A COLORED PETRI NET MODEL
OF INTELLIGENT NODES

Alexander H. Levis

Paper in the Proc. 1991 IMACS Symposium on Modeling and Control of

Center of Excellence in C3I
George Mason University
Fairfax, VA 22030
Distributed Intelligence Systems (DIS) consist of interconnected intelligent nodes that carry out tasks in coordination. The intelligence in the nodes may be due to humans, to machines, or to a combination of human and machine, such as a human with a workstation. An earlier model of an intelligent node was based on ordinary Petri Nets with switches and led to several approaches for the design of DIS with fixed structures. That model has been extended to accommodate the execution of multiple roles by the same intelligent node; this leads to architectures with variable structures. The Colored Petri Net formalism is used to develop a compact representation of this model.

1. INTRODUCTION

Distributed intelligence systems (DIS) constitute an ever expanding class of systems that is being designed and implemented. Local area networks and other communication systems are used to interconnect new nodes, or to strengthen the connectivity of existing nodes.

Distributed intelligence systems may be defined as those in which the capacity for reasoning is dispersed across its component subsystems. This definition, however, begs the issue by relating intelligence to reasoning, a term that requires definition in its own right. A more appropriate pair of definitions which may not be formal, but capture the concept of distributed intelligence used in this paper can be found in Minsky [1]. In a distributed system, each function is spread over a number of nodes so that each node's activity contributes a little to each of several different functions.

The notion that a function is decomposed and its components (or sub-functions) assigned to different nodes is an old one. What this definition of "distributed" makes explicit is that each node contributes to the execution of several different functions. The types of systems being considered carry out a number of functions, sometimes in sequence and sometimes concurrently. Thus, the problem is not solved by doing a simple allocation of a decomposed function to the available resources - human and machine ones. One must allocate several decomposed functions in such a manner that the resulting workload does not exceed the capacity of each node.

The modeling of (cognitive) workload in distributed intelligence systems must take into account not only the workload associated with each sub-function carried out by a node, but also the workload associated with coordinating the execution of the sub-functions and with coordinating the interactions between intelligent nodes. An information theoretic framework has been used to model cognitive workload. [2]

The concept that will be introduced as the basis for modeling the distributed aspects of the system is that of a role; a role is capable of executing a sub-function. The role represents the lowest level of functional decomposition for a particular application; a role must be executed in its entirety by a single node. Two types of interactions among roles will be defined: (a) those that are among roles within the same node, called internal interactions, and (b) those that are between roles in different nodes, called external interactions. The latter are the ones that determine the organizational structure or architecture of the DIS.

The origins of the model of the role can be traced back to the four stage model of the interacting decision maker with bounded rationality introduced by Boettcher and Levis [2]. The formal specification of the allowable interactions between interacting decision makers was made by Remy [3]. This specification led in turn to an algorithm, the Lattice algorithm [3], which generates all feasible fixed structure architectures that meet a number of structural and user constraints. Andreadakis [4] introduced an alternative model that was not based on the decision maker model, but on the function carried out by a resource, whether that represented a human or a machine. While this was a five stage model, it was very similar to the four stage one in terms of the allowable interactions. That model formed the basis for a different algorithm for organization design, the Data Flow Structure (DFS) algorithm [4]. Andreadakis' approach to organization design (determine first the required data flow structure for the system and then assign functions to resources) led to the formulation of the notion of role by Demeule [5]. In a parallel effort, Monguillet [6] formulated a model of variable structure architectures. A synthesis of the idea of the role and of a variable structure led to the generalization of the Lattice algorithm to a special class of variable structure architectures: those who adapt their structure to the task or input they have to process [5].

The various models mentioned in the previous paragraph have been used to address a number of problems in the design, analysis, and evaluation of distributed decision making organizations supported by decision aids and decision support systems. The analytical and experimental work of
the last ten years, when combined with theoretical developments in the representation of discrete event systems through Colored Petri Nets as defined by Jensen [7], has led to the reassessment of the various models and their variants and to the conclusion that a slightly more general model could subsume all previous ones without invalidating any of the cognitive modeling or the design algorithms.

This model is presented in this paper. The formalism of ordinary Petri Nets is used to introduce the model in the next section; while in the third section, interactions between roles are defined. In the final section, the most general form of the model is folded into a Colored Petri Net representation.

2. MODEL OF ROLE

A distributed intelligence system, such as a decision making organization, is designed to carry out a mission. Following accepted terminology, a mission is decomposed into functions and functions are decomposed into tasks. This hierarchical decomposition, Mission → Function → Task, shown in Figure 1, has some interesting properties [8].

![Figure 1. Functional decomposition.](image)

One property of interest is that it is a nested functional decomposition. What may be considered a function from a certain viewpoint, it becomes a mission from another. This is shown in Figure 2.

Consequently, this functional decomposition can be carried to any level of detail. In practical terms, it is carried out to the point that the lowest level tasks (the leaves of the tree) must be executed by a single resource. A role is used to model the execution of such a task by a single resource. The basic model of the role can be represented in block diagram form as shown in Figure 3. It consists of three processing stages and two interaction stages for a total of five stages.

![Figure 2. Nested Decomposition of Functions](image)

A role receives inputs or data x from the external environment (sensors) or from other nodes of a system. The incoming data are processed in the first block marked situation assessment (SA) to obtain the assessed situation z. This variable may be sent to other nodes, as shown by the outgoing arrow. If the role receives data about the assessed situation form other nodes, these data z' are fused together with its own assessment z in the information fusion (IF) stage to obtain the revised assessed situation z'. The assessed situation is processed further in the task processing (TP) stage to determine the strategy v to be used to select a response. However, if hierarchies exist in the organization, the particular role may receive command inputs v' from superordinate nodes (or higher echelon decision makers) that may restrict the strategies available for selecting a response. This is depicted by the use of the command interpretation (CI) stage. The output of that stage is the variable w which contains both the revised situation assessment data and the response selection strategy. Finally, the output or response of the role y is generated by the response selection (RS) stage.

Note that if there are no situation assessment and command inputs from other nodes, the five stage model reduces to the common two-stage situation assessment and response selection model of the single, non-interacting decision maker [2]. If there is no information fusion, then the IF stage disappears and the SA and TP stages can be merged into a single SA stage. Finally, if there are no command inputs, then the CI stage disappears and the TP and RS stages can be merged into a single RS stage. The five-stage model and its special cases correspond very closely to the flow types defined by Andreadakis [4] with the main difference being the explicit meaning given to each stage.

![Figure 3. The five-stage model of a role.](image)
Therefore, by introducing the middle processing stage, Task Processing, a new model is obtained that contains the features of both the four-stage interacting decision maker model of Boettcher and Levis [2] and the Andreadakis [4] model.

If ordinary Petri Net notation is used, the model of Figure 3 takes the form shown in Figure 4. The rounded box enclosing the five stages has no formal meaning; it is used to indicate the internal structure of a role and show explicitly the inputs and outputs of the role.

Consider Role i. The first question to be answered is whether it receives data from the environment, from the sensors. This is denoted by the coefficient $e_i$. If $e_i$ is equal to 1, then Role i receives data from sensors, if it is equal to zero, it does not. The coefficient $C_{ij}$ indicates whether the output of Role i is an input to the situation assessment stage of Role j. This type of interconnection is needed to represent the tandem (series) connection of roles. The coefficient $F_{ij}$ represents the sharing of the assessed situation $z$ by Role i with Role j. This is one type of information sharing. The second type, the sharing of results, is represented by $H_{ij}$. In this case, Role i communicates to Role j the output of the response selection stage. The link $H_{ij}$ is used in place of $G_{ij}$ when there is an input from the external environment to Role j. Whether to communicate the assessed situation or to communicate the decision a role has made is an interesting design question that has been addressed by many researchers. The basic trade-off is the amount of data that need to be transmitted – the situation assessment usually requires more bits than the decision. On the other hand, under some rather restrictive conditions, it is possible to reconstruct the assessed situation when the decision is known. The final type of interaction is the issuing of a command from Role i to Role j, as shown by $C_{ij}$. Finally, the coefficient $s_i$ denotes whether Role i produces an output to the environment.

When fixed organizational structures are considered, then each node contains only one role and these six coefficients are constant and take values in $\{0, 1\}$. If there are $n$ roles/nodes in a fixed structure organizational form, then all the interactions can be represented by a set of six arrays:

$$\Sigma = \{e, F, G, H, C, s\}$$

where $e$ and $s$ are $n \times 1$ and $F, G, H, \text{ and } C$ are of dimension $n \times n$. From the definition of the arrays (see Fig. 5) it is apparent that the diagonal elements of $F, G, H, \text{ and } C$ are identically zero. Therefore, any possible set of interconnections among $n$ roles can be represented by assigning the value of 0 or 1 to $q$ elements where:

$$q = \frac{n(n-1)}{2}$$
The number of admissible organizational forms is 2^q. Note that all these structures may not necessarily represent feasible organizations, i.e., organizations that satisfy a number of structural constraints.

The basic structural constraints are:

1. The Petri Net that corresponds to Σ should be connected; a directed path should exist from the single source place to every node of the net and from every node of the net to the single sink place.

The single source and single sink places are modeling artifacts used to coordinate the source model and to collect all the outputs of the net into a single node.

2. The net Σ should have no loops; it should be acyclic.

3. There can be at most one link from the RS stage of node i to another node j, i.e.,
   \[ G_{ij} + H_{ij} + C_{ij} \leq 1 \]

4. Information fusion can take place only at the IF and CI stages. Consequently, the SA stage can receive either inputs from the source model or from a single other node.

5. This last constraint is not necessary; its inclusion eliminates some awkward interactions between nodes.
   \[ G_{ij} + F_{ij} \leq 1 \]

Additional constraints may be introduced to reflect the specifics of a particular application. The Lattice algorithm [3] can be used in conjunction with this model to generate alternative fixed-structure architectures.

4. INTELLIGENT NODES

One way of representing an intelligent node without resorting to the use of Colored Petri Nets is to use a special type of transition, namely, a switch. A switch is a transition with multiple output places and some decision rule which directs the generation of a token after firing in one and only one of its output places.

Since a switch is really a transition, the firing rules for a switch are identical to the firing rules for a transition: a switch will fire if all its input places contain at least one token. Unlike regular transitions however, all the output places of a switch will not receive a token. Only one of them will. This place will be chosen by the internal decision rule associated with the switch. An example of a switch with two input places and three output places is shown in Figure 6. The output places of the switch are called the branches of the switch. In Fig. 6a, the switch is enabled. The rule embedded in s1 is used to determine in which output place a token will be generated. In this case, the token is generated in p11. Note that it is necessary to associate attributes with the tokens so that the rule embedded in the switch can differentiate among inputs.

Consider an intelligent node that can instantiate any one of several roles depending on the input it receives. A switch can be introduced between the source generating tasks (inputs) and the M roles with a rule that directs the particular task to the appropriate role. This is shown schematically in Figure 7 for the case when there are four roles.

![Figure 6. Example of a three-branch switch.](image)

![Figure 7. Model of Intelligent Node](image)

Consider the changes that are needed in the modeling of the source. Let the source generate n distinct tasks \( x_i \); then the set of tasks \( X \) can be partitioned into four disjoint subsets \( X_1 \) to \( X_4 \). Then the rule embedded in the switch takes the form:

If the input \( x_j \in X_j, j = 1 \) to \( 4 \), then Role \( j \) is activated.

This particular generalization is straightforward. A task is generated by the source, it is directed by the switch to the appropriate role and an output is generated. The complications arise when interactions between intelligent nodes are considered. As Monguillet [6] has shown, the rules governing the switches are not independent. If the interactions between nodes depend on the task (i.e., type of token that is being processed) then a node \( i \) that interacts with node \( j \) must know what role node \( j \) has assumed so that it can select a compatible one. If this does not happen, then deadlocks can occur. For example, node \( i \) chooses a role that does not include the transmission of situation assessment information to node \( j \). On the other hand, node \( j \) chooses a role that requires situation assessment...
information from node i. The IF transition of node j will not be enabled and that node will be deadlocked. To address this problem, a table needs to be created that contains all the admissible combinations (the inter-correlations) of switch settings. The difficulty is that this information is not included as part of the Petri Net formalism.

The answer to this problem is the introduction of Colored Petri Nets and the adding of several more structural constraints to the list of five that was presented in the previous section. A simple form of Colored Petri nets is described next.

In Colored Petri Nets, the form of Hugh Level Nets introduced by Jensen [7], tokens are distinguishable. A set of attributes is associated with each token, where each attribute can take a number of values. The color of the token denotes a particular choice of attribute values. All the possible combinations of attribute values (all the colors) constitute the universal color set for a particular problem. There is a Color Set associated with each place in the Petri Net: the Color Set specifies the colors of the tokens that may reside in that place. This Color Set is a subset of the universal color set. Similarly, an Occurrence Color Set is associated with each transition. Arcs are inscribed with expressions of the form:

$$[\text{Boolean}] \cdot \text{Expression}$$

where Boolean is a Boolean expression. When this expression evaluates a true, then the arc inscription evaluates to a set of colors according to the normal expression Expression. A transition is enabled, if there exists one at least one binding for the variables in the inscriptions of the arcs from its input places such that each input place contains at least as many color tokens as specified by the arc inscription. In addition, when the transition contains a guard function, the condition indicated by the guard function must also be satisfied. The following example, Fig. 8, demonstrates these definitions.

The universal Color Set contains three colors, red, white, and blue. All three places have as their Color Set the universal set; they can hold all types of tokens. A variable x has been defined that can take values (be bound to) the same universal set S. The transition has a guard function that restricts the variable x from taking the value w if the transition is to be enabled. The inscriptions translate as follows: If the variable x takes the value r and there is at least one r token in the input place on the right, and if there are two blue tokens in the place on the left, then the transition will fire. Two blue tokens will be removed from the left input place, one red token from the right input place, and two red tokens will be generated in the output place. If x takes the value b, then the right place must have at least one blue token. However, the arc inscription on the left will not evaluate as True and, therefore, there is no enablement condition for that binding of the variable. Finally, because of the guard function, there is no point checking what will happen to the arc inscriptions when x takes the value w.

Consider now the example if Figure 7. There are n distinct inputs (token colors) in the set X and four Color Sets, each set corresponding to one of the partitions X_k of X. The arcs from the input place to each one of the roles are inscribed as follows:

$$[x_j \in X_k] \% x_j \text{ for } i = 1 \text{ to } 4 \text{ and } j = 1 \text{ to } n.$$ 

Note that instead of an arc inscription, a guard function of the form $<x_j \in X_i>$ could have been placed in a transition preceding the $i$-th role. The places on the right in Figure 9 are the input places of the individual roles; a particular role will have an input and will be activated only if a token or tokens belonging to the right Color Set appear in its input place.

The Colored Petri Net (CPN) model of each role is then attached to the model of Fig. 9 to represent the intelligent node that can instantiate a number of nodes. Note that the CPN model makes it very simple to model concurrent operation of several roles; if tokens $x_2$, $x_6$, and $x_{17}$ are present in the source place, then the transitions corresponding to roles 1, 2, and 3 will be concurrently enabled and can fire. The output place of the intelligent node is identical in structure to that shown in Figure 7. The Color Set for the output place contains all the output colors of each role.

One condition that must be maintained though is that the roles within a node do not interact directly with each other. There can be resource constraints that may force the execution of the roles to be serialized, but the roles cannot exchange information about the task they are processing. Consequently, the model of the interactions between roles presented in Section 3 still holds, but with an additional important condition: the roles must belong to different nodes. However, the introduction of the color-dependent conditions for the enablement of transitions, brings back the issue that led to the discarding of the switch-transition as an acceptable solution - the coordination between conditions.
This is addressed by the introduction of one more condition [5] to the five discussed in Section 3.

To formulate the new condition, several concepts must be introduced. First, it is recognized that the variable structure architecture $\Pi$ that can be constructed with this CPN model of the node is nothing else but a superposition, or folding, of a number of fixed structure architectures $\Sigma$ of the type defined earlier: to each input $x_i$ corresponds some fixed architecture $\Sigma_i$. For the example of Fig. 9,

$$\Sigma_j = \Pi(x_i \in X_j)$$

i.e., role $j$ is instantiated when the input is an element of the subset $X_j$. A color set $X_j$ is said to be accessible at a transition $t$ if and only if there is a directed path from the source to the transition $t$ in the fixed structure obtained when $x_i \in X_j$. The reason for introducing this concept is that the variable interaction between two stages of roles in different nodes must be based only on information that is accessed jointly by the roles that interact. For example, the SA stage of Role 1 in node 3 must determine, based on some information it has accessed, whether or not it must wait for a message from Role 2 in node 5. Similarly, Role 2 in node 5 must infer from some information it has already received, whether or not it must wait for a message from Role 1 in node 3 before initiating information fusion at the IF stage. These conditions can be expressed by inscribing the arcs in the manner described in Figure 8. In that example, the new condition is manifested by the assumption that the left arc knows that the binding on the right is $x = r$, even though no red tokens are required on the left for the transition to be enabled. Thus, constraint 6 takes the form:

6. In an architecture that contains intelligent nodes, the following condition must hold:
   - If there is an interaction between the SA stage of a role in a node and the IF stage of a role in another node, then in every $\Pi(x)$, for any $x \in X_j$ that activates those roles, there must exist directed paths from the source to these two transitions.
   - If there is an interaction between the RS stage of a role in a node and the IF stage of a role in another node, then in every $\Pi(x)$, for any $x \in X_j$ that activates those roles, there must exist directed paths from the source to these two transitions.
   - If there is an interaction between the RS stage of a role in a node and the CI stage of a role in another node, then in every $\Pi(x)$, for any $x \in X_j$ that activates those roles, there must exist directed paths from the source to these two transitions.

For a formal description of these conditions in the context of a special class of Colored Petri Nets, see [5].

The software used to model the intelligent nodes is DesignCPN by Meta Software Corporation. The software has embedded in it syntax checks that ascertain that the information needed by condition 6 is accessible to the arcs representing interactions between nodes. They are the conditions needed for the model to be able to execute, i.e., for a simulation to take place.

5. CONCLUSION

In this paper, a generalization of the model of the interacting intelligent node has been presented. This model, based on the Colored Petri Net formalism, subsumes the model used by Dëmaël [5], and is supported by commercially available software. It is forming the basis for on-going research on variable-structure architectures of distributed intelligence systems.

REFERENCES