DOES ORGANIZATIONAL STRUCTURE MATTER? AN ADAPTIVE SIMULATION APPROACH FOR INVESTIGATING INFORMATION PROCESSING STRUCTURES IN ORGANIZATIONS

Roger Stein
Department of Information Systems, 0266
New York University
Leonard N. Stern School of Business
44 West 4th Street, Suite 9-181
New York, NY 10012-1126
(212) 998-0800
fax: (212) 995-4228
rstein@stern.nyu.edu

July 1996

Working Paper Series
Stern #IS-96-4

Abstract

We adopt the view of organizations as information processing entities. As such, we propose that various exogenous and endogenous factors should affect the performance of organizations with respect to information processing tasks. We present a methodology for modeling organizational structures and for determining which organizational structures, if any, distinguish themselves given various constraints. Our methodology relies upon computer simulations that combine Monte Carlo methods and genetic algorithms to represent dynamic organizational operating environments and competition among firms, respectively.

1.0 Introduction[1,2]

As a field, how much do we really understand about how the structure of organizations affects how they process information?

A stream of OT literature has dealt with the perspective of organizations as acting to process informational cues from their environments (March and Simon, 1958; Galbraith, 1974; Radner, 1992) or acting to make sense out of the situations with which they are confronted (Weick, 1979). These views hold that organizations arise primarily to deal with the uncertainty inherent in complex environments and with the cognitive limitations on human information processing power. March and Simon (1958), for example, state that

> Organizations will have structure... insofar as there are boundaries of rationality - insofar as there are elements of the situation that must be or are in fact taken as givens, and that do not enter into rational calculations as potential strategic factors. If there were not boundaries to rationality, or if the boundaries varied in a rapid and unpredictable manner, there could be no stable organization structure. (p. 170)

Yet clearly, there are costs associated with implementing and maintaining organizations (Tushman and Nadler, 1978; Williamson, 1979). Activities must be coordinated, members must be monitored, and contingencies must be sized. Furthermore, assuming that structure does make a difference, then given the search space of possible organizational structures, effort must be expended to determine which structures are appropriate to which organizations in which situations, (March and Simon, 1958; Tushman and Nadler, 1978).

Organization theorists have debated issues relating to organizational structure and efficiency for several years. To date, however, there has been little empirical evidence to support these arguments. Radner (1992) notes:

> Finally, I shall have to admit that research to date has not provided an adequate explanation on economic grounds alone of the conditions under which one expects to see a hierarchical organization of business firms. (p. 1384)

What are the "conditions under which one expects to see a hierarchical organization?" Does the structure of an organization matter? What are the factors that mediate the efficiency of a particular organizational structure?

In this paper, we present a proposal for using computer simulation as one method to understand structural dynamics of organizations. We develop a series of propositions growing out of our interpretation of the literature on organizations as information processing entities. We then discuss how our modeling approach, based on a combination of Monte Carlo methods and genetic algorithms, provides a tool for testing these propositions.

Our basic premise is that under certain sets of environmental conditions certain organizational forms process information more effectively than others. Furthermore, the effective form of an organization cannot be determined without considering the information processing capabilities of the individuals that make up the organization as well as the environment in which the organization operates.

The remainder of this paper is organized as follows: Section 2 discusses various types of prototypical organizational structures and how these structures affect the information processing capabilities of organizations. Section 3 discusses an abstract model of organizations, their members, and the tasks with which they are faced. Section 4 discusses the nature of information processing efficiency in organizations and model of the search for efficient structures by organizations. Section 5 presents a simple metric that provides a proxy for quantifying the organizational structure. Section 6 we develops three sets of propositions that arise out of our discussions of structure and information processing. Section 7 presents an overview of Monte Carlo simulation and genetic algorithms and goes on present an organizational simulation methodology that encompasses both. Section 8 discusses how the methodology can be used to

---

[1] This is a working paper. As such several areas of the paper and the thinking underlying it are still being revised and reworked.

[2] A portion of the thinking underlying this paper arose from work done by the author at the Santa Fe Institute during the summer of 1995. The material has, however, never been formally presented before. The author benefited greatly from discussions with Ajit Kambil, Hank Lucas, and Michael Burtka, all of the Stern School at New York University. Any errors are, however, the author's own.

test for effects implied by the propositions. Finally, in the concluding section, we discuss limitations of our methodology.

2.0 Organizational structure and information processing capabilities

In this section, we consider the relationship between the structure of an organization and its information processing capabilities. We draw heavily on the work of Tushman and Nadler (1978) and Radner (1992).

Tushman and Nadler describe organizations in terms of being organismic or mechanistic. Mechanistic organizations correspond to hierarchies, and organismic organizations correspond to fully connected networks. Tushman and Nadler distinguish mechanistic from organismic structures on the basis of the degree to which members of an organization are interconnected, saying:

> ...One way of thinking about the impact of subunit structure on information processing capacity is by focusing on the impact of subunit structure on patterns of communication (p. 617).

Organismic structures are characterized by high degrees of connectivity with members communicating freely among themselves along multiple channels. Mechanistic structures, in contrast are characterized by a small number of specialized connections between specific organization members. Information travels along much more highly specified routes and each member communicates with only one or a few other members.

How should these architectures affect organizational information processing? According to Tushman and Nadler,

> Subunit structure has an important impact on the subunit's ability to process information and deal with uncertainty... While organismic structures are able to deal effectively with greater amounts of uncertainty than more mechanistic structures, there are costs associated with this increased information processing capacity. Organismic structures consume more time, effort energy, and are less amenable to managerial control. *Thus, the benefits of increased information processing capacity must be weighted against potentially increased response time* (p. 618). (Emphasis added.)

We would expect, then that in noisy, uncertain environments organismic structures would have an advantage due to their error tolerance, but in stable predictable environments they would suffer unnecessary processing delays and inefficiencies. This would be the mirror image of the behavior of mechanistic structures which should be quite efficient in stable low-noise environments, but unreliable in high-noise, high-variability environments.

Radner (1992) supports this view saying that, in situations where "the amount of environmental data is given" (p. 1393) (i.e.: stable environments), "It is not economical for all decisions to be based on all the information available to the firm." (p. 1384).

This suggests that hierarchical structures are more efficient than mechanistic ones under these conditions due to their partitioning of computing tasks among sub-units of the organization. Radner, in fact, goes on to say that in some situations hierarchies are, in fact, the optimal configuration of organizational members in that they minimize the number of processors required by the organization for certain classes of problems.

A view of organizational structuring as a crucial factor in information processing efficiency thus emerges. In the remainder of this paper we discuss how this view might be validated through simulation experiments.

3.0 Abstract organizations

In order to operationalize our concept of organizational structuring for our simulations, we allow several abstractions in our model. In the next sections we discuss the abstract form of organizations, their members, and their environments.

3.1 The nature of organizational environments

In the literature, organizational environments are typically described as being *certain* or *uncertain* with respect to their information flows and the operations necessary to process environmental cues (e.g. Galbraith, 1974). Furthermore, organizations must deal with information flowing at differing velocities. Some environments (ex: financial markets) require very rapid processing of and responses to new information, while others (ex: automotive design) permit somewhat slower processing of and response to new information.

If the arrival speed of new information is faster than the delay required to process it, organizations face the unhappy prospect of making decisions based on outdated information. (Radner, 1992).

For the purposes of this paper, we will focus on the *velocity of information arrival* as a main environmental consideration for organizations planning their structures.

3.2 The nature of organization members

Following Radner (1992) we define organization members (or processors) as having a *register*, an *in-box* (memory), and a *clock*. Processors move items from their in-box to their register for computation.

We extend Rander's representation by also allowing varying computing speeds (IQ) for processors. As a result, rather than allowing a processor to retrieve only one item per clock tick, we say that a processor may take as many items per clock tick from its memory as its computing speed will allow[3].

We also allow varying memory sizes for processors. The processor may maintain a queue of as many items as its memory will allow. Once memory is exceeded, the oldest unit of information is pushed out and lost to make room for the next unit of new information, and so on.

Finally, each processor has an *address list* which it uses to notify other members of the organization when it completes its calculations. Note that this means members can be ignorant of the overall structure and processes of the organization while still performing their tasks and processing information in the organizational context (Cohen, James and March, 1972).

From an individual perspective, the cognitive power of members within the organization will affect the manner in which information is processed. Two of these factors, processing speed and memory capacity will be important. Individuals with high processing speeds are able to interpret information more quickly than those with lower speeds. Similarly, individuals with large memory capacities, can remember more information to use in their processing tasks; they can hold more information in memory for simultaneous usage.

For the purposes of this paper, the *average memory size* and *average computation speed* of organization members are the factors relating to organization members that must be considered in organizational structuring.

3.3 The nature of organizational tasks

March and Simon (1958) propose that most of what organizations do can be considered pattern matching. Similarly, Radner (1992) suggests that organizations engage in two types of activity (1) linear decision rules, in which environmental cues are transformed into decisions via linear superposition; and (2) pattern matching in which environmental cues are examined and compared with a store of exemplars.

For purposes of this paper, we will consider an organizational task to be either a *linear combination of environmental cues* such as averaging or summing, or a *pattern matching problem* such as calculating the minimum Hamming distance between incoming data and a member's of set of exemplars.

This view of organizational tasks is useful in this paper for two reasons. Firstly, these definitions lend themselves conveniently to computer simulation. Secondly, such definitions can be generalized to a large class of organizational problems.

4.0 Organizational information processing efficiency

Several factors can differentiate organizational information processing. We are proposing that these factors also constrain the performance of the various tasks that an organization might undertake. *Ceteris paribus*, some organizations are more readily suited than others to performing certain tasks. The role of an organization's structure is,

[3] Without losing generality, the computing speeds may also be thought of as the amount of attention (Cohen, James and March, 1972) that individuals can afford to expend evaluate incoming information.

therefore, to make optimal use of the firm's basic cognitive assets for performing the set of tasks with which it is faced, given its cognitive resources.

4.1 A model of information processing efficiency seeking by organizations

Given the above discussion, we can now describe more formally a method by which organizations seek to structure themselves for efficient information processing.

Specifically, we say that the information processing efficiency, E, of an organization using a particular organizational structure, S_i, can be given as:

$$E=f(S_i,m,p,R) \quad \text{(Eq. 1)}$$

where, m is the average memory of members, p is the average processing speed of members, and R is the information flow rate in the environment[4].

The reader will note that for our purposes, the members of the organization (and their attributes) are considered fixed. Organizations with fixed members seeking to optimize their information processing capabilities should therefore seek the structure, S_i, which maximizes the expected efficiency, E. This amounts to seeking the structure that maximizes the objective function $f(.)$ while holding m, p, and R constant.

Unfortunately, most organizations do not know the form of the objective function $f(.)$ described in Eq. 1; it is hidden. As a result, their search method must be a closed-form type of optimization in which they experiment with different organizational structures, S_i, and observe $\partial f / \partial S$ as they do so[5].

Note that for purposes of this paper, it is irrelevant what strategy organizations pursue in searching for structures. Some organizations may use heuristics, some may use random search, and some may use sophisticated organizational theoretic models of structurization. We are only trying to determine the effect of organizational structure, not the path by which that structure is achieved.

5.0 Organization connectivity: A measure of structure

Before we can present propositions about the relationship between structure and information processing, it is necessary to provide some objective evaluation of an organization's structural description. How can one determine whether an organization is more mechanistic or organismic? We propose that *connectivity*, defined below, is one such measure.

First consider how many connections can exist between two organization members. In our model, there are at most two communication relationships possible between any two members of an organization. There can exist no relationship; there can exist a communication line from A to B; and there can exist a communication line from B to A[6].

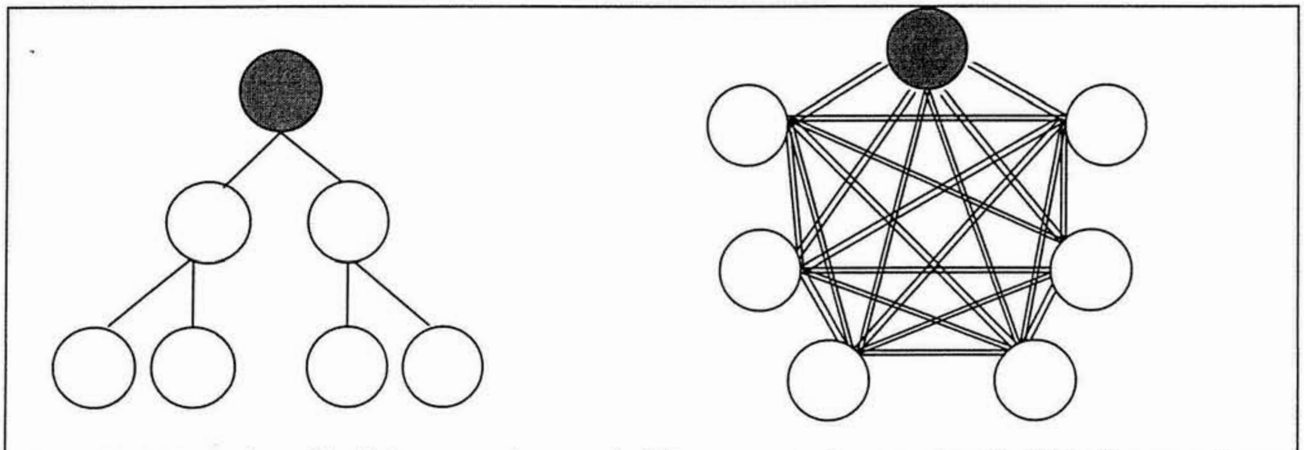
---

[4] More accurately, m and p should be vectors of size N (where N is the number of members in the organization), and R should be a conditional probability distribution. However, to simplify the model we have chosen to use the mean values of these variables over their respective domains.

[5] This is similar in spirit to what Tushman and Nadler (1972) propose when they say, "...[T]he essence of organizational design is: subunits must choose from a feasible set of structural alternatives, a particular set of organizational arrangements, to most effectively deal with their information processing requirements." (p. 622)

[6] This may be thought of intuitively as follows: if A is the president of an organization and B is a production line worker, it is likely that there will be a fast line of communication between A and B (under A's control). It is also highly unlikely that there will be a communication link from B to A. On the other hand, if C is a fellow production line worker, it is likely that there will be a connection from B to C and a corresponding connection from C to B. Finally, if D is a production worker in another plant several states away, it is unlikely that there will be any connection at all between C and D.

Figure 1, below demonstrates examples of both a hierarchy and a fully connected organization. Although both organizations have seven members, we can see visually that the number of connections is quite different. The fully connected network is much denser than the network in terms of inter-member connections.



A perfect hierarchy with N-1 connections   A fully connected network with $N(N$-$1)$ connections

Figure 1

In the case of the hierarchy shown in the Figure 1, there are seven members in the organization and there are 6 links in the network. In contrast, the fully connected organization has seven members but contains 42 links. The more connections exist in an organization, the more likely it is that the organization is organismic. Conversely, the fewer connections exist, the more likely it is that the organization is hierarchical.

For reasons we will discuss below, the formulation of organizational structure as solely characterized by the degree of connectivity is problematic. Nonetheless, for purposes of our experiments, we propose a this simple proxy of organization structure. We term the metric *connectivity* and define it as:

$$C = \frac{\sum_i \sum_j \delta(c_{i,j})}{N}, \qquad \text{(Eq. 2)}$$

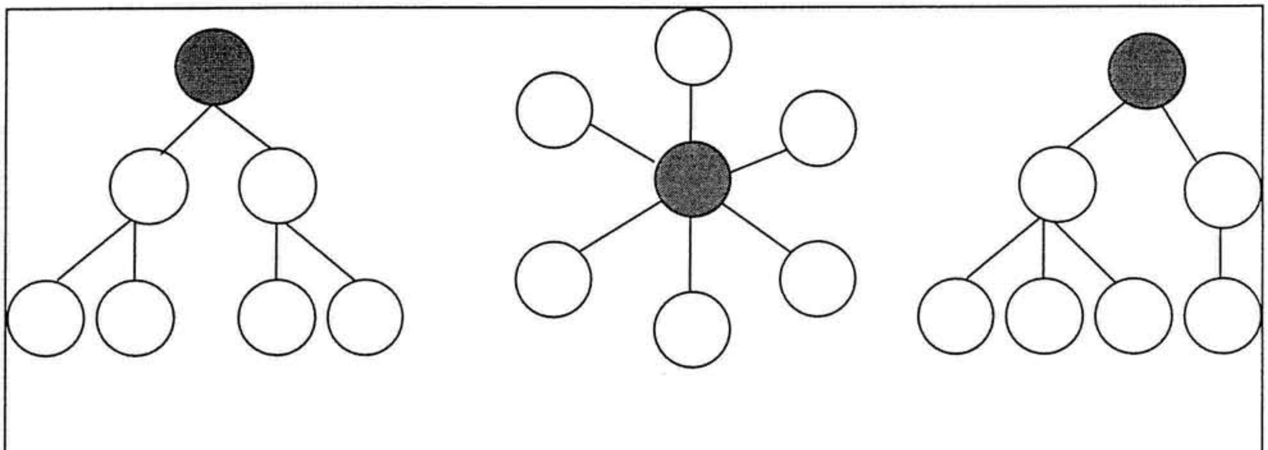
where, C is connectivity, N is the number of members in the organization, $c_{i,j}$ is the communication relationship between the i^{th} and j^{th} member of the organization, and $\delta(c_{i,j})$ is unity if a communication link exists and zero otherwise. This measure is quite similar to the measure connectedness used by Tichy and Fombrun (1979) as a dimension of organizational network structure[7].

This measure is simply the average number of connections per member in the network. Note that for organizations with N members, C will take on a value near one in the case of a hierarchy since each member (except the root node) will have exactly one link between it and its superior, and the root node will have no links. Similarly, C will take on the value of $(N\text{-}1)$ in the case of a fully connected organismic structure since each of the N members of the organization will have a connection to each of the $(N\text{-}1)$ other members of the organization.

[7] Our measure of connectivity is somewhat different in that it allows for uni-directional links between members of a network. Since Tichy and Fombrun (p. 957) propose only bi-directional links, they formulate connectedeness as: $2L/[N(N\text{-}1)]$ where L is the number of bi-directional links possible in the network $(L \leq [N(N\text{-}1)/2])$.

This metric for describing organizational form is also convenient in that it reduces an organizational structure into a single integer which makes it amenable for treatment with standard statistics. The utility of such a measure will become apparent in the experimental section of this paper.

But there are drawbacks in using such a proxy as well. The measure is one of organizational density not topology. For example, Figure 2, below, shows three possible topologies for a network with seven members and six connections. All three of these networks would have the same measure of connectivity. Although these are, in fact, each variants on a hierarchy, they nonetheless represent different topological forms.



Three different topologies for a network with seven members and six connections

Figure 2

In fact, since there is a maximum of $N(N-1)$ possible connections in a network of N nodes, then, for any organization with N members and k connections, there are

$$\binom{N(N-1)}{k} = \frac{[N(N-1)]!}{k![N(N-1)-k]!} \qquad \text{(Eq. 3)}$$

possible topologies. For example, in addition to the three networks shown in Figure 2, there are approximately five and a quarter million other configurations of seven nodes and six connections (assuming uni-directional connectivity). The connectivity metric described in Eq. 2 does not distinguish between these different structures.

At first, this seems daunting. Nonetheless, this may not be overly concerning for two reasons. Firstly, as we have discussed there is a correspondence between structure and connectivity. For example hierarchies will always have connectivities approaching one, and fully connected organizations will always have connectivities of $(N-1)$. The fact that the converse is not true should not trouble us for the purposes of this paper. This is particularly so given the fact that the simulation method we propose later on which serves to "weed out" networks which do not perform well.

Secondly, more complete descriptions of the structure, which better differentiated these topologies could easily be substituted for connectivity without fundamentally altering the methodology described herein[8].

[8] Tichy and Fombrun (1979), for example, propose a more complex analysis method involving the use of two software packages (SOCK and COMPLT) for analyzing network structures, sub-structures, and clustering. These procedures evaluate networks based on the number of shared connections among nodes in the network, segmenting networks into "clusters" of related activity which can then be evaluated both individually and with respect to the overall network in terms of density, hierarchy, overlap, etc.

The reader may question how memory and computing speed enter into the calculations. Recall, however, that the analysis we describe herein proposes that the members of the organization (and their attributes) are fixed at various levels and that the organization seeks to structure itself as efficiently as possible, given its current environment and its resources.

6.0 Propositions

We may now describe propositions that indicate the expected relationships between an organization's structure, the attributes of its individuals, and its environment.

Earlier we discussed the impact that high speed information flow can have on an organization's ability to process information. We hypothesize that hierarchies should perform better than organismic structures in high information intensity environments since information can be processed incrementally at lower levels allowing higher level members to make faster decisions.

We would expect "information overload" to be a potential problem in high information intensity environments. Hierarchical structures lend themselves to information filtering. On the other hand, if the information flow is fairly slow, organismic structures should not suffer overly since their members will have ample time to process information before new information arrives. This leads us to state our first set of propositions:

> *P1A: In low information intensity environments, all other factors held constant, most organization structures should be able to perform well.*
>
> *P1B: In high information intensity environments, all other factors held constant, the performance of high connectivity structures should dominate that of low connectivity structures.*

We can also make statements about the impact of individuals' memory capacities on information processing. If individuals have very small memory capacities, then no matter how quickly they can calculate, they will only be able to base decisions on a few pieces of data.

In organismic structures, since organization members are unable to maintain very large stores of information, they cannot afford to attend to all the information queues present in the environment or in the organization. Since, hierarchical organizations allow members to partition memory requirements, we would expect to see benefits from the filtering and aggregation potential of hierarchical structures.

If, in contrast, memory is ample, filtering is less essential and most structures should compete equally. This leads us to our second set of propositions:

> *P2A: In organizations whose members have high memory capacity, all other factors held constant, most organization structures should be able to perform well.*
>
> *P2B: In organizations whose members have low memory capacity, all other factors held constant, the performance of low connectivity structures should dominate that of high connectivity structures.*

Finally, we can discuss the impact of individual's computing speed on information processing tasks. Slow computers will benefit from organizational partitioning of information processing in much the same way as those with limited memory. If computing speed is slow, the organization members cannot process very much information per time period.

While these procedures are much more complex than the simple measure of connectivity we propose, there is no reason, in principle, why they could not be applied to the analysis of networks generated by the simulation processes described in this paper. This would, of course, require considerably more programming skill and analysis time than the simple use of connectivity, but is certainly feasible. See Tichy and Fombrun (1979) for a more complete discussion.

As a result, they are forever behind until eventually their memories overflow because they cannot process information fast enough.

Furthermore, even when their memories are filled below capacity, slow computers will often be late in their computations resulting in a similar problem as that described in the first set of propositions. In effect for slow computers, *any* speed of information flow is too fast.

We would expect hierarchies to perform better than organismic structures under these circumstances due to the benefits accruing from partitioning of computing tasks. On the other hand, if computing speed is fast, this is much less of a problem and we would expect that organismic structures should not suffer overly since their members will have ample time to process information before new information arrives. This leads us to state our final set of propositions:

> *P3A: In organizations whose members have high computing speed, all other factors held constant, most organization structures should be able to perform well.*
>
> *P3B: In organizations whose members have low computing speed, all other factors held constant, the performance of low connectivity structures should dominate that of high connectivity structures.*

7.0 Testing the propositions through simulation

Having offered the three sets of propositions in the previous section, we now turn to the task of testing them through simulation. We chose simulation methods for several reasons.

Firstly, as we discussed in Section 4.2, our assumption is that organizations do not know the functional form of Eq. 1. As a result, they perform a closed-form optimization on their organizational structures. Simulation methods allow us to mimic this search process. Secondly, simulation methods allow us to test the propositions while making only a limited number assumptions about the parameters of the problem. Thirdly, simulation techniques allow us to deal with the complexity inherent in organizational structures. Finally, as we will describe below, simulation allows us to create an environment in which differing organizational structures compete with each other to efficiently perform similar tasks.

7.1 Modeling the organization members and their behavior

We have created a generic toolbox, called OrgNet, for representing organizations. The toolbox is an object library written in C++. The toolbox allows us to represent organizations, their members, and their structures. In this sub-section, we present the properties of OrgNet. (See: *Appendix A* for a simple example of how the toolbox is used.)

7.1.1 Representing organization members

The library represents organization members as having a fixed amount of memory, a particular computing speed, a task function, and a message queue that holds input information that the member receives. The computing speed parameter indicates how many units of time the member requires to execute its task function. The task function itself defines how the member processes the information in its message queue. For purposes of this paper a task function takes on one of the forms described in Section 3.3.

The message queue can hold as many units of information as the member has memory. Once memory is exceeded, the oldest unit of information is pushed out to make room for the next unit of new information,. In addition, each member has a list of other members to which messages about the results of performing the task are sent[9]. Organization members wait idly until a message is received in their queue, at which point they begin processing the message according to their task function.

7.1.2 Representing the organization

The organization itself is composed of a number of members (as described in Section 7.1.1) and a matrix which defines the relationships among the members. This matrix is defined more fully in Section 7.2.2.

---

[9]Although not required for this project, additional communications attributes and capabilities may be defined for the relationships between organization members.

The organization has a message queue as well. The message queue serves as a post-office or clearing house for information. Members communicate with each other by posting an unambiguously addressed message to the message queue. The organization coordinates delivery of these messages to its members. Finally, the organization has a clock which synchronizes the activity of the organization members and message delivery. Each message is time-stamped with its scheduled delivery time and is delivered at the time indicated.

For example, the message

```
#1    [3->2]      at 5 <0.000000>
```

would indicate that message number one in organizational message queue is a message from member 3 to member 2 to be delivered in time period 5 and containing the value 0.000000.

Organizational tasks can take on a variety of forms. For simplicity, we adopt a definition of tasks as either 1) linear decision rules; 2) or pattern matching (Radner, 1992). In the first case, the goal of the organization is the make decisions based on a linear combination of environmental inputs. In the second, the decision makers in the organization must compare environmental input with a set of known exemplar patterns and make decisions based on the nearness of the inputs to a particular pattern.

7.2 Modeling the organizational environment

Once an organization has been defined, we need to create an environment in which it can operate. In order to create such an environment, we perform stochastic (Monte Carlo) simulation. This subsection describes that process.

7.2.1 Monte Carlo simulation

Monte Carlo methods involve simulating a wide range of the behaviors of a system of interest by designing a computer-based representation of the system. Stochastic components of the system are represented using pseudo-random number generation. The range of behaviors of the system is then sampled by executing a large number of iterations of various possible random paths or trajectories through time (Ripley, 1987).

Simulation is often used when more traditional open-form mathematical and statistical methods are unavailable or prove intractable. Monte Carlo methods have been used modeling a wide variety of processes including economic markets (Weber , 1994; Kirman and Salmon,1995; Clemons and Weber, 1996), the behavior of financial instruments (Hull, 1993), queuing processes (Rubinstein, 1986) and organizational dynamics (Cohen, James, and March, 1972; Starbuck, 1983), as well as a in wide variety of other areas.

7.2.2 Using Monte Carlo methods to model the organizational environment

Using random number generators, we produce a time-series of inputs (environmental cues) which the simulated organization must process.

The speed at which these inputs arrive is dependent on the parameter, R, of the random number generator. If the speed is fast, there will be new vectors of input at almost every tick in the time-series. If it is slow, most of the time-series will not contain new inputs.

The time-series is processed by the members as follows:

1. At each tick of the organization clock, the time-series is evaluated.
2. If an input is not present in the time-series, no action is taken.
3. If an input is present it is placed into the organization queue addressed to all organization members eligible to receive input from the environment
4. The input is delivered to the queue of individual organization members who process it according to their task functions

Only the timing of the inputs is random, the actual form of the inputs is a simple periodic function, either additive or pattern-matching, as described in Sections 3.3 and 7.1.2.

7.3 Modeling inter-organizational competition

Since real organizations compete with one another, we would like to incorporate this feature into our simulations.

By comparing the actual output of an organization with the ideal output that the organization would have gotten if it processed the information perfectly, we can get a feel for how well the organization processes the input information from the environment.

Since the input data represent functions that are deterministic (Section 7.1.2), we know the ideal outputs the organization should make, given the input data. The difference between the actual output of an organization, and the desired output form the error of an organization in performing its tasks. The larger the error, the poorer the organization's performance.

By comparing any two organizations, we can determine which, if either, is more efficient than the other by comparing the error rates of the two. The organization with the lower error rate is *de facto* more efficient at solving the particular problem at hand in the particular environment at hand.

Note that this does not necessarily mean that the favored organization is most efficient at solving other problems. In fact, it often means that it will not be. (We will return to this point when we discuss the caveats associated with our method.) In this sub-section we discuss how competition can be modeled using genetic algorithms.

7.3.1 Genetic algorithms

We will use genetic algorithms (GAs) to simulate interorganizational competition. GAs have their basis in the biological metaphor of survival of the fittest. GAs have been found to be useful for finding good solutions for a wide variety of problems, including classes of problems that were previously computationally prohibitive (Holland, 1970/1992; Davis, 1991; Goldberg, 1989).

A genetic algorithm attempts to solve a problem by creating a range (population) of possible solutions. These usually take the form of strings. Each member of the population (an individual) is then interpreted and evaluated in the context of the problem and ranked in terms of its fitness. Fitness is an assessment of how well a particular individual solves the problem at hand. (In a biological context, the problem specifications can be seen as analogous to the environmental constraints brought to bear on an organism, and fitness as a measure of how well the organism survives in its environment.)

The individuals are then matched with other individuals in the population such in a way that those with higher fitness are more likely to be selected. The results of this mating form the offspring that make up the population of the next generation and the process can be repeated with this new population.

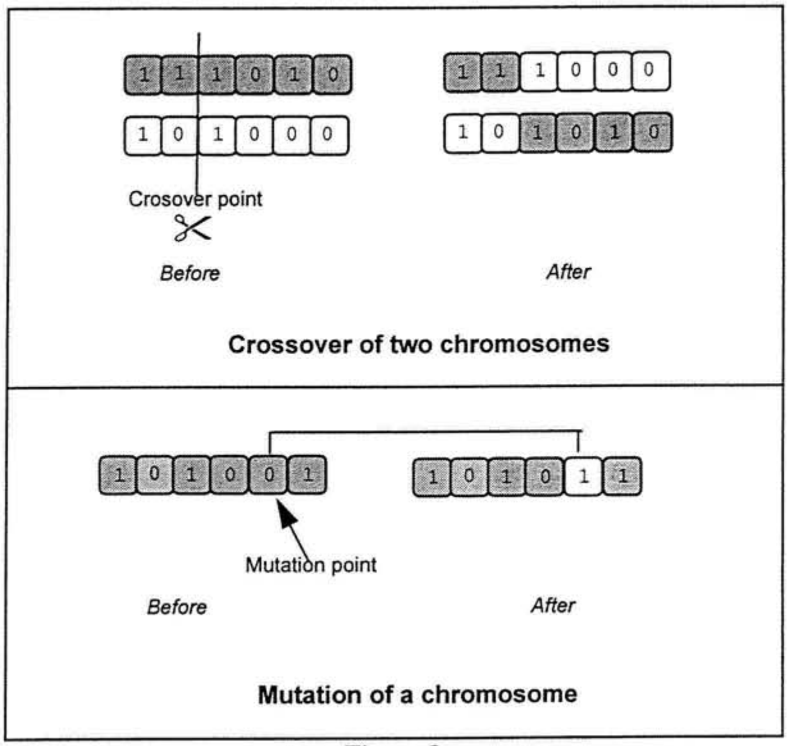


Figure 3

During the reproduction process two operations take place: mutation and crossover. Mutation involves changing the value of an information unit in an individual (an allele). Crossover involves the exchange of portions information between two individuals.

By mutating and crossing over, the GA is, in effect, experimenting with new solutions while preserving potentially valuable interim results or building blocks (Holland, 1970/1991, Davis, 1991; Goldberg, 1989; Goldberg, et al, 1992). If an experiment (crossover or mutation) fails (that is, produces a relatively unfit offspring), then the offspring will, in all likelihood, be dropped from the population within a few generations due to its inferior fitness. On the other hand, if the experiment is successful, then these new interim results can be passed on to the future generations for further refinement. Thus the more promising areas of a solution space are explored, and lower payoff areas are examined in a more cursory manner.

The top and bottom figures in Figure 3 show how crossover and mutation respectively would work for a chromosome composed of Boolean alleles. In our simulation, we represent chromosomes as strings of Boolean digits. As we will show in the next section, such strings can be converted into representations of organizational structure.

The genetic algorithm paradigm allows the search of potentially huge problem spaces in a parallel and efficient manner (Goldberg, 1989). Because of the constant adjustments due to mutations and crossovers, the risk of converging to a local minima or maxima is low in comparison to many other methods, provided that the problem is coded sufficiently (Goldberg, et al, 1991).

In addition to straightforward optimization problems, GAs have been used to model a variety of social systems including money markets (Boehme, 1994), the evolution of behavioral norms (Axelrod, 1985), the analysis of foreign policy (Schrodt, 1986), and the dynamics of arms races (Forest and Meyer-Kress, 1991).

7.3.2 Representing an organization as a chromosome in a genetic algorithm

To represent an organizational structure inside a GA, we first need to conceive of an organizations connections as entries in an $N \times N$ matrix where entry i,j represents the value of $\delta(c_{i,j})$ from Eq. 2. In other words, entry i,j is unity if there is a communication relationship between the i^{th} and j^{th} member of the organization, and is zero otherwise. The middle section of Figure 4 shows how various organizations might be represented using such a matrix scheme[10].

To map this structure onto a chromosome in a genetic algorithm, we start by observing that we can convert each column of the matrix into a string by merely transposing it. To create a chromosome, we take each of the columns of the matrix and concatenate the transposed columns into a single string of length N^2. Each chromosome in the GA now represents an separate organizational structure. This is shown graphically in the bottom portion of Figure 4.

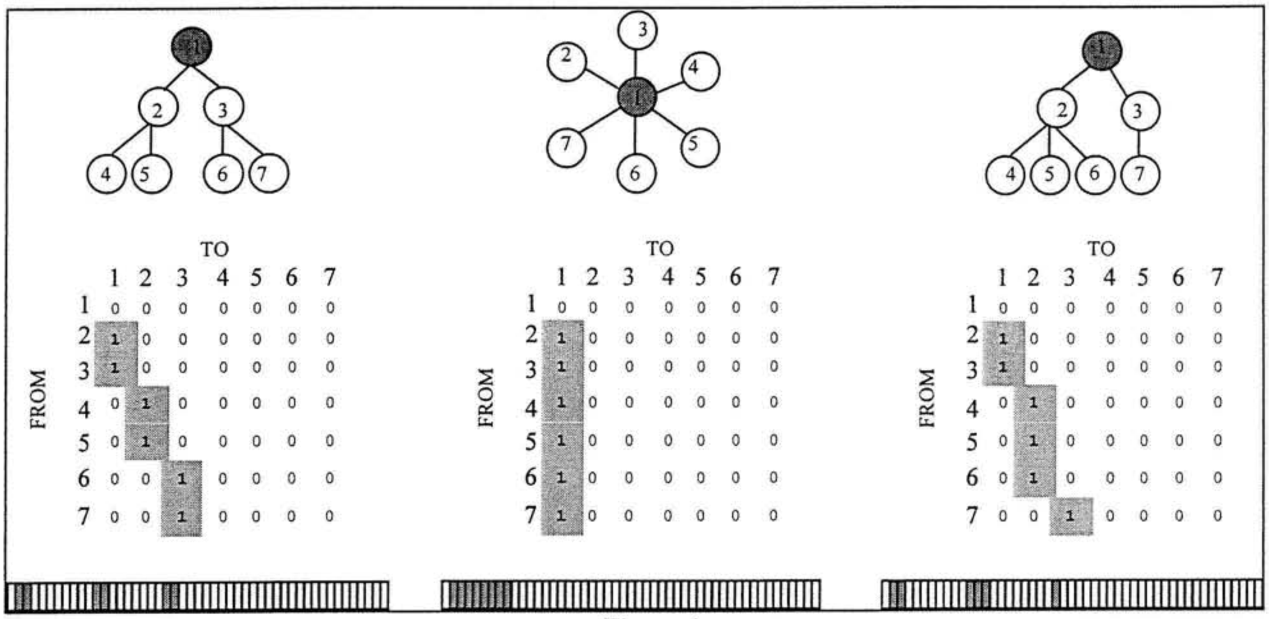


Figure 4

In this context, mutation amounts to changing the value of a matrix entry which translates into creating (or destroying) an communication link between two members of an organization. Crossover, on the other hand, corresponds to swapping a range of entries in one matrix with those in another matrix which translates to an organization adopting a sub-structure from another organization.

7.3.3 Using a GA to model inter-organizational competition

By representing an organization as described in the previous section, a GA can create a population of organizations, each with a different structure. Initially, the GA will generate a random population of organizations. The GA can allow these organizational structures to compete with each other at solving a task (in our case, the task represented by the Monte Carlo simulation).

By periodically evaluating the performance of each organization at the task (i.e.: the accuracy of a linear decision or the correct classification rate for a pattern matching problem) the GA is able to rank each of the organizations based on its performance. The better performing organizations will have a higher likelihood of being selected to continue on to the next generation.

Crossover allows successful organizations to swap sub-structure designs and mutation allows them to make random changes to their structures. This continues for several generations. Since the GA periodically evaluates the organizations performance, over time, the successful sub-strings should be retained and unsuccessful ones should be flushed out of the population of organizations (Goldberg, 1989).

---

[10] This is similar to the scheme proposed by Cohen, James and March (1972) for representing what they term the access structure of individuals in an organization.

8.0 Simple experimental design

For simplicity, we assume a generic organization made up of members with homogeneous memory and processing speeds[11].

The idea of these simulations is to test the propositions by exploring the range of organization dynamics at different levels of the variables m, p, and R. To do this we will perform a large number of runs of individual genetic algorithms. The output of each run of the genetic algorithm will be a "most fit" organizational structure, given a particular set of values for m, p, and R.

The connectivities of each organization structure resulting from each run of the genetic algorithm form a distribution. It is our hope that by comparing the distributions of connectivity levels conditioned on any particular set of the parameters, we will be able to establish support for, or reject the propositions. Of course, this requires that we simulate structures at many different levels of m, p, and R.

[11] This assumption is not a requirement but making it reduces the complexity of the analysis of the results by several orders of magnitude.

We propose the following algorithm for gathering the simulation data:

```
ALGORITHM 1
a)    For each set of parameter values
      i)    Perform a large number of GA runs (repeat 7.3.3 many times):
            a)    Create a genetic algorithm population of random connectivity
                  matrices for the organization (7.3.2)
            b)    REPEAT for each generation of the genetic algorithm
                  (1)   Create one or more time-series of inputs (7.1.1)
                  (2)   Use the time-series to simulate the organizational
                        environment for each organizational structure (7.1.2)
                  (3)   Evaluate each organizational structure's performance by
                        determining how much error it produced
                  (4)   Evolve the structures using the genetic operators
                        (7.3.1)
            c)    UNTIL max_generations reached (GA run is done)
            d)    At the end of the GA's run, select the fittest (most
                  efficient) organization structure
            e)    Calculate the structures connectivity using Eq. 2.
b)    Record the distribution of connectivities
c)    Use statistical tests of significance (F-tests, t-tests, or χ²) to compare the
      distributions along the variable of interest
```

Initially we might choose values of the parameters that were prototypically large or small. The results of this might be represented in a 4x2 table as follows:

COMP SPEED/ MEM	INFO. INTENSITY Low	High
Low / Low	C_1	C_5
Low / High	C_2	C_6
High / Low	C_3	C_7
High / High	C_4	C_8

As we began to understand the process better, we can allow each of the parameters to step up in a nested fashion so that we explore more fully the dynamics of the parameter space. In this case a graphical representation might also prove useful.

An algorithm for this process would be as follows:

```
ALGORITHM 2
a)    Set parameters to minimum values
b)    WHILE R is less than max_R
      i)    While m is less than max_m
            a)    While p is less than max_p
                  (1)   Do (a) from ALGORITHM 1
                  (2)   Record the distribution of connectivities
                  (3)   p=p+p_step
            b)    m=m+m_step
      ii)   R=R+R_step
c)    ENDWHILE
d)    Use statistical tests of significance (F-tests, t-tests, or χ²) to compare the
      distributions along the variable of interest
```

9.0 Conclusion

The methodology we propose offers one means to testing empirically the effects of limitations on individual cognitive powers and the severity of information environment on information processing in various organizational structures. By testing the sensitivity of competing organizational structures to changes in environment or limitations on internal resources, we can determine, through simulation and simulated competition, which structures are efficient under which conditions.

But the method is not without its drawbacks. Foremost among these is the use of simulation in place of open-form analysis as an investigative method. Some researchers object to simulation methods on the grounds that they result from arbitrary assumptions about an environment and lead to unclear relationships between inputs and outputs. Starbuck (1983), for example, warns that:

> Simulators have to specify activity sequences and sufficient assumptions even when they lack information about them...Very large models are too large to validate in detail... Many simulations are hard for their creators to understand...Relations that are too complex to analyze algebraically tend to remain stubbornly incomprehensible by alternative means (p. 156-7)

While these concerns are not without merit, they are also not unique to simulation approaches. With respect to the validity of assumptions, as Clemons and Weber (1996) point out, most statistical methods require researchers to make (sometimes questionable) assumptions about distributional properties of data, errors, etc., a fact which is often ignored in the literature. Furthermore, researchers are often forced to assume that samples they obtain through surveys or archival research are random samples from some larger universe, an assumption that is often not supported by the data collection methods used.

With respect to the concern about interpretation of outputs, here again, the concern is not limited to simulation projects. Sophisticated statistical methods and diagnostics are only useful when the users of these methods understand the implications of their use and consider issues beyond the immediate gratification produced by a high level of significance.

While simulation does rely on assumptions and researcher interpretation, these concerns are common to many research methods and should not be used to rule out simulation *a priori.*

On the other hand, simulation does have much to recommend it. Starbuck (1983) admits that:

> ...[S]imulation offers deductive capabilities that, in principle, can extend well beyond those of algebraic analyses. Because computers can accommodate very complex assumptions about multitude variables, simulators...can discover the consequences of many, nonlinear, discontinuous interacting assumptions that no one knows how to analyze algebraically. Simulators can make assumptions they believe to be realistic, even if the assumptions are not mathematically tractable...Although computer simulation is no panacea, its significant capabilities make it the only effective methodology for some research tasks, and the best methodology for others. (p. 156 - 159)

Our goal in presenting this methodology is to capture the benefits of simulation, while, to as large a degree as possible, reducing its negative effects. One way in which we attempt to do this is by limiting the number of parameters in the simulation. In fact, we use only four parameters:[12] N, the number of individuals in an organization; m, the average memory of the members of the organization; p, the average computation speed of the individuals in the organization, and R the rate at which information flows into the organization.

Furthermore, we have designed our experiments in keeping with the recommendation that "a new simulation model should differ only marginally from a previous simulation model that one understands," so that one can gradually, "build up complexity while always being able to identify the assumptions triggering changes in the models' behaviors."

[12] Technically there are also four parameters associated with the GA: the mutation rate; the crossover rate; the population size, and the number of generations. We do not vary these over the course of the simulation and instead adopt standard values recommended in the literature (e.g. Goldberg, 1989).

(Starbuck, 1983). By running simulations at various levels of each parameter while holding all of the other the model parameters constant, we are able to understand the behavior of the system over the range of its values.

In this regard, the reader may question why we chose to hold constant the values of the parameters m, p, and R, while varying the organizational structure as opposed to fixing the structure and allowing other parameters to change. This is not an arbitrary decision. As we discussed in Section 5.0, for any set of nodes, there is a combinatorially large number of possible structures. The other parameters, in contrast, lend themselves much more easily to sequential systematic sampling for experimental design.

As regards the more fundamental assumptions about the nature of organizational environments, the attributes of organization members, and so forth, we feel that these have support in the literature sited. But does such a simple model capture the complexity of the organizational contexts we have described in the motivating literature? We feel it does, although the model of organizations that we present is obviously limited. This is not uncommon in simulation. Cohen, James and March (1972) for example state:

> Though the specifications are quite simple, their interaction is extremely complex so that investigation of the probable behavior of the system fully characterized by [the model] and previous specifications requires computer simulation. (p. 16)

While they concede that "No real system can be fully characterized in this way [simulation]," (p. 16), they support the method contending that their simulation does exhibit behaviors which are observed in organizations, sometimes frequently.

On the other hand, it is not clear that simulating the two types of tasks we have chosen provides general enough results to extend beyond tasks of (linearly) additive or pattern matching forms. For example, some classes of non-linear problems may be much more suited to organismic representations than hierarchical representations[13].

Furthermore, while the results may be generalizable to other tasks that fall within the same basic structure as the two discussed here, it is also probable that the organizational structures that are most efficient for solving one particular problem (say, executing a specific linear decision rule) will not be efficient for solving another different problem of the same class (Radner, 1992). This is true since the second problem's solution may be based on different sets of data than the first.

Despite these limitations, we feel that the method we propose has the potential to give organization theorists a general simulation approach for experimenting with the impact of various environmental and organizational factors on information processing within organizations in the presence of competition.

Overall, however, we feel that the simulation method discussed in this paper provides one possible solution to the dilemma of determining empirically, "the conditions under which one expects to see a hierarchical organization of business firms." In fact, the same methodology, with minor modifications, could be used to examine the types of environments that encourage organismic structures as well as a variety of other organizational phenomena.

[13] Connectionist (neural network) approaches, for example, rely on high levels of connectivity for solving high-dimensional non-linear problems. These problems are often insoluble using simple linear approaches (e.g.: Haykin, 1994).

References

1. Boehme, Frank, "Adaptive Coordination Mechanisms in the Economic Modeling of Money Markets," in *Many Agent Simulation and Artificial Life*, E. Hillebrand and J. Stender, Eds., IOS Press, Amsterdam, 1994.
2. Clemons, Eric, and Weber, B., "Competition Between Stock Exchanges and New Trading Systems: A Demonstration of Adverse Selection and an Exchange Response, Working Paper, *Information Systems Working Paper Series*, Stern School of Business, 1996.
3. Cohen, Michael D., March, James G., and Olsen, Johan P., "A Garbage Can Model of Organizational Choice," *Administrative Science Quarterly*, Vol. 17, 1972.
4. Davis, L.. Ed., *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, 1991.
5. Forest, Stephanie, and Mayer-Kress, Gottfried, "Genetic Algorithms, Nonlinear Dynamical Systems, and Models of International Security", in L. Davis, Ed., *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, 1991.
6. Galbraith, Jay R., "Organization Design: An Information Processing View," *Interfaces*, Vol. 4, No. 3, May 1994.
7. Goldberg, D. E., *Genetic Algorithms in Search, Optimization, & Machine Learning*, Addison-Wesley, 1989.
8. Goldberg, D. E., Deb, K., and Horn, J., Massive multimodality, deception, and genetic algorithms, *Parallel Problem Solving from Nature, 2*, Manner, R and Manderik B., editors, Elevier Science, 1992.
9. Haykin, Simon, *Neural Networks: A Comprehensive Foundation*, Macmillan, New Jersey, 1994.
10. Holland, J. H., *Adaptation in Natural and Artificial Systems*, MIT Press, 1970 / 1995.
11. Hull, John C., *Options Futures and Other Derivative Securities*, Second Ed., Prentice-Hall, New Jersey, 1993.
12. Kirman, Alan P. and Salmon, M., *Learning and Rationality in Economics*, Oxford, 1995.
13. March, James G., and Simon, Herbert A., *Organizations*, John Wiley & Sons, New York, 1958.
14. Radner, Roy, "Hierarchy: The Economics of Managing," *Journal of Economics Literature*, Vol. XXX, September 1992.
15. Ripley, Brian, D., *Stochastic Simulation*, John Wiley & Sons, New York, 1987.
16. Rubinstein, Reuven Y. *Monte Carlo Optimization, Simulation and Sensitivity of Queuing Networks*, John Wiley & Sons, New York, 1986.
17. Schrodt, R. A., "Predicting International Events," *Byte*, December, 1986.
18. Starbuck, William H., "Computer Simulation of Human Behavior," *Behavioral Science*, Vol. 28, 1983.
19. Tichy, Noel and Fombrun, Charles, "Network Analysis in Organizational Settings," *Human Relations*, Vol. 32, No. 11, 1979.
20. Tushman, Michael L., and Nadler, David A., "Information Processing as an Integrating Concept in Organizational Design," *Academy of Management Review*, July, 1978.
21. Wiliamson, Oliver E., "Transaction-Cost Economics: The Governance of Contractual Relations," *Journal of Law and Economics*, 1979.
22. Weber, Bruce W., "Bypass and Market Quality in Electronic Securities Exchanges," *Journal of Organizational Computing*, Special Issue on I.S. and Economics, 1994.
23. Weick, Karl, *The Social Psychology of Organizing*, Random House, New York, 1979.

Appendix A: A sample simulation using the OrgNet.cpp Library

This sample program demonstrates the organizational modeling library discussed in the paper. For clarity, the simulation module is not presented here. In place of the simulation, this sample program simply inserts data into nodes in the network by invoking the `AddToMemory()` method of the organization member. For brevity the GA is also not presented since its implementation is standard. For details on the GA see, for example, Goldberg (1989).

```
//
//This simulation sets up a two level three member organization and processes
//input information until the clock time exceeds TIME_OUT ticks  Members
//process inputs based on their computation functions.  In the example the two
//"STAFF" members simply gather and pass on inputs, while the manager member
//averages the inputs. The organization structure is a pure hierarchy.
//


#include <stdio.h>
#include "orgnet.hpp"

// ------------------

double linear(double *memory, int mem_size)
{
      // pass on contents of first memory location

      if(mem_size) return memory[0];
      else return -1;
}

// ------------------

double average(double *memory, int mem_size)
{
      //calculate arithmetic average of memory contents

      if(!mem_size) return -1;

      double value=0;
      for(int i=0; i<mem_size; i++) value+=memory[i];

      return (value/(double)memory_size);
}

// ------------------
```

```
void main()
{
      int TIME_OUT=6;
      int num_nodes = 3;

      Net org(num_nodes);
      Node *n;

      // Set up organization hierarchy
      //
      //                         MANAGER
      //                        /       \
      //                  STAFF1         STAFF2
      //

      org.connection->element(0,2)=1.0;
      org.access_time->element(0,2)=1.0;
      int a0[3]={2}; int na0=1;
      n=&(org.nodes->GetNode(0));
      n->SetName("STAFF 0");
      n->SetNode(10, 1,0.0, linear, &(org.msg_q),a0, na0);
      // mem, comp_time, st_time, func, msq_q, addresses, nadresss

      org.connection->element(1,2)=1.0;
      org.access_time->element(1,2)=1.0;
      int a1[3]={2}; int na1=1;
      n=&(org.nodes->GetNode(1));
      n->SetName("STAFF 1");
      n->SetNode(10, 1,0.0, linear, &(org.msg_q),a1, na1);
      // mem, comp_time, st_time, func, msq_q, addresses, nadresss

      int a2[3];  int na2=0;
      n=&(org.nodes->GetNode(2));
      n->SetName("MANAGER");
      n->SetNode(10, 1,0.0, average, &(org.msg_q),a2, na2);
      // mem, comp_time, st_time, func, msq_q, addresses, nadresss


      // Input some data

      n=&(org.nodes->GetNode(0));
      n->AddToMemory(0.0);

      n=&(org.nodes->GetNode(1));
      n->AddToMemory(1.0);

      // Start the organization

      org.clock=0;
      for(org.clock=0; org.clock<TIME_OUT; org.clock++)
      {
            printf("\n\n\n############## T=%i\n", org.clock);
            org.OutputCurrentState();
            org.ExecuteOneClockTick();
      }

      getchar();
}
```

Appendix B: Output of Sample Program

```
############## T=0
3 NODES AT TIME 0:

MESSAGES:
. Message queue is empty.

NODE STATES
0
======================================

(I)     STAFF 0  0.000  0:00
        s=1.000000      m=10
MEMORY
M:0.000000
------------
WORKING MEMORY
------------
ADDRESS LIST
TO:2

======================================

1
======================================

(I)     STAFF 1  0.000  0:00
        s=1.000000      m=10
MEMORY
M:1.000000
------------
WORKING MEMORY
------------
ADDRESS LIST
TO:2

======================================

2
======================================

(I)     MANAGER  0.000  0:00
        s=1.000000      m=10
MEMORY
------------
WORKING MEMORY
------------
ADDRESS LIST

======================================


CONNECTIONS
  0.00    0.00    1.00
  0.00    0.00    1.00
  0.00    0.00    0.00
ACCESS TIME
  0.00    0.00    1.00
  0.00    0.00    1.00
  0.00    0.00    0.00



############## T=1
3 NODES AT TIME 1:

MESSAGES:
Message queue is empty.

NODE STATES
0
======================================
(C)     STAFF 0  0.000  0:00
        s=1.000000      m=10
MEMORY
------------
WORKING MEMORY
WM:0.000000
------------
ADDRESS LIST
TO:2

======================================

1
======================================

(C)     STAFF 1  0.000  0:00
        s=1.000000      m=10
MEMORY
------------
WORKING MEMORY
WM:1.000000
------------
ADDRESS LIST
TO:2

======================================

2
======================================

(I)     MANAGER  0.000  0:00
        s=1.000000      m=10
MEMORY
------------
WORKING MEMORY
------------
ADDRESS LIST

======================================


CONNECTIONS
  0.00    0.00    1.00
  0.00    0.00    1.00
  0.00    0.00    0.00


############## T=2
3 NODES AT TIME 2:

MESSAGES:
#0      [0->2]  at -1 <0.000000>
#1      [1->2]  at -1 <1.000000>

NODE STATES
0
======================================

(I)     STAFF 0  0.000  0:00
        s=1.000000      m=10
MEMORY
------------
WORKING MEMORY
------------
ADDRESS LIST
TO:2

======================================

1
======================================
```

```
(I)     STAFF 1  1.000  0:00
        s=1.000000     m=10
MEMORY
------------
WORKING MEMORY
------------
ADDRESS LIST
TO:2

======================================

2
======================================

(I)     MANAGER  0.000  0:00
        s=1.000000     m=10
MEMORY
------------
WORKING MEMORY
------------
ADDRESS LIST

======================================


CONNECTIONS
  0.00    0.00    1.00
  0.00    0.00    1.00
  0.00    0.00    0.00

############## T=3
3 NODES AT TIME 3:

MESSAGES:
#0      [0->2]  at 3 <0.000000>
#1      [1->2]  at 3 <1.000000>

NODE STATES
0
======================================

(I)     STAFF 0  0.000  0:00
        s=1.000000     m=10
MEMORY
------------
WORKING MEMORY
------------
ADDRESS LIST
TO:2

======================================

1
======================================

(I)     STAFF 1  1.000  0:00
        s=1.000000     m=10
MEMORY
------------
WORKING MEMORY
------------
ADDRESS LIST
TO:2

======================================

2
======================================

(I)     MANAGER  0.000  0:00
        s=1.000000     m=10
MEMORY
------------
WORKING MEMORY
------------
ADDRESS LIST

======================================


CONNECTIONS
  0.00    0.00    1.00
  0.00    0.00    1.00
  0.00    0.00    0.00

############## T=4
3 NODES AT TIME 4:

MESSAGES:
Message queue is empty.

NODE STATES
0
======================================

(I)     STAFF 0  0.000  0:00
        s=1.000000     m=10
MEMORY
------------
WORKING MEMORY
------------
ADDRESS LIST
TO:2

======================================

1
======================================

(I)     STAFF 1  1.000  0:00
        s=1.000000     m=10
MEMORY
------------
WORKING MEMORY
------------
ADDRESS LIST
TO:2

======================================

2
======================================

(C)     MANAGER  0.000  0:00
        s=1.000000     m=10
MEMORY
------------
WORKING MEMORY
WM:1.000000
WM:0.000000
------------
ADDRESS LIST

======================================


CONNECTIONS
  0.00    0.00    1.00
  0.00    0.00    1.00
  0.00    0.00    0.00

############## T=5
3 NODES AT TIME 5:

MESSAGES:
Message queue is empty.

NODE STATES
0
======================================
```

```
(I)     STAFF 0  0.000  0:00
         s=1.000000      m=10
MEMORY
------------
WORKING MEMORY
------------
ADDRESS LIST
TO:2

======================================

1
======================================

(I)     STAFF 1  1.000  0:00
         s=1.000000      m=10
MEMORY
------------
WORKING MEMORY
------------
ADDRESS LIST
TO:2

======================================

2
======================================

(I)     MANAGER  0.500  0:00
         s=1.000000      m=10
MEMORY
------------
WORKING MEMORY
------------
ADDRESS LIST

======================================

CONNECTIONS
  0.00     0.00     1.00
  0.00     0.00     1.00
  0.00     0.00     0.00
```