, which the FU's either cannot do, or can do but have to wait for the HQ's command. HQ, therefore, sets the interactions between the FU's to be as in FDMO1 when the threat occurs in the extreme sectors [0, 1] and [2, 3], and as in FDMO2 when the threat is in]1, 2[. In the former case, there is no real need for the Field Units to interact since at least one of them has an accurate measurement of the position of the threat. In the latter case, however, the precision of the measurement is increased because the FU's fuse their information, and, in doing so, reduce the interval of uncertainty of their respective measurements.

When compared to FDMO1, VDMO is likely to have an improved accuracy of response when the threat appears in]1,2[. When compared to FDMO2, VDMO will have a lower response time when the threat appears in the extreme sectors. The results for Accuracy and Timeliness for the VDMO are shown in Table 7.3, for the eight Pure Strategies.

TABLE 7.3 Accuracy and Timeliness for the Pure Strategies.

				FDMO1		FDMO2		VDMO	
	FU1	FU2	HQ	T	J	T	J	T	J
S_1	SA1	SA1	SA1	5.00	3.5900	9.00	2.4838	6.67	3.3944
S_2	SA1	SA2	SA1	6.00	2.8167	10.00	1.9583	7.67	2.6271
S_3	SA2	SA1	SA1	6.00	2.8167	10.00	1.9583	7.67	2.6271
S_4	SA2	SA2	SA1	6.00	2.1759	10.00	1.4167	7.67	2.000
S_5	SA1	SA1	SA2	7.00	3.0500	11.00	2.4167	8.67	2.8056
S_6	SA1	SA2	SA2	7.00	2.0833	11.00	1.1250	8.67	1.7292
S_7	SA2	SA1	SA2	7.00	2.0833	11.00	1.1250	8.67	1.7292
S_8	SA2	SA2	SA2	7.00	1.3056	11.00	0.5625	8.67	1.0625

7.2.2 System Locus and Comments

A behavioral Organizational Strategy is constructed by considering the probability distributions of choosing a particular algorithm at each switch. In the present case, such a strategy is completely defined by the triplet $\lambda = (\lambda_1, \lambda_2, \lambda_3)$, where the λ_i's are the parameters describing the (binary) mixed strategy of each decisionmaking unit (Boettcher and Levis, 1982):

$$\lambda_i = p(u_i = 1).$$

The resulting strategy space for the organization is the set $[0, 1]^3$. The performance measures of the organization for the Pure Strategies S_i, for i = 1 to 8, are denoted by T_i and J_i. Timeliness and Accuracy, as defined in chapter 2, are linear functions of the mixed strategies of the individual decisionmakers. $J(\lambda)$ (and accordingly $T(\lambda)$) is computed for any behavioral strategy λ as follows:

$$\begin{aligned} J(\lambda) = \; & \lambda_1.\lambda_2.\lambda_3.J(S_1) + \lambda_1.(1-\lambda_2).\lambda_3.J(S_2) + (1-\lambda_1).\lambda_2.\lambda_3.J(S_3) \\ & + (1-\lambda_1).(1-\lambda_2).\lambda_3.J(S_4) + \lambda_1.\lambda_2.(1-\lambda_3).J(S_5) + \lambda_1.(1-\lambda_2).(1-\lambda_3).J(S_6) \\ & + (1-\lambda_1).\lambda_2.(1-\lambda_3).J(S_7) + (1-\lambda_1).(1-\lambda_2).(1-\lambda_3).J(S_8). \end{aligned}$$

The system loci for the two organizations with a fixed structure, i.e., FDMO1 and FDMO2, are depicted in Fig. 7.5. They are disjoint, and no matter what Organizational Strategy is used in any of the two organizations, FDMO2 needs more time to respond. As

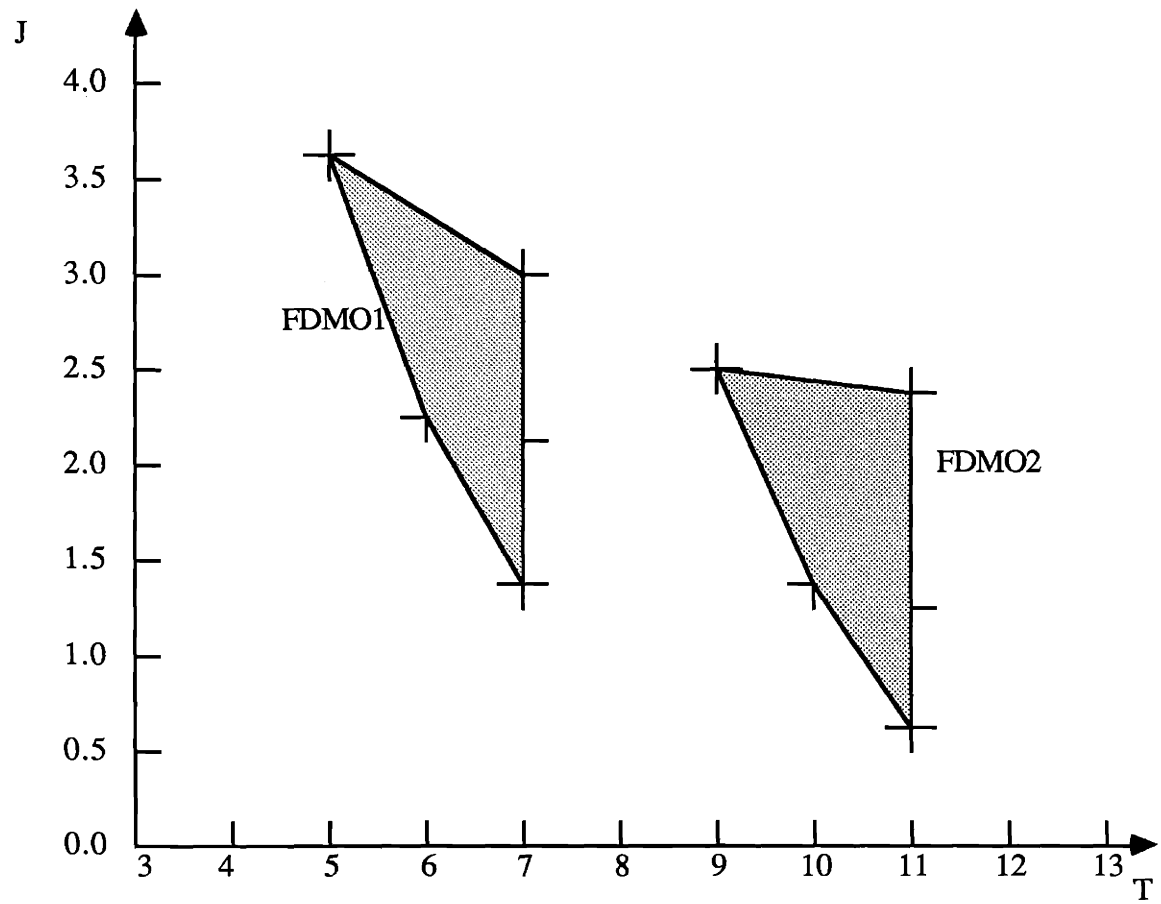


Figure 7.5 System Loci for FDMO1 and FDMO2.

indicated in Fig. 7.5, the whole locus for FDMO1 is to the left of the line T = 7 units of time, whereas the one for FDMO2 is to the right of the line T = 9 units of time.

The same methodology as for the organizations with fixed structure applies for the organization with a variable structure VDMO. The system locus of VDMO is shown in Fig. 7.6. As expected, the variable structure organization is, on the average, faster to respond than the fixed structure organization in which the Field Units have to interact (FDMO2), precisely because they do not always interact in VDMO. VDMO is also, on the average, more accurate than FDMO1, since the FU's in the VDMO interact as needed to improve their measurements of the position of the target.

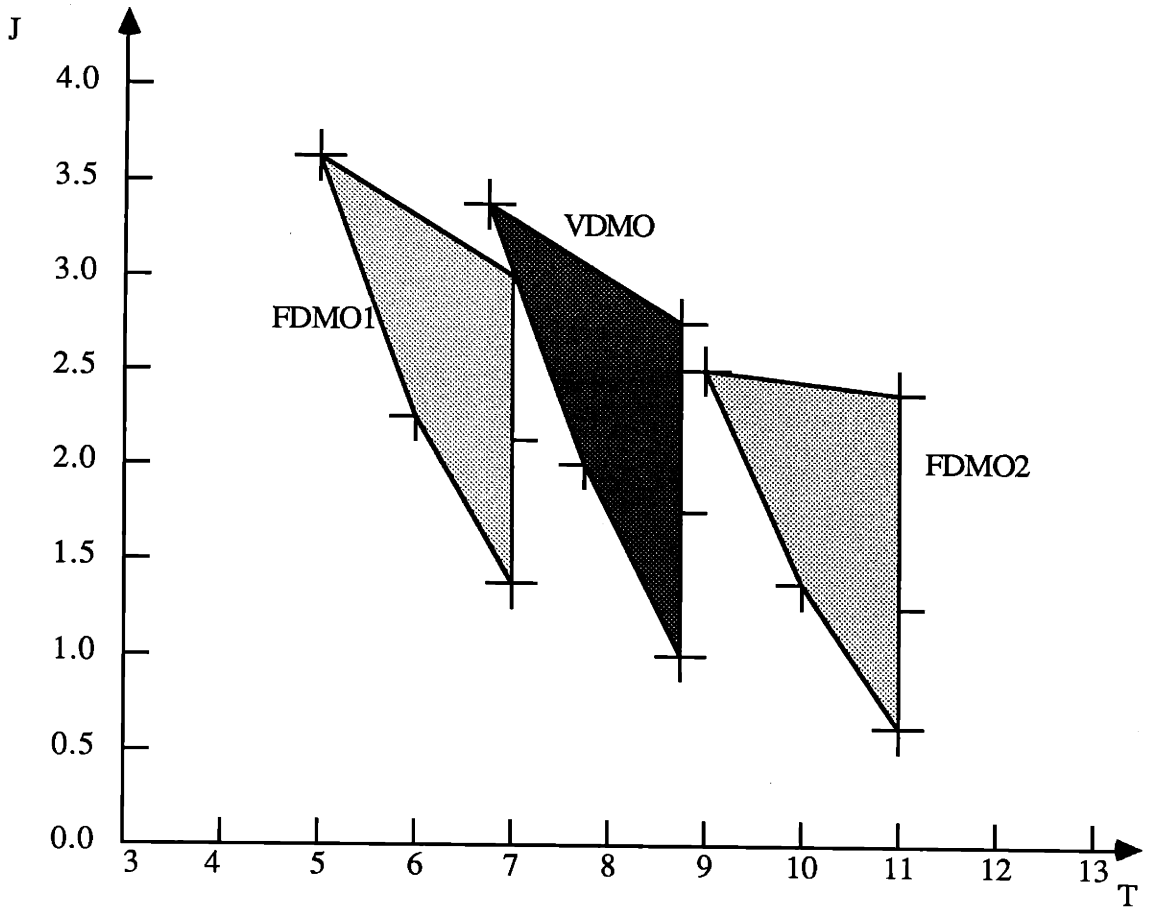


Figure 7.6 System Loci for VDMO.

In the observations that can be made on these plots, one has to be aware that each system locus has associated a probability distribution f (see section 2.1.4), which indicates how probable it is to attain a given set of the MOP's, T and J.

The computation of the performance of an organization for any behavioral strategy and the representation of its system locus are not sufficient to allow the designer to select the best organization among a set of candidates. The mission the organization has to fulfill has to be taken into account. In the next section, the Effectiveness of the three organizations that we have considered in the achievement of their common mission will be evaluated.

7.3 MEASURES OF EFFECTIVENESS

7.3.1 Diagrams of Consistency

The mission of an organization is described in terms of a pair (T^0, J^0) of constraints on its performance. As described in chapter 2 (see section 2.1.4), a convenient representation of the Effectiveness of a DMO is a three dimensional locus $(T^0, J^0, E(T^0, J^0))$, called **diagram of consistency**. In such a locus, $E(T^0, J^0)$ is the percentage of strategies for which the performance of the DMO (T, J) meets the requirements of the mission $(T \leq T^0, J \leq J^0)$. $E(T^0, J^0)$ takes a value between 0 and 1, 0 corresponding to no strategy at all satisfying the mission, and 1 meaning that all admissible strategies lead to satisficing performance.

The diagrams of consistency for the three candidate organizational structures are depicted in Fig. 7.7 (for FDMO1), Fig. 7.8 (for FDMO2) and Fig. 7.9 (for VDMO). They have been obtained with the Locus module of the CAESAR system. In these diagrams, the variables X, Y and Z are matched respectively to J, T, and E. The figures show clearly that FDMO1 has a higher effectiveness than any of the other two organizations in the region of stringent constraint on Timeliness and loose constraint on Accuracy. FDMO2, on the contrary, is the most effective when the mission requires high Accuracy.

For any given design candidate, such locus provides some insight on the shape of the probability distribution f(T, J) (as defined in section 2.1.4). However, it does not allow an easy comparison of different designs of organizations achieving the same mission. A tool for that comparison will be derived in the next section.

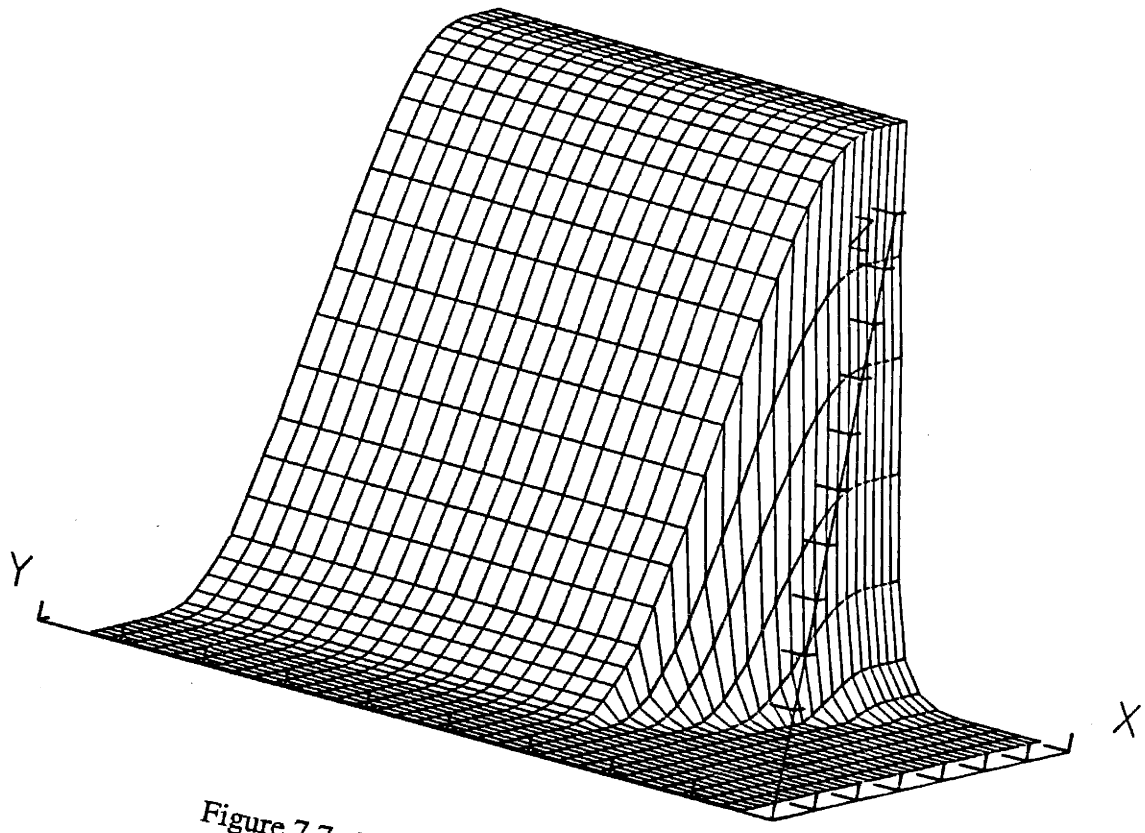


Figure 7.7 Diagram of Consistency for FDMO1.

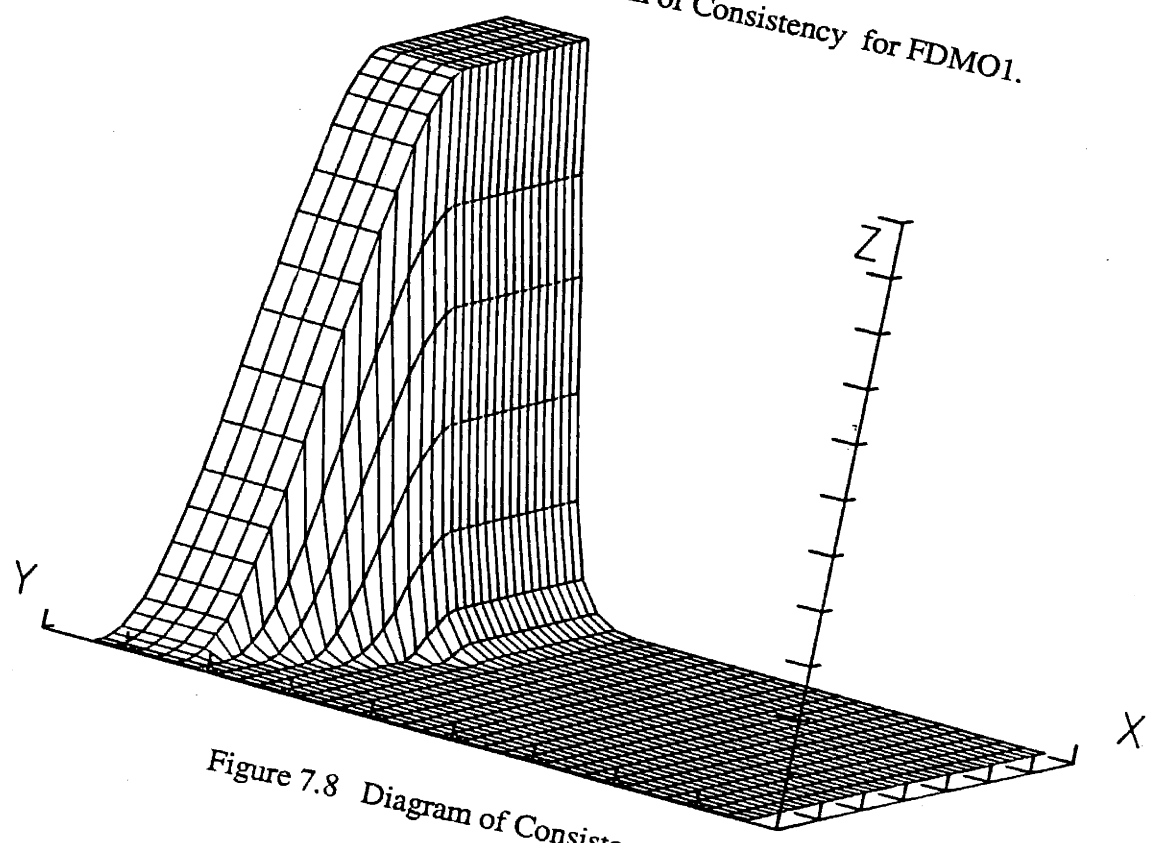


Figure 7.8 Diagram of Consistency for FDMO2.

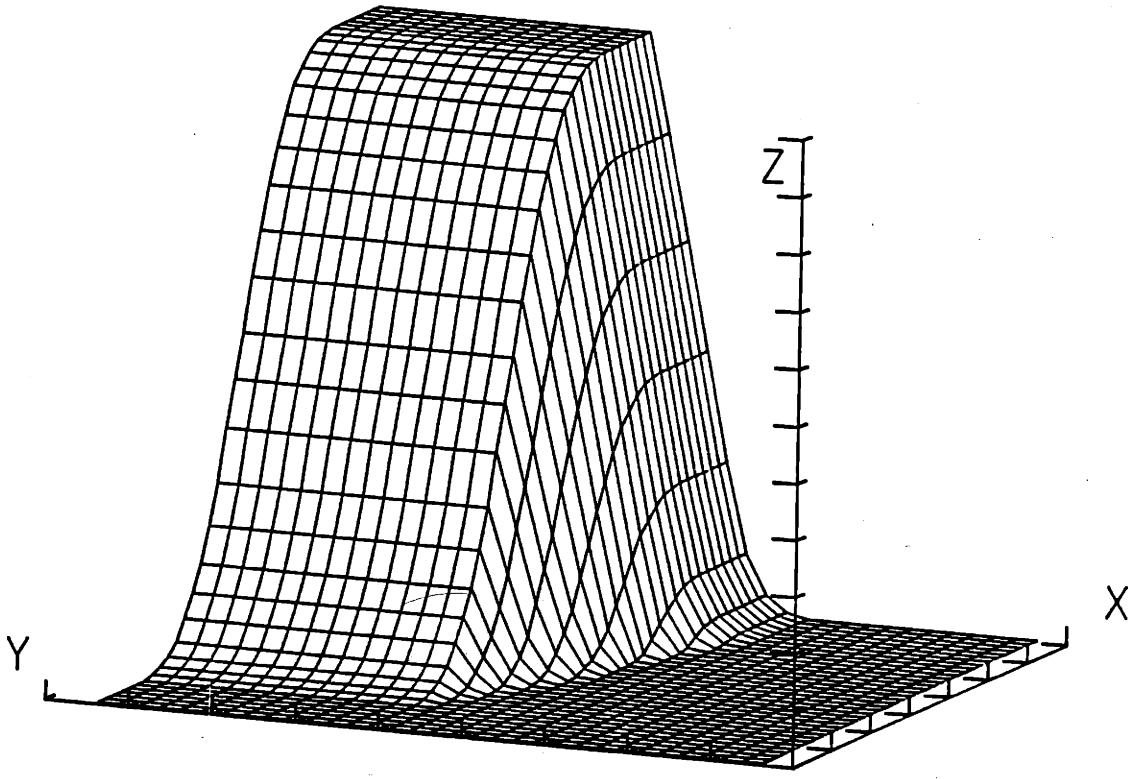


Figure 7.9 Diagram of consistency of VDMO.

7.3.2 Comparison of Designs

In section 7.3.1, the Effectiveness of each of the three design candidates has been computed, for any given mission defined by its requirements (T^0, J^0). For each pair (T^0, J^0) of mission requirements, the organization which has the highest effectiveness can be selected. More than one organization can, of course, achieve the same Effectiveness. Then each organization has associated a range of mission requirements (T^0, J^0) in the MOP space, such that for any mission requirements (T^0, J^0) within that subset, that organization will have higher effectiveness than all the other candidates. This defines a partitioning of the requirements space (T, J) in areas corresponding to each organization, or set of organizations, if the maximum effectiveness is obtained for several designs for the same mission requirements. This happens when more than one system locus is included into the mission locus.

The computation of the measure of effectiveness E for each design candidate has been done for discrete values of T^0 and J^0. Thirty three values for the Timeliness requirement T^0, ranging from 4.00 to 12.00, and thirty six values for the Accuracy requirement J^0, ranging from 0.50 to 4.00, have been used. This resulting grid of 33x36 values for the effectiveness of each candidate was then used to determine the ranges of mission requirements for which each candidate is the most effective. The precision of the determination of these ranges is of course a function of the size of the grid. This explains, for instance, the occasional piecewise linear border between zones.

Such a partitioning is represented first for the organizations with a fixed structure, FDMO1 and FDMO2. In Figure 7.10, the methodology outlined in the previous paragraph yields four different areas. The first area, with no shading pattern, corresponds to the set of mission requirements for which both FDMO1 and FDMO2 have an effectiveness equal to 0. The system locus and the mission locus are completely disjoint, i.e., there is no organizational strategy that can meet the mission requirements. The area labeled FDMO1 is the one in which FDMO1 is the most effective; its non-zero measure of effectiveness is higher or equal to the measure of effectiveness of FDMO2. The reverse is true for the area labeled FDMO2; here FDMO2 is more effective than FDMO1. In the fourth area, which is labeled FDMO1+FDMO2, both organization have an effectiveness of 1, which means that for both any organizational strategy will meet completely the requirements of the mission. There is no rationale in that case to pick up one organization rather than the other. Note that, in this case, the FDMO1+FDMO2 region is defined by the worst accuracy of FDMO1 and the worst timeliness of FDMO2: $J^0 = 3.59$ and $T^0 = 11.00$.

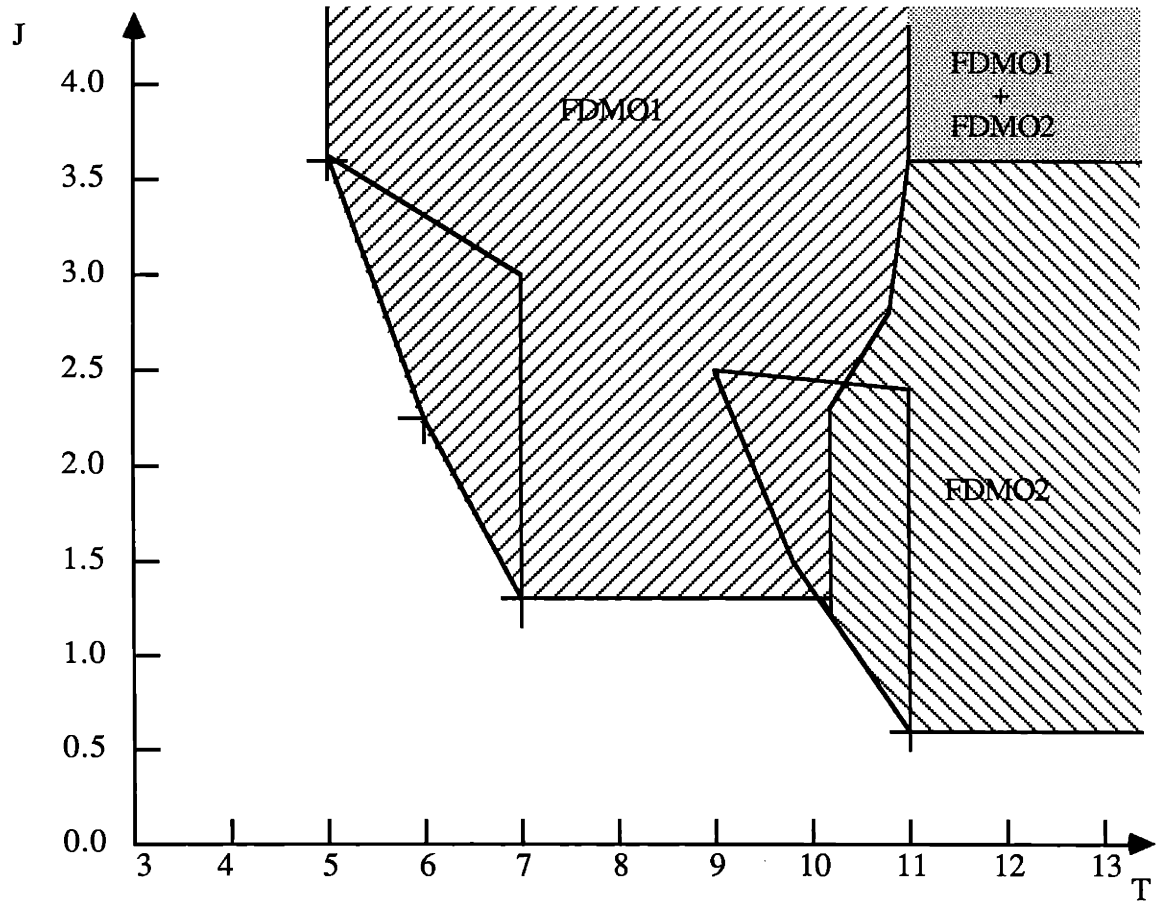


Figure 7.10 Partitioning of the requirements space for fixed structure DMO's.

The same methodology applies when considering the three designs together, namely FDMO1, FDMO2, and VDMO. The results are shown in Fig. 7.11. There are seven subsets of the requirements space, one corresponding to effectiveness equal to zero, when no design meets the mission requirements, one corresponding to each one of the three designs, in which that particular design is clearly the most effective, and three others associated to more than one design. In (FDMO1+VDMO), for instance, FDMO1 and VDMO have an effectiveness equal to 1, whereas FDMO2 is less effective. There is no region corresponding to FDMO1+FDMO2. In the region (FDMO1+FDMO2+VDMO), all three designs meet totally the requirements.

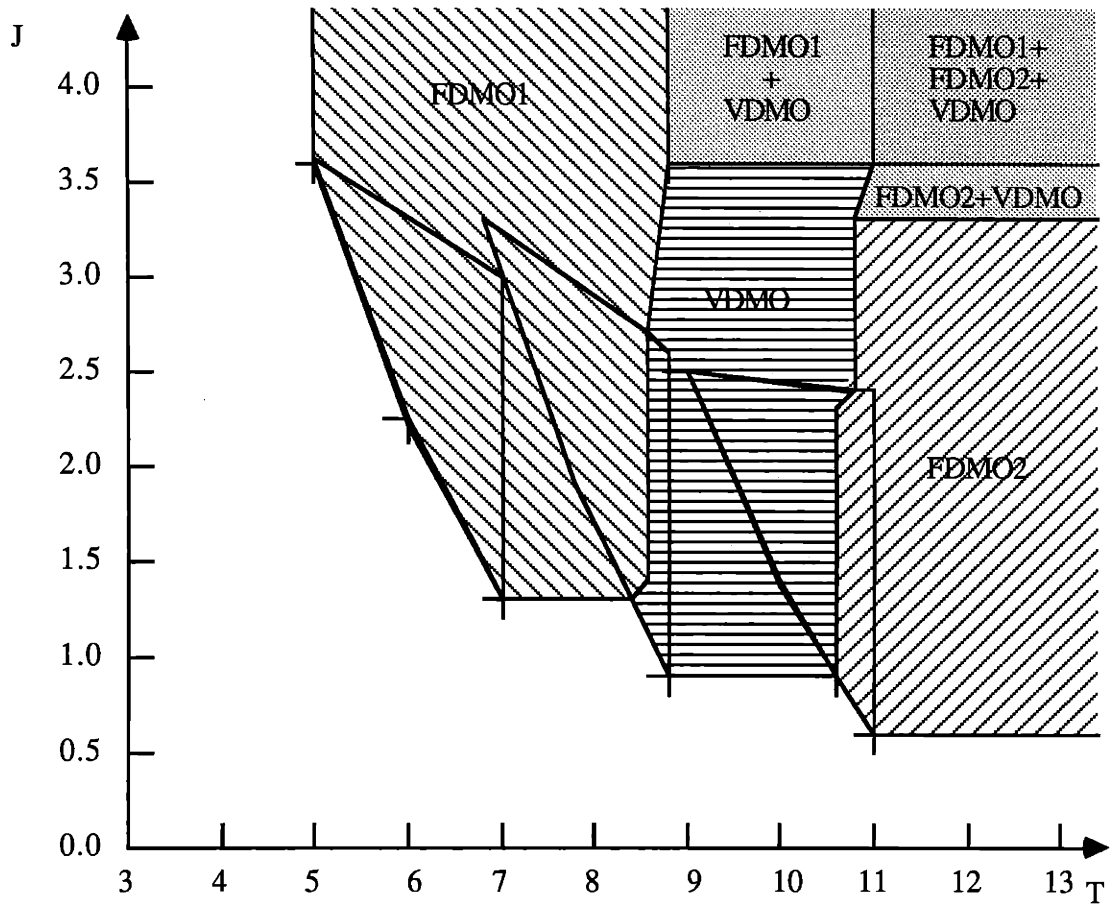


Figure 7.11 Partitioning of the requirements space for fixed and variable structure DMO's

7.3.4 Conclusion

The previous sections have successively modeled a variable structure organization, plotted its system locus, and its diagram of consistency. It has been shown that one can not decide whether a VDMO performs better than an organization with a fixed structure, unless the specific mission requirements are taken into consideration. Then ranges of mission requirements have been identified for which specific organizational designs are most effective. If the requirements are such that the best design is the one with variable patterns of interactions, then the VDMO should be considered. If they do not, there is no need to introduce variability in the DMO's, since a VDMO would not perform any better. A Fixed organizational structure would require a simpler C^3 system to support it.

If the requirements are met both by a variable structure organization and an organization with a fixed structure, then other criteria may be used at this point, such as, for instance, the robustness of a design, which would favor a fixed structure DMO since it is less sensitive to noise or jamming. These criteria have not been addressed in this Thesis, but would constitute the next step toward the modeling of more realistic decisionmaking organizations.

CHAPTER VIII

CONCLUSIONS AND DIRECTIONS FOR FURTHER RESEARCH

8.1 CONCLUSIONS

In this Thesis, variability in decisionmaking organizations has first been defined. More precisely, three different types of variability have been distinguished: a type 1 variable DMO adapts the pattern of its interactions to the class of inputs it has to process. A type 2 variable DMO adapts that pattern to changes in the environment. It has been assumed at this point that the organization has a way to perceive such a change. Finally, a type 3 variable organization adapts that pattern to changes in the nature of the components that it is constituted of, and again the same assumption has been made. This artificial distinction between types of variability has been introduced to facilitate its description, but clearly a given organization can exhibit these three types simultaneously.

The System Effectiveness Analysis methodology has been extended to account for variable organizations. A Measure of Effectiveness has been proposed for each type of variable DMO's. A mathematical formulation for the computation of that MOE has been established.

A modeling methodology has been described providing a representation of DMO's by functions. The main features of that methodology is the decoupling between the pattern of interactions and the identity of decisionmakers, who are modeled by tokens and treated like any other resources. The Predicate Transition Nets formalism has been adapted to allow such representation.

An example of the overall procedure has been presented. It consists of three candidate designs for an air defense task. Each of these candidates is composed of three decisionmakers, namely one Headquarters and two Field Units. Two organizations have a fixed structure, and the third one is type 1 variable; for some tasks, it adapts the pattern of interactions to a pattern comparable to that of the first fixed structure DMO. For some others, it takes the other pattern. The results of the comparison of these designs are that a particular one cannot be selected in general on the basis of its system locus only. The Effectiveness of

each candidate has to be evaluated quantitatively for each set of mission requirements; then zones can be defined in the requirements space which characterize for each organization the ranges of mission requirements for which it is the most effective. In that particular case, the set of mission requirements for which the variable structure organization has the highest Effectiveness has been computed and represented. It shows clearly that a variable structure organization is only preferable to fixed structure ones when the requirements are such that one design is not timely enough, whereas the other is not accurate enough. Type 1 variability seems then to provide a compromise between extreme performance of organizations with fixed structure.

8.2 DIRECTIONS FOR FURTHER RESEARCH

At this point, research could be pursued in many directions to improve and extend the methodology developed in this Thesis. Four different areas for future research have been considered.

8.2.1 Improvement of the Present Model

The present example of the three design candidates can be investigated in more detail. The HQ in the organization, which has been described in chapter VII, has a single strategy to determine the sector in which the threat is located. The impact of alternatives that the HQ could have in that determination would be interesting to assess. In such a configuration, HQ would have several ways to set up the interactions of the rest of the organization and, as a result, the system locus of the variable organization would be likely to expand. In the extreme case where HQ associates a class to the command that he sends to the Field Units at random, i.e., with no rationale for that decision, that locus would include both of those of the organizations with fixed structure.

The same result is likely to occur when there is some noise either in the determination of the position of the target by the HQ or in the communication of the commands to the Field Units.

In either case, the area of the requirements space in which the variable organization is the most effective would shrink, up to a point where it is no more worth it having a variable structure. A substantial effort would be required for the quantification of these qualitative

conjectures.

The investigation of changes in these results when the probability of occurence of the threat is no longer uniform, but when for instance the probability of a threat occuring in any sector is (p_1, p_2, p_3) instead of (1/3, 1/3, 1/3) could give some insight on what a type 2 variable organization would be like. The methodology for the evaluation of effectiveness for type 2 variable organization as developed in chapter II could then be applied in that context.

8.2.2 Impact of the Bounded Rationality Constraint

Another Measure of Performance has been defined in the context of decisionmaking organizations, which is the Workload, or Activity G_i of each decisionmaker DM_i (Boettcher and Levis, 1982). G_i measures the amount of mental effort expended by DM_i in order to perform his task. G_i depends on several factors, including the probability distribution of the inputs, the algorithms used to represent the various processing stages, and the interactions between decisionmakers. The qualitative notion that the decisionmakers are not perfectly rational has been modeled as a constraint (known as the bounded rationality constraint) on their activity G_i:

$$G_i \leq F_i \, . \, \tau$$

where τ is the mean interarrival time of the inputs and F_i the information processing rate that characterizes DM_i. Any given organizational strategy yields a set of MOP's (T, J, G); the bounded rationality constraint determines a part of the system locus that should be avoided.

The methodology which has been used so far in the quantitative evaluation of the Activities of the DM's (Boettcher and Levis, 1982; Andreadakis and Levis, 1987; Weingaertner, 1986) has to be adapted to the particular case of variable organization. The first part of this task would be to set up the right formalism to do it. The second part would be to examine the impact of the bounded rationality constraint on the measure of effectiveness of a variable organization as opposed to the ones of organizations with a fixed structure. The effort required to change the configuration of the interactions between DM's could possibly drive the type-1 variable organizations out of the competition with fixed structure ones.

8.2.3 Dynamics of Variable Structure Organizations

A deeper insight on the improvement that variability can bring in the effectiveness of an organization can be provided when the DMO is considered from a dynamic point of view, as in Hillion, (1986). The order of arrival of the classes of inputs would be as important as the interarrival times. In that context, a type 1 variable organization would allocate dynamically its resources to the class of the incoming input. It is suspected at this point of the research that the introduction of variability in decisionmaking organizations would produce a greater improvement of performance, when compared with the static case. A whole set of problems arises, however, among which is the design of preprocessor which would reorder the inputs to minimize the number of switchings. This area of research would relate closely to queueing theory.

8.2.4 Computation of Invariants

Invariants can be computed on the Predicate Transition Nets, using a methodology introduced originally by Genrich, (1986). An invariant can be defined as a linear function $\mathbf{1}$, from the set of the markings of the net, to the set of the symbolic sums of all the individual tokens which can be found in the net; given a net Π and an initial marking M^0, the value $\mathbf{1}(M)$ of the function $\mathbf{1}$ is the same for each marking M in the reachability set of M^0. It represents a certain content of tokens which remains constant during the firing process.

Not all PrTN's can be obtained through the methodology developed in chapter V. An interesting area of research would be to characterize these nets, and then to develop a methodology for the computation of their invariants. It is suspected at this point that the relations on the markings of the different places of the net that the invariants will provide would correspond to the conservation of the initial markings of the resource places of these nets; in terms of the variable DMO's that these nets model, these relations would correspond to the conservation of the resources utilized by the organization when it functions.

REFERENCES

Andreadakis, S. K. and A. H. Levis (1987). "Accuracy and Timeliness in Decision-Making Organizations," *Proc. 10th IFAC World Congress*, Pergamon Press, New York.

Boettcher, K. L. and A. H. Levis (1982). "Modeling the Interacting Decisionmaker with Bounded Rationality," *IEEE Transactions on Systems, Man and Cybernetics*, Vol. SMC-12, No.3.

Bouthonnier, V. (1982). "System Effectiveness Analysis for Command and Control," S.M. Thesis, LIDS-TH-1231, Laboratory for Information and Decision Systems, MIT, Cambridge, MA.

Bouthonnier, V. and A. H. Levis (1984). "Effectiveness Analysis for C^3 Systems," *IEEE Trans. on Systems, Man, and Cybernetics*, SMC-14.

Brams, G. W. (1983). *Réseaux de Petri: Théorie et Pratique*, Paris.

Genrich, H. J. and K. Lautenbach (1981). "System Modeling with High-Level Petri Nets," *Theoretical Computer Science*, No.13, pp. 109-136.

Genrich, H. J. (1986). "Predicate Transition Nets," *Proc. of the Advanced Course on Petri Nets*, Bad Honnef, Germany.

Grevet, J. L. (1987). "Decision Aiding and Coordination in Decisionmaking Organizations," S.M. Thesis, Laboratory for Information and Decision Systems, MIT, Cambridge, MA.

Hillion, H. (1986). "Performance Evaluation of Decision Making Organizations using Timed Petri Nets," S.M. Thesis, LIDS-TH-1590, Laboratory for Information and Decision Systems, MIT, Cambridge, MA.

Kyratzoglou, J. (1987). "Computer Aided Design for Petri Nets," M.E. Thesis, LIDS-TH-1694, Laboratory for Information and Decision Systems, MIT, Cambridge, MA.

Levis, A. H. (1984). "Information Processing and Decision-Making Organizations: a Mathematical Description," *Large Scale Systems*, No.7, pp. 155-163.

Martin, P. J. F. and A. H. Levis (1987). "Mesures of Effectiveness and C^3 Testbed Experiments," *Proc. 1987 Symposium on C^2 Research*, National Defense University, Fort McNair, Washington DC.

Peterson, J. L. (1981). *Petri Net Theory and the Modeling of Systems*, Prentice Hall, Englewood Cliffs, NJ.

Reisig, W. (1985). *Petri Nets, An Introduction*, Springer Verlag, Berlin.

Remy, P. and A. H. Levis (1987). "On the Generation of Organizational Architectures Using Petri Nets," *Proc. 8th European Workshop on Applications and Theory of Petri Nets*, Zaragoza, Spain.

Remy, P., A. H. Levis, and V. Y. Y. Jin (1987). "On the Design of Distributed Organizational Structures," *Proc. 10th IFAC World Congress*, Pergamon Press, New York.

Rona, Th. P. (1977). "Conceptual Framework for Military C^3 Assessment," Boeing Aerospace Co..

Tabak, D. and A. H. Levis (1985). "Petri Net representation of Decision Models," *IEEE Transactions on Systems, Man and Cybernetics*, Vol. SMC-15, No 6, pp. 812-818.

Weingaertner, S.T. (1986). "A Model of Submarine Emergency Decisionmaking and Decision Aiding," S.M. Thesis, LIDS-TH-1612, Laboratory for Information and Decision Systems, MIT, Cambridge, MA.