2001

A Model to Evaluate the Effect of Organizational Adaptation

Holly A. H. Handley
Old Dominion University

Alexander H. Levis

Original Publication Citation

Handley, H. A., & Levis, A. H. (2001). A model to evaluate the effect of organizational adaptation. *Computational & Mathematical Organization Theory, 7*(1), 5-44. doi: 10.1023/A:101134261517


Computational & Mathematical Organization Theory 7, 5–44, 2001.

A Model to Evaluate the Effect of Organizational Adaptation

HOLLY A.H. HANDLEY AND ALEXANDER H. LEVIS
George Mason University, 4400 University Drive, MSN 4D2, Fairfax, VA 22030, USA
email: hhandley@gmu.edu
email: alevis@gmu.edu

Abstract

When an organization's output declines due to either internal changes or changes in its external environment, it needs to adapt. In order to evaluate the effectiveness of different adaptation strategies on organizational performance, an organizational model composed of individual models of a five stage interacting decision maker was designed using an object oriented design approach and implemented as a Colored Petri net. The concept of entropy is used to calculate the total activity value, a surrogate for decision maker workload, based on the functional partition and the adaptation strategy being implemented. The individual decision maker's total activity is monitored, as overloaded decision makers constrain organizational performance. A virtual experiment was conducted; organizations implementing local and global adaptation strategies were compared to a control organization with no adaptation. The level of tolerance of the organization, the workload limit based on the concept of the bounded rationality constraint, was used to determined when a decision maker was overloaded: the limiting effect of the workload on performance. The timeliness of the organization's response was used in order to evaluate organizational output as a function of adaptation strategy.

Keywords: organizational model, adaptation strategy, decision maker workload, organizational performance

1. Introduction

An organization can be considered a system composed of individual decision makers. The organization has a set of tasks that it must do to accomplish its mission. A task is such that it is too complex for a single decision maker to accomplish alone, however the task can be decomposed into functions that can be completed by the individual decision makers. A task graph is developed that represents the information pattern between these functions necessary to complete the task. This task graph is referred to as a process: the series of functions required for each task to produce a response. When changes occur, either within the organization or its environment, to such an extent that the organization is no longer performing adequately or can not perform in its current configuration, the organization is forced to adapt, or reconfigure itself, in response to these changing circumstances, while still continuing to operate.

Three different strategies for organizational adaptation can be identified: none, local, and global. In some cases, no changes in the organizational structure may be the appropriate response to change. Since any type of change imposes an overhead on the organization, for example having more than one decision maker available for a single function,

an organization that does not accommodate change can be considered a "low cost" organization; the workload of the individual decision makers is lower than in organizations that can adapt. Alternatively, local changes within the organizational structure can be made by identifying secondary or backup decision makers allowing tasks or resources to be transferred among the decision makers (Serfaty and Entin, 1995; Perdu and Levis, 1998). This strategy maintains the process or sequence of functions necessary to complete the task. When the decision maker who has primary responsibility for a function is over loaded, the back up decision maker for that function is selected to decrease the primary decision maker's workload. In order for this method to be effective, there must be alternative decision makers available who are able to complete the same function. This additional information on secondary decision makers is the overhead associated with local adaptation and can be shown to increase a decision maker's workload.

By releasing the constraint on maintaining a fixed process associated with a task, a third adaptation strategy can be implemented by identifying alternative processes capable of completing the same task. This global adaptation represents a change in the organization's strategy for accomplishing a given task (Handley, 1999). When the organizational output no longer meets the performance requirements of a task, a new process is implemented to complete that type of task. For this method to be effective, there must be alternative task processes identified. This information on alternative process is the overhead associated with global adaptation and can also be shown to increase the decision maker's workload. While the local strategy advocates offsetting individual decision maker's workload while maintaining the same task process, the global strategy implies that selecting a new process will result in improved performance. Carley (1998) discriminates between adaptation at the strategic and at the operational levels.

The first method of adaptation, implementing local changes, is similar in concept to workload balancing. Perdu (1998) used this method to off load tasks to other decision makers when the number of waiting tasks reached a maximum value. He calculated a Load Balance parameter, which measured how far the normalized load distribution was from a uniform distribution. The second method of adaptation, global change, releases the constraint on maintaining a fixed process associated with a task, representing a change in strategy for completing a task. Although this is similar to the concept of business process reengineer, those changes are made off line; as a method of adaptation process change is accomplished dynamically. This may redistribute the workload, as a new process may introduce new functions and new decision makers. However, the impetus for process change is not the workload of the decision makers directly, but rather the degradation of the task outcome, which is induced by overloaded decision makers.

This paper presents a model with which to evaluate the performance of an organization under different adaptation strategies, which in turn induce different workload overheads. Two alternative approaches have been used to design organizational models for subject experiments: structured analysis and object oriented. The structured analysis approach results in a rigid model of the organization (Handley et al., 1997) while the object oriented approach leads to an organizational model that can exhibit change (Handley et al., 1998; Perdu and Levis, 1998). An object oriented approach was used to design this organizational model; objects facilitate modularity and reconfiguration by incorporating data and functions

together leaving only the interface visible to the designer. Modeling the decision maker as an object class allows different interactions to be instantiated based on the process needs; decision makers within an organization coordinate to complete tasks based on the indicated process. The decision maker's total activity, as he receives and completes functions, is monitored to determine his workload status. The organizational output is also monitored. An executable model of the organization was created as a Colored Petri net, a discrete event system modeling tool which captures the precedence relations and structural interactions of concurrent and asynchronous events, using the objects and relationships indicated in the collaboration diagram from the object oriented design.

The prominent component of the organizational model is the decision maker. The five stage interacting decision maker model of Levis (1992) was used as the basis of the decision maker model. Two other decision maker models have been implemented recently, an adaptive decision maker (Perdu and Levis, 1998) and a reconfigurable decision maker (Handley et al., 1998). However, there is a need to have a consistent model of the decision maker throughout the study of adaptive organizations and the five stage model has such a history of development and validation (Louvet et al., 1988; Jin and Levis, 1992). Different roles for the five stage decision maker model were identified by noting the allowable input and output patterns required for different functions. These roles were then folded back into one model and implemented as a Colored Petri net, a sub page in the hierarchical organizational model. Stored within the model is an execution table, which returns the output data value for an input data value for a given function; this represents the algorithm implementation of each function. Also included in the model are a process rulebase, which indicates the next function(s) in a specified process, and a next decision maker table, which identifies which decision maker can perform the next function(s). There is also a backup decision maker table that provides secondary decision maker information for overloaded primary decision makers.

The algorithms within each stage of the model can be used to calculate the total activity of the decision maker, a surrogate for the decision maker's workload. Since the adaptation strategy being implemented activates different algorithms in the model, different adaptation strategies result in different total activity values for the decision makers. While a previous study of local adaptation limited the total number of functions a decision maker could accept in order to avoid overloading (Perdu and Levis, 1998), using total activity as an indicator of cognitive workload provides a better indicator of the decision maker's status based on his functional abilities and the adaptation strategy; it is the cognitive limitation of the decision makers that manifests itself as a bound on the organizational performance. Under different levels of tolerance, the overhead for maintaining the adaptation strategy, as indicated by decision maker workload, may limit organizational performance.

Models of organizations can be used to study organizational behavior under different conditions in order to address questions on adaptation. Pre-experimental modeling is an important step in the experimental design process of subject experiments (Handley et al., 1997, 1998). By constructing a software model of the experimental situation, the behavior of the organization under design can be observed before the subject experiment is conducted. Organizational models can also be used to conduct virtual experiments (Carley, 1995). The results from the virtual experiment can then be used to design experimental organizations that behave similarly for further study. The consequences of the different

adaptation strategies are illustrated by conducting a virtual experiment which simulates the organizational model under three different adaptation strategies: none, local, and global. Data from a recent subject experiment was used to populate the model and create input scenarios of tasks for the organizational model to respond to; four task processes and two alternative organizational structures were used. Stress on the organization was induced by increasing the interarrival time of the tasks or the number of arriving tasks. As the decision maker's total activity became greater than the workload bound, functions become delayed. During normal operation the task processing stage of all decision makers is equally delayed at one time unit, however functions causing a decision maker to become overloaded incur additional delay, thus degrading the organizational performance. The workload limit was varied among three different values, low, medium and high, during different iterations of the simulations.

The organization output was evaluated on performance measures that relate organizational performance to adaptation strategy. For a given task, there is only a limited period of time during which the organizational response will be effective. This time period is called a window of opportunity. The timeliness of each organizational response was scored based on the task's window of opportunity; if the response was within the window, it was given a score of one, otherwise it received a score of zero. Based on the score, the percentage of timely tasks can be calculated and used to compare the performance of different organizations under different adaptation strategies.

Studies of teams using local change as an adaptation mechanism have concentrated on changing team structure to maintain performance (Serfaty and Entin, 1995) or allowing tasks or resources to be transferred among the decision makers (Perdu and Levis, 1998). These teams have the ability to adapt to task demands in order to maintain the perceived stress at tolerable levels while maintaining team performance. Superior command teams adapt their coordination processes to changes in both the task environment and the organizational structure in an attempt to achieve acceptable performance while not exceeding workload tolerances (Serfaty, 1996). Carley (1995) found through computational models that organizations often alter their structure in response to their performance. However, she concluded that altering the organizational structure in response to minor shifts in performance tends to limit organizational performance. The rate of response has more impact on performance than the personnel change strategy. She also found that the type of change is more important than the amount of change, if organizations are to be adaptive (Carley and Lee, 1997). Some changes made by the organization did not serve to improve performance or even maintain current levels of performance. In this study, performance was used as an indicator of the effectiveness of the change or adaptation. If performance degraded after the change, the change was not considered an adaptation.

Aldrich (1999) applied evolutionary theory to organizational emergence and focused on emergence and change instead of structure and stability. Evolutionary models treat origins and persistence as inseparable issues, encompassing many levels and units of analysis, typically taking an interdisciplinary approach. Carley (1995a) presents a pertinent review of the computational and mathematical models that address organizational evolution and change; she describes the interdisciplinary approach to organizational theory using formal models and the relationship between the organization and its environment.

In Section 2, the organizational model is developed using an object oriented approach. Section 3 describes the five stage interacting decision maker; the structure and functionality of each stage are described in detail. Section 4 describes how ideas from information theory can be used to compute the individual decision maker's total activity, which will be used to monitor the decision maker's workload. The virtual experiment is presented in Section 5 as an example of the model, methodology, and effects of different adaptation strategies.

2. Organizational Model

An organization is formed in order to perform a set of tasks that individual decision makers cannot perform alone. The tasks to be performed by the organization consist of receiving signals or inputs from one or more sources, processing them, and producing outputs which can be actions or signals (Boettcher and Levis, 1983). The structure of the organization is defined by the interactions between the decision makers. Models of organizations are used in order to answer a set of specific questions about its behavior. As the questions asked about organizations start to deal with change and adaptation, the models designed to answer these questions must be able to exhibit change. Many of the early models of organizations used a structured analysis approach in order to design the behavioral models. These models are based on a fixed, functional decomposition of the tasks in the system (Levis, 1999). The functions are then arranged in a specific partial order that when activated will achieve the goal. This partial order represents the process that the system invokes to respond to the task. This model of the system is fixed; any change made to the organization or to the process requires a change in the structure of the model, therefore this type of model can not answer questions about change.

An alternative modeling approach used for organizational modeling is the object oriented approach, which borrows techniques from software engineering. The object oriented methodology begins with the concept of an *object* (Rumbaugh et al., 1991). An object is an abstraction of the real world that captures a set of variables which correspond to actual real world behavior (Sage, 1993). These objects are instantiated by the model as they are needed to represent the behavior of the organization. This approach can be used to create a model where the process is not fixed, but is represented by the order of the functions as they are invoked.

This work builds on a stream of research of variable structure and adaptive organizations modeled as Petri nets (see Remy and Levis, 1988; Levis, 1992). While the earlier efforts tried to use the structured analysis approach, the rigid models soon showed their limitations. By experimenting with the object oriented approach, models were developed that could exhibit change (Handley et al., 1998; Perdu and Levis, 1998). In either case the static models were converted to a Petri Net for an executable model. This work extends the previous object oriented work by using the Unified Modeling Language and identifying the methodology with which to create the executable model. By continuing to use Petri nets for the executable model, previously developed tools can be used for the analysis of results.

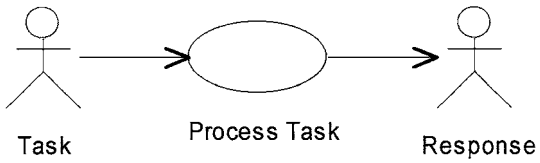


Figure 1. Use case diagram.

2.1. Object Oriented Design

The organizational model was developed using the Rational Rose® case tool (Rational Software Corporation, 1998). The notation used throughout the model is the Unified Modeling Language (UML) of Booch et al. (1998). The model development followed that of Gomaa (1998) for reusable software architectures. Note that the objective of this object oriented approach is not to generate code, but rather to complete a high level design of the organizational model.

The first step in creating a model of a system is to define the use case views. A use case is a snapshot of one aspect of a system; it documents the interaction between a user, or the environment, and the system. In the use case diagram, a circle represents a description of a major flow of events within the system and is labeled with the system goal that is achieved. The actors, shown as stick figures, represent external objects that initiate or are effected by the use case. Figure 1 shows the use case, Process Task. It represents the functionality of the organization that occurs when a Task initiates a process and the Response that is created when the process is complete.

In order to describe the functionality of the system associated with the use case, the logical view of the model, or class diagram, is created which describes the types of objects in the system and the various kinds of static relationships, associations, between them. Figure 2 shows the logical view of the system. The object class Organization is an aggregation of four object classes: Communications, DM_Monitor, Decision_Maker, and Org_Monitor. The Communication object class allows messaging within the Organization, the DM_Monitor object class provides the status of the Decision_Maker objects to the Organization, the Org_Monitor provides the score to the Organization, and the Organization is composed of Decision_Maker objects. The DM_Monitor monitors the Decision_Maker as he performs Functions. The Organization implements a Process which consists of Functions. The interactions of these objects implements the use case described above.

Now that the object classes have been identified, interaction diagrams can be defined that show the specific objects involved in the use case. A sequence diagram is one type of interaction diagram. In a sequence diagram, vertical lines represent objects and horizontal arrows represent messages. Only the source and destination of the arrow are relevant; the message is sent from the source object to the destination object. Time increases from the top of the page to the bottom, however spacing between messages is not relevant. Figure 3 shows a sequence diagram for the organization completing a process in the case where no adaptation is allowed. The actors from the use case initiate and conclude the sequence diagram. When a process is initiated by the arrival of a Task, the Process indicates the

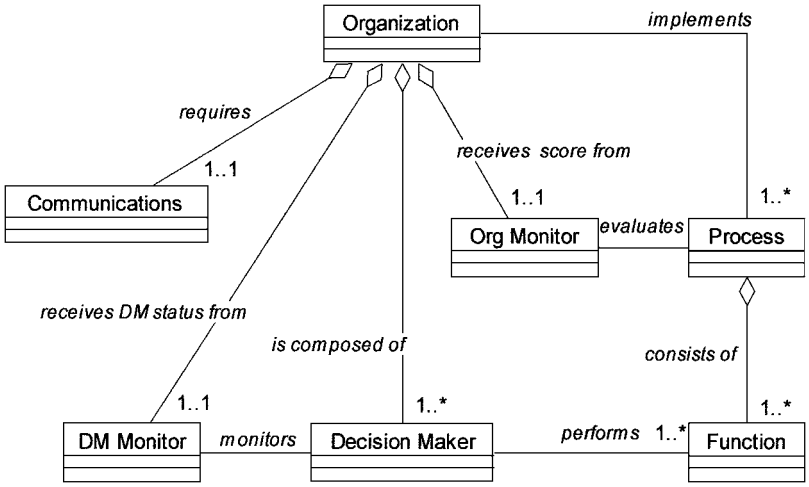


Figure 2. Logical view of the organization.

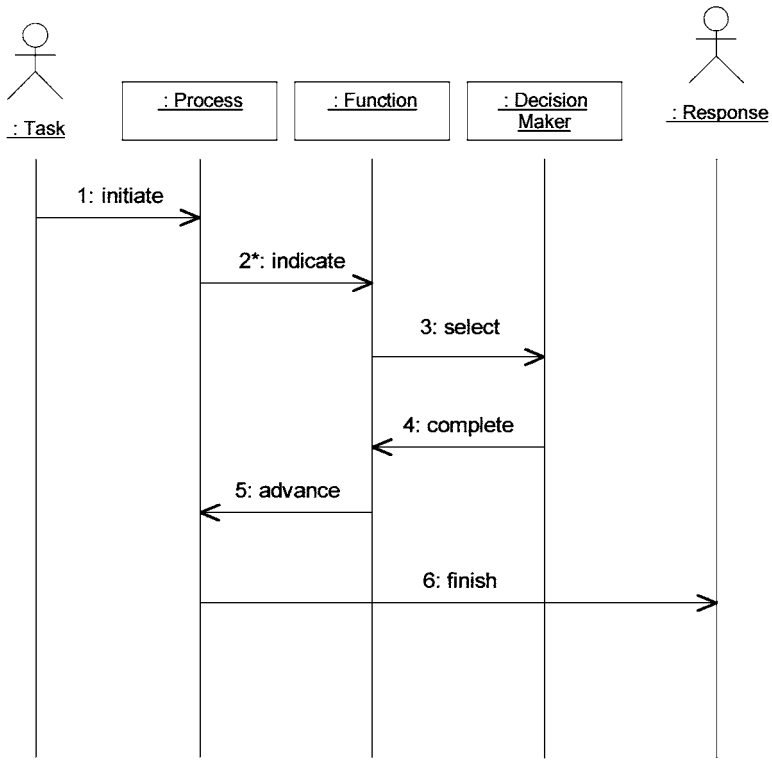


Figure 3. Sequence diagram for organization with no adaptation.

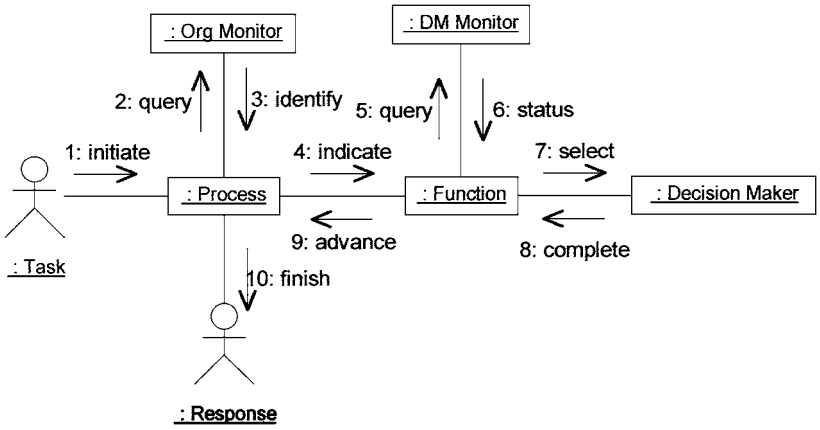


Figure 4. Combined collaboration diagram.

Function to be sent to the Decision_Maker. This stimulates the Decision_Maker's internal operation until he completes the Function. The Function then advances the Process, and the next Function and the next Decision_Maker are identified. This cycle repeats until the Process indicates that it is complete and a Response is made. Likewise sequence diagrams for organizations using the local and global adaptation strategies can be developed. In the case of local change, the output of the DM_Monitor is checked before the next Decision_Maker is chosen. In the case of global change, the Org_Monitor is checked for the correct process before a process for a new task is initiated.

The way objects dynamically cooperate with each other can also be depicted in an object collaboration diagram. The collaboration diagram is another view of the sequence diagram. The sequence in which the objects participate is maintained through message sequence numbers, however the time lines are collapsed. A single collaboration diagram can be made that represents the interactions of the object classes for all modes of adaptation. This is shown in figure 4.

The combined collaboration diagram is now a good approximation for the design of the organizational model. However, one of the objectives of this research is to create an adaptable version of the five stage decision maker model; this five stage decision maker model must include the process, function, and decision maker information as will be described in the next section. Therefore the three objects Process, Function, and Decision_Maker can be replaced by a representation of the Five_Stage_DM_Model. Still missing from the model is the Communications object class. The Communications model simply contains the logic for routing messages to the decision makers to perform the iterative functions in order to complete the process; it can be represented as a rule set. The complete organizational model is now shown in figure 5. The organization model is composed of four submodels: the Five_Stage_DM Model, the Org_Monitor model, the DM_Monitor model, and the Communications model.

The use of Petri nets to represent object oriented specifications is an on going field of research. Many frameworks have been proposed, often with intermediate Petri net formalisms

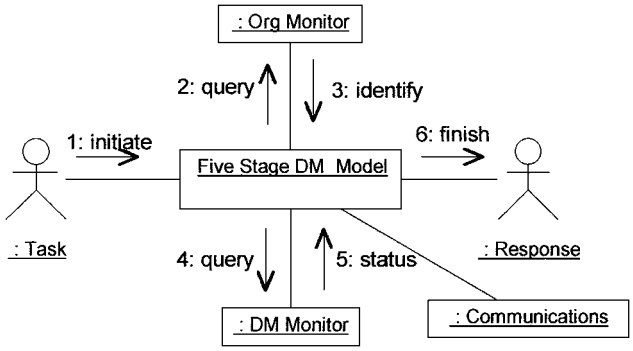


Figure 5. Complete organizational model.

or code generators, in order to create truly Object Oriented Petri nets. The area of inheritance and the associated polymorphism and dynamic binding continue to pose a problem (see Lakos, 1995). For this work a Colored Petri net was used to implement an object oriented design without including these properties; the modularity and reconfigurability were the most important attributes of the model. The individual decision makers are instantiated by the model as they are needed to represent the behavior of the organization. By modeling the decision maker as an object class different interactions could be instantiated between the decision makers based on the needs of the process; in previous approaches the interactions and ordering of the decision makers was fixed.

2.2. Colored Petri Net Executable Model

The object oriented design of the organization provides the top page in a hierarchical, Colored Petri net model. Petri nets provide a graphical modeling language with which to represent a system and an underlying mathematical theory for rigorous analysis (Murata, 1989). They can represent the external interactions of the decision makers as well as any internal algorithms the decision maker must perform. Ordinary Petri nets are bipartitie directed graphs (Peterson, 1981; Reisig, 1985). There are two sets of nodes: places denoted by a circle node and transitions modeled by a bar node. The arcs or connectors that connect these nodes are directed and fixed. They can only connect a place to a transition or a transition to a place. A Petri net also contains tokens. Tokens are depicted graphically by indistinguishable dots and reside in places. A marking of a Petri Net is a mapping that assigns a non negative integer, representing the number of tokens, to each place. A transition is enabled by a marking, if and only if all of its input places contain at least one token. An enabled transition can fire. When the firing takes place, a new marking is obtained by removing a token from each input place and adding a token to each output place. The dynamical behavior of the system is embedded in the changing of the markings.

Colored Petri nets are an extension of Petri nets (Jensen, 1990). Instead of indistinguishable tokens, tokens now carry attributes or colors. Tokens of a specific color can only reside in places that have the same color set associated with them. The requirements to fire a

transition are now specified through arc inscriptions; each input arc inscription specifies the number and type of tokens that need to be in the place for the transition to be enabled. Likewise, output arc inscriptions indicate what tokens will be generated in an output place when the transition fires. Code segments can also be associated with a transition; these are blocks of code that execute when the transition fires, representing the functionality of the transition. A global declaration node of the Colored Petri net contains definitions of all variables, color sets, and domains for the model.

The top page of the Colored Petri net model of the organization is shown in figure 6. The place with the color set *Scenario* must contain an initial marking of the list of tasks and their entrance times used to simulate the model. Each task enters the model at its indicated time through the first transition. Each task is initiated by a decision maker who completes the first function and launches the process for that task. When one decision maker completes a function in the process, he uses his internal algorithms to identify the next function and the next decision maker in the process until the task is complete.

The two places with color sets *Super* and *Bckup* are given initial markings to indicate the adaptation strategy to be used by the organization. The Communication sub page routes information among the decision makers and the monitors. The Org_Monitor provides the organization's state information by generating a score for each task when it is completed. The initial decision maker in a task uses this information to select the process for a new task based on the last similar task's score, when the organization is operating under global adaptation. The DM_Monitor provides decision maker status information by evaluating each decision maker's workload as functions are initiated and completed. The decision makers use this information to select alternative decision makers for overloaded decision makers when the organization is operating under local adaptation. The overloaded decision maker information is also used to insert a processing delay whenever an overloaded decision maker receives an additional function. The model continues to process tasks until the task list is exhausted and there is a score for each task.

3. Decision Maker Model

The main component of the organizational model is the five stage decision maker model. March and Simon (1958) hypothesized that decision makers follow a two step process: first determining the situation and then determining a response. This led to a two stage decision maker model by Wohl (1981) and was expanded to four stages by Boettcher and Levis (1982). Remy and Levis (1988) formalized the interactions between decision makers based on the four stage model. Levis (1992) presented a model of a five stage interacting decision maker that subsumed the previous models. The model presupposes that the decision makers are executing well-defined tasks for which they have been trained and that there is a limit to the amount of processing a decision maker can perform (Boettcher and Levis, 1982).

3.1. Five Stage Decision Maker Model

The five stage decision maker model of Levis (1992) is shown in figure 7. The decision maker receives a signal, x, from the external environment or from another organization

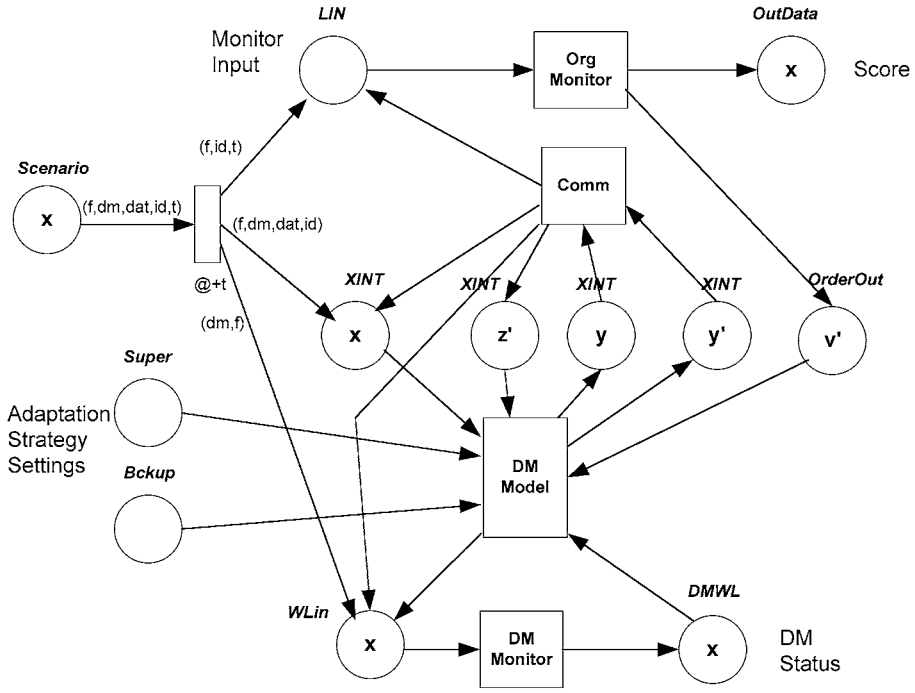


Figure 6. Colored Petri net model of the organization.

member. The situation assessment stage (SA) represents the processing of the incoming signal to obtain the assessed situation, z, which may be shared with other decision makers. The decision maker can also receive a signal z' from another decision maker; z' and z are then fused together in the information fusion (IF) stage to produce z''. The fused information is then processed at the task processing (TP) stage to produce v. A command or control

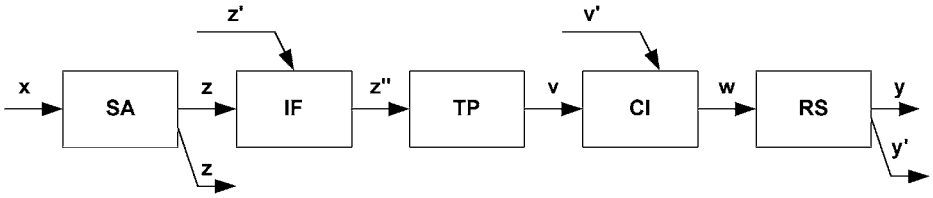


Figure 7. Five stage interacting decision maker (Levis, 1992).

information from another decision makers is received as v'. The command interpretation (CI) stage then combines v and v' to produce the variable w, which is input to the response selection (RS) stage. The RS stage then produces the output y to the environment, or the output y' to other decision makers.

The model depicts the stages at which a decision maker can interact with other decision makers or the environment. A decision maker can receive inputs from the external environment only at the SA stage. However, this input x can also be from another decision maker's y' from within the organization. A decision maker can share his assessed input through the z output at this stage. The z' input to the IF stage is used when the decision maker is receiving a second data input. This input must be generated from within the organization and can be the output of another decision maker's SA or RS stage. The fused information from the IF stage, z', is the input to the TP stage. The decision maker's function is performed at this stage and results in the output v. In the CI stage, the decision maker can receive control information as the input v'. This is also internally generated and must originate from another decision maker's RS stage. In the RS stage, an output is produced; y is the output to the environment and y' is the output to another decision maker. Thus the interactions between two decision makers are limited by the constraints enumerated above: the output from the SA stage, z, can only be an input to another decision maker's IF stage as z', and an internal output from the RS stage, y', can only be input to another decision maker's SA stage as x, IF stage as z', or CI stage as v'.

Subsets of the five stage decision maker model, called *roles*, can be defined depending on the requirements of the function being performed by the decision maker. Since a single decision maker may be assigned to a variety of functions, he may be required to perform a variety of roles with different frequency during the execution of a process. By defining the different types of roles that may be required for the different functions as subsets of the five stage model, these individual role models can then be folded back into one decision maker model. This allows one decision maker model to represent the different roles that the decision maker may require while completing functions during the course of a process.

Since there are strong constraints in the allowable interactions between a decision maker and the external environment, as well as between a decision maker and the rest of the organization, as described in the preceding section, one way to define the required roles is in terms of the allowable interactions for each role. In other words, the different roles a decision maker needs to assume in the course of a process can be defined by the inputs he receives and the outputs he produces for each specified function he performs. The model has three inputs, x, z', and v' which are inputs to the SA, IF, and CI stage respectively. It also has

Table 1. Roles and characteristic vectors.

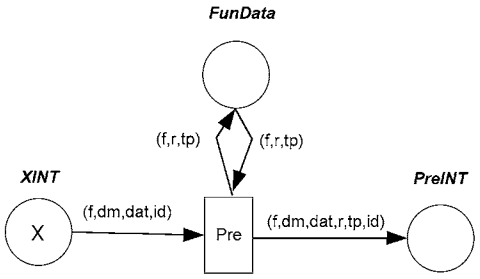
Role	$(x, z', v' : z, y, y')$
Independent (I)	$(1, 0, x : 0, 1, 0)$
Output coordinating (OC)	$(1, 0, x : 0, 1, 1)$
Input coordinating (IC)	$(1, 1, x : 0, 1, 0)$

three outputs, z, y, and y' which are outputs from the SA, RS, and RS stages respectively. Therefore the role required by a function can be specified by a vector representing these inputs and outputs as $[x, z', v' : z, y, y']$. A one indicates that input or output is required, a zero indicates that it is not. Note that not all combinations are acceptable roles; the acceptable roles for each function in the process must be defined.

In the *independent role* (I) a decision maker can complete the function without interacting with other decision makers. This role only requires the x input and produces a y output. In the *Output Coordinating* role (OC) the decision maker needs to send his output to two or more subsequent decision makers. This role requires the input data x and both outputs y and y'. In the *Input Coordinating* role (IC) the decision maker needs to receive inputs from two or more prior decision makers to complete his function. A decision maker in this role requires the input x and the information fusion input, z' and produces the output y. Each of these roles is defined as a unsupervised mode, since none of them receive a v' input. If the process is being completed autonomously by the decision makers, then these are the roles that are necessary. However, by adding the v' input to each role, the roles are now supervised. The supervised mode is necessary to insert control information into the decision maker model when adaptation is present. Table 1 summarizes the three roles with their characteristic vectors. Note that the x in the v' column represents that the role can be unsupervised if the value of x is zero and supervised if the value of x is one.

3.2. Colored Petri Net Decision Maker Model

The complete decision maker model is implemented using Colored Petri nets. The executable model of the decision maker must fold together all the roles described above into one model and yet allow any one role to be active at a time. The role required by the decision maker to perform the requested function defines the correct structure and functionality of the model; the internal processing, or algorithms of each stage of the model must also occur correctly depending on both the function and the role. Thus the model includes a pre stage that identifies the role and the type for the requested function. Figure 8 shows the Colored Petri net and the declaration node for the pre stage. The input to the model is in the place labeled "x" with the color set *XINT*. It is a tuple of four variables: function (**f**), decision maker (**dm**), data (**dat**), and task identifier (**id**). The function is the function being requested of the specified decision maker. The data variable holds the input data to be used by the function and the identifier tracks which task is being processed. Each function has attributes **r** and **tp**: **r** indicates the role required for the function, and **tp** indicates if the function has

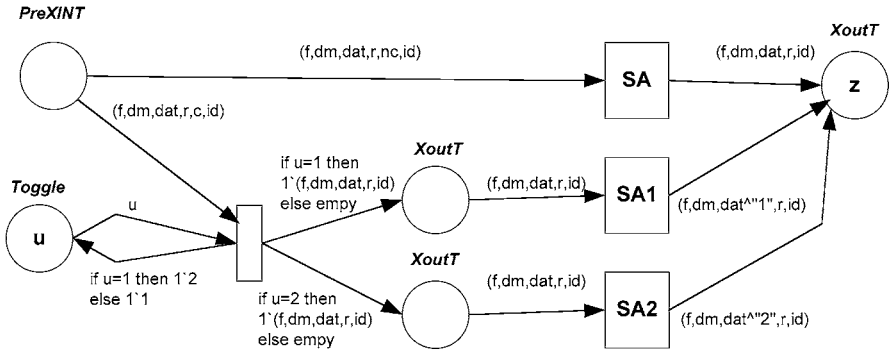



```
color Function = with f1|f2|f3|f4|f5|f6|f7|f8|f9|f10;
var f : Function;
color DecMaker = with DM0|DM1|DM2|DM3|DM4|DM5|DML;
var dm : DecMaker;
color Data = string timed;
var dat,id : Data;
color Role = with s|ic|oc;
var r : Role;
color Type = with c|nc;
var tp : Type;
color XINT = product Function*DecMaker*Data*Data;
color PreINT = product Function*DecMaker*Data*Role*Type*Data;
color FunData = product Function*Role*Type;
```

Figure 8. Pre stage.

alternative algorithms available; **r** can take the values *ic* (input coordinating), *oc* (output coordinating) and *i* (independent) while **tp** can take the values *c*, indicating a choice of algorithms, or *nc*, indicating no choice of algorithms. The place with the color set *FunData* must have an initial marking which specifies the **r** and **tp** values for each function. The pre transition returns the values of these function attributes, **r** and **tp**, which are then added to the output tuple of this stage. The color set, *PreXINT*, at the output place includes these variables into the output tuple. This stage is considered a preparation stage as it does not do any processing of the function, it simply identifies attributes necessary for the correct implementation of the role model.

In the situation assessment stage, the decision maker receives an input x from the environment, or another decision maker, and processes this information to produce a z output to the next stage. The Colored Petri net and declaration node for this stage are shown in figure 9. The x input is the output from the pre stage at the place with the color set *PreXINT*. The situation assessment stage uses the value of **tp** to determine the path through the stage. If the value of **tp** is *nc*, no choice, the top path of the stage to transition SA is active and the input tuple is copied to the output tuple at the place labeled "z". The color set, *XoutT*, drops the type variable from the tuple as it is no longer needed. If the value of **tp** is *c*, the lower path through the stage is active. The choice of lower paths, through either transition SA1 or SA2, depends on the decision switch, **u**. The decision switch can take various forms, i.e., it can be a probability distribution on u or on $u \mid x$ where the knowledge of x conditions the selection of the algorithm. The decision switch represents the fact that

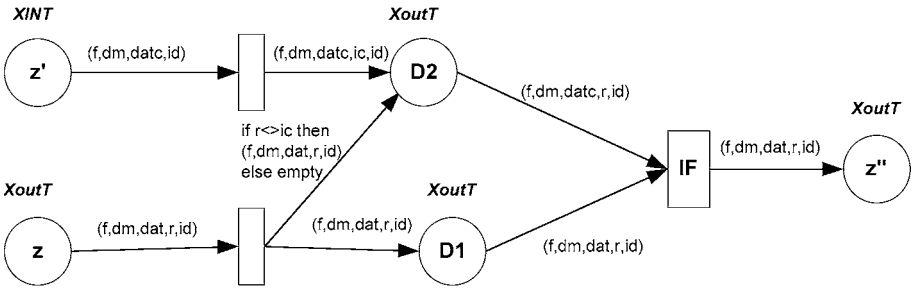



```
color Function = with f1|f2|f3|f4|f5|f6|f7|f8|f9|f10;
var f : Function;
color DecMaker = with DM0|DM1|DM2|DM3|DM4|DM5|DML;
var dm : DecMaker;
color Data = string timed;
var dat,id : Data;
color Role = with s|ic|oc;
var r : Role;
color Type = with c|nc;
var tp : Type;
color XINT = product Function*DecMaker*Data*Data;
color PreINT = product Function*DecMaker*Data*Role*Type*Data;
color XoutT = product Function*DecMaker*Data*Role*Data;
color Toggle = int;
var u : Toggle;
```

Figure 9. Situation assessment stage.

in some functions, alternative algorithms are available for the decision maker to use. Both algorithms are appropriate, and the decision maker simply makes a choice between them. In this model, the decision variable **u** is an alternating deterministic sequence, indicating that the decision maker alternates his choice of algorithms and the choice is independent of the input. If **u** is 1, the path to transition SA1 is active. If **u** is 2, the lower path to transition SA2 is active. The SA1 and SA2 transitions modify the data variable by appending a "1" or a "2" to the data, indicating which path was taken. The value of the decision variable, **u**, is toggled each pass through the stage by the self loop. Since only one path through the stage is active, the place labeled "z" has only one output tuple.

In the information fusion stage the input z from the situation assessment stage is merged with another input z' to produce the output z'' to the task processing stage. The model for this stage is shown in figure 10. The input tuple is at the place labeled "z", the output from the previous stage. If the role variable, **r**, is *oc* or *i*, the function does not require additional input at this stage and the place labeled "z'" will be empty. In this case, since **r** does not equal *ic*, both the D1 place and D2 places hold a copy of the input tuple. However, if the value of **r** is *ic*, a second, external input is required. The input tuple from the place "z" is again put at the D1 place, but the IF transition waits until the tuple from the "z' " is available at the D2 place from the top path of the model. The tuple at the place "z' ", with color set *XINT*



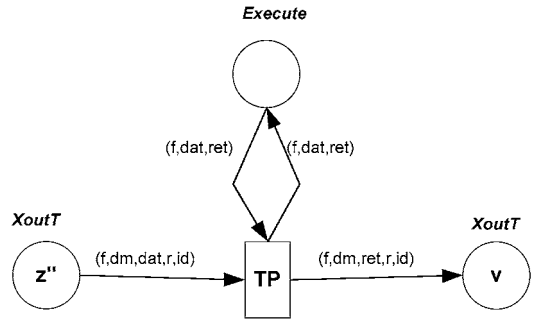

```
color Function = with f1|f2|f3|f4|f5|f6|f7|f8|f9|f10;
var f : Function;
color DecMaker = with DM0|DM1|DM2|DM3|DM4|DM5|DML;
var dm : DecMaker;
color Data = string timed;
var dat,id : Data;
color Role = with s|ic|oc;
var r : Role;
color XINT = product Function*DecMaker*Data*Data;
color XoutT = product Function*DecMaker*Data*Role*Data;
```

Figure 10. Information fusion stage.

has the variables **f, dm**, and **id** as previously described, and the variable **datc** holding the second data input. The input transition inserts the role value, *ic*, so that the tuple is consistent with the color set *XoutT* at place D1. The IF transition then fuses the inputs into a single output tuple at the "z'' " output place with the same color set as the input places, *XoutT*. In order for the IF transition to fire, all the variables, except the data, must be identical. This insures that it is the correct coordinating data for the same function within the same task id process.

The task processing stage is where the algorithm resides for the particular function that the decision maker is performing. There are no additional inputs or outputs at this stage, only the input from the previous stage, at the place "z'' ", and the output to the next stage, at the place "v". Figure 11 shows the task processing stage. At the TP transition, the output data, **ret**, is returned for the function, **f**, and the input data, **dat**. The place with the color set *Execute* must have an initial marking that indicates the output data for each input data of each function in order to implement the task processing algorithm.

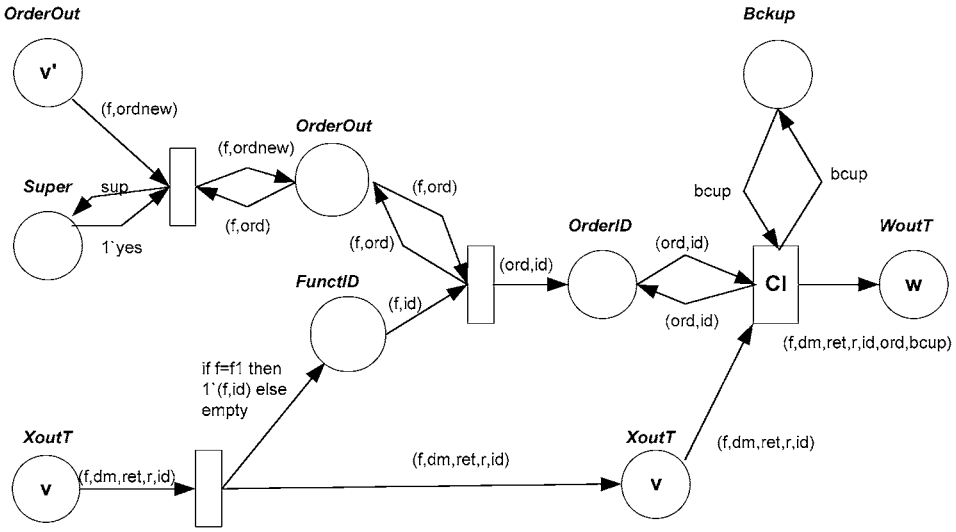
In the command interpretation stage, a decision maker receives external control information on which his final output is based. If the global adaptation strategy is in effect, the v' input carries the process switch information. The new process information is synchronized to the start of the next process. If the local adaptation strategy is in effect, the v' input carries backup status information. When new control information is entered via the v' input, the old control information is over written. If no adaptation is allowed, the nominal control




```
color Function = with f1|f2|f3|f4|f5|f6|f7|f8|f9|f10;
var f : Function;
color DecMaker = with DM0|DM1|DM2|DM3|DM4|DM5|DML;
var dm : DecMaker;
color Data = string timed;
var dat,id,ret : Data;
color Role = with s|ic|oc;
var r : Role;
color XINT = product Function*DecMaker*Data*Data;
color XoutT = product Function*DecMaker*Data*Role*Data;
color Execute = product Function*Data*Data
```

Figure 11. Task processing stage.

information remains unchanged. Figure 12 shows the command interpretation stage. The input tuple at the place labeled "v′" is the pair (**f, ordnew**), indicating the first function of the type of task to be switched and the new process to implement. When the place with color set *Super* holds a token with value *yes* indicating global adaptation, the new process information enters the model and waits in the place with color set *OrderOut* for the next task of this type to be started. The first transition after the input place labeled "*v*" checks to see if the function in the input tuple is *f1*, indicating the start of a new task process. If the function is *f1*, and if a process switch is indicated for this task type by a new order waiting at the place with *OrderOut*, the new process indicator is associated with this initial function by the task identifier, **id**. This logic serves to synchronize the process switch to the start of the next task. The process indicator, **ord**, and the task identifier, **id**, reside at the place with color set *OrderID* at the CI transition. If the model is not implementing global adaptation, new orders are not entered into the model and the nominal process will be associated with the start of each new task. The nominal process values are the initial marking of the place *OrderOut*. The backup status, *loc* or *glob*, indicating whether local backup is allowed, is available at the *Backup* place of the CI transition. The CI transition now produces the output tuple at the place labeled "w" which includes the process indicator and the local backup status. The output tuple now includes the following variables as indicated by the color set *WoutT*: function (**f**), decision maker (**dm**), output data (**ret**), role (**r**), task identifier (**id**), process (**ord**), and backup (**bcup**).




```
color Function = with f1|f2|f3|f4|f5|f6|f7|f8|f9|f10;
var f : Function;
color DecMaker = with DM0|DM1|DM2|DM3|DM4|DM5|DML;
var dm : DecMaker;
color Data = string timed;
var dat,id,ret : Data;
color Role = with s|ic|oc;
var r : Role;
color XoutT = product Function*DecMaker*Data*Role*Data;
color FunctID = product Function*Data;
color Order = with p0|p1;
var ordnew,ord : Order;
color OrderOut = product Function*Order;
color OrderID = product Order*Data;
color Super = with yes|no;
var sup : Super;
color Bckup = with loc|glob;
var bcup : Bckup;
color WoutT = product Function*DecMaker*Data*Role*Data*Order*Bckup;
```

Figure 12. Command interpretation.

In the response selection stage the decision maker produces his output by interpreting the data and control information from the input tuple at the place labeled "w". Figure 13 shows the response selection stage. The first transition RS1 returns the next function to be implemented in the task process by using the current function, **f**, and the process indicator, **ord**, to retrieve the variable, **fnxt**, which is the next function in the indicated process. The place with the color set *NextFunction* must have an initial marking that contains the next function information based on the possible processes, i.e., the process rulebase. The tuple at the place with the color set *WoutT2* now includes both the completed function, **f**, and the next function, **fnxt**. The process information is no longer needed and has been dropped from

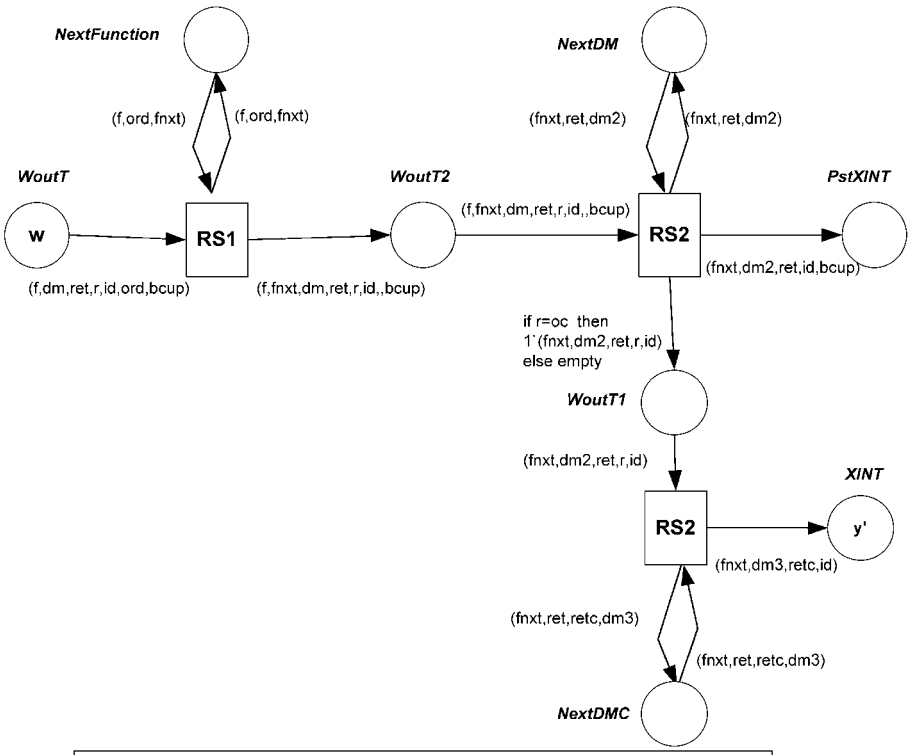
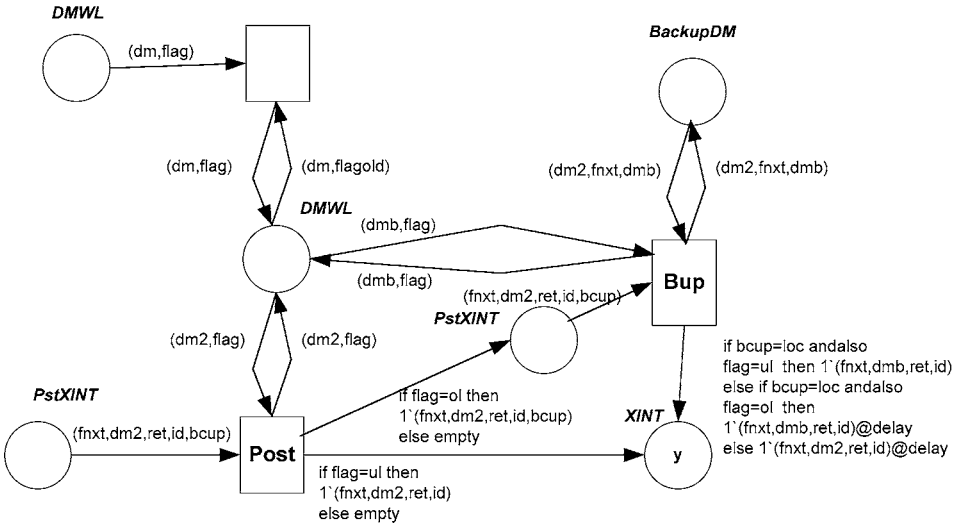


Figure 13. Response selection stage.

the tuple. The next transition RS2 uses the **fnxt** and **ret** variables to retrieve the primary decision maker, **dm2**, to receive the next function. The place with the color set *NextDM* must have an initial marking that indicates the primary decision maker for each function. The output of this transition is in the place with the color set *PstXINT*. The tuple now contains




```
color Function = with f1|f2|f3|f4|f5|f6|f7|f8|f9|f10;
var f, fnxt: Function;
color DecMaker = with DM0|DM1|DM2|DM3|DM4|DM5|DML;
var dm,dm2,dm3,dmb : DecMaker;
color Data = string timed;
var dat,datc,id,ret,retc : Data;
color XINT = product Function*DecMaker*Data*Data;
color Bckup = with loc|glob;
var bcup : Bckup;
color PstXINT = product Function*DecMaker*Data*Data*Bckup;
color OLFLag = with ol|ul timed;
var flag, flagold : OLFlag;
color DMWL = product DecMaker *OLFlag;
color BackupDM = product DecMaker*Function*DecMaker;
```

Figure 14. Post stage.

only the variables next function (**fnxt**), next decision maker (**dm2**), output data (**ret**), task identifier (**id**), and the backup status (**bcup**). If the role is *oc*, indicated by the **r** variable, a second RS2 transition returns the coordinating decision maker, **dm3**, to also receive this function. This becomes the output at the place labeled "y′". Note that the output data in this case is appended with a "*c*" to indicate it is the coordinating data.

After the RS2 transition a valid *y* output is available, however, the overloaded status of the primary decision maker has not yet been checked. A post stage is used to check the workload status of the primary decision maker and, if the **bcup** variable is *loc*, to assign a secondary decision maker to the function if he is overloaded. The post stage is shown in figure 14. The workload status of the decision makers is maintained and updated at the *DMWL* place. The Post transition returns the flag variable for the decision maker. If the flag of the primary decision maker indicates he is underloaded, *ul*, the stage is finished and the output tuple is put at the place labeled "y". If the flag indicates he is overloaded, *ol*, then an alternative decision maker is chosen using the Bup transition. The place with the color set

BackupDm must have an initial marking with the secondary decision maker information for each function. If the value of **bcup** is *loc*, then the function is sent to the secondary decision maker and the stage is complete. However, if the primary decision maker is overloaded, but the value of **bcup** is *glob*, then the tuple at the "*y*" output remains with the primary decision maker in the decision maker (**dm2**) variable. When the function is sent to a decision maker, whether it is the primary or the secondary, and that decision maker is overloaded, a delay is incurred which represents the degradation in performance due to overloaded decision makers. The workload status check and secondary decision maker selection occurs in a post stage because the workload status of the individual decision makers is updated through the input place with the *DMWL* color set.

The model of the five stage decision maker model is in its most general form, meaning that any of the roles or interactions mentioned can be implemented. This model extends the five stage decision maker model by including the ability to perform a set of functions, not just one, and allowing adaptation through any of the three strategies: none, local, or global. In some stages the structure of the Petri net model depends on the role the decision maker assumes, e.g. situation assessment, in other stages the structure depends on the adaptation strategy, e.g. command interpretation. In simpler cases, where some roles or interactions are not required the model could be simplified. For example, if only the independent role were needed the five stage model could reduce the two stage model of Wohl (1979).

Likewise, in its most general form all three adaptation strategies can be modeled. If for example, the global adaptation strategy was not implemented, the command interpretation stage could be merged with the response selection stage for this model. The variables used within the model allow for a wide variety of tasks and functions. If only one type of task was included or if each decision maker only performed one specific function, the variable set could be simplified. Because the Petri net model of the decision maker represents the object class, it has to allow all possible roles, strategies, and variables that any one instantiation of a decision maker could be; narrowing any of these aspects would simplify and reduce the complexity of the model.

4. Decision Maker Workload

One of the main causes of error in an organization's output, and therefore a main driver of organizational change, is overloaded decision makers. Decision makers can only handle a finite amount of activity in a given time period. If this amount of activity increases or the amount of time available to do the required activity decreases, the decision maker reaches a point where he can no longer accomplish his functions in the given time. This, in turn, manifests itself in the form of delayed output and/or incomplete or incorrect output. At an individual level, this may be sensed immediately by the subsequent decision maker in the task process. At the organizational level, the delay and/or error propagates through the process and effects the organizational output. When the organizational output no longer meets the requirements, it is a signal that the current process is no longer appropriate and an organizational change should be made.

In order to provide a consistent model of the decision maker throughout the study of variable structure and adaptive organizations, the five stage interacting decision maker

model of Levis (1992) was chosen as the basis of the decision maker model; this model has a history of development and validation (Louvert et al., 1988; Jin and Levis, 1992). It was also proposed to use decision maker workload to indicate the overloaded status of the individual decision makers; it is the cognitive limitation of the decision makers that manifests itself as a bound on the organizational performance. While a previous study of local adaptation limited the total number of functions a decision maker could accept in order to avoid overloading (Perdu and Levis, 1998), using total activity as an indicator of cognitive workload provides a better indicator of the decision maker's status based on his functional abilities and the adaptation strategy. This model allows the active algorithms in each stage to be defined and the total activity of the decision maker, a surrogate for decision maker workload to be calculated and incorporated in the model in a consistent manner.

Previous research has restricted each decision maker to one function in order to calculate the workload (Andreadakis, 1988), this research extends the workload calculation to include decision makers performing multiple functions; it also extends the use of the partitioning matrix to include inputs from the environment and from other decision makers. The calculated value represents differences due to both adaptation strategy and the number and complexity of functions.

4.1. Total Activity as a Workload Surrogate

An organization completes a task by using a specified process to produce the appropriate output in response to the task input; the process consists of a series of functions assigned to individual decision makers. Within this organizational context each decision maker acts as an information processing system. He receives signals or messages that contain information relevant to the functions he has to perform in the processing of the organizational task. He performs his requested function by interpreting his input information and produces output information. Therefore, it is postulated that, an information theoretic model of the decision maker can be used as a means to monitor the workload incurred by a decision maker as he performs different functions within the organizational task process (Boettcher and Levis, 1982).

A primary quantity of interest in information theory is entropy: given a variable x, which is an element of the alphabet $\mathbf{X}$, and occurs with probability $p(x)$, the entropy of x, $H(x)$, is defined to be

$$H(x) \equiv -\sum_{x} p(x)\ \log p(x).$$

Entropy is measured in bits per symbol when the base of the logarithm is two, as this is the number of binary digits needed to represent the value. Entropy represents the uncertainty associated with the variable: if one value is much more likely than the others, the entropy value will be low, however, if all values are equally likely, the entropy value will be at a maximum. The entropy, $H(x)$, is associated with each occurrence of x. As long as the probability distribution of x remains the same, each occurrence of x has the same entropy value. Total activity of a system, G, can be defined as the sum of the entropy of all the

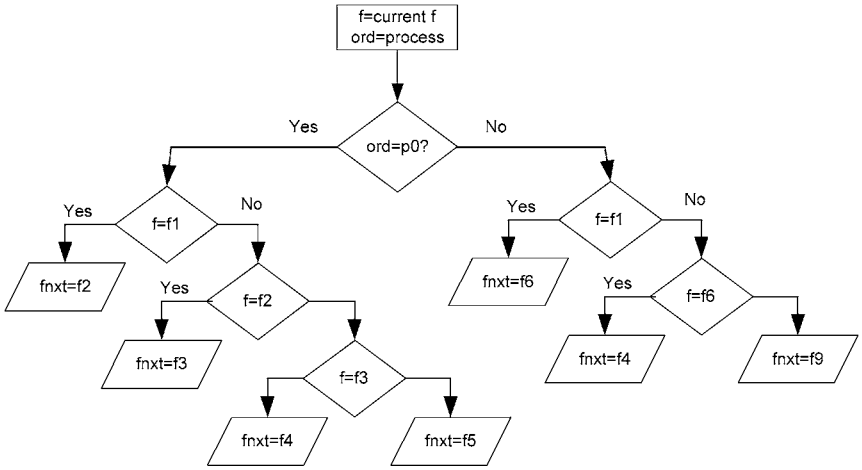


Figure 15. The next function algorithm of the response selection stage.

variables, w_i, in the system:

$$G = \Sigma_i H(w_i).$$

The total activity of the decision maker, G, has been used as a surrogate measure for the workload of the decision maker performing a function, which can not be measured directly (Levis, 1992). Several steps are used to compute the total activity for each decision maker performing a function. First the algorithms present in each stage of the five stage model are defined in order to identify all the variables in the model. For each function, the algorithms active in each stage of the model must be identified and probabilities associated with the variables. The algorithms are also dependent on which adaptation strategy is being implemented. Then the total activity for a decision maker completing that function under a specified adaptation strategy can be calculated by summing the entropy of each stage; the total activity of the decision maker is equal to the sum of the total activities of each stage:

$$G = \sum_{i=1}^{N} G^i$$

where N is the number of stages (Conant, 1976).

The total activity due to the adaptation strategy is represented by the algorithms in the response selection and post stages. In the response selection stage the next function for a given process is found. When an organization is utilizing a global adaptation strategy, the algorithm must include all the alternative processes, as shown in figure 15. If the organization is using none or local backup, this algorithm potentially contains fewer variables as it only maintains one process; thus the entropy calculation for this stage includes the overhead for the global adaptation strategy. When an organization implements the global adaptation

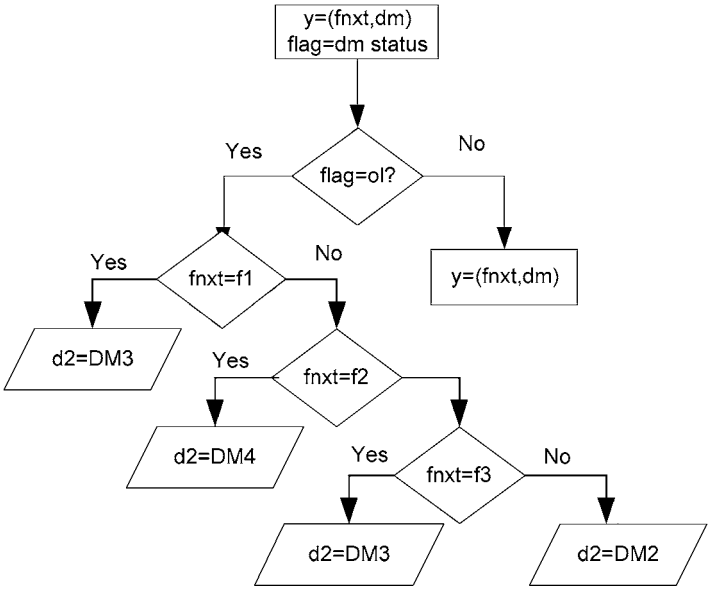


Figure 16. The backup decision maker algorithm of the post stage.

strategy, the decision makers' response selection stage includes additional entropy due to maintaining information on all possible processes.

If the adaptation strategy is none or global, a secondary decision maker is not chosen, regardless of the status of the primary decision maker. In this case the output produced at the end of the response selection stage is the final output. However, if the strategy is local adaptation, a secondary decision maker is chosen in the post stage if the primary decision maker is overloaded; this algorithm is shown in figure 16. The entropy calculation for this stage includes the overhead for the local adaptation strategy. When an organization implements the local adaptation strategy, the decision makers' post stage includes additional entropy due to maintaining information on backup decision makers.

Total activity based on the internal algorithms of the decision maker model has been previously used as the workload surrogate for a decision maker performing a single, repetitive function (Louvet et. al., 1988; Andreadakis and Levis, 1988; and Jin and Levis, 1992). However, in an organization, a decision maker is often assigned to more than one type of function. He may be assigned a function early in the process and then another function later on. The set of functions an organization must perform within a process can be represented by a vector $\mathbf{x}$ which takes values from an alphabet X, the set of organizational functions. Each component of the vector represents one of the functions, i.e.

$$\mathbf{x} \equiv (f_1, f_2, \ldots, f_N) \quad \mathbf{x} \in X.$$

This vector can then be partitioned into groups of functions that are assigned to different decision makers through partitioning matrices. The partitioning matrix for the ith decision

Table 2. Methodology to compute decision maker total activity.

Step 1:	Define task processing algorithm for each function.
Step 2:	Compute total activity for each function: – Include additional entropy due to role, – Consider inputs to function independent for input entropy calculation.
Step 3:	Determine each decision maker's function input vector $\mathbf{x}^i$: – Enumerate each decision maker's function based on the process(es), – Consider each adaptation strategy.
Step 4:	Compute each decision maker's total activity: – Determine the frequency of each function in the set of processes, – Add an additional decision variable for each additional function.
Step 5:	Compute the additional entropy due to adaptation strategy.
Step 6:	Compute each decision maker's final total activity.

maker is denoted Π^i, and results in the partition x^i:

$$\mathbf{x}^i = \Pi^i \mathbf{x}.$$

The set of partitioning matrices $\{\Pi^1, \Pi^2, \ldots, \Pi^i\}$ has been used to specify the inputs received by each decision maker from the organizational environment by Stabile and Levis (1984) and is now modified to represent the partition of the functions of a process to the individual decision makers. The resulting vectors, x^i, may have some, all, or no components in common. If there are duplicate components, this indicates that more than one decision maker is qualified to complete that function. This redundancy can be used later to determine the primary and secondary decision maker for the function necessary for the local adaptation strategy. Each decision maker is assigned a set of specific functions to perform based on the x^i partitions. The number and type of functions performed by the organization are determined by the process definition. The number of occurrences of each function within the set of processes the organization is performing can be used to assign probabilities to the input vector of each decision maker.

When a decision maker receives a function, he first determines which function it is. Because no learning takes place during the performance of a sequence of functions, the successive values taken by the variables of the model are uncorrelated, i.e., the model is memoryless (Boettcher and Levis, 1982). The value of an arriving input is not known to the decision maker; rather, it is known only that an input is present. When a decision maker is assigned to more than one type of function, his task processing stage must include the alternative algorithms. Intuitively, a decision maker is assigned to functions based on his skills; these skills are represented as the set of function algorithms in his task processing stage.

The final total activity for a decision maker performing a function is determined based on his set of assigned functions x^i and the adaptation strategy. Table 2 summarizes the steps to calculate each decision maker's total activity. The first step is to define the task processing algorithm for each function. The second step calculates the total activity for each function by calculating the entropy of the each stage's algorithm required by the role and summing

the entropy of the stages. The inputs to the functions are considered independent for this calculation, unless there is reason not to. At the end of this step, there is a total activity value for each function.

Step three considers the set of functions being performed by each decision maker. This set changes, based on what process(es) are being considered and what adaptation strategy is being implemented. For example, different processes contain different sets of functions. The set of functions should include only the functions required by the processes being considered for that organizational configuration. Likewise local adaptation requires backup functions which may not be included in global adaptation. The frequency of each function within each decision maker's set of functions can be used to calculate the input entropy associated with the choice of task processing algorithm to use in the decision maker's task processing stage; likewise, for each function beyond one, additional entropy must be included to determine which task processing algorithm to use.

Step five computes the additional entropy due to the existence of the adaptation strategy. The next function algorithm in the response selection stage and the backup decision maker algorithm in the post stage both depend on the adaptation strategy. The entropy associated with both of these algorithms must be calculated under the different strategies. Finally each decision maker's total activity can be calculated by using the total activity value from step four and adding the appropriate additional entropy based on the adaptation strategy from step five. This total activity value can now be used as an estimate of the workload a decision maker experiences each time he is asked to perform one of his assigned functions.

Now, assume the decision maker is asked to do two functions from his set of functions. His total activity increases. If he consistently uses the same algorithmic structure, that is his internal variables are the same, the total activity for two functions, G_2, is (Louvet et. al., 1988):

$$G_2 = H(x') + \sum_{i=1}^{k} H(w_i) + \sum_{i=k+1}^{2k+1} H(w_i) + H(y')$$

where x' and y' represent the dual input and dual output respectively. If the probability distributions for the input and output variables composing the input and output pair are the same as in the single task case then

$$G_2 = 2H(x) + 2\sum_{i=1}^{k} H(w_i) + 2H(y) \leq 2G_1$$

where G_1 is the total activity of the single task case. This indicates that the total activity value calculated for a decision maker performing a single function from his set of functions can be doubled and used to represent the total activity he experiences when he performs two functions from his set of functions. This property is the independence bound on entropy which states that

$$H(X_1, X_2, \ldots, X_n) \leq \sum_{i=1}^{n} H(X_i)$$

with equality if and only if the X_i are independent (Cover and Thomas, 1991). This can be used to monitor the workload of the decision maker: each time the decision maker receives a function to perform, his current workload is increased by his total activity value. Each time he completes a function, his workload is decreased by this amount. Thus the current workload of the decision maker depends on the number of functions he performs simultaneously. As long as the probability distribution of the input variable remains constant, the total activity of the aggregate functions a decision maker is responsible for can be used as a workload monitor for the decision maker.

4.2. The Effect of Adaptation Strategy on Total Activity

The majority of the total activity value comes from three stages, the Task Processing stage, which holds the algorithms for all the functions the decision maker is responsible for, and the Response Selection and Post stages, where the next function and next decision maker in the process are chosen. This dependency can be shown as

$$G_{\mathrm{TOT}} \propto (G_{\mathrm{TP}}, G_{\mathrm{RS}}, G_{\mathrm{P}})$$

where G_{TOT} is the total activity of the decision maker, G_{TP} is the total activity of the Task Processing Stage, G_{RS} is the total activity of the Response Selection stage, and G_{P} is the total activity of the Post stage. G_{TP} is dependent on the number of functions a decision maker is responsible for, n_{f}, assuming they have a comparable number of internal decision variables:

$$G_{\mathrm{TP}} = f(n_f).$$

G_{RS} and G_{P} are dependent on the adaptation strategy. First consider the case of no adaptation strategy. G_{RS} contains the algorithm to select the next function for a single process P0; the total activity associated with this algorithm depends on the internal variables of the P0 decision tree, w_{P0}:

$$G_{\mathrm{RS-NONE}} = f(w_{\mathrm{P0}}).$$

Since no backup decision maker information is necessary for this adaptation strategy, the total activity of the Post stage is zero:

$$G_{\mathrm{P-NONE}} = 0.$$

Therefore the total activity for a decision maker under the none adaptation strategy can be represented as

$$G_{\mathrm{TOT-NONE}} = f(n_f, w_{\mathrm{P0}}, 0).$$

The local adaptation strategy increases the total entropy of the decision maker in two ways. First, it may increase the number of functions a decision maker is responsible for by adding

additional functions due to backup responsibilities, n_{Bupf}, which increases G_{TP}:

$$G_{\text{TP–LOCAL}} = f(n_f, n_{\text{Bupf}}).$$

These are backup functions that the decision maker is responsible for that were not previously included in his partition. Secondly, it adds the backup decision maker algorithm w_{Bup} to the Post stage, increasing G_{P}:

$$G_{\text{P–LOCAL}} = f(w_{\text{Bup}}).$$

The activity of the Response Selection stage is unchanged, as local adaptation only supports one process. Therefore the total activity for a decision maker under the local adaptation strategy can be represented as

$$G_{\text{TOT–LOCAL}} = f(n_f, n_{\text{Bupf}}, w_{\text{P0}}, w_{\text{Bup}}).$$

The global adaptation strategy also increases the total entropy in two ways. First, it also may increase the number of functions a decision maker is responsible for by adding additional functions, n_{P1f}, which increases G_{TP}:

$$G_{\text{TP–GLOBAL}} = f(n_f + n_{\text{P1f}}).$$

These are additional functions due to the second process that the decision maker is now responsible for that were not previously included in his partition. Secondly, it adds a second next function algorithm to the Response Selection stage, w_{P1}, representing the alternative process, increasing G_{RS}:

$$G_{\text{RS–GLOBAL}} = f(w_{\text{P0}} + w_{\text{P1}}).$$

Similar to the none adaptation strategy, global adaptation does not support decision maker backup, therefore the activity of the Post stage remains at zero. Therefore the total activity for a decision maker under the global adaptation strategy can be represented as

$$G_{\text{TOTAL–GLOBAL}} = f(n_f + n_{\text{P1f}}, w_{\text{P0}} + w_{\text{P1}}, 0).$$

Both adaptation strategies increase the decision maker's total activity value greater than the none strategy. They both may increase the number of functions a decision maker is responsible for which increases the total activity of the Task Processing stage. The principal difference is in the Response Selection and Post stages: local adaptation adds an additional algorithm to the Post stage, while global adaptation adds an additional algorithm to the Response Selection stage. Local and global adaptation are orthogonal; which strategy has a greater effect depends on the specific organization and its task graphs. It cannot be determined by inspection, an executable model is necessary to determine the more effective strategy.

4.3. Monitoring Decision Maker Workload and Organizational Peformance

The total activity value for each decision maker represents the workload he incurs each time he receives any one of the functions from his partition; one total activity value is associated with each decision maker for each organizational design, i.e. function allocation and adaptation strategy. If the adaptation strategy is changed or the function partition is changed, the total activity value must be recalculated. The independence bound property of entropy was used to show that if the decision maker receives two functions at once, from his partition of functions, his total activity value doubles. This can be generalized to multiple occurrences of simultaneous functions: each time the decision maker receives a function, his current workload is increased by the total activity value and each time he completes a function his total workload is decreased by the total activity value. The current total activity of the decision maker, G_{CUR} can be expressed as

$$G_{\mathrm{CUR}} = \sum_{1}^{n} G = G * n$$

This is possible because the single total activity value for each decision maker includes the uncertainty associated with the *set* of functions partitioned to the decision maker. This assumes that there is no interactions between functions, that is they are independent; the decision maker has no knowledge of what his next function will be.

However, in order to use total activity as a decision maker monitor, it is necessary to determine the decision maker's workload limit. Perdu and Levis (1998) set a limit on the number of tasks a decision maker could simultaneously process; each decision maker and task were treated identically. However, in this case, each decision maker's total activity is different based on his functional mapping; it can be assumed that a decision maker with a lower total activity can assume more tasks than a decision maker with a higher total activity. By using total activity as a means to monitor the workload status of a decision maker during the course of repeated processes, concepts similar to the bounded rationality constraint can be used to establish a workload limit.

In studies of the cognitive demands of command and control on decision making organizations, it was found that a performance decrease occurs when the task demands exceed some limit (Levis, 1993). The mathematical model that is used to represent this phenomenon is that of the bounded rationality constraint, which states that if the workload rate exceeds some value, rapid degradation of performance occurs. Decision makers normally operate at a level where the bounded rationality constraint is not in effect. The bounded rationality constraint can not be determined analytically but can be established experimentally. Consider the situation in which a decision maker is performing a sequence of functions with interarrival time τ. The workload experienced by the decision maker can be expressed as:

$$G = Ft; t \leq \tau$$

where G is the decision maker's workload or total activity as previously calculated (bits per symbol), F is the decision maker's processing rate (bits per symbol time), and t is the portion of the inter-arrival time during which the decision maker is processing the function.

If τ is more than ample, various tradeoffs between F and t are possible, as G for the function is fixed. However, for sufficiently small values of τ, t will approach τ, and the decision maker must increase F in order to maintain G and thereby avoid degradation of performance. When further increases in F are not possible, at the interarrival time t', the decision maker has reached the bounded rationality constraint, $F_{\max}$ (Levis, 1993).

Since the bounded rationality constraint can not be determined analytically, an adjustable workload limit is used in the model. The workload limit places an upper bound on the amount of tasks a decision maker can accept in one processing interval. If the decision maker receives a set of inputs that exceeds the workload limit, the decision maker's status changes to "overloaded". If new functions are sent to the decision maker while he is in the overloaded status, they are delayed one processing unit, representing the degradation in performance associated with overloaded decision makers. When the decision maker's status returns to "underloaded", functions are once again completed in a single processing unit. The local or decision maker monitor provides the mechanism to monitor the workload of the individual decision makers. The output of the monitor, the decision makers' workload status, is used in the post stage of the decision maker model to determine if a backup decision maker is necessary when the organization is operating under local adaptation, and to indicate when functions should be delayed due to overloaded decision makers. The Colored Petri net model for this function is shown in figure 17.

Timeliness expresses the organization's ability to respond to an incoming task within an allotted time. The allotted time is the time interval over which the output produced by the organization is effective in its environment. This allotted time can be described as a window of opportunity whose parameters are determined a priori by the requirements of the task. Different task types may have different windows of opportunity. Two quantities are needed to specify the window of opportunity: the lower and the upper bounds of the time interval, t_s and t_f, respectively, or one of the bounds and the length of the interval, e.g. t_s and Δt (Cothier and Levis, 1986). In the situation where repetitive tasks will be processed, where the start time of each task will be dependent on the scenario driving the model, the window of opportunity for each task type is defined a priori with the second method with t_s set to zero. The organization's response time is represented by the first method: the lower bound is the initial time of the task as received by the organization and the upper bound is the finish time of the task. The organization's response time is the time delay between the moment when the organization receives a task and the moment it produces an output.

In order to evaluate if the organization's output is timely, the organization's response time is compared to the window of opportunity of the task. The initial time of the task, t_s, is sent to the organization monitor function. When the last function of the process is complete and the final output is made, the final time of the task t_f is also sent to the organizational monitor. The comparison of the difference between these two times is compared to the window of opportunity by the function $d((0, \Delta t), (t_s, t_f))$ which represents the timeliness of the output. A binary measure can be assigned to each task to represent the organization's score, **s**, for this task based on timeliness. If the response time is less than or equal to the window of opportunity, the organization receives a score of one, indicating a timely response. If the output is outside this window, indicating the response took too long, the

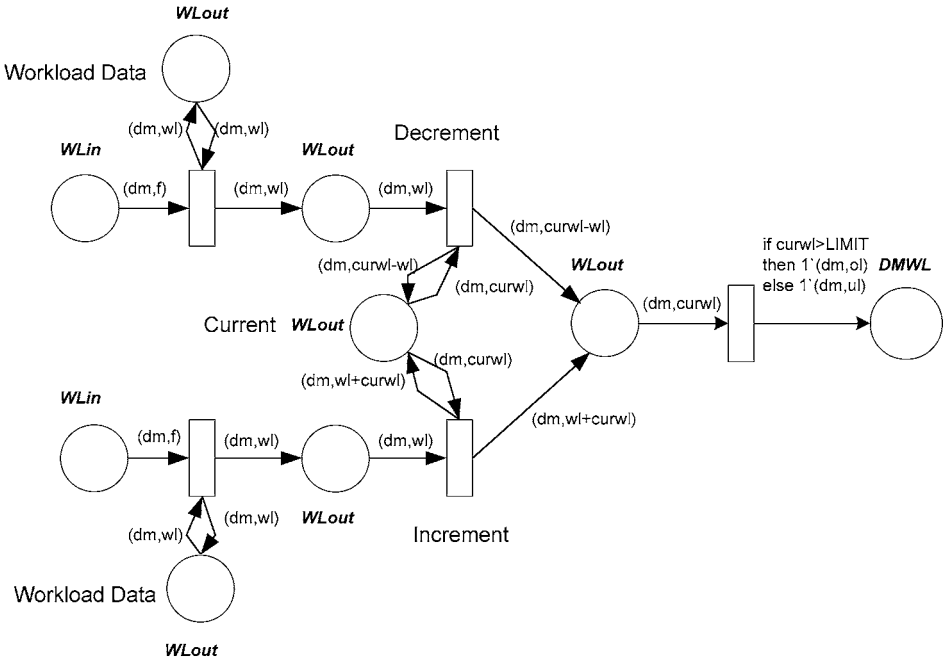


Figure 17. Colored Petri net model of the decision maker monitor.

organization receives a score of zero.

$$\mathbf{s} = d((0, \Delta t), (t_s, t_f)) = \begin{cases} 1 : t_f - t_s \leq \Delta t \\ 0 : t_f - t_s > \Delta t \end{cases}$$

The score can be used as both a performance measure for the organizational output and as a trigger of adaptation for process selection. For example, if a task receives a score of zero, it is a signal that the current process is not performing adequately. The organization should switch processes for that task type in order to correct the untimely response. The organizational monitor provides task feedback to use to determine the process for each task type when the organization is in a global adaptation mode, and provides the score for

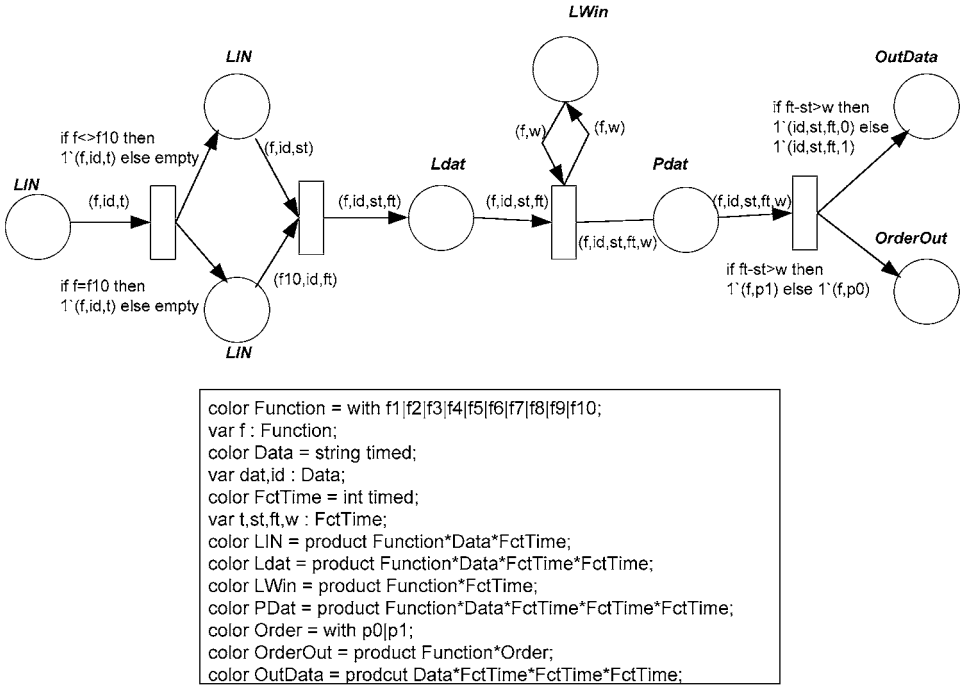


Figure 18. Colored Petri net model of the organization monitor.

each task processed based on the timeliness of the organization's response, regardless of the mode. It receives the first and last time stamps of the process; the start time is when the task is received by the system and the finish time is when the last function is completed. It subtracts these two times and compares the difference to the task window of opportunity. It must then assign a score to the task, and if the organization is implementing a global adaptation strategy, indicate a process switch, if necessary. The Colored Petri net model and declaration node are shown in figure 18.

5. Virtual Experiment

A virtual experiment was conducted by configuring the organizational model to actual experimental organizations and evaluating the effectiveness of the alternative strategies. In order to configure the model, three inputs are required: the task graph, the organizational design, and the experimental design. The task graphs and the organizational designs were obtained from a recent subject experiment studying organizational adaptation, while the experimental design was created specifically to evaluate the different adaptation strategies. The organizational model was configured into two different organizational designs and executed under the three different adaptation mechanisms: no adaptation allowed, local adaptation using alternative decision makers, and global adaptation using alternative

process selection. The resulting performance of the different organizations under different adaptation strategies was then compared.

5.1. Experimental Design

The Adaptive Architectures for Command and Control (A2C2) is a research initiative to investigate the process of adaptation in Joint Command and Control architectures at the Commander Joint Task Force Level. Organizations are designed to carry out a predefined mission under different stress levels: when full resources are available to complete the mission the organization experiences low stress; when the number of resources is reduced, the organization experiences higher stress as it strives to complete the same mission with less resources. The organizations that are designed are modeled and executed in a virtual environment: scenarios are created that represent the mission and threats that the organization will experience. The organizations are then used in subject experiments where humans respond to a simulated environment through a computer simulation.

The organizations designed for the A2C2 second experiment had a sequence of tasks to perform based on an overall mission: to secure the port and airfield of a friendly nation invaded by a hostile neighbor. The sequence of tasks was represented by a task graph that included beach landings, clearing roads, and coordinating the storming of the port and airfield. It also included defensive tasks, i.e., responding to threats launched by the enemy. Each of these defensive tasks can be represented by its own task graph: the series of functions that must be performed in order to respond to the threat. Two specific threat tasks that occurred during the "clearing roads" portion of the mission were identified for the virtual experiment: Frog launchers (FROG) and Silkworm missiles (SILK). Both of these threats had a window of opportunity for the organization's response: if the response was made within a certain time period the enemy would withdraw the threat and retreat back into their bunkers. Both a nominal and alternative process was defined for both tasks, resulting in a total of nine different functions as shown in Table 3.

Table 3. Organizational functions and characteristics.

Function	Description	Role required	Function type
f1	Threat notification	Independent	Non choice
f2	Choose platform	Independent	Choice
f3	Contact launch	Independent	Non choice
f4	Launch platform	Independent	Non choice
f5	Confirm launch	Independent	Non choice
f6	Coordinate attack	Output coordinating	Non choice
f7	Choose verification	Independent	Choice
f8	Threat confirmation	Independent	Non choice
f9	Confirm coord launch	Input coordinating	Non choice

Two distinct organizational structures, termed "Traditional" (TA) and "Non Traditional" (NTA) were designed for the subject experiment; each had six decision makers. In the A2C2 program, decision makers are defined by the resources they are assigned rather than the skills they possess; however, the decision maker can be assumed to have the skills necessary to operate the resources. The TA organization allocated resources to decision makers in a geographic manner, while the NTA organization allocated resources in order to balance the decision maker workload. The total activity value to use for each decision maker, under two different process and the three different adaptation strategies, was calculated for the two different organizational designs as shown in Table 4. Note that the number of decision makers is the same in both architectures and the functions to be performed are identical, however the distribution of functions to decision makers is different.

In order to stimulate the organizational models under different conditions, scenarios of input tasks were generated to provide a series of threat inputs to the organization. The organization was stimulated with two sets of scenarios: one set varies the interarrival time of the tasks, it consist of three series of 32 tasks, the second set varies the number of tasks arriving at one time interval, it consists of three series of 32, 40, and 48 tasks. The nominal process was selected for the initial configuration. An organization adapting through local change will maintain this process, and will select secondary decision makers for overloaded decision makers. An organization using strategic adaptation will continue to use the primary decision makers but will be allowed to implement alternative processes at the start of a new task. In order to cause the organization to change, it must be stressed; it can be stressed in two ways: increasing the frequency of arriving tasks or increasing the number of tasks arriving per time. In the case of local change, adaptation should occur as individual decision makers become overloaded. In the case of global change, adaptation should not occur until the organizational output deteriorates. The level of tolerance of the organization can be varied through the value of the workload limit. An organization with a low level of tolerance will change at a minimal level of stress while an organization with a high level of tolerance will not change until a high level of stress is reached.

The independent variables of the experiment are those controlled by the experimental setup: the organizational configuration, the adaptation strategy, and the level of stress tolerated by the decision makers varied through the value of the workload limit. The dependent variable, or the variable to be observed and measured, is the score of each task based on the

Table 4. Decision maker total activity (bits per symbol).

Organization	Traditional			Non traditional		
Adaptation	None	Local	Global	None	Local	Global
DM0	20.6	28.4	35.1	34.7	43.7	49.2
DM1	50.0	57.8	71.7	42.1	51.1	56.6
DM2	29.8	37.6	44.3	28.0	37.0	42.5
DM3	50.0	57.8	71.7	37.8	47.6	52.3
DM4	28.6	40.8	63.9	34.2	43.2	48.7
DM5	32.9	40.8	63.9	34.2	43.2	48.7

timeliness of the organizational response. For each scenario of tasks used to simulate the organization under a specific strategy, the percentage of tasks with a score of one, or the timeliness percentage S, can be calculated:

$$S_{\text{Scenario}} = n_{s=1}/n$$

where n is the number of tasks in the scenario and $n_{s=1}$ is the number of tasks that received a score of one. This represents the percentage of tasks that the organization completed in a timely manner. The experimental sample space is shown in Table 5.

For both organizations, the model was first simulated with no adaptation strategy. In this mode, each time the next function in a process is assigned to an overloaded decision maker, the function incurs a processing delay of one time unit, thereby affecting the timeliness of the completion of the process. The next strategy implemented was the local adaptation strategy. In this mode, each time the next function in a process is assigned to an overloaded decision maker, an alternative decision maker is chosen. A processing delay of one time unit is incurred only if the alternative decision maker chosen is overloaded as well. In global adaptation, if the next decision maker chosen is overloaded, the function occurs a processing delay of one time unit, the same as with no adaptation. However, if the task receives a score of zero upon completion of the process, indicating it was not completed in a timely manner, the next task of this same type will be implemented with the alternative process.

Because the workload limit, the total activity a decision maker can sustain without performance degradation, can not be calculated analytically, three different values of the workload limit were used in the experiment. The high tolerance level was determined by taking the highest decision maker total activity value and doubling it. The low tolerance level was determined by taking the lowest decision maker total activity value and doubling it, the medium level is the median between the two. For this experiment the tolerance level values, the workload limit, were set as follows: low = 50, medium = 100, and high = 150, all in bits per symbol. The workload limit determines the number of tasks each decision maker can accept in a processing interval without being overloaded.

Table 5. Experimental sample space.

Organization	Traditional			Non traditional		
Adaptation	None	Local	Global	None	Local	Global
Tolerance	L|M|H	L|M|H	L|M|H	L|M|H	L|M|H	L|M|H
Task interarrival:						
Low						
Medium						
High						
Simultaneous tasks:						
Low						
Medium						
High						

5.2. *Experimental Results*

The experimental results indicate that under some conditions, there are no significant differences between the adaptation strategies. For the conditions where the differences in performance were significant, the strategy with the higher performance can be identified. The results are summarized in figure 19. When an organization does not experience stress, the organization with no adaptation strategy performs as well or better under all three workload tolerance levels. In this case the decision makers do not have the additional total activity associated with an adaptation strategy and therefore are less likely to become overloaded. However, as the stress on an organization is increased, at low levels of workload tolerance, the local adaptation strategy performs as well or better under medium stressed conditions and out performs an organization with no adaptation strategy under the most stressed conditions. As the workload limit is increased, increasing the tolerance level, the local adaptation strategy performs as well as no adaptation strategy under low and medium stressed conditions, but the local and global adaptation strategies perform the best under the most stressed condition. Finally when the workload limit is at its maximum, all three strategies perform usually as well under low stress conditions, however as the stress is increased, the global adaptation strategy performs the best.

The numbers in parenthesis under the adaptation strategy in figure 19 represent the performance measure range, the timeliness percentage values for the region. While the None strategy outperforms the other strategies in the low tolerance, high stress conditions, its performance is still quite low as indicated by the range 0.2 to 0.4 of the tasks completed in a timely manner. In comparison, in the high tolerance, high stress condition under the Global adaptation strategy, 0.7 to 0.8 of the tasks were completed in a timely manner.

A second analysis compared the performance of the two different organizations, as shown in Table 6. Based on the value of the timeliness percentage performance measure, the Non

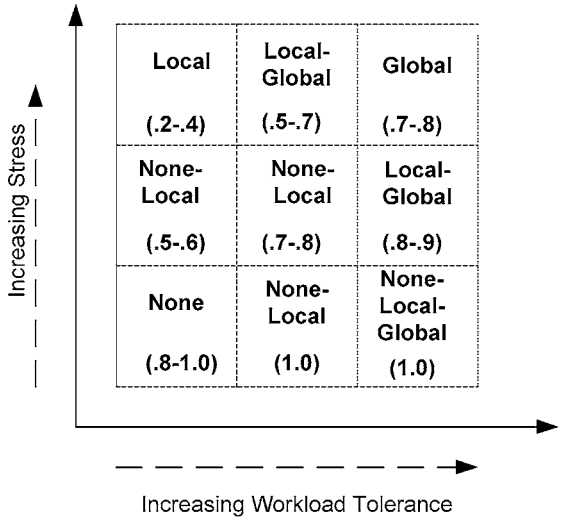


Figure 19. Experimental results.

Table 6. Organizational structure with best performance.

Tolerance	Low			Medium			High		
Strategy	Global	Local	None	Global	Local	None	Global	Local	None
Low stress	NTA	NTA							
Medium stress	NTA	NTA	TA	NTA			TA		
High stress	NTA			NTA	NTA	TA	NTA	NTA	NTA

Traditional organization usually performed better or the same as the Traditional organization, where significant differences exist. This reflects the balance workload among the decision makers in the Non Traditional organizational design, indicated by the total activity values of the individual decision makers. The balanced workload implies that the decision makers are all operating under similar stress conditions. In the Traditional organization, two decision makers have a very high total activity value, while one has a very low total activity value. This imbalance causes the performance of some decision makers to degrade earlier than the others, affecting the performance of the entire organization; this degradation happens earlier in the Traditional organization than in the Non Traditional organization.

Models of organizations can be used to study organizational behavior under different conditions in order to address questions on adaptation. Pre experimental modeling is an important step in the experimental design process of subject experiments (Handley et al., 1997, 1998). By constructing a software model the experimental situation, the behavior of the organization under design can be observed before the subject experiment is conducted. Organizational models can also be used to conduct virtual experiments (Carley, 1995). The results from the virtual experiment can then be used to design experimental organizations that behave similarly for further study. Models of adaptive organizations must be able to exhibit changing behavior as adaptation is a dynamic process, organizations that adapt do so in real time, operating and changing concurrently.

This model is appropriate for a stressful, high performance environment in which there is a severe time constraint in generating responses to external events. Organizations, not only military command and control centers, but also air traffic control centers, space mission control centers, and emergency action centers face a variety of missions in a very stressful environment characterized by strong time constraints and uncertainty. No single organizational structure is optimal for all situations and the teams that constitute the organization need to be adaptive to meet the changing situations. While these organizations may have a person in charge, time constraints are usually such that indirect coordination is required between team members to identify the need for change, select a new configuration, and implement it in a timely manner.

6. Conclusions

This paper developed and presented the results of an organizational model used to evaluate the effect on performance of different adaptation strategies, which includes the adaptation

strategy's effect on decision maker workload. An organizational model was developed through an object oriented design approach and represented the top page of a Colored Petri net model. Colored Petri nets were also developed for the decision maker model, the workload monitor, and the organizational monitor. This organizational model can be configured to different organizations implementing various processes and different adaptation strategies; it is a dynamically adaptable, reconfigurable model which can be used to evaluate the performance of organizations implementing different adaptation strategies.

The five stage decision maker model of was used to model the decision makers within the organization. A detailed executable model the five stage decision maker that can dynamically adapt via either local or global adaptation was presented; the role assumed by the decision maker indicates the active structure in each stage. Since the model allows the active algorithms in each stage to be defined, the total activity, a surrogate for decision maker workload, can be calculated. A procedure to compute a decision maker's total activity based on the entropy of the set of functions he is required to perform and the active algorithms in each of his stages is presented; the process used to complete the task and the adaptation strategy implemented by the organization also effects the total activity calculation.

By using entropy to calculate each decision maker's total activity, the impact of different functions, processes and adaptation strategies are reflected in the value used to evaluate overloaded decision makers. As the number of different functions a decision maker is responsible for increases, the number of task processing algorithms he must maintain increases. The total activity value also includes the overhead due to adaptation strategy. Both local and global adaptation increase the number of functions assigned to each decision maker: local adaptation includes backup functions and global adaptation adds functions present in alternative processes. Both strategies affect the final output of the decision maker: global adaptation forces the decision maker to maintain the rulebase of alternative process for use in the next function selection, while local adaptation requires secondary decision maker information for use in the next decision maker selection.

Because different combinations of the three variables function, process, and adaptation strategy may lead to different conclusions as to which strategy is more effective, the dynamically adaptable, reconfigurable model should be used to evaluate the alternative strategies for particular organizational designs. A virtual experiment was conducted using data from recent subject experiments in order to evaluate the different adaptation strategies on two different organizations: one with a balanced workload and one without. The model was configured to each organization and simulated with scenarios that stressed the organizations. The simulation results indicate that the global adaptation strategy was most effective under high stress and high workload tolerance conditions, and the organization with balanced workload generally performed better.

Acknowledgments

This research was supported by the Office of Naval Research under grants no. N00014-93-1-0912 and N00014-94-1-0840.

References

Aldrich, H. (1999), *Organizations Evolving*. Sage Publications, London, UK.

Andreadakis, S. (1988), "Analysis and Synthesis of Decision-Making Organizations," LIDS-TH-1740, Ph.D. Thesis, Laboratory for Information and Decision Systems, MIT, Cambridge, MA.

Boettcher, K.L. and A.H. Levis (1982), "Modeling the Interacting Decisionmaker with Bounded Rationality," *IEEE Transactions on Systems, Man and Cybernetics*, SMC 12(3), 334–344.

Boettcher, K.L. and A.H. Levis (1983), "Decisionmaking Organizations with Acyclical Information Structures," *IEEE Transactions on Systems, Man and Cybernetics*, SMC 13(3), 384–391.

Booch, G., I. Jacobson and J. Rumbaugh (1998), *The Unified Modeling Language User Guide*, The Addison-Wesley Object Technology Series, Addison-Wesley, Boston, MA.

Carley, K.M. (1995), "Automatic Restructuring of the C2 Structure and Performance," in *Proceedings of the First International Symposium on Command and Control Research and Technology*, National Defense University, Washington, DC, pp. 227–237.

Carley, K.M. (1995b), "Computational and Mathematical Organization Theory: Perspective and Directions," *Computational and Mathematical Organization Theory*, 1(1), 39–56.

Carley, K.M. (1998), "Organizational Adaptation," *Annals of Operations Research*, 75, 25–47.

Conant, R.C. (1976), "Laws of Information which Govern Systems," *IEEE Transactions on Systems, Man, and Cybernetics*, SMC 6(4), 240–255.

Cothier, P.H. and A.H. Levis (1986), "Timeliness and Measures of Effectiveness in Command and Control," *IEEE Transactions on Systems, Man, and Cybernetics*, SMC 16(6), 844–853.

Cover, T.M. and J.A. Thomas (1991), *Elements of Information Theory*, Wiley Series in Telecommunications, John Wiley and Sons, New York, NY.

Gomaa, H. (1998), *Course Notes on Software Design: Object Oriented Analysis and Modeling*, Department of Information and Software Systems Engineering, George Mason University, Fairfax, VA.

Handley, H.A.H. (1999), "Effecting and Evaluating Organization Adaptation via Dynamic Process Selection," Ph.D. Dissertation, School of Information Technology and Engineering, George Mason University, Fairfax, VA.

Handley, H.A.H., D.M. Perdu, I. Shin and A.H. Levis (1997), "Architectural Modeling of the A2C2 Second Experiment," Technical Report, GMU/C3I-183-R, C3I Center of Excellence, George Mason University, Fairfax, VA.

Handley, H.A.H., Z.R. Zaidi and A.H. Levis (1998), "Pre-Experimental Modeling of Adaptive Organizational Architectures," in *Proceedings of the 1998 Command and Control Research and Technology Symposium*, Naval Postgraduate School, Monterey, CA, pp. 44–53.

Jensen, K. (1992), *Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use*. Springer-Verlag, Berlin, Germany.

Jin, V.Y. and A.H. Levis (1992), "Impact of Organizational Structure on Team Performance: Experimental Findings," in C.R. Jones (Ed.) *Toward a Science of Command, Control, and Communications*, AIAA Press, Washington, DC.

Lakos, C.A. (1995), "From Coloured Petri Nets to Object Petri Nets," in *Proceedings of the 16th International Conference on the Application and Theory of Petri Nets*, Torino, Italy, pp. 278–297.

Levis, A.H. (1992), "A Colored Petri net Model of Intelligent Nodes," in J.C. Gentina and S.G. Tzafestas (Eds.) *Robotics and Flexible Manufacturing Systems*, Elsevier Science Publishers, The Netherlands.

Levis, A.H. (1993), "Human Interaction with Decision Aids: A Mathematical Approach," in W.B. Rouse (Ed.) *Human/Technology Interaction in Complex Systems*, JAI Press.

Levis, A.H. (1999), "System Architectures," in A.P. Sage and W.B. Rouse (Eds.) *The Handbook on Systems Engineering and Management*, Wiley, NY.

Louvet, A.C., J.T. Casey and A.H. Levis (1988), "Experimental Investigation of the Bounded Rationality Constraint," in S.E. Johnson and A.H. Levis (Eds.) *Science of Command and Control: Coping with Uncertainty*, AFCEA International Press, Fairfax, VA.

March, J.G. and H.A. Simon (1958), *Organizations*. John Wiley and Sons, NY.

Murata, T. (1989), "Petri Nets: Properties, Analysis and Applications," in *Proceedings of the IEEE*, 77(4) 541–579.

Perdu, D.M. and A.H. Levis (1998), "Adaptation as a Morphing Process: A Methodology for the Design and Evaluation of Adaptive Organization Structures," *Computational and Mathematical Organization Theory*, 4(1), 5–41.

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

Rational Software Corporation (1998), *Rational Rose® 1998: Using Rational Rose*. Rational Software Corporation.

Reisig, W. (1985), *Petri Nets, an Introduction*. Springer-Verlag, Berlin, Germany.

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

Rumbaugh, J., M. Blaha, W. Premerlani, F. Eddy and W. Lorensen (1991), *Object-Oriented Modeling and Design*, Prentice-Hall, Inc. Englewood Cliffs, NJ.

Sage, A.P. (1993), "Object Oriented Methodologies in Decision and Information Technologies," *Information and Decision Technologies*, 19(1), 31–54.

Serfaty, D. and E.E. Entin (1995), "Shared Mental Models and Adaptive Team Coordination," in *Proceedings of the First International Symposium on Command and Control Research and Technology*, National Defense University, Washington, DC, pp. 289–294.

Stabile, D.A. and A.H. Levis (1984), "The Design of Information Structures; Basic Allocation Strategies for Organizations," *Large Scale Systems*, 6, 123–132.

Wohl, J.G. (1981), "Force Management Decision Requirements for Air Force Tactical Command and Control," *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-11(9).

Holly Ann Heine Handley received her Bachelor of Science in Electrical Engineering with Great Distinction from Clarkson College in 1984. She received her Master of Science in Electrical Engineering from the University of California at Berkeley in 1987, which she attended under the Miccioli Scholar Program sponsored by Raytheon Company. She earned a Master in Business Administration from the University of Hawaii at Manoa in 1995; she was named the Outstanding Student in Management Information Systems. She completed her Ph.D in Information Technology and Engineering from George Mason University in 1999. She is a member of Tau Beta Pi and Beta Gamma Sigma, and she is a Professional Engineer licensed from the state of Washington. Dr. Handley worked as a design engineer for Raytheon Company's Submarine Signal Division from 1984 to 1988 where she was responsible for the design of various digital modules for military systems. She transferred to the Semiconductor Division in 1988 where she assumed responsibility for customer applications utilizing the CMOS 1.25u VLSI technology. In 1989 she was promoted to Senior Engineer and worked as a Field Engineer providing technical support to the European sales staff on Raytheon's digital, linear, and mixed signal gate arrays and standard cell products. She left Raytheon in 1993 to pursue academics full time.

Alexander H. Levis is University Professor of Electrical, Computer, and Systems Engineering. He is associated with the C31 Center where he heads the System Architectures Laboratory. He served as Chair of the Systems Engineering Department in 1992–1994, and then again in 1996–1997. He was educated at MIT where he received the BS (1965), MS (1965), ME (1967) and Sc.D. (1968) degrees in Mechanical Engineering with control systems as his area of specialization. He also attended Ripon College where he received the AB degree (1963) in Mathematics and Physics. He taught systems and control theory at the Ploytechnic Institute of Brooklyn from 1968 to 1973 and, for the following six years, he was with Systems Control, Inc. of Palo Alto, CA, where he was manager of the systems research department. From 1979 to 1990 he was a Senior Research Scientist at the MIT Laboratory for Information and Decision Systems. He joined George Mason University and the C31 Center in 1990. In 1992, he received the Distinguished Faculty award. Dr. Levis is a Fellow of the Institute of Electrical and Electronic Engineers (IEEE) and past president of the IEEE Control Systems Society from which he received in 1987 the Distinguished Member award. He is also a Fellow of the American Association for the Advancement of Science (AAAS), a member of AFCEA, and a senior member of AIAA.