JULY 1988 LIDS-P-1795

MODELING AND EVALUATION OF VARIABLE STRUCTURE COMMAND AND CONTROL ORGANIZATIONS*

by

Jean-Marc Monguillet
Alexander H. Levis

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

ABSTRACT

Distributed decisionmaking organizations with variable structure are those in which the interactions between the members can change, or which can process the same task with different combinations of resources. Variable structure could be a possible design solution when no fixed structure organization can meet the requirements of the mission. A modeling methodology is introduced to represent variable structure organizations that is based on the theory of Predicate Transition Nets. Decisionmaking organizations are then viewed from a new perspective in which 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 modeled independently, i.e., the representation of the interactions, resources, and tasks is done separately in separate modules, and modifications in one module can be made without affecting the others. The methodology is illustrated by an example of a three member decisionmaking organization carrying out an air defense task.

*This work 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).

MODELING AND EVALUATION OF VARIABLE STRUCTURE COMMAND AND CONTROL ORGANIZATIONS*

Jean-Marc Monguillet

Alexander H. Levis

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

ABSTRACT

Distributed decisionmaking organizations with variable structure are those in which the interactions between the members can change, or which can process the same task with different combinations of resources. Variable structure could be a possible design solution when no fixed structure organization can meet the requirements of the mission. A modeling methodology is introduced to represent variable structure organizations that is based on the theory of Predicate Transition Nets. Decisionmaking organizations are then viewed from a new perspective in which 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 modeled independently, i.e., the representation of the interactions, resources, and tasks is done separately in separate modules, and modifications in one module can be made without affecting the others. The methodology is illustrated by an example of a three member decisionmaking organization carrying out an air defense task.

1. INTRODUCTION

The need to meet ever increasing performance levels and to satisfy conflicting requirements has led to the investigation of organizations whose structure is variable. Variable structure organizations could be a possible design solution when no fixed structure organization can meet such requirements as robustness or survivability. The modeling of variability in the structure of organizations constitutes another step towards the representation of more realistic decisionmaking organizations.

The mathematical formulation of the modeling and analysis problem 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). The information processing and decisionmaking organizations that have been modeled and analyzed in earlier work (Levis, 1984; 1988) have been depicted as systems performing tasks in order to achieve a mission. These organizations are now 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 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 in separate modules, and that modifications in one module can be made without affecting the others.

In the next section, variable structure organizations are defined, while in the following one the modeling methodology is described. A case study is presented in the fourth section; it illustrates the whole procedure through the design of a set of three candidate structures for a given mission, one of which is variable. Measures of Effectiveness are used to select the most effective candidate for a specific mission.

2. VARIABLE STRUCTURE ORGANIZATIONS

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

The relationships which tie the components together are defined at three different levels: physical arrangements, links between components, and protocols ruling the arrangements of these links. The architecture of the organization allows the topology of interactions to vary. The way it does vary is implemented in the protocols themselves. 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; an actual 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**. Some patterns of interactions may be more suitable for the processing of a given input than others.

* **Type 2 variability**: The VDMO adapts its structure of interactions to **the environment**. The performance of a DMO depends strongly on the characteristics of the environment as perceived by the organization. For example, an air defense organization may be optimized for some types of threats and

*This work 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).

their probabilities of occurence. Now, if the adversary's doctrine changes, or the deployment of his assets changes, then the probability distribution of the occurence of the threats is modified. The organization (with the interactions set as before the changes in the environment) may not meet the mission requirements any more.

* **Type 3 variability**: The VDMO adapts its structure of interactions to **the system's parameters**. The performance of a system changes when assets are destroyed or become unavailable because of countermeasures such as jamming of communications.

These three different types of variability can be related to the properties of **Flexibility, Reconfigurability**, and **Survivability**. A DMO is survivable when it can achieve prescribed levels of performance under some wide range of changes either in the environment, or in the characteristics of the organization, or in the mission itself. The extent to which a DMO is survivable depends on the extent to which it is flexible, and reconfigurable. Flexibility means that the DMO may adapt to the tasks it has to process, to their relative frequency , or to its mission(s) Reconfigurability means that it can adapt to changes in its resources. Both properties overlap, and their quantitative evaluation clearly falls outside the scope of this paper.

The organizations under consideration are restricted to the class of teams of boundedly rational decisionmakers (DM's) (Boettcher and Levis, 1982). Each DM is well trained and memoryless. 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 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. 1.

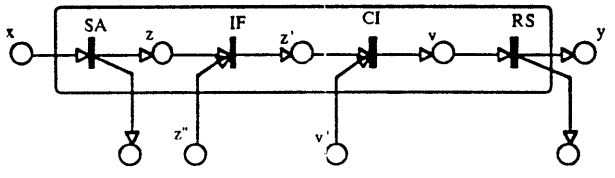

Figure 1 Four stage Petri Net model of a DM.

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. 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 leads to the 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 - the response of the decisionmaker - which is sent to the environment or to other DMs.

As shown in Fig. 1, the decisionmakers can only receive inputs at the SA, IF, and CI stages, and send outputs from the SA and RS stages (Remy and Levis, 1987). The interactions which are the most significant are shown in Fig. 2. For the sake of clarity, however, this figure only accounts for the interactions as directed links from DM_i to DM_j. Symmetrical links from DM_j to DM_i e exist as well.

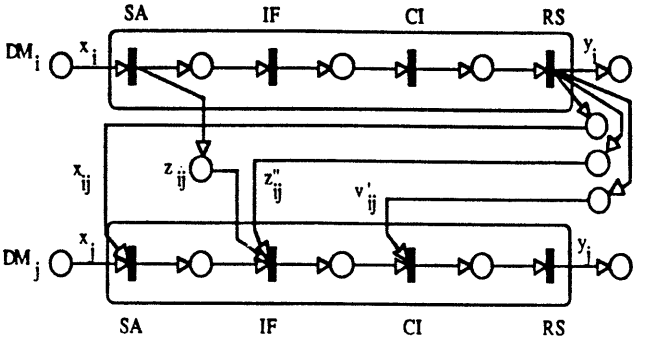

Figure 2 Allowable interactions from DM_i to DM_j.

Two kinds of places can be distinguished: **internal places**, or **memory places**, where the decisionmaker 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**. 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.

In the Petri Net representation, 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.

In the earlier model, the SA and RS stages contained several algorithms and a switch that detrmined the choice of algorithm. The switch position was in turn determined by the decision strategies of each individual DM. The extension of the concept of a switch to model the changing interactions in a variable structure organization turned out not be useful; it introduced a set of problems :

- many switches are needed.

- the needed intercorrelation between the switches cannot be indicated on the net. A table has to be attached to it. Thus, the net representation is not complete.

- the relation between the inputs and the patterns of interactions is not shown explicitly. The 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.

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

Attributes can be used to describe what the tokens represent. For instance, if the decisionmaker has to identify an incoming threat and to respond 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.

What is needed 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.

3. MODELING METHODOLOGY FOR VARIABLE DMO'S

In this section, a step-by-step procedure for the modeling of VDMO's using Predicate Transition Nets is developed. An example of a three member organization with type 1 variability illustrates the methodology. The methodology has a modular architecture (Fig. 3):

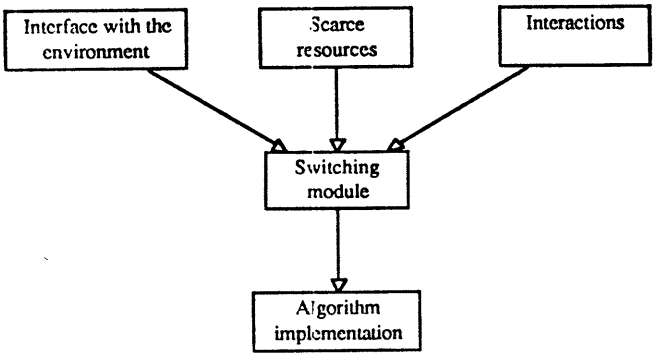

Figure 3 Architecture of the modeling methodology.

There are five modules,

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

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

When the first three modules have been completed, the switching module is executed. The switching module is the part of the model where the logic, which controls the variability of the organization, is implemented. For each incoming input, this is the part of the model which decides what particular resources, and what particular set 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 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 gives precise meaning to what the individual tokens stand for (i.e., the list of their attributes), depending on the places which host them, and on what algorithm, or what set of algorithms, a particular transition models. The processing time of the different algorithms is also specified. The steps of that methodology are independent enough to allow 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.

This 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 Monguillet (1988).

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, the discrete representation of information sets is done in the form of 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; no matter what these values were, the treatment of the token was the same: the interactions between the components were the same.

In the case of type 1 VDMOs, the alphabet X of inputs is 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 implies that it can only be processed with one specific set of resources, and one specific kind of interactions. The identity of a token is the class X_i to which it belongs; it is denoted by the index number i. The variable "class of inputs" is denoted by x and has the following set of allowable identities:

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

Since the environment is not modeled, the tokens which model the outputs of the organization need not have an identity. They are instances of the 0-ary variable ¢.

Example: Step 1

The example consists of 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. 4.

Figure 4 Example - Step 1.

Scarce Resources

Resource is a generic name which designates elements needed for the processing of a task. A resource is scarce when it cannot be allocated freely to the processing of any incoming input because of insufficient or limited supply. 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 multiple input transitions and multiple output transitions, and non-zero initial marking. 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 this modeling methodology, the decisionmakers are treated as scarce 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. 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 indistinguishable tokens. The other resources that the organization may need are partitioned and associated with variables and places in the same way.

Connectors are labeled with a formal sum of variables, which indicates the kinds of tokens they can carry. The input and output connectors of a given resource place R where the corresponding variable is x are labeled by elements of $L^+(x)$, the set of all applications from $\underline{x}$ to the non-negative integers.

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 unique 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 at this point is shown in Fig. 5.

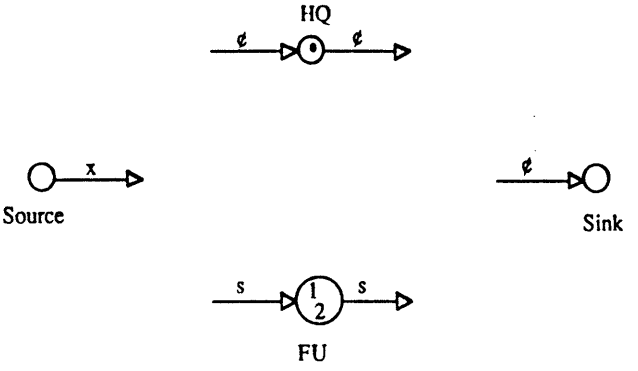

Figure 5 Example - Step 2.

Interactions

The allowable interactions between components are represented without considering 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 typical model obtained at this point is shown in Fig. 6; it is a list of the possible patterns of interactions depicted in their most aggregated form. Had these interactions been considered alone as DMO's with fixed structure, the input and output places would have been the source and sink places.

The possible interactions can be partitioned in four generic types, as illustrated in Fig. 6:

- *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. The Predicate Transition Net which is obtained 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 resource R_2 is not used during the processing of the inputs. In the particular case of Fig. 6(d), it is only needed at the beginning. The ordinary Petri Net modeling that pattern is then aggregated in two super-nodes, (Int#4,1) and (Int#4,2).The former stands for the part of the processing that uses resource R_2, while the latter accounts for the remaining processing.

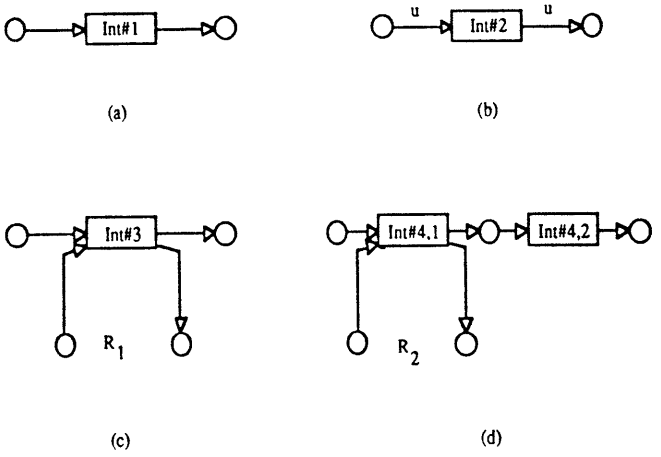

Figure 6 Allowable interactions.

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 make aggregation in super-nodes inappropriate. In that case, the net which would appear in Fig. 6 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 Figure 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). If each switch is aggregated in a macro transition, then the ordinary Petri Nets which are obtained are all marked graphs, i.e., a place can have only one input transition, and only one output transition.

Example: Step 3

In the example, 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. 7). Then an aggregated model comparable to Fig. 6 can eventually be produced.

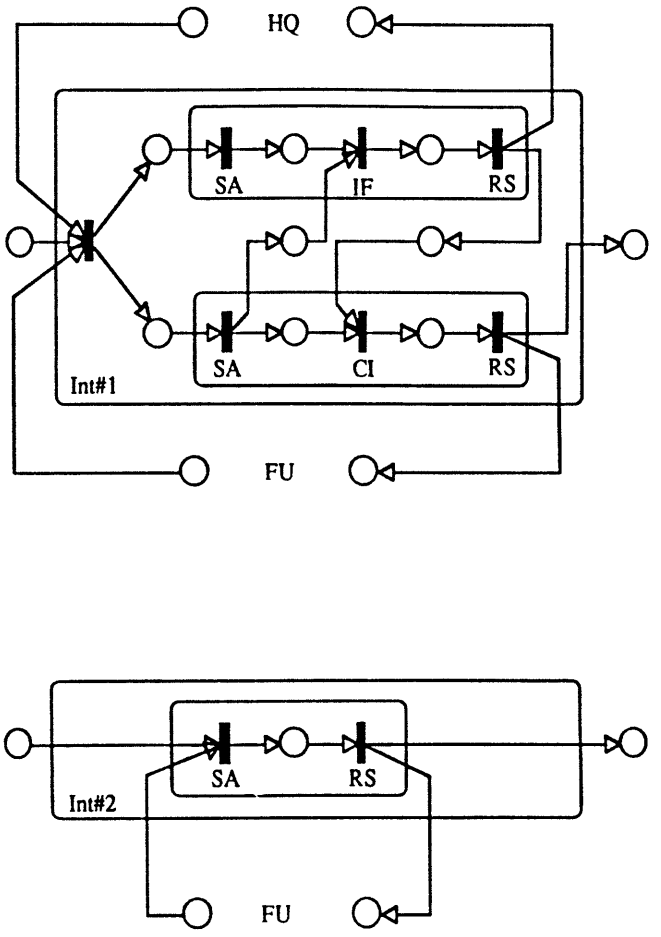

Figure 7 Example - Step 3.

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.

Switching Module

The objective of this module is the representation of the decision rule which determines, for any incoming input, what the actual configuration of the organization will be. The switching module is the part 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 consists of a set of transitions with operators, whose arguments are the individual tokens in the source and resource places. Recall that a DMO with type 1 variability is being modeled, 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 specific 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 i.e., 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:

$$(\exists\, x \in SO) \wedge (\gamma(x) = \phi(j)) \wedge (\exists\, res(k, x), R_k \supseteq res(k, x)) \quad (1)$$

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 (1) 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:

$$Op_j = \bigvee [(\exists\, x \in SO) \wedge (\exists\, res(k, x),\ R_k \supseteq res(k, x))] \quad (2)$$

$$x \in \gamma^{-1}(\phi(j))$$

The operators $(Op_j)_{j=1,...,J}$ which are attached to the transitions t_j - the branches of the switch - are such that the following conflict resolution rule 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 to transition t_j are labeled by the set $L_{conn}(R_k, t_j)$ 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:

$$L_{conn}(R_k, t_j) = \left\{ \lambda \in L^+(s_k) \mid \lambda = \sum_{n=1}^{N(x)} s_{k,n}(x) \text{ and } \gamma(x) = \phi(j) \right\} \quad (3)$$

Example: Step 4

In the example, the switching module contains two transitions t_1 and t_2. Therefore,

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-2)].

In the net obtained up to this point the patterns of interactions, the resources, the source, the sink, and the transitions of the

switch are connected together, and the transitions show the operators assigned to them. The patterns of interactions, however, are still in their most aggregated form, and the connectors are not all labeled (Fig. 8). This net is not yet fully defined. The purpose of the next module will be precisely to make this net functional by completing its annotation.

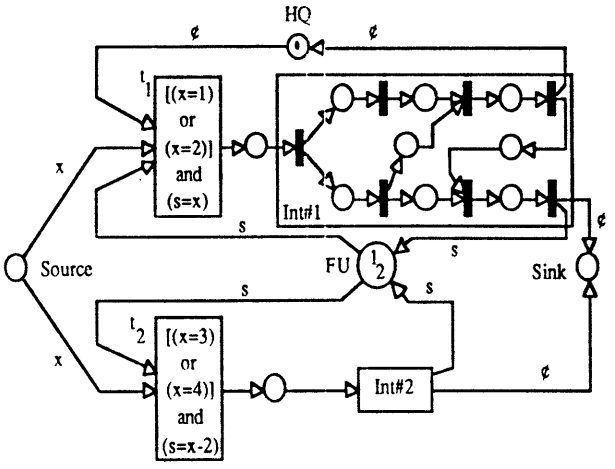

Figure 8 Example - Step 4.

Algorithm Implementation

This fifth module 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 must also be established.

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 Eq. (3). 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 $\gamma(x) = \gamma$. 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:

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

These labeling rules are the most general that can be presented, and can be applied to any case.

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, can allow simultaneous firing; depending on the circumstances, two tokens in the same place can enable the same transition at the same time and leave simultaneously the same place.

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

Example: Step 5

The final representation of the example is given in Fig. 9. Since the organization is fairly simple, a simplified and self-explanatory labeling has been adopted.

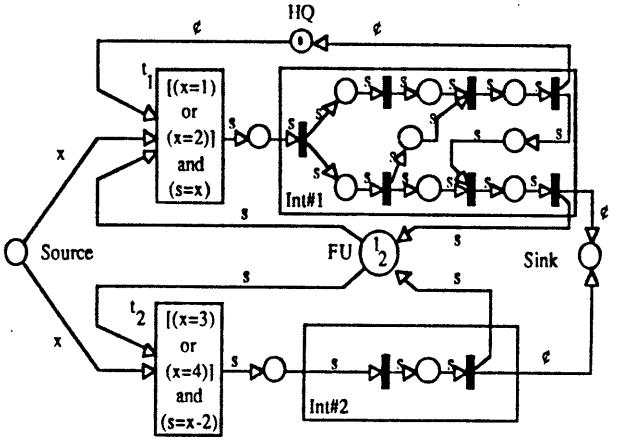

Figure 9 Example - Step 5.

4. EFFECTIVENESS OF A TYPE 1 VARIABLE DMO

In the previous section, 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 structure organization for which this assumption is relaxed is considered in this section.

The organization and its model

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. 10 and 11).

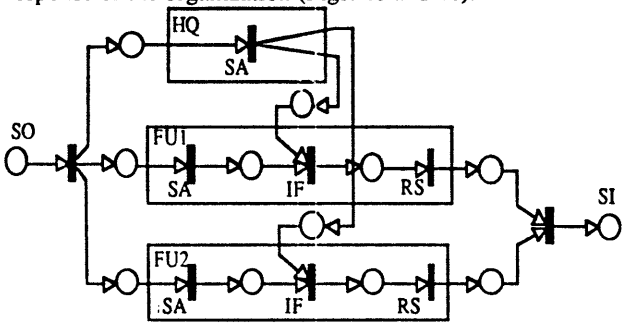

Figure 10 Candidate #1: FDMO1.

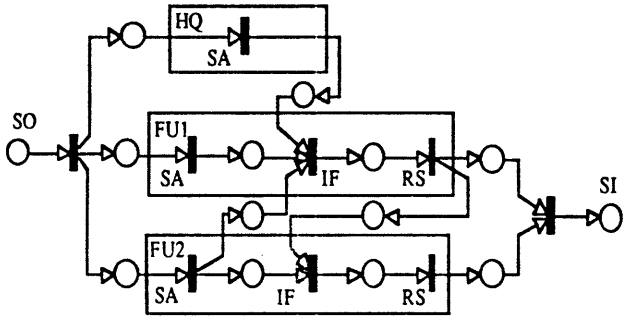

Figure 11 Candidate #2: FDMO2.

Type 1 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; i.e., lower processing delay and higher accuracy. The organization which would be obtained that way would be **type-1 variable**, and the Headquarters in that case would play the role of a preprocessor. 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 the previous section. The Situation Assessment stage of the HQ acts as a source of information and associates an attribute u to the incoming token. What results is an hybrid representation, using the formalisms of both ordinary Petri Nets and Predicate Transition Nets. The VDMO is shown in Fig. 12. The variable controling the variability is called u, whose set of allowable values is {0,1}. The Situation Assessment stages of 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}, with token 1 (resp. 2) standing for FU1 (resp. FU2).

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, namely [0,1[,]1,2[, and [2,3]. Since the Field Units are placed close to the extreme sectors, they perform the same algorithm that 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 analogous to FU1's.

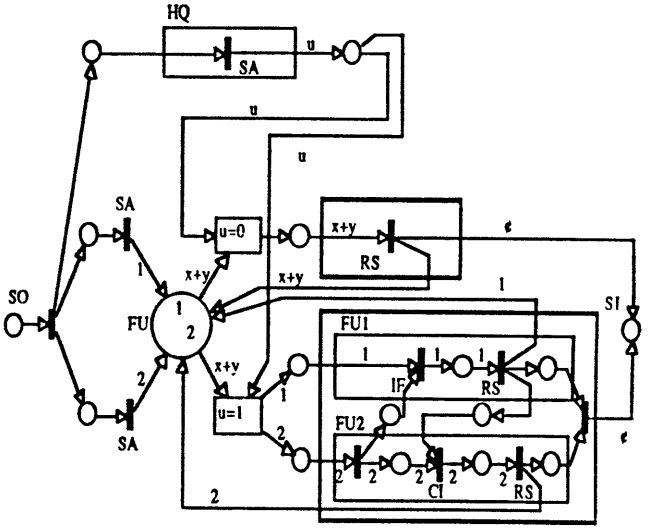

Figure 12 Candidate #3 VDMO.

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 remoteness. 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.

Strategies of the DM's and Cost Matrix

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

Situation Assessement

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 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 any 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 they can achieve. In this model, precision 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.

- Algorithm SA1(FU) for FU1 :

$$\begin{aligned} 1 \le i \le 10 &\Rightarrow \delta = 1 \\ 11 \le i \le 20 &\Rightarrow \delta = 3 \\ 21 \le i \le 30 &\Rightarrow \delta = 5 \end{aligned}$$

- Algorithm SA2(FU) for FU1:

$$\begin{aligned} 1 \le i \le 10 &\Rightarrow \delta = 3 \\ 11 \le i \le 20 &\Rightarrow \delta = 5 \\ 21 \le i \le 30 &\Rightarrow \delta = 10 \end{aligned}$$

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

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 its input with the algorithm SA1.

$u_1 = 2$ if FU1 processes its input with the algorithm SA2.

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

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

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 a function 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, precision is increased, if these measurements are consistent. If two measurements of the same input with precision 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 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 Stage and Cost Matrix

The decisionmaker in each Field Unit can either allocate a missile to the target, or do nothing. If he sends a missile to the position 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 by y, the target coordinates: y can take the values x, if the missile is sent exactly where the target is, $\overline{\text{x}}$, if a missile is sent to a wrong position, and † if no missile is sent.

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

TABLE 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 actual responses of the organization. For example, the ideal response for a Friend input is for the Field Units to take no action, i.e., x_1 and x_2 to be inactive (†). If one missile is targetted to the wrong coordinates, 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 targetting accurately a Friend is three. These values can be modified to account for any other set of beliefs.

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:

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

Measures of Performance

Measures of Performance (MOP's) are quantities which describe the system properties. The MOP's are functions of the system parameters and of the organizational strategy adopted by the organization. The two MOP's considered here are *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:

$$J = \sum_{i=1}^{n} p(x_i) \sum_{j=1}^{p} C(y_d(x_i), y_{aj}(x_i)) \cdot p(y_{aj}(x_i) \mid x_i) \qquad (4)$$

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 the expected value of the processing delay:

$$T = \sum_{i=1}^{n} p(x_i) \cdot T_d(x_i) \qquad (5)$$

The performance of the organization is a function of the strategy of the organization as a whole, or organizational strategy, which 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 Eqs. (4) and (5) gives immediately the values for Accuracy J and Timeliness T for FDMO1 and FDMO2, for each Pure Strategy S_i. The results are shown in Table 3, with T in units of time.

The type 1 VDMO beeing considered adapts the interactions between the Field Units to the inputs that they have to process. We consider 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 3, for the eight Pure Strategies.

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 {p(u1), p(u2), p(u3)}. The resulting strategy space for the organization is the set $[0, 1]^3$. The system loci for the two organizations with a fixed structure,

i.e., FDMO1 and FDMO2, are depicted in Fig. 13. They are disjoint, and no matter what Organizational Strategy is used in any of the two organizations, FDMO2 needs more time to respond. As indicated in Fig. 13, 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.

TABLE 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

The same methodology for evaluating the MOPs applies to the organization with a variable structure, the VDMO. The system locus of VDMO is shown also in Fig. 13. 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.

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. This mission is described in terms of a pair (T^0, J^0) of constraints on performance. 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 $(T \leq T^0, J \leq J^0)$. The effectiveness measure $E(T^0, J^0)$ takes a value between 0 and 1, with 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. 14 (for FDMO1), Fig. 15 (for FDMO2) and Fig. 16(for VDMO). In these diagrams, the variables X, Y and Z correspond to J, T, and E, respectively. The figures show clearly that FDMO1 has a higher effectiveness than any of the other two organizations in the region of stringent constraints on Timeliness but not on Accuracy. Conversely, FDMO2 is the most effective when the mission requires high Accuracy.

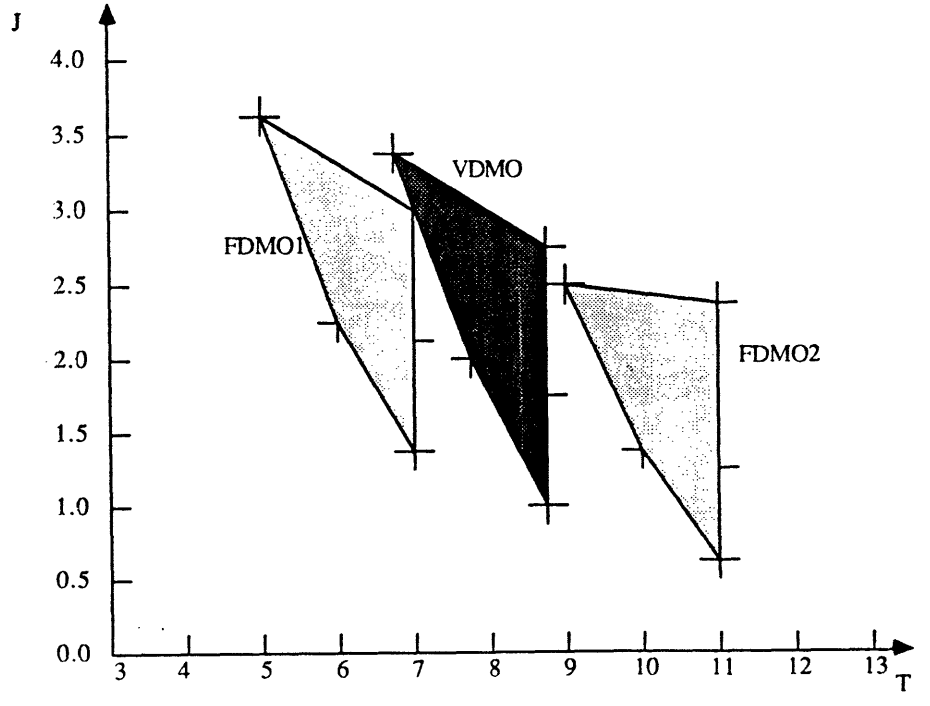

Figure 13 System Loci for VDMO.

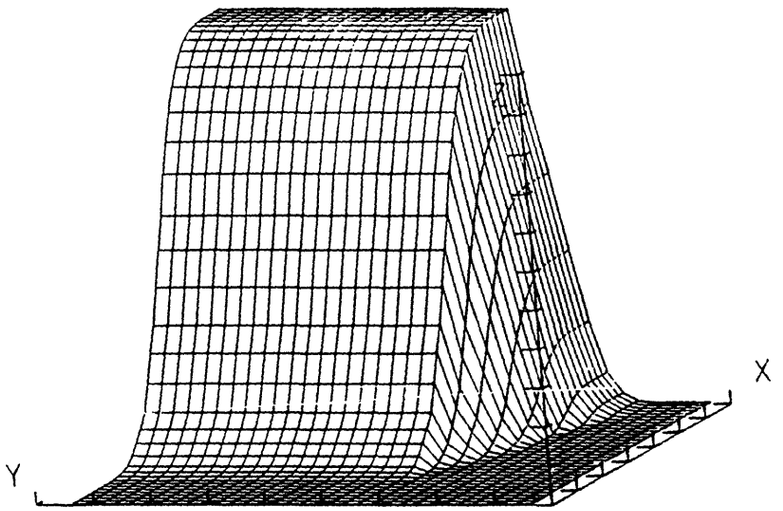

Figure 14 Diagram of Consistency for FDMO1.

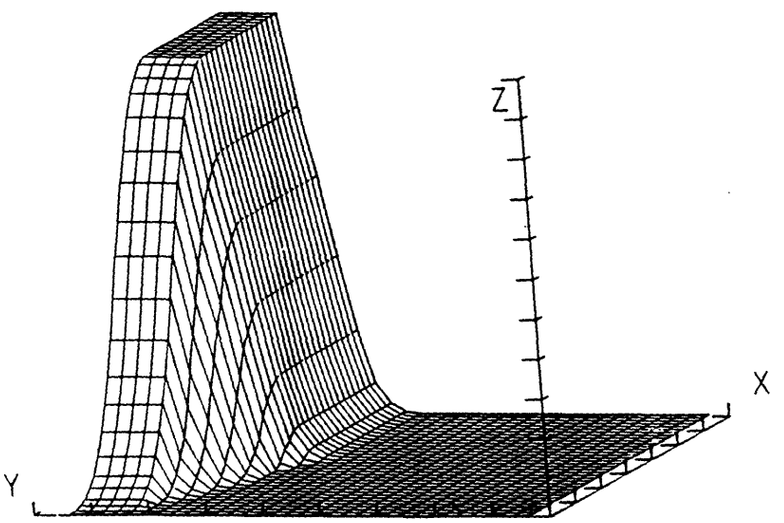

Figure 15 Diagram of Consistency for FDMO2.

Since the Effectiveness of each of the three design candidates has been computed, for any given mission defined by its requirements (T^0, J^0), the organization which has the highest effectiveness for a specific mission 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.

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 33 x 36 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.

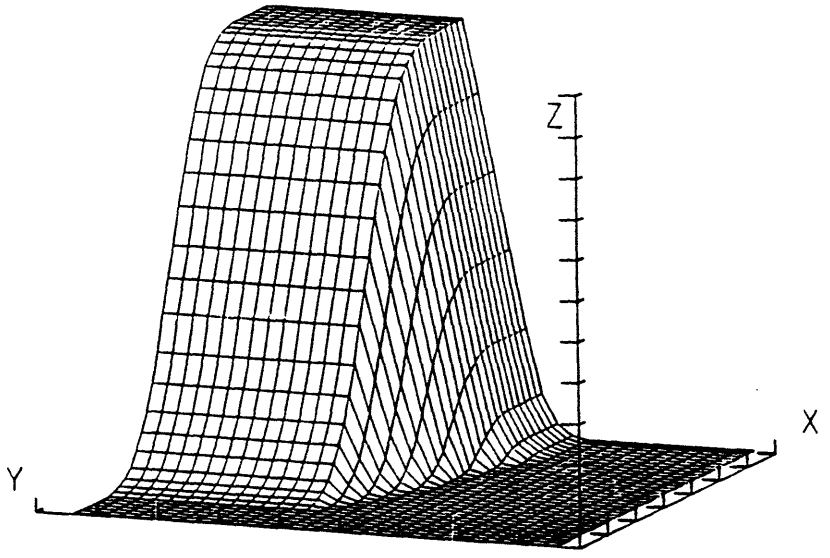

Figure 16 Diagram of consistency of VDMO.

Such a partitioning is represented in Figure 17. There are seven distinct areas. The first area, with no shading pattern, corresponds to the set of mission requirements for which all organizations have an effectiveness equal to 0, 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 areas labeled VDMO and FDMO2 are the ones for which VDMO and FDMO2 are most effective. In the fifth area, which is labeled FDMO1+VDMO, both organizations 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 select one organization over the other. There is no region corresponding to FDMO1+FDMO2. In the region (FDMO1+FDMO2+VDMO), all three designs meet totally the requirements.

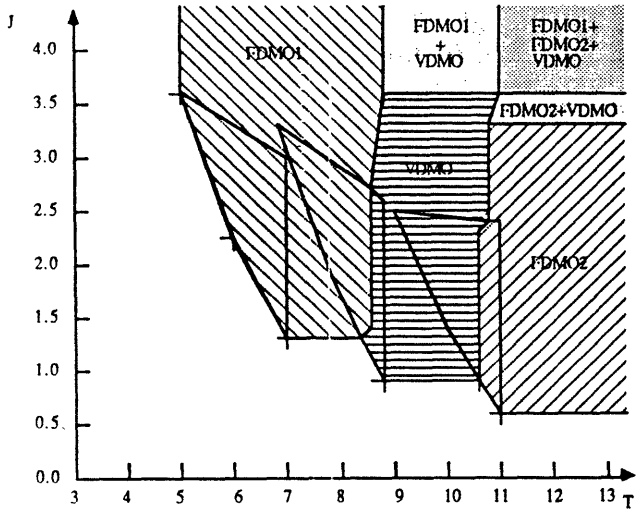

Figure 17 Partitioning of the requirements space for fixed and variable structure DMO's

5. CONCLUSIONS

In the previous sections, the need for variable structure organizations has been described and the concept of variability discussed. A methodology for modeling variable structure decisionmaking organizations that is based on Predicate Transition Nets has been presented. The approach was then used to model a variable structure organization and then analyze it with the tools that have been developed earlier for fixed structure organizations. 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 are not, then there is no need to introduce variability, 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 paper, but would constitute the next step toward the modeling of more realistic decisionmaking organizations.

6. REFERENCES

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

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

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

Levis, A. H., (1988). "Human Organizations as Distributed Intelligence Systems," *Proc. IFAC Symposium on Distributed Intelligence Systems*, Pergamon Press, Oxford.

Monguillet, J. M., (1988). "Modeling and Evaluation of Variable Structure Organizations," MS Thesis, Report LIDS-TH-1730, Lab. for Information and decision Systems, MIT, Cambridge, MA.

Remy, P. and A. H. Levis (1988). "On the Generation of Organizational Architectures Using Petri Nets," in *Advances in Petri Nets*, G. Rozenberg, Ed., Springer Verlag, Berlin.