OBJECT ORIENTED DESIGN OF
DECISION MAKING ORGANIZATIONS

Alexander H. Levis
Didier M. Perdu

Paper to be presented at the First Workshop in Command, Control, and
Informations Systems Research, May 3-5, 1994, Eynsham Hall, Oxon. UK.

This work was carried out with support from the Office of Naval Research under
contract No. N00014-93-1-0912.

Center of Excellence in C3I
George Mason University
Fairfax, VA 22030
Object Oriented Design of Decision Making Organizations

Alexander H. Levis  Didier M. Perdu

C3I Center
George Mason University
Fairfax, VA 22030
USA

ABSTRACT

Object oriented analysis and design can be applied to the problem of organizational design and the Colored Petri Net formalism, while the supporting software can be used effectively to implement the approach. Furthermore, existing results can be mapped easily in this paradigm. A dynamical model representation of the organization is obtained that can be analyzed using both Petri Net theory and the associated algorithms, as well as simulation.

1. Introduction

The organizations in Command Centers assess and make decisions about situations that are very uncertain. Decisions have to be made quickly based on a variety of evidence that are often incomplete, noisy, and inconsistent. The manner in which a Command Center is organized to support the Commander affects its performance. Therefore, a command center design needs to be evaluated to make sure that the organization design meets the requirements.

A systematic way to derive organizations based on Petri Nets and Lattice theory was proposed by Remy and Levis (1988) and extended by Zaidi and Levis (1994). The Petri Net model of an organization is obtained by interconnecting models of a single interacting decision maker (Boettcher and Levis, 1982; Levis, 1993). The basic idea of the approach is to generate the set of all organizations that satisfy organizational and structural constraints. This solution set is very large, but is in the form of a lattice of which only the smallest and the largest element need to be characterized. Performance analysis can be conducted on selected elements of the solution set to determine those that satisfy performance requirements.

The problem in this approach is that the organization design is not evolutionary. The incorporation of new technologies that provide alternative options in terms of task allocation, and the sharing of information or the modification of any generic model, requires restating and resolving the problem. Given that many new systems will not be designed or become operational in the next few years, it has become essential to consider approaches that allow for the graceful evolution of existing systems. Designs should be adaptable and be able to incorporate new technologies easily. Furthermore, the recent broadening of the scope of military activities, from providing humanitarian aid in disasters, to limited police actions, to coalition warfare, requires command centers that are easily reconfigurable to meet the needs of the specific situation. The methodology to meet these goals presented in this paper is drawn from the object oriented analysis and design approach used in software systems engineering. Indeed, it turns out that the object oriented approach can have a rigorous instantiation in the problem of decision making organization design using the mathematical framework of Colored Petri Nets.

In object oriented analysis, classes are defined and an object is an instance of a class. An object is a complete entity: it has attributes, behavior, semantics, and defined relationships to other objects. In the object oriented paradigm, concepts such as abstraction, encapsulation, modularity, and hierarchy are well defined. In particular, a hierarchy represents an ordering or ranking of abstractions. This ordering can be based on inheritance (simple or multiple) which allows for specialization or generalization, or aggregation, which allows for decompositions. The difficulty with object oriented design comes in the
handling of dynamics. While event traces represent specific sequences of events, there is no equivalent to the dynamic model description of the organization which allows for analytical as well as computational investigation of the properties of the designed organization. At best, simulation is used to understand behavior.

It is shown in this paper that:
- The object oriented approach can be applied to the problem of organizational design and that the Colored Petri Net formalism and
- The supporting software can be used effectively to implement the approach.

A dynamical model representation of the organization is obtained that can be analyzed using both Petri Net theory and the associated algorithms (e.g., invariant analysis and occurrence graph analysis) and simulation.

An organization is designed using as object classes human decision makers, data bases, processing units, display units, and communications systems, i.e., C3I systems. Sensors and weapons systems can be added, if the application requires it. It is shown that instances of those classes can be used to construct organizations that meet sets of structural and functional constraints. Emphasis is placed in the software objects that represent the organization components. For example, one single class represents the decision makers; the data determine which decision makers are instantiated (objects) in a particular organization.

2. Object Oriented Modeling

There is extensive literature in Object Oriented Programming; more recently, the literature has begun to address problems in Object Oriented Analysis and Design, as applied to software systems engineering. (Booch, 1994; Rumbaugh et al, 1991; Berard, 1993). Even more recently, Sage (1993) reviewed the basic concepts of object oriented design and formulated a methodology for the design of decision support systems. There are several major ongoing efforts to extend the software systems engineering constructs to a full scale systems engineering approach, particularly of software intensive systems such as Battle Management systems, but they have not been documented yet.

The fundamental notion in object oriented design is that of an object: an object is an abstraction of the real world that captures a set of variables which correspond to actual real world behavior (Sage, 1993). An object is a complete entity; it has attributes, behavior, semantics, and relationships to other objects. The boundary of the object is clearly defined. This boundary separates or hides the inner workings of the object from other objects. Interactions between objects occur only at the boundary through the clearly stated relationships with the other objects. The selection of objects is domain specific. In organization design, candidates for objects are decision makers (DMs), communication systems, data bases, workstations, etc.

A class is a template, description, or pattern for a group of very similar objects, namely, objects that have similar attributes, common behavior, common semantics, and common relationship to other objects. In that sense, an object is an instance of a class. For example, 'air traffic controller' is an object class; the specific individual that controls air traffic during a particular shift at an ATC center is an object - an instantiation of the abstraction air traffic controller. The concept of object class is particularly relevant in the design of military organizations, where specific organizational positions are defined, but a wide variety of individuals may serve in those positions over a period of time. Indeed, one of the requirements of military organizations is that they be robust with respect to individual differences.

Encapsulation is the process of separating the external aspects of an object from the internal ones; in engineering terms, this is defining the boundary and the interactions that cross the boundary - the old black-box paradigm. This is a very natural concept in organization design; it allows the separation of the internal cognitive processes of a decision maker from the interactions with other decision makers, either directly or through communication systems.

Modularity is another key concept in object oriented design that has a direct, intuitive meaning in organization design. Modularity, according to Booch (1994) is the property of a system that has been decomposed into a set of cohesive and loosely coupled modules. Consider, for example, the planning section, the operations section, and the intel section in a command center. Each one consists of objects and their interactions; the assumption here is that the objects within a section (module) have a higher level of interactions than there is a cross the modules.

It is fairly clear that encapsulation, modularity, and object class are closely related concepts. Encapsulation is the process by which the boundary characterizing an object class is defined; the nature of the object classes lead to the identification of modules. This process can be top down (Zaidi and Levis, 1994) as well as bottom up (Remy and Levis, 1988).
The concept of hierarchy has its roots in organization theory. The meaning of the term itself refers to the rule of the high priest in ancient Greece and the tree-like organizational structure of the lesser priests below him. In the context of object oriented design, hierarchy refers to the ranking or ordering of abstractions, with the more general one at the top and the more specific one at the bottom. An ordering is induced by a relation and the ordering can be strict or partial. In the object oriented paradigm, two types of relations are recognized: aggregation and inheritance. Aggregation refers to the ability to create objects composed of other objects with each a part of the aggregate object. This concept is well known and understood in organization design and has been exploited as the means to design large organizations. The "line" organizations of the military are constructed in this way: the individual infantryman, a squad, a platoon, a company, a battalion, a regiment, a division, etc. This is a useful construct, even though it tends to oversimplify the organizational structure. As one moves up the hierarchy, the organizational units need to add components or sections that provide services across the organization (staff functions). The concept of aggregation provides the means of incorporating functional decompositions from structured analysis in the object oriented construct.

The second concept, inheritance, has some interesting implications for organization design. Inheritance is the means by which an object acquires characteristics (attributes, behaviors, semantics, and relationships) from one or more other objects. (Berard, 1993) In single inheritance, an object inherits characteristics from only one other object; in multiple inheritance, from more than one object. Inheritance is a way of representing generalization and specialization. The "navigator" in an air crew inherits all the attributes of the "air crew member" object class, but has additional attributes that specialize the object class. The "pilot" and the "copilot" are different siblings of the air crew object class. The inheritance concept has not been explored fully in organizational design where specialization has been thought more in terms of functional decompositions which are partitions of a function.

There are additional concepts in object oriented design, such as polymorphism, persistence, reuse, message passing, and dynamic binding. These refer more to the software implementation aspects; they are considered in describing the Petri Net implementation of the model of the organization design.

3. The Interacting Decision Maker as an Object Class

The basic model of the interacting decision maker (Boettcher and Levis, 1982; Levis, 1993) can be represented in block diagram form as shown in Figure 1. It consists of three processing stages and two interaction stages for a total of five stages. In the structured analysis approach to organization design (Levis, 1993), this model is referred to as a role, since the role can be assigned to different physical assets (decision makers). In the object oriented paradigm, the object class incorporates the role and the particular object, an instantiation of the class, represents that actual DM who implements this role.

A DM receives inputs or data x from the external environment (e.g., from sensors) or from other DMs in the organization. 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 DMs, as shown by the outgoing arrow. If the DM receives data about the assessed situation from other DMs, these data z' are fused together with his 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 DM may receive command inputs v' from 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

![Fig. 1 The five-stage model of a DM.](image-url)
both the revised situation assessment data and the response selection strategy. Finally, the output or response of the DM $y$ 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. 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.

If ordinary Petri Net notation is used, the model of Figure 1 takes the form shown in Figure 2. The rounded box enclosing the five stages has no formal meaning in Petri Net theory; it is used to indicate the internal structure of a DM and show explicitly the inputs and outputs of the DM. However, in the object oriented paradigm, the rounded box denotes the encapsulation of the DM's internal structure and the identification of the six interactions with other objects. The model in Figure 2 may be thought as the notional model of the object class Interacting_DM.

In accordance with Petri Net conventions, transitions are denoted by solid bars and they represent processes or events. Signals or conditions are depicted by places which are shown as circles. Directed arcs indicate the relationships between the two types of nodes. Since Petri Nets are bipartite directed graphs, arcs can exist only from a transition to a place or a place to a transition. A detailed description of Petri net modeling of decision making organizations is given in Levis (1993). This model, however, is too restrictive and does not serve well as the basis for defining the object class Interacting_DM. To accomplish this, a Colored Petri Net model is defined, as shown in Figure 3.

This model has the same five stages as the Ordinary Petri Net model. It has the same six interactions with the environment; all places have the same color set $\text{Info}$. However, it has an additional place with color set $\text{DMp}$ with parameters $\text{sae}$, $\text{ifi}$, $\text{ife}$, $\text{cii}$, and $\text{cie}$. The first parameter, "sae," specifies the number of different information streams coming to the SA stage of the decision maker. Since no fusion of information is done at the SA stage, "sae" can take the values 0 or 1. The second parameter, identified by the variable "ifi," defines the number of internal links between the SA and the IF stage and can take the values 0 or 1 (0 is used when there is no SA stage).

The third parameter, identified by the variable "ife," defines the number of external information streams coming from other decision makers that needs to be fused in the IF stage of the decision maker. The fourth parameter is used if the decision maker does not have the SA and the IF stages, i.e., the DM does not do situation assessment but only executes orders. The corresponding variable "cii" takes the value 0 if this is the case and 1 otherwise. By specifying the five parameters in the color set, not only is a particular decision maker object instantiated, but one of four allowable specialized DMs is obtained.

This example illustrates the inheritance property. All DM objects instantiated from the Interacting_DM class have the same basic structure and properties. However, by setting the five parameter values of $\text{DMp}$, i.e., by marking the place $\text{DMp}$, one of the four siblings of the parent class are obtained. The siblings are shown in Figure 4. The first one represents the DM who receives external inputs as well as processed information from within the organization and produces a response. This could be a model of an intelligence officer or a planner. The second model in Figure 4 represents a forward observer who collects information (acts as a sensor) and communicates it to the rest of the organization. The third model represents a DM who receives some sensor data and then acts while constrained by the prevailing rules of engagement. A fighter pilot in close air support could be modeled by this object. Finally, the fourth model is the general one.
Fig. 3 The Object Class Interacting_DM

Fig. 4 Four specialized DM objects
The interactions between two decision makers can be of two different types: information sharing and coordination of actions. The latter can take two distinct forms: direct coordination through the issuing of commands by the commander to the subordinate, and by results sharing in which each decision maker informs the other of his response, the decision he has made. This model of interactions is shown in Figure 5. It illustrates the modularity of object-oriented constructs. The two-person organization can be decomposed into two a set of two cohesive and loosely coupled modules; in this case, the two shaded objects representing the individual DMs.

However, for organization design, a more detailed view of the interactions is required, a view that is consistent with the model of the object class defined in Fig. 3. Consider two DMs, DMi and DMj. Figure 6 shows the possible flow of data from DMi to DMj. A symmetric set of flows are defined from DMj to DMi. For simplicity of representation, the ordinary Petri Net model is used.

Consider DMi. The first question to be answered is whether he receives data from the external environment, from the sensors. This is denoted by the coefficient $e_i$. If $e_i$ is equal to 1, then DMi receives data from sensors, if it is equal to zero, it does not. The coefficient $g_{ij}$ indicates whether the output of DMi is an input to the situation assessment stage of DMj. This type of interconnection is needed to represent the tandem (series) connection of objects. The coefficient $f_{ij}$ represents the sharing of the assessed situation $z$ by DMi with DMj. This is one type of information sharing. The second type, the sharing of results, is represented by $h_{ij}$. In this case, DMi communicates to DMj 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 DMj. Whether to communicate the assessed situation or to communicate the decision a DM 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 DMi to DMj, as shown by $c_{ij}$ from the response selection stage of i to the command interpretation stage of j. Finally, the coefficient $s_i$ denotes whether DMi produces an output to the environment.

When command center organizational designs are considered, then each node contains only one DM and these six coefficients are constant and take values in \{0, 1\}. If there are $n$ DMs 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, and c are of dimension $n \times n$. From the definition of the arrays (see Fig. 6) it is apparent that the diagonal elements of $F, G, H, and C$ are identically zero. Therefore, any possible set of interconnections among $n$ DMs can be represented by assigning the value of 0 or 1 to $q$ elements where:

$$q = 2n + (4n^2 - 4n) = 4n^2 - 2n$$

(1)
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 $\Sigma$ should be connected; a directed path should exist from the single source place to every node of the net (place or transition) and from every node of the net to the single sink place.

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

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 (Remy and Levis, 1988) can be used in conjunction with this model to generate alternative fixed-structure architectures.

4. The Lattice Algorithm

The analytical description of the possible interactions between DMs forms the basis for an algorithm that generates all the architectures 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 designs. The most important constraint addresses the connectivity of the organization - it eliminates information structures that do not represent a single integrated organization.

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 designs 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 $\Psi_n$; its cardinality is given by $2^q$, where $q$ is given by Eqn. (1). 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 $\Pi$ is a subnet of $\Pi'$ if and only if

\[
e' \leq e; \quad F' \leq F; \quad G' \leq G; \\
s' \leq s; \quad H' \leq H; \quad C' \leq C
\]

where the inequality between arrays is interpreted element by element. In other words, $\Pi'$ is a subnet of

Fig. 6 Admissible data flows from DMi to DMj
If any interaction in \( \Pi' \), i.e., a 1 in any of the arrays \( e', s', F', G', H', C' \), is also an interaction in \( \Pi \). The union of two subnets \( \Pi_1 \) and \( \Pi_2 \) of a WDN \( \Pi \), is a new net that contains all the interactions that appear in either \( \Pi_1 \) or \( \Pi_2 \) or both.

A WDN can be represented in two different ways: (i) The matrix representation, i.e., the set of arrays \( \{ e, s, F, G, H, C \} \), and (ii) The Petri Net representation, given by the graph or the incidence matrix of the net, with the associated labeling of the transitions. These two representations of a WDN are equivalent, i.e., a one to one correspondence exists between them.

Let the organizational structure 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 organizational structures consisting of interconnections of DM objects will be designed, each WDN is not a valid organizational structure. Additional constraints to restrict the set of WDNs to useful information structures are needed. User-defined constraints are introduced to address this issue: 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. 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 \( \Psi^n \) of all WDNs of dimension \( n \). The concepts of maximal and minimal elements can therefore 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, an additional interaction between objects, without violating the set of constraints. Similarly, a MINO is a WDN such that it is not possible to remove a single link without violating the set of constraints. The following proposition is a direct consequence of the definition of maximal and minimal elements.

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

Let \( \Pi \) be a WDN that satisfies constraint 1 and whose source and sink have been merged together into a single external place. A simple path of \( \Pi \) is a directed elementary circuit which includes the (merged) source and sink places. Since the Petri Net representing \( \Pi \) is a marked graph, a simple path is a minimal support S-invariant of \( \Pi \) 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 \( \text{Sp}(\Psi^n) \) the set of all simple paths of the WDN that satisfies the user constraints \( \Psi^n \):

\[
\text{Sp}(\Psi^n) = \{ sp_1, \ldots, sp_r \}
\]

We will denote by \( \cup \text{Sp}(\Psi^n) \) the set of all possible unions of elements of \( \text{Sp}(\Psi^n) \), augmented with the null element \( \varnothing \) of \( \Psi^n \), i.e., the WDN with all elements identically equal to zero. The union of two elements of \( \cup \text{Sp}(\Psi^n) \) is the WDN composed of all the simple paths included in either one of the two considered elements. Every WDN, element of the set \( \cup \text{Sp}(\Psi^n) \), satisfies the connectivity constraint 1; furthermore, a feasible structure that fulfills this constraint is an element of \( \cup \text{Sp}(\Psi^n) \).

The following proposition characterizes the set of all feasible organizations. \( \Pi \) is a feasible structure if and only if

- \( \Pi \) is a union of simple paths, i.e., \( \Pi \in \cup \text{Sp}(\Psi^n) \).
- \( \Pi \) 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 super ordinate is a simple path and not an individual link. In generating organizational structures with simple paths, the connectivity constraint 1 is automatically satisfied.

The Lattice algorithm 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. Then the MINOs and the MAXOs are placed 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.

5. The Lattice Algorithm and Object Oriented Design

A Naval air defense example is used to illustrate the methodology: a Command and Control organization for the outer air battle (Andreadakis, 1988).

Problem Description

The objective of air defense for a naval battle group is to deploy aircraft and set up a screen to locate incoming enemy aircraft, and engage them before they reach a distance from which they can fire their missiles and endanger the carrier, or its escorts. If the enemy missiles have a range \(R_m\), the circumference of the circle of radius \(R_m\), with center the Battle Group, is the weapons release line. The mission is to keep the enemy aircraft (threats) as far as possible from the weapons release line. The engagement outside the weapons release line is called the outer air battle.

Two E2C Hawkeye (airborne warning radar) aircraft patrol two sectors outside the weapons release line. Each E2C commands one squadron of interceptors which are directed to engage the threats in the corresponding sector. The E2C is equipped with passive and active radar. Active radar receives the reflection of its beam by objects (such as other aircraft) while passive radar receives the radar transmission of other aircraft. Each E2C initially operates only the passive radar so as not to reveal its position to enemy units. When enemy aircraft approach the E2C and are about to acquire the E2C on their radar, the E2C turns its active radar on.

Enemy aircraft may be identified on the basis of their emitter signature and their velocity. Based on the assessment of incoming threats in each sector, the decision makers develop courses of action considering the strength of the incoming raid and the type of enemy aircraft. Finally, a response is selected from the available alternatives based on the availability of resources.

Use of Lattice Algorithm

The Lattice algorithm is applied to generate two families of organizational designs. In the first design, three DM objects are interconnected; in the second, four objects are interconnected.

Let the number of DM objects be 3, i.e., \(n = 3\) (hierarchical structure). Further assume that DM1 and DM2 are on the E2Cs and DM3 is on the carrier. DM1 and DM2 sense the environment and have to send their assessments to DM3, who can issue orders to both DM1 and DM2. From these orders, DM1 and DM2 produce the organization output.

Interconnection matrices:

\[
\begin{align*}
F & \quad \text{SA} \rightarrow \text{IF} \\
1 & \quad 2 & \quad 3 \\
1 & \quad 0 & \quad 0 \\
\end{align*}
\]

\[
\begin{align*}
G & \quad \text{RS} \rightarrow \text{SA} \\
1 & \quad 2 & \quad 3 \\
1 & \quad 0 & \quad 0 \\
0 & \quad 0 & \quad 0 \\
\end{align*}
\]

\[
\begin{align*}
H & \quad \text{RS} \rightarrow \text{CI} \\
1 & \quad 2 & \quad 3 \\
1 & \quad 0 & \quad 0 \\
0 & \quad 0 & \quad 0 \\
\end{align*}
\]

When the Lattice Algorithm is applied on this set of connection constraints, it leads to a set of 12 possible organizations bounded by 1 MINO and 2 MAXOs. These three organizational forms are shown in Figures 7, 8, and 9.

The second case considered has four DM objects and represents a parallel structure. DM1 and DM2 are in the first E2C, while DM3 and DM4 are in the second E2C. The two E2C operate independently of the carrier. DM1 and DM3 sense the environment, but DM2 and DM4 produce the organization output. There is no hierarchy, only exchange of information to share local assessments and notify each other of local course of actions so that coordination can be maintained.
When the Lattice Algorithm is applied on the set of connection constraints for the 4 DM case, it leads to a set of 9 possible organizations bounded by 1 MINO and 4 MAXOs. The MINO and one MAXO are shown in Figures 10 and 11. The other three MAXOs are just variants of this one with respect to the link from the RS stage to the IF stage between DM1 and DM3, and DM2 and DM4.

Once these organizational designs are obtained, then they are analyzed to determine their performance characteristics. Since this is an illustrative example, only the two MINOs are analyzed.

6. Adaptation of the Lattice Algorithm to the Object-Oriented Paradigm.

The Lattice algorithm allows the design of decision making organization by interconnecting properly different instances of a model class of decision maker and different instances of a Communications class. Interactions between DMs in this example can only take place through a communications system. The models of the Communications object class is shown in Figure 12.
The interconnection constraints take the form:

The input port is the data to be exchanged from a decision maker; the output port places contain tokens representing exchanged information. Each decision maker is assumed to share the same communications channel and therefore the DMs can only access the communications network one at a time. This is included in the model through the consideration of a Global Fusion Place of color set "CommRes" that contains initially only one token which is shared by the different instance of the Communications model. It takes one unit of time for a message to be exchanged as shown by the time region of the transition ("@+1").
Setting the number of decision makers \( n \) instantiates an equal number of DM objects from the Interacting_DM class. In the Hierarchical Colored Petri Net representation, each DM is represented by a substitution transition - this substitution transition corresponds to the shaded box in Figure 4. The actual DM model, as shown in Figure 3, appears in subordinate pages. Furthermore, the results of the algorithm specify which interconnections to other DMs are used and which ones are absent. This information is used to "specialize" the DM object, as described in Sections 2 and 3.

The set of interconnection matrices that define the interactions between decision makers instantiates places in the organization page (the upper page in the hierarchy) connected to the relevant transitions. In this way, port and matching socket nodes are created on corresponding higher and lower level pages in the hierarchical Colored Petri Net. The following procedure is followed:

- From the chosen MINO, MAXO or an intermediate design between a MINO and MAXO, identify the active places, and erase the non active ones.

- Combine (fold) the places that are inputs to the same stage of other decision makers. The number of such places defines the parameters "ife" or "cie" of the decision maker to be included in the place DMp of the corresponding instance.

- Insert a communications object so that every message produced by a DM is "broadcast" to those decision makers specified by the Lattice algorithm solution. For example, the message produced at the RS stage of a DM has to be sent to the IF stage of two other DMs. A substitution transition that has as its page model an instance of the Communication object class is added; its input socket is a single place corresponding to the output place of the RS stage, the output places are the places that define inputs (shared information) to the other decision makers.

- Perform the port assignment between the places (sockets) of the upper page and the port node at the lower pages of the different DM objects (instances of the Interacting_DM class) and Communication objects.

- Use the port assignment process to derive the parameters necessary for the specialization of each decision maker using the following rules:

  - No SA input => "sae" = 0 and "ifi" = 0,
  - SA input => "sae" = 1 and "ifi" = 1.
  - No SA input and No IF input => "cii" = 0

**Results**

The organization design for the three DM case, when the MINO architecture generated by the lattice algorithm is used is shown in Figure 13. In this case, the specialization parameters for DM1 and DM2 are \((1,1,0,1,1)\) (they have the same structure). The parameters for DM3 are \((0,0,2,1,0)\). The two decision makers in the E2C aircraft send their assessed situation to the carrier; the coordinator in the Tactical Control center on the carrier fuses the information with information available at the TCC and issues instructions/guidance to the E2C DMs. The two E2Cs do not communicate directly; all coordination is done by the TCC on the carrier.

The four DM case design, using again the MINO solution of Figure 10, is shown in Figure 14. The parameters for each decision maker are the same: \((1,1,1,1,0)\). All four decision makers have the same structure. In this case, when there is no carrier, each E2C has two DM aboard. DM1 and DM3 carry out situation assessment and information fusion tasks; they share information and produce a consistent tactical picture for DM2 and DM4. The last two choose courses of action and execute them; however, they provide additional coordination by informing each other of their course of action. In this design, there is no leader or follower; both E2C operate as equals (parallel) structure), but they coordinate fully with each other.

These are complete designs. Each model contains a source and a sink, all interactions (message passing) between DMs is defined precisely, the individual DMs are represented as objects, and the technical means of communication between them are represented by instances of a generic communications object. The objects themselves, the DMs, are specialized by the initial marking in the place DMp. This marking, which represents system data, defines which interconnections are active and specifies the message passing rules. The methods, the procedures or
functions performed at each decision making stage, are embedded in the code segments associated with each transition and the annotations of the Colored Petri Net arcs. The specifics of constructing code segments and of writing arc inscriptions can be found in Jensen (1993) and in the Design/CPN™ Reference manual (1993).

The advantage of having used Colored Petri Nets as the means of modeling organizations becomes apparent now. The models shown on Figures 13 and 14 are fully executable: they represent dynamic discrete event systems. They contain more information than state diagrams and can generate, in simulation mode, event traces as the result of initial markings.

7. Conclusions

A mathematical theory for organization design has been recast in the framework of object oriented design. It has been shown that this is a more natural framework that the structured analysis one for this application. Furthermore, the formalism of Hierarchical Colored Petri Nets supports easily the constructs of object oriented analysis and design and leads, without additional effort to a fully executable model of the design.

References


Fig. 14 The 4 DM Organization Design


