FEBRUARY 1989 LIDS-P-1849

GENERATION OF ARCHITECTURES FOR DISTRIBUTED INTELLIGENCE SYSTEMS

Alexander H. Levis

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

ABSTRACT

An approach to the modeling of discrete event distributed intelligence systems is presented that uses ordinary Petri Nets for fixed structure architectures. The model leads to the precise formulation of the problem of generating the complete set of architectures that satisfy a number of resource and design constraints. Two algorithms are presented: the first one (the DFS algorithm) starts with the specification of the desired functionality and generates data flow structures with prescribed redundancy and complexity; then the various functions are allocated to resources; the second one (the Lattice algorithm) starts with a given set of decision making units and obtains the partially ordered sets that contain all the feasible architectures. The resulting architectures can be analyzed to evaluate their performance characteristics.

* Invited paper to appear in the Proceedings of the 1989 IEEE International Conference on Control and Applications (ICCON '89), Jerusalem, Israel, April 1989. This work was supported by the Office of Naval Research under contract no. N00014-84-K-0519.

INTRODUCTION

A special class of systems, called distributed intelligence systems, is the focus of this paper because it promises to provide models, analysis tools, and evaluation procedures that can be used in the design of the architecture of the decision systems needed to manage large scale physical systems. The term *distributed intelligence* systems is used in the sense of Minsky [1] who defined as *distributed* the property that each function performed by a system is spread over a range of locations so that each part's activity contributes a little to each of several different functions, and as *intelligence* all the mental skills that at any moment we admire but do not understand. These qualitative definitions are very appropriate: First, they include the concept that each entity in the system, each node, is involved in the carrying out of more than one function. Second, they allow for the boundary between machine intelligence and human intelligence not to be fixed, but vary as the state of knowledge about a system changes.

We are considering the decision making or management level of large scale physical systems, such as air traffic control centers, energy control centers of large scale power systems, the operations management of flexible manufacturing plants, or a command, control and communications system, either a military one, such as the one used in a naval battle group, or a civilian one, such as the one used by a fire department in a large city. In all these cases, there are some common characteristics or features: The assets managed by the decision system are geographically distributed; there is some hierarchical structure either in the monitoring and control system, the human decision system, or both, that introduces delays; there is some level of decision making at each node; there is a fast tempo of operations that requires on-line decision making; and there is some type of decision support system that interconnects the various nodes allowing for information transmission and for the dissemination of decisions or commands. The support system may also include decision aids that assist individual nodes in their decision making functions. These decision aids are usually introduced to compensate for the bounded rationality of human decision makers.

Since we will be dealing with the management or decision making layer, it is appropriate to model the process as a discrete event dynamical system. If a boundary is drawn around the system to be modeled, then its interactions with the environment can be of two types: inputs and outputs. The system receives signals, or stimuli, or tasks, and produces outputs, or responses, or decisions. It is assumed that when an event occurs in the environment, it generates a signal that is sensed by the system, by one or more sensors. The formulation is general enough to allow for a single event to be sensed by different sensors at different times, or for different events to be sensed by different sensors asynchronously. Thus, we can have either a single source producing discrete events, selected from some finite set with some probability, at times also characterized by a probability distribution, or we can have multiple sources producing independent events. Similarly, the output or outputs are selected from finite sets.

The problem addressed in this paper is the generation of distributed architectures for the class of systems described in the previous paragraphs. The design problem will be formulated in two ways. In the first case, the functions to be performed by the system are given along with certain desirable structural characteristics. In the second case, the decision making entities or intelligent nodes are specified a priori. Both approaches are based on Petri Net theory and concepts; some relevant definitions and results are reviewed in the next section.

OVERVIEW OF PETRI NET THEORY

A Petri Net - denoted by PN - is a bipartite directed graph represented by a quadruple PN = (P, T, I, O), where P is a finite set of places, denoted by circles, and T is a finite set of transitions, denoted by bars [2]. A node will refer to either a place or a transition of PN. The mappings I and O correspond to the set of directed arcs from places to transitions and to the set of directed arcs from transitions to places, respectively. When I and O take values in $\{0,1\}$, the resulting nets are called ordinary Petri Nets.

A marking of a Petri Net is a mapping M which assigns a non-negative integer number of tokens to each place of the net. A marking can be represented by a n-dimensional integer vector, also denoted by M, whose components correspond to the places of the net. A transition t is enabled by a given marking M if and only if for each input place p of t, $M(p) \geq I(p,t)$. When a transition is enabled it can fire. The new marking M' reached after the firing of t is defined as follows :

$$(\forall p \in P) \quad M'(p) = M(p) - I(p,t) + O(t,p). \tag{1}$$

A place p and a transition t are on a self-loop if p is both an input and an output place of t. A Petri Net will be pure, if it does not contain self-loops. Petri Nets under consideration in this article will all be pure; this is a reasonable assumption for information structures.

Certain basic graph-theoretic definitions apply readily to Petri Nets. A Petri Net is connected if and only if there exists a path – not necessarily directed – from any node to any other node; it is strongly connected if and only if there exists a directed path from any node to any other node. A directed circuit is a directed path from one node back to itself. A directed elementary circuit is a directed circuit in which no node appears more than once. Directed elementary circuits play a key role in the theory of net invariants.

The topological structure of a pure Petri Net can be represented by an integer matrix C called an incidence or flow matrix. C is a nxm matrix whose columns correspond to the m transitions and whose rows correspond to the n places of the net. C is defined as follows:

$$C_{ij} = O(t_j,p_i) - I(p_i,t_j) \quad \text{for } 1 \le i \le n \text{ and } 1 \le j \le m. \qquad (2)$$

Note that the definition is restricted to pure Petri Nets. There is actually a problem with non-pure Petri Nets in the sense that self-loops cannot be represented in the incidence matrix: a 1 and a -1 cancel each other to yield a zero in the matrix, thus losing track of the existence of the self-loop.

The mappings O and I can be reconstructed from the matrix C in the following trivial way:

$$O(t_j,p_i) = \max \{ C_{ij} , 0 \}; \qquad (3a)$$

$$I(p_i,t_j) = \min \{ C_{ij} , 0 \}. \qquad (3b)$$

An initial marking M_o is bounded if there exists a positive integer k such that, for every reachable marking M, the number of tokens in each place is bounded by k. If k equals one, the marking is said to be safe. A Petri Net PN is structurally bounded, if any initial marking of PN is bounded. A marking M_o is live if for any transition t and for every reachable marking M there exists a firing sequence from M that includes t. In other words, every transition of the net can fire an infinite number of times. A Petri Net PN is structurally live, if any initial marking of PN is live.

A marked graph is a strongly connected Petri Net in which each place has exactly one input and one output transition. There is a useful relationship between the circuits of a marked graph and the S-invariants. An S-invariant is an n x 1 non-negative integer vector **x**, element of the kernel of the transpose of the incidence matrix **C**, i.e., **x** verifies the relation:

$$C' \mathbf{x} = 0. \qquad (4)$$

Similarly, a T-invariant is an m x 1 non-negative integer vector **y**, element of the kernel of C, i.e., **y** verifies the relation:

$$C\ \mathbf{y} = 0. \qquad (5)$$

The set of places (resp. transitions) whose corresponding components in **x** (resp.**y**) are strictly positive is called the support of the invariant and is denoted <x> (resp.<y>). The support of an invariant is said to be minimal if and only if it does not contain the support of another invariant but itself and the empty set. Let **x** be an S-invariant of a Petri Net PN and let <**x**> be its support. <**x**> is a set of places of PN, i.e., a subset of P. We call S-component associated with **x** - denoted [**x**] - the subnet of PN whose set of places is <**x**> and whose transitions are the input and output transitions of the places of <**x**> in PN. The T-components are defined in a similar way.

Some useful properties of S- and T-invariants are listed without proof. The first property establishes the conservation of the number of tokens belonging to the support <**x**> of an S-invariant of a Petri Net. **x** is a S-invariant of PN, if and only if for any initial marking M_0 of PN and for any reachable marking M,

$$\mathbf{x}' M = \mathbf{x}' M_0 \qquad (6)$$

Marked graphs play a key role in the quantitative modeling of information structures. The following two results, due to Commoner and Holt [3], are of primary importance. In a marked graph, the number of tokens in any elementary directed circuit - the token content of the circuit - remains invariant by transition firings. Furthermore, a marking of a marked graph is live if and only if the token content of every directed elementary circuit is strictly positive. Another result relates directed circuits and S-components of a marked graph. It gives an algebraic characterization of a topological concept and will be extensively used in the sequel: The minimal S-components of a marked graph are exactly its elementary directed circuits.

THE DATA FLOW STRUCTURE ALGORITHM[4]

In the first approach to the generation of architectures of distributed intelligence systems, the basic element is a five stage generic information and decision making process that represents a functionality. By the term functionality, we refer to a sequence of processes that accomplish a specific task. This sequence can be represented by a Petri Net that takes the form of a line of alternating places and transitions; this structure leads to the observation that a functionality is represented by an information flow path. The path starts with a place that depicts the input or task to be performed. The first transition, IP for Initial processing, represents the first function prior to any interaction with other information flow paths. Indeed, two types of interactions are postulated. The first, Data Fusion [DF], provides for the sharing of data among different flow paths. The second, Results Fusion [RF], allows the communication not of data, but of decisions or results from one flow path to another. The Middle Processing [MP] and Final Processing [FP] transitions model the processes or functions appropriate to each functionality. The line ends at a place that represents the output.

If the functionality represents an information processing and decision making sequence, e. g., human decision making, then the following interpretation can be given to the five stages. As data are received they are processed in the IP stage to obtain the situation assessment. Information (local or partial situation assessments) of several IP stages are fused in the DF stage to produce a global situation assessment. This is processed in the MP stage to generate options or alternative courses of action. These results are fused together in the RF stage to eliminate conflicting or infeasible options. Finally, a response is selected from the available options in the FP stage.

Not all stages need be present in an information structure. An information flow path with all five stages defines Flow Type 1 (Fig. 1). If there is no results fusion, then by convention we merge the MP transition into the FP one, as shown in Flow Type 2. Similarly, if there is no data fusion, then the MP transition is merged into the IP one as in Flow Type 3.

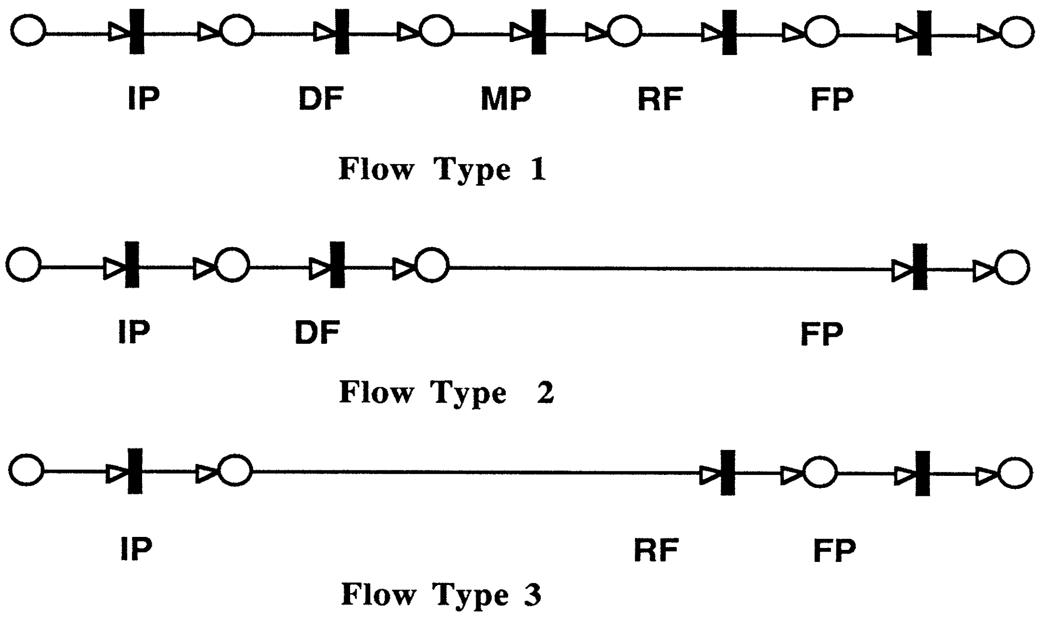

Figure 1. Basic Flow Types

A general data flow structure (DFS) is classified according to the flow types of the information flow paths it contains. If all the paths are of flow type 1, then the DFS belongs to class 1. If some paths are of flow type 1 and some of flow type 2, the DFS class is 12. The feasible classes are: 1, 2, 3, 12, 13, and 123. Class 23 is infeasible because flow type 2 and flow type 3 paths cannot exchange information: the flow type 2 information paths have data for fusion and DF transitions, while the flow type 3 information paths have results for fusion and RF transitions. A DFS with all three flow types (class 123) is shown in Figure 2.

This approach has allowed the introduction of two additional design specifications: the *degree of complexity* of the organization and the *degree of redundancy* (Figure 3). The first measure addresses the complexity that results from transitions needing many different inputs to be enabled. The second is a measure of the number of times outputs are replicated.

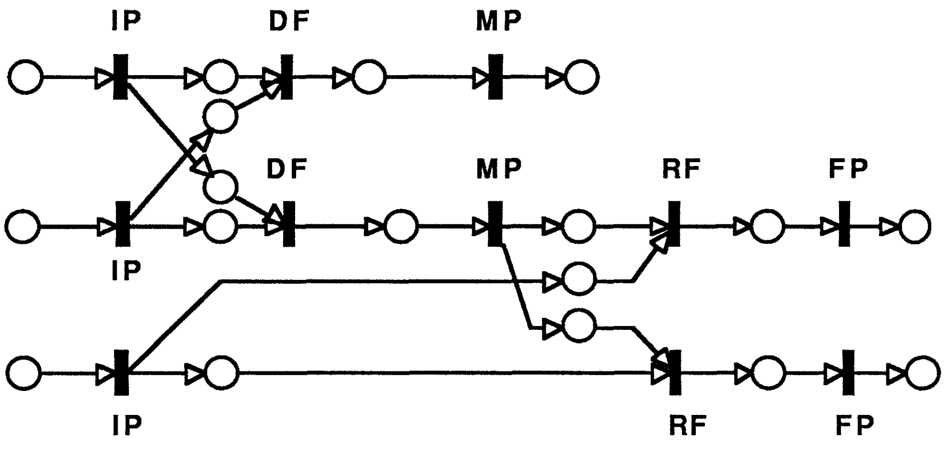

Figure 2. Data Flow Structure with all three Flow Types.

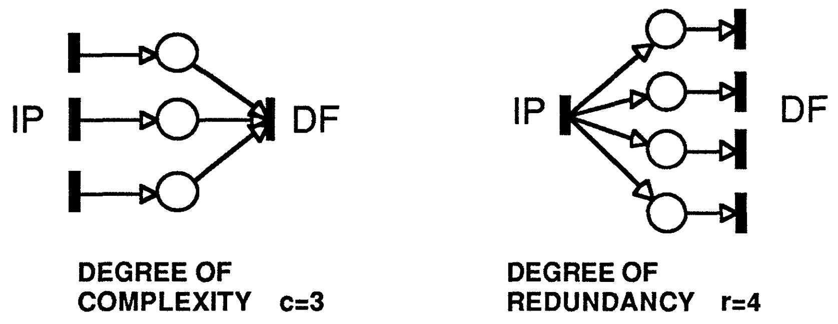

Figure 3. Examples of complexity and redundancy

The *degree of complexity* of a DF transition is defined as the number of input places that feed data to the DF transition. The degree of complexity of the DF stage is defined as the maximum of the degrees of complexity of the DF transitions. The term complexity is justified by the observation that the more sources that feed data to a fusion node, the more complex the processing that takes place.

The need for redundancy of information within the structure arises from reliability considerations and topological factors. The *degree of redundancy* of an IP transition is defined as the number of fusion stages that receive the output data of the IP transition. The degree of redundancy of the IP stage is defined as the maximum of the degrees of redundancy of the IP transitions. The term redundancy is justified by the fact that the same information is communicated to more than one fusion node, and is therefore redundant in the data flow structure.

The degree of complexity of a RF transition, the degree of redundancy of a MP transition, and the degrees of complexity and redundancy of the RF stage are similarly defined. A data flow structure with degree of complexity $c_1 = 2$ and redundancy $r_1 = 2$ of the DF stage, and degree of complexity $c_2 = 3$ and redundancy $r_2 = 3$ of the RF stage is shown in Figure 4.

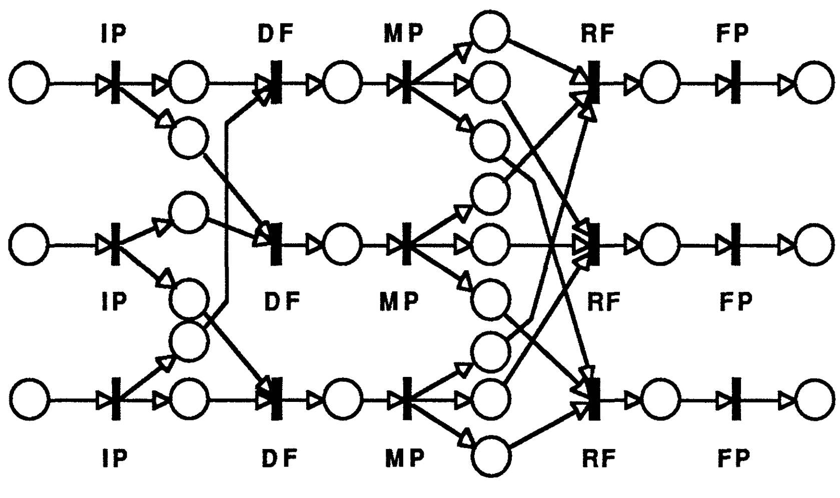

Figure 4. Data Flow Structure $c_1 = 2$, $r_1 = 2$, $c_2 = 3$, $r_2 = 3$

Having defined the basic elements of the architectures, i.e., the information flow paths that are derived from the requirements that the distributed system exhibit certain functionalities, it is now necessary to introduce the grammar rules for the connectivity of the processing transitions. They are:

- exactly one MP node can receive data from a DF node;
- exactly one FP node can receive data from an RF node;
- one IP transition for each input to the organization;
- one FP transition for each output of the organization.

These are practical rules that are consistent with the definitions of the five stages. The first two indicate that fusion processes produce outputs that must receive some additional processing before being distributed to other nodes. Such distribution can be made only by the MP and FP transitions. The third rule associates a processor - or preprocessor in some cases - with each input source. Similarly, each distinct output is produced by a different FP transition. Recall, that there has been no restriction that a decision making unit contain only one IP or FP transition.

The algorithm for the generation of data flow structures is based on the grammar rules and is parameterized by the choice of flow types that should exist in the design. It produces the incidence matrix of the corresponding Petri Net. In order to generate data flow structures in a consistent, methodical way, the design parameters are varied between the minimum and maximum value they may obtain. The DFS algorithm consists of seven steps:

Step 1: On the basis of the requirements of the application, determine the required functionality and represent it in terms of information flow paths. Consider all possible DFS classes.

Step 2: Select the number n_1 of initial processing (IP) transitions that provide data for fusion (DF stage). Let n_2 be the number of initial processing (IP) transitions that provide results for fusion (RF stage). The total number n of IP transitions is:

$$n = n_1 + n_2. \tag{7}$$

Step 3: Select the degree of complexity c_1 and the degree of redundancy r_1 of the DF stage. The number p of output places of the IP transitions that are input place to DF transitions is:

$$p = n_1 r_1 \tag{8}$$

and the number k of data fusion transitions is:

$$k = n_1(r_1 / c_1). \tag{9}$$

For the pair (r_1, c_1) to be feasible, i.e., for all transitions of this stage to have the same degree of complexity and degree of redundancy, the number k must be integer. (Another practical constraint on k is that it be no larger than the upper bound of available processing assets.) Since each DF transition is connected to one middle processing (MP) transition, the number of MP transitions is also k.

Step 4: Since one IP transition is connected to each place that represents an input to the organization, and exactly one MP transition is connected to each output place of a DF transition, the corresponding elements of the incidence matrix can be assigned the values of 1 or 0.

Step 5: Select the number k_2 of MP transitions that provide results for fusion (at the RF stage). Let k_1 be the number of middle processing transitions that produce outputs. The total number of MP transitions is:

$$k = k_1 + k_2. \tag{10}$$

Step 6: Select the degree of complexity c_2 and the degree of redundancy r_2 of the RF stage. The number q of output places of the IP transitions and MP transitions is:

$$q = (n_2 + k_2)\, r_2 \tag{11}$$

and the number of results fusion transitions, m, is:

$$m = (n_2 + k_2)\,(r_2 / c_2). \tag{12}$$

For the pair (r_2, c_2) to be feasible, i.e., for all transitions of this stage to have the same degree of complexity and degree of redundancy, m must be integer. The second constraint on m is

$$m \leq a \tag{13}$$

where a is the maximum number of available information processing units. Since each RF transition is connected to one FP transition, the number of FP transitions is also m.

Step 7: Since each RF transition has exactly one output place; each FP transition has exactly one input place; and, finally, exactly one output place is connected to each FP transition, the remaining entries of the incidence matrix are determined.

These seven steps generate the incidence matrix of a number of ordinary Petri Nets that form the basis for designing system architectures. The precise number depends on the requirements and the parameters. The following figure (Figure 5) indicates, as an illustrative example, the number of possible data flow structures one can obtain as a function of the number of inputs (sources), parameterized by the number of assets, which in this case means the maximum number of parallel information flow paths. In one particular application, where the number of assets was three, and there were two external sources, consideration of all possible classes of structures led to 14 distinct Petri Nets.[5]

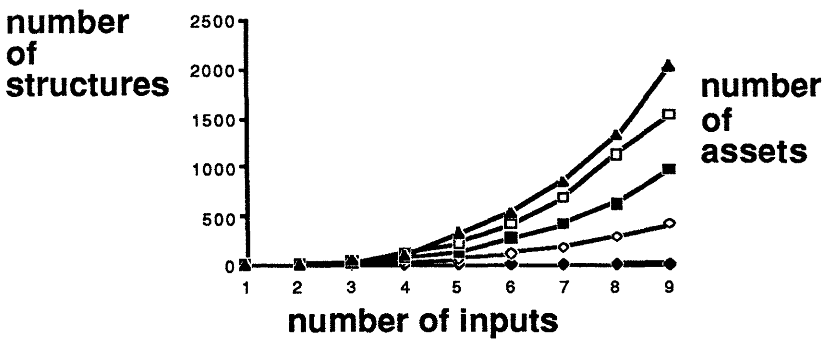

Figure 5. Number of data flow structures generated by algorithm

The next step in the design procedure consists of selecting from among the generated DFS those that seem most promising in view of the various resource constraints and such criteria as reliability, symmetry, hierarchical patterns, etc. Let us assume that the following structure (Figure 6) has been chosen as the basis for designing architectures.

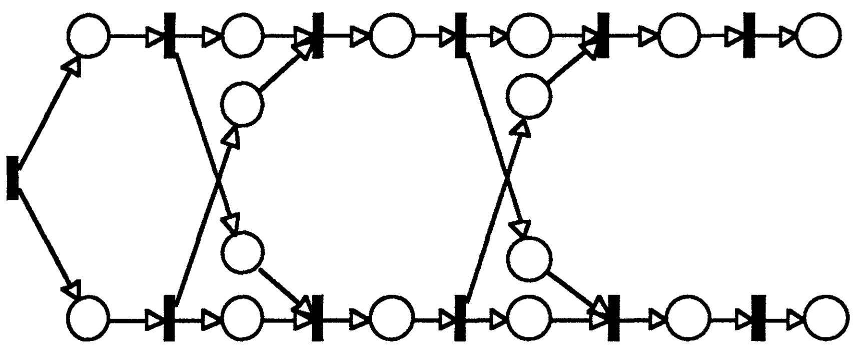

Figure 6. Candidate data flow structure

Next, this net will be divided into entities that correspond to the nodes of the distributed architecture. This is accomplished by the introduction of resource or control circuits that group together a number of transitions. Two such alternatives are shown in Figures 7 and 8.

In Figure 7, two resource places have been introduced, symmetrically, with each one defining one of the parallel information flow paths as a distinct entity or node. The resulting structure is a parallel one with both data fusion and results fusion interconnections between the two nodes.

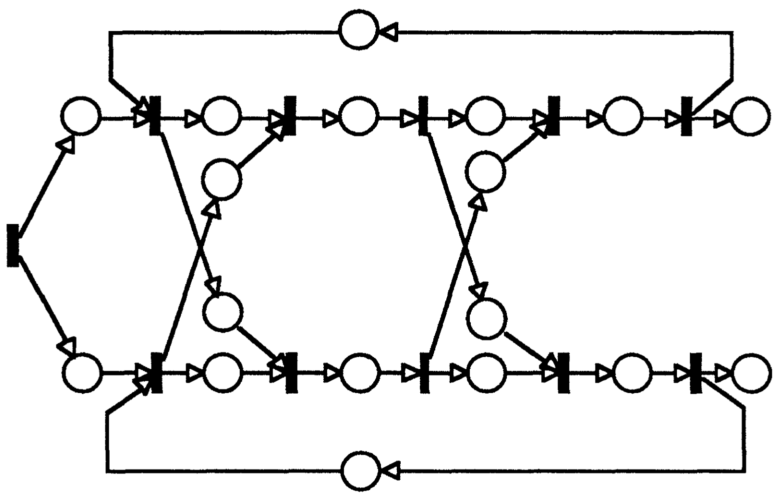

Figure 7. Two-node parallel structure

In Figure 8, four resource place have been introduced. However, each one of the information flow paths has been partitioned into two nodes in tandem. The overall structure is still parallel, but the information exchanges are restricted as follows: the two leftmost nodes exchange data which are then fused; the two rightmost nodes exchange decisions or results. The two figures show rather vividly the differences in the procedures and protocols that are needed to convert these two structures into architectures.

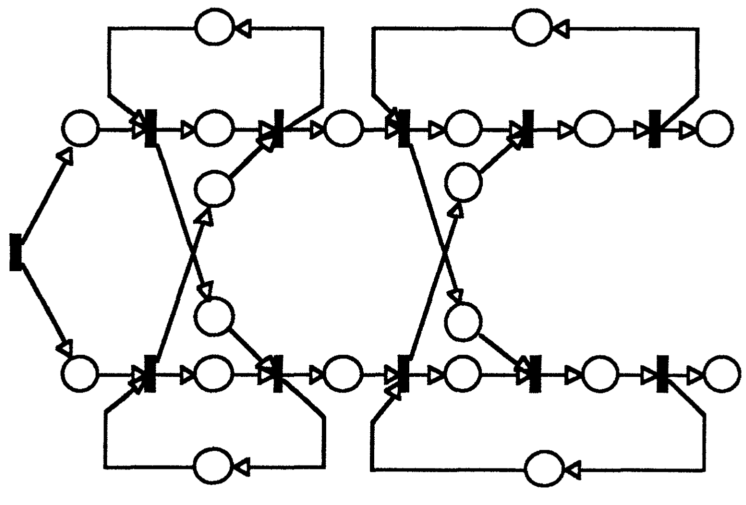

Figure 8. Four-node parallel structure

The last step in the process is the drawing of the boundaries around the nodes and then introducing the places and transitions that represent the physical system that affects the interconnections between the nodes (the communications system.) Two such architectures are shown in Figures 9 and 10.

Some more places and connectors have been added around some nodes to implement a synchronization constraint, mandated by the specific application. Also note in both figures the new transitions and places between nodes that model the communications or decision support system.

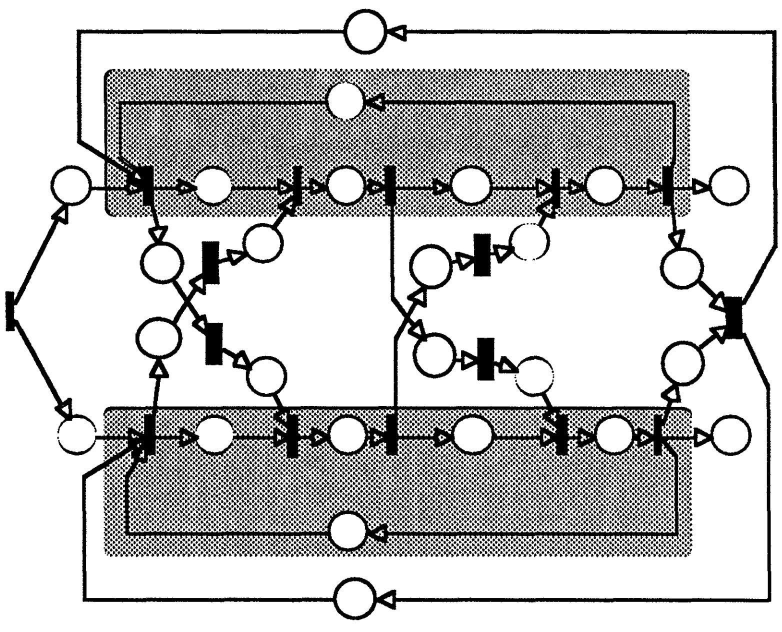

Figure 9. A two-node parallel architecture

Many more architectures can be generated by considering the possible variations in allocating resource places and in grouping entities together in one node. Selection of the preferred architectures - all of which satisfy functionality and resource requirements - is accomplished by analyzing the resulting structure and obtaining suitable measures of performance. Such measures include accuracy, which is a measure of the quality of the information processing, response time, which is a measure of the of timeliness of the response, throughput rate, and still others are information consistency, synchronization, and coordination. [6,7]

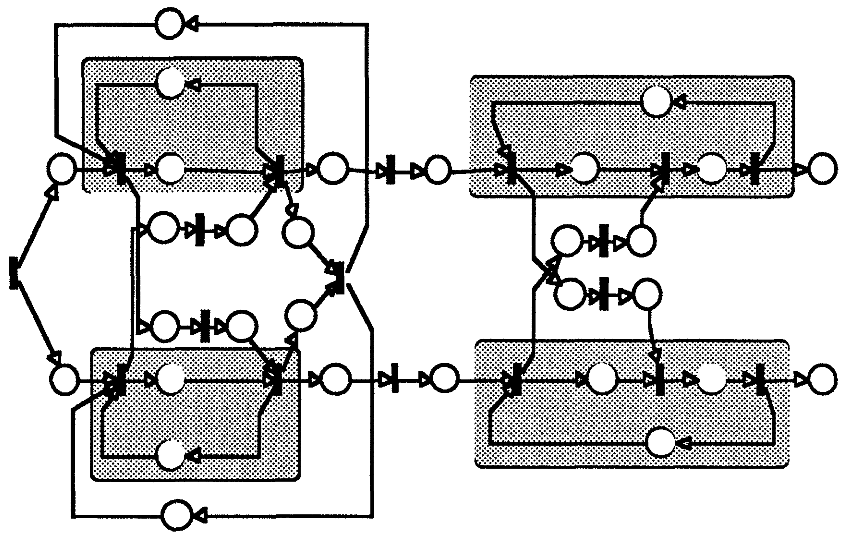

Figure 10. A four-node parallel architecture

THE LATTICE ALGORITHM

In contrast to the DFS algorithm, the Lattice algorithm, due to Remy [8], starts with a generic model of an intelligent node and considers the interconnections among a specified number of them. These nodes represent humans, machines, or humans supported by machines. The original Petri Net model of the human decision maker, who interacts with other components and with the environment, is shown in Figure 11 [9]. The same model is used to represent all intelligent nodes.

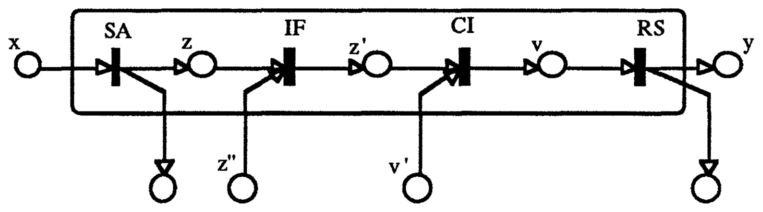

Figure 11. Four stage model of a intelligent node

The decision maker (DM) receives input signals x at the Situation Assessment (SA) stage from a variety of sources: from the environment, from a decision support system (DSS), or from the rest of the organization. He processes this input, with or without use of information stored in a data base (memory), to obtain an estimate of x and of z, the "assessed situation", which he may share with other DMs. He may also receive at this point other information, z", from the rest of the organization. He combines this information with his own assessment in the Information Fusion (IF) stage, which contains a data fusion algorithm, to obtain the final assessment of the situation, labeled z'. The next step is the consideration of inputs v' from other DMs which could result in a restriction of his set of alternatives for generating the response to the given input. This is the Command Interpretation (CI) stage. The outcome of the CI stage is a signal v, which contains the data z' and the rule v', to be used in the Response Selection (RS) stage to select the procedure or algorithm for generating the output y. This is the response of the decision maker; it may be sent to the environment or to other DMs within the organization. Note that the main difference between this model and the basic flow type (#1) in the DFS model is the absence of the MP transition between the two fusion transitions. In this model, the middle processing has been allocated to the IF and CI transitions.

Other organization components can be modeled using the same basic four stage model, but eliminating one or more of the stages. For example, a processor that receives sensor data and converts it to an estimate of a vector variable can be modeled by a single SA transition, while a data fusion algorithm can be modeled by an IF transition. With this model and its variants used to model other components, it is now possible to formulate the problem of designing distributed architectures.

The intelligent node can only receive inputs at the SA, IF, and CI stages, and produce outputs at the SA and RS stages. These conditions lead to the set of admissible interactions between two nodes, or two DMs, that is shown in Fig. 11. For clarity, only the connectors from DM^i to DM^j are shown; the interactions from DM^j to DM^i are identical.

The mathematical representation of the interactions between DMs is based on the connector labels e_i, s_i, F_{ij}, G_{ij}, H_{ij}, and C_{ij} of Fig. 12; they are integer variables taking values in $\{0,1\}$ where 1 indicates that the corresponding directed link is actually present in the organization, while 0 reflects the absence of the link. These variables can be aggregated into two vectors e and s, and four matrices F, G, H, and C. The interaction structure of an n-decision maker organization may be represented by the following six arrays: Two n x 1 vectors e and s, representing the interactions between the external environment and the organization:

$$e \equiv [e_i]\ ; \qquad s \equiv [s_i]; \qquad \text{for} \quad i = 1, 2,..., n.$$

and four n x n matrices F, G, H, C representing the interactions between decision makers inside the organization:

$$\begin{array}{lll} F \equiv [F_{ij}]\ ; & G \equiv [G_{ij}]\ ; & \text{for} \quad i = 1, 2,..., n \\ H \equiv [H_{ij}]\ ; & C \equiv [C_{ij}] & \text{and} \quad j = 1, 2,..., n. \end{array}$$

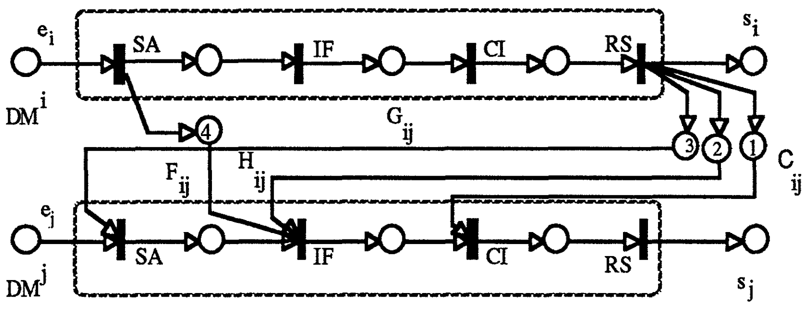

Figure 12. Allowable interactions between DMs.

Since there are four possible links between any DM and any other DM except himself, then the maximum number of interconnecting links that an n-decision maker organization can have is

$$k_{max} = 4n^2 - 2n. \qquad (14)$$

Consequently, if no other considerations were taken into account, there could be $2^{k_{max}}$ alternative structures. This is a very large number: 2^{90} for a five intelligent node structure.

The analytical description of the possible interactions between nodes forms the basis for an algorithm that generates all the nets that meet some structural constraints as well as application–specific constraints that may be present. The set of structural constraints that has been introduced rules out a large number of nets. The most important constraint addresses the connectivity of the structure - it eliminates information structures that do not represent a single integrated distributed system.

The Lattice algorithm determines the maximal and minimal elements of the set of designs that satisfy all the constraints; the entire set can then be generated from its boundaries. The algorithm is based on the notion of a simple path - a directed path without loops from the source to the sink. Feasible structures are obtained as unions of simple paths. Consequently, they constitute a partially ordered set.

The six-tuple {e, s, F, G, H, C} is called a Well Defined Net (WDN) of dimension n, where n is the number of components in the organization. The set of all Well Defined Nets of dimension n is denoted Ψ^n; its cardinality is given by $2^{k_{max}}$, where k_{max} is given by Eqn. (14). The notion of a subnet of a WDN can be defined as follows. Let $\Pi = \{e, s, F, G, H, C\}$ and

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

be two WDNs. The WDN Π is a subnet of Π' if and only if

$$e' \leq e; \qquad F' \leq F; \qquad G' \leq G;$$
$$s' \leq s; \qquad H' \leq H; \qquad C' \leq C$$

where the inequality between arrays is interpreted element by element. In other words, Π' is a subnet of Π if any interaction in Π', i.e., a 1 in any of the arrays e', s', F', G', H', C', is also an interaction in Π. The union of two subnets Π_1 and Π_2 of a WDN Π, is a new net that contains all the interactions that appear in either Π_1 or Π_2 or both.

The matrix representation, i.e., the set of arrays {e, s, F, G, H, C}, and the Petri Net representation, given by the graph or the incidence matrix of the net, with the associated labeling of the transitions, are equivalent, i.e., a one to one correspondence exists between them.

Let the system be modeled as having a single source and a single sink place. Each internal place of a WDN has exactly one input and one output transition. The sink of a WDN has one input but no output transitions, while the opposite stands for the source. If source and sink are merged into one place, every place in the net will have, therefore, one input and one output transition. Since the net is strongly connected, it is a marked graph. Note that considering the source and the sink of a WDN as the same place has no bearing on the internal topology of the net. The assumption becomes important however when the dynamic behavior of a WDN is studied. The merging of source and sink limits indeed the amount of information a given organization can process simultaneously. The initial marking of the place representing the external environment will define this bound. At this stage, a WDN may contain circuits.

While WDNs constitute the framework within which information structures will be designed, each WDN is not a valid structure. Additional constraints to restrict the set of WDNs to useful information structures are needed. First, there are some WDNs corresponding to combinations of interactions between components that do not have a physical interpretation, e.g., DMs can exchange information - F_{ij} and F_{ji} can coexist - but commands are unilateral - either C_{ij} or C_{ji} or none, but not both. Those WDNs should be eliminated, if realistic structures are to be generated. The structural constraints define what kinds of combinations of interactions need to be ruled out. Second, any realistic design procedure should allow the designer to introduce specific structural features appropriate to the particular design problem. User-defined constraints are introduced to address this issue. A set of four different structural constraints R_s is formulated that applies to all organizational structures being considered:

R_1 A directed path should exist from the source to every node of the structure and from every node to the sink.

R_2 The structure should have no loop, i.e., be acyclical.

R_3 There can be at most one link from the RS stage of a DM to each one of the other DMs, i.e., for each i and j, only one element of the triplet $\{G_{ij}, H_{ij}, C_{ij}\}$ can be nonzero.

R_4 Information fusion can take place only at the IF and CI stages. Consequently, the SA and RS stages of each DM can have only one input.

Constraint R_1 eliminates structures that do not represent a single integrated system and ensures that the flow of information is continuous within the system. Constraint R_2 allows acyclical structures only. This restriction is made to avoid deadlock and circulation of messages within the system. It also restricts the marked graphs to occurrence nets, which makes analysis much simpler. Particularly, liveness and safety are easily treated: if the source has initially exactly one token, then each transition fires exactly once, and eventually the sink is marked. There is never more than one token in a place. Constraint R_3 states that the output of the RS stage of one DM or component can be transmitted to another DM or component only once. It does indeed not make much sense to send the same information to the same node at several different stages. Constraint R_4 prevents a node from receiving more than one input at the SA stage. The rationale behind this limitation is that information cannot be merged at the SA stage; the IF stage has been specifically introduced to perform such a fusion.

To introduce constraints that will reflect the specific application, the organization designer can place the appropriate 0's and 1's in the arrays {e, s, F, G, H, C} defining a WDN. The other elements will remain unspecified and will constitute the degrees of freedom of the design. The set of user-defined constraints is denoted R_u, while the complete set of constraints is denoted R.

A *feasible structure* is a Well Defined Net that satisfies both the structural and the user-defined constraints. The design problem is to determine the set of all feasible structures corresponding to a specific set of constraints.

The notion of subnet introduced earlier defines an order (denoted $\leq$) on the set Ψ^n of all WDNs of dimension n. Therefore, maximal and minimal elements can be defined. A maximal element of the set of all feasible structures is called a Maximally Connected Organization (MAXO). Similarly, a minimal element is called a Minimally Connected Organization (MINO). Maximally and minimally connected organizations can be interpreted as follows. A MAXO is a WDN such that it is not possible to add a single link without violating the set of constraints R. Similarly, a MINO is a WDN such that it is not possible to remove a single link without violating the set of constraints R. The following proposition is a direct consequence of the definition of maximal and minimal elements.

For any given feasible structure Π there is at least one MINO Π_{min} and one MAXO Π_{max} such that $\Pi_{min} \leq \Pi \leq \Pi_{max}$. Note that the net Π need not be a feasible . There is indeed no guarantee that a WDN located between a MAXO and a MINO will fulfill the constraints R, since such a net need not be connected. To address this problem, the concept of a simple path is used.

Let Π be a WDN that satisfies constraint R_1 and whose source and sink have been merged together into a single external place. A simple path of Π is a directed elementary circuit which includes the (merged) source and sink places. Since the Petri Net representing Π is a marked graph, a simple path is a minimal support S-invariant of Π whose component corresponding to the external place is equal to 1. Note that if the latter property is not satisfied, the S-invariant is an internal loop of the net. The simple paths of a given WDN are themselves WDNs. We will denote by $Sp(R_u)$ the set of all simple paths of the WDN that satisfies the user constraints R_u:

$$Sp(R_u) = \{sp_1, \ldots, sp_r\}. \tag{15}$$

We will denote by $\cup Sp(R_u)$ the set of all possible unions of elements of $Sp(R_u)$, augmented with the null element φ of Ψ^n, i.e., the WDN with all elements identically equal to zero. The union of two elements of $\cup Sp(R_u)$ is the WDN composed of all the simple paths included in either one of the two considered elements. Every WDN, element of the set $\cup Sp(R_u)$, satisfies the connectivity constraint R_1; furthermore, a feasible structure that fulfills the constraint R_1 is an element of $\cup Sp(R_u)$.

The following proposition characterizes the set of all feasible organizations. Π is a feasible structure if and only if

- Π is a union of simple paths, i.e., $\Pi \in \cup Sp(R_u)$.
- Π is bounded by at least one MINO and one MAXO.

Note that in this approach, the incremental unit leading from a WDN to its immediate superordinate is a simple path and not an individual link. In generating organizational structures with simple paths, the connectivity constraint R_1 is automatically satisfied.

An algorithm has been developed [10] which generates, once the user-defined constraints are specified, the MINOs and the MAXOs which characterize the set of all organizational structures that satisfy the designer's requirements. The solution can be expressed in the form of a set of lattices, with each lattice being defined by a MAXO, a MINO, and the feasible structures in between. The complete set can be represented by a Hasse diagram that also specifies the the construction of any structure by showing the simple paths that comprise it.

The next step of the analysis consists of putting the MINOs and the MAXOs in their actual context, to give them a physical interpretation. If the organization designer is interested in a given pair of MINO and MAXO, because they contain interactions that are deemed desirable for the specific application, he can further investigate the intermediate nets by considering the chain of nets that are obtained by adding simple paths to the MINO until the MAXO is reached.

This methodology provides the designer of distributed intelligence systems with a rational way to handle a problem whose combinatorial complexity is very large.

CONCLUSION

Two quantitative models of distributed intelligence systems have been presented; the formalism of Petri Nets has been used to develop explicit graphical representations of the structures that are supported by algebraic representations. Two different algorithms for generating alternative structures that meet a variety of structural and application-specific constraints have been presented. The first one parameterizes the design with respect to the degree of complexity and the degree of redundancy required in the data flow structure. The second one, the lattice algorithm, generates the complete set of partially ordered structures.

The mathematical framework and the algorithms characterize organizations that have fixed structures, i.e., the interconnections do not depend on the information processing task itself, or on changes in the resources available to the organization. Current research is focused on the characterization of variable information structures using Colored Petri Nets and Predicate Transition Nets, forms of High Level Nets [11].

REFERENCES

[1] M. Minsky, *The Society of Mind,* Simon and Schuster, New York, 1986.

[2] W. Reisig, *Petri Nets: An Introduction,* Springer Verlag, Berlin 1985.

[3] F.Commoner and A. Holt, "Marked directed graphs," *J. Computer and Sys. Science* **5**, 511-523, 1971.

[4] S. K. Andreadakis, "Analysis and synthesis of decisionmaking organizations," Ph.D. Thesis. Lab. for Information and Decision Systems, MIT, Cambridge MA, 1988.

[5] S. K. Andreadakis and A. H. Levis, "Synthesis of distributed command and control for the outer air battle," *Proc. 1988 Symposium on C2 Research,* SAIC, McLean, VA, 1988.

[6] S. K. Andreadakis and A. H. Levis, "Accuracy and Timeliness in Decision-Making Organizations," *Proc. 10th IFAC World Congress*, Pergamon Press, Oxford, 1987.

[7] J. L. Grevet and A. H. Levis, "Coordination in organizations with decision support systems," *Proc. 1988 Symposium on C2 Research,* SAIC, McLean, VA, 1988.

[8] P. A. Remy and A. H. Levis, "On the generation of organizational architectures using Petri Nets," In *Advances in Petri Nets,* G. Rozenberg (ed.). Springer Verlag, Berlin 1989

[9] A. H. Levis, "Information processing and decision-making organizations: A mathematical description," *Large Scale Systems,* **7,** 1984, pp. 151-163.

[10] P. A. Remy, A. H. Levis, V. Y. Jin, "On the design of distributed organizational structures," *Automatica* **24,** 81-86.

[11] J. M. Monguillet and A. H. Levis, "Modeling and evaluation of variable structure command and control organizations," *Proc.1988 Symposium on C2 Research,* SAIC, McLean VA, 1988.