An Organizational Approach to Planning in Distributed Problem Solving Systems

Daniel D. Corkill

Computer and Information Science
University of Massachusetts
Amherst, Massachusetts, 01003

COINS Technical Report 80-13

May 1980

ABSTRACT

The organizational structure of a distributed problem solving system is the pattern of information and control relationships that exist between the nodes in the system and the distribution of problem solving capabilities (expertise) among the nodes in the system. Organizational design is the explicit planning of these internode relationships.

This report discusses the need for organizational self-design in complex distributed problem solving systems and outlines the design of a prototypical distributed planning system which integrates both the coordination of "domain-level" activities and the construction and maintenance of an appropriate organizational structure.

This research is supported in part by National Science Foundation Grant MCS78-04212 and Office of Naval Research Grant N00014-79-C-0439. The views and conclusions contained in this document are the author's and should not be interpreted as representing the offical opinion or policy of the University of Massachusetts or any agency of the U.S. Government.

1.0 INTRODUCTION

How do intelligent individuals work together to achieve shared goals? Creating effective cooperative behavior within a society of interacting, semi-autonomous, intelligent systems is an important aspect of successful distributed problem solving. Planning cooperative behavior in distributed problem solving systems is the subject of this report.

1.1 Planning and Distributed Problem Solving Systems

Informally, a distributed problem solving system is a system which performs its problem solving activity in a logically or physically distributed fashion. Distributed problem solving systems are both complex and dynamic, and therefore require planning of their own activities. Planning, the generation and evaluation of alternative courses of action without actually executing them, is needed to allocate shared resources (such as communication media), to decide what is to be done and by which node, and to respond to internal and external environmental events (such as partial system failure and localized processing overloads). In addition, certain potential distributed problem solving applications (such as air traffic control and tasks involving mobile robots) are all basically planning applications.

Although centralized planning techniques can be appropriate in some distributed problem solving situations, they are basically incongruous with the characteristics of distributed systems. If planning is localized to a single node, environmental data needed for planning would have to be transmitted to that node, potentially requiring significant internode communication. Once a plan is generated, commands would have to be transmitted to appropriate nodes for execution, again potentially requiring significant internode communication. Planning localized to a single node makes loss of the planning node an important system reliability issue. The use of redundant planning nodes can improve reliability at the cost of further communication requirements.

A preliminary investigation of distributed planning [Corkill 1979] focused on the generalization of a centralized planning system (Sacerdoti's NOAH system [Sacerdoti 1977]) to accomodate multiple and distributed centers of planning control. A key assumption in this investigation was the existence of a mechanism for allocating planning activity to individual nodes. In fact, allocation of planning activity is part of the larger issue of determining an appropriate organizational structure for the particular problem solving situation.

The organizational structure of a distributed problem solving system is the pattern of information and control relationships that exist between the nodes in the system and the distribution of problem solving capabilities (expertise) among the nodes in the system. Organizational structures include hierarchies, heterarchies or flat structures, matrix organizations, groups or teams, and market or price systems. Organizational design is the

explicit planning of these internode relationships.

The organizational structure of a distributed problem solving system relates strongly to its effectiveness in a given problem solving situation. This effectiveness is a multi-valued measure incorporating such parameters as processing resources, communication requirements, timeliness of activity, accuracy of activity, etc. An organizational structure may lose effectiveness as the internal or external environment of the distributed system or the nature of the problem solving task changes. In order to respond to such a change, the distributed system must detect the decreased effectiveness of its organizational structure, determine plausible alternative structures, evaluate the cost of continuing with its current structure versus the cost of reorganizing itself into a more appropriate structure, and carry out such reorganization if appropriate. In general, these activities should be accomplished in a distributed fashion.

Organizational design decisions are faced regularly in human organizations, especially those in the business community where pressures of efficiency are most severe. Theories of organizational design which attempt to explain the art of organizational structuring in these human organizations are highly relevant to the development of organizational design knowledge for distributed problem solving systems.

The proposed research bypasses further refinement of domain-level planning techniques, such as those emphasized in distributing NOAH, in favor of the development of a framework which encompasses both domain-level planning and organizational design. Although many issues in distributed domain-level planning remain to be solved, the development of a distributed organizational design system seems the more salient research direction. A rephrasing of the programming adage:

"Don't optimize a bad algorithm -- rewrite it."

seems appropriate:

"Don't improve plans within a bad organizational structure -- reorganize."

1.2 The Organizational Structure of the Report

The following section serves as an introduction to the important characteristics of distributed problem solving systems and to the problems posed by these characteristics. In Section 3, a particular problem solving task is described: distributed traffic monitoring. Section 4 outlines the development of a prototypical distributed planning system for the distributed traffic monitoring task which addresses both the coordination of domain-level activities and the construction and maintenance of an appropriate organizational structure. Section 5 outlines a research strategy for developing the distributed organizational design system and delineates those aspects which are outside the scope of the proposed research. The potential contributions of

this research are also discussed.

2.0 AN INTRODUCTION TO DISTRIBUTED PROBLEM SOLVING SYSTEMS

The phrase "distributed system" invokes an intuitive feeling of the class of system under consideration. Notions of processors scattered about, interconnected in some fashion by cables or radio links, come readily to mind. Such physically distributed systems are now economically realistic due to developments in microprocessor technology [Noyce 1967; Scientific American 1977] and network technology [Cerf & Kahn 1974; Kimbleton & Schneider 1975]. While much of the original motivation for distributed problem solving system research has stemmed from the availiability of physically distributed architectures, it is important to emphasize that logically distributed systems embody many of the properties of their physically distributed cousins.

2.1 Important Characteristics of Distributed Systems

If mere physical distribution is not a suitable litmus for defining the class of distributed systems, what are the significant characteristics of these systems?

One significant characteristic is system decomposition. A distributed system is composed of a collection of modules or nodes and can be viewed from two perspectives [Lesser & Erman 1979, p. 557]. From a reductionist perspective, a distributed system is considered to be a centralized system that is decomposed over a number of nodes, each of which is a part in the overall system. From a constructionist perspective, a distributed system is synthesized from individual systems operating at each node. These two perspectives arise from the inherently hierarchic (in the systems-composed-of-systems sense of hierarchy [Simon 1969, p. 87]) nature of a distributed system. While both perspectives view the same reality, the reductionist viewpoint tends to encourage a search for appropriate ways of pulling apart existing centralized systems. The constructionist viewpoint tends to encourage a search for ways of organizing individually "complete" systems into a society of cooperating nodes. The constructionist viewpoint is taken in this report.

A second significant characteristic of distributed systems is restricted internode interaction. Interaction between nodes in a distributed system is expensive (in a cost/benefit sense) relative to local computation. In physically distributed systems this expense can arise from a number of factors:

- Internode communication is limited by the physical properties of the communication channel(s). This often results in a cost which grows with the amount of information to be communicated and the distance between interacting nodes. Due to communication channel management issues, information which is transferred in large units may be more cost effective than the same

information transferred in small units.

- Internode communication may have significant time delays in transmission. Such delays arise from propagation delays in the communication media, queueing delays for accessing the communication channel, and buffering delays in forming larger units of information from smaller units.

- Internode communication may be incorrectly exchanged or lost altogether. Errors in communication can result from encoding/decoding errors and errors during transmission. Loss can result from hardware failure in communication channels or processors (nodes).

In logically distributed systems this expense can arise from the bounded rationality [Simon 1957] of programming modules. Bounded rationality limits the amount of information which can be absorbed by a programming module and the amount of control which a module can exercise. Processing resources are required to respond to an incoming message or to construct an outgoing message. Because processing resources are limited, a module sending or receiving many messages spends significant processing resources in communication that would otherwise be available for problem solving activities. As the number of interacting modules becomes larger, increasingly general communication protocols may be required to handle all types of intermodule messages. Such generality comes at the cost of increasingly complex module interfaces which require additional processing resources and at the cost of additional complexity to be handled by the system designers.

Physically distributed systems are also logically distributed (the two distributions need not coincide) and incur these same logical interaction expenses.

The effect of the high cost of internode interaction in distributed systems is to force a restriction of the view an individual node has of the state of the rest of the system. This restriction can be handled in several ways:

- By decomposing the system task in such a way that a node usually has in its local database the information required to complete its processing. External information required by a node is assumed to be relatively small in quantity.

- By providing a node with algorithms which can perform significant processing despite the presence of incomplete and abstracted information in its local database.

Conventional distributed system design methodologies have concentrated on the first approach to handling restricted local views. The distributed system is organized so that each node's local database contains an exact copy of the appropriate portions of the overall problem solving database. In these systems, a node rarely needs the assistance of another node in carrying out its problem solving activities. Only when information is not locally available does node interaction occur. This interaction takes the form of a request for information from another node which is returned as a complete and correct result. This type of distributed approach has been termed completely-accurate, nearly-autonomous (CA/NA) [Lesser & Corkill 1980] because of the maintenance of complete and correct information ("completely-accurate") and the emphasis on unassisted local processing ("nearly-autonomous").

While the CA/NA approach is well suited to conventional distributed system applications, a number of important potential applications for physically distributed systems cannot be partitioned effectively due to the physical distribution of data and the type of processing required at each node. In these situations, the CA/NA approach is expensive (this time in monetary terms) because of the amount of communication and synchronization required to guarantee completeness and consistency of the local databases. The CA/NA approach can also be expensive in logically distributed systems when the number of modules and their interactions become so complex that it is conceptually difficult to design and computationally expensive to maintain a complete and consistent intermodule interaction structure.

An alternative and relatively new type of distributed system uses the second approach to handling restricted local views. In these systems, a node's local database is not required to be either complete or consistent with the databases of other nodes in order for the node to function. Because a node's algorithms and control structures operate on information which may be incomplete, inconsistent, and even incorrect, these systems are inherently more complex than CA/NA distributed systems. Since information may be based on processing which used uncertain data, an iterative, coroutine type of node interaction in which revised decisions are periodically exchanged is used to problem solve. This type of system has been termed functionally-accurate, cooperative (FA/C) [Lesser & Corkill 1980]. "Functionally-accurate" refers to the requirement that the system exhibit accurate input/output behavior without requiring all intermediate aspects of the computation to be correct and consistent. "Cooperative" refers to the property that nodes in the system form a cooperative network to achieve the overall system task.

Networks of cooperating nodes are not new to Artifical Intelligence (AI) research. The ACTOR formalism of Hewitt [1977] and the BEINGS system of Lenat [1975] are examples of the "cooperating experts" problem solving paradigm. In this paradigm a system is composed of a network of communicating problem solving "experts". FA/C distributed systems extend the "cooperating experts" paradigm (as implemented to date) in two important

directions: intelligence and motivation.

In "cooperating experts" systems, knowledge is distributed among the "experts" to the degree that each "expert" becomes a specialist in one particular aspect of the overall problem solving task. An "expert" has little or no knowledge of the problem solving task as a whole or of general techniques for communicating and cooperating with the rest of the system. As a result, an "expert" cannot function outside the context of the other "experts" in the system nor outside the communication and cooperation protocols specified (in advance) by the system designer.

A node in an FA/C distributed system is assumed to possess sufficient overall problem solving knowledge that its particular expertise (resulting from either specialist knowledge or a unique perspective of the problem solving situation) can be applied and communicated outside the context of the other nodes in the system. This does not imply that a node functions as well alone as when cooperating with other nodes -- internode cooperation is often the only way of developing an accurate solution -- but the node can at least formulate a solution using only its own knowledge.

A node in an FA/C system also possesses significant expertise in communication and control. This knowledge frees the system from the bounds of designed protocols and places its nodes in the situation of developing their own communication and cooperation strategies [Nilsson 1980, p. 419].

The second significant difference between cooperating "experts" and FA/C distributed systems is motivation. Simply stated, cooperating "experts" are message-driven in their behavior. The "expert" awaits receipt of a message, performs activites based upon that message, communicates results of those activites, and awaits receipt of a new message. A node in an FA/C distributed system is self-directed in its activity. For instance, if a node does not receive an appropriate piece of information from another node, it is able to continue processing using whatever data are available at that time. As will be discussed in Section 4.2, a node can choose whether to engage in activities suggested by other nodes or pursue its own "best interests." The importance of such "self-motivation" in program modules has been suggested by Fox [1979a, p. 359].

The result of these differences is that FA/C distributed systems can resolve uncertainty and error in data, communication, control, or algorithm as an integral part of their problem solving activities. This capability is especially important in a distributed environment where uncertainty and error resulting from the characteristics of the distributed system can be significant. In fact, additional mechanisms traditionally required to handle hardware, communication, and processing errors may be superfluous, given that uncertainty resolving mechanisms are already a part of the distributed system architecture [Lesser & Corkill 1979, 1980].

The cooperating "experts" paradigm remains an important and useful design and programming methodology for constructing certain AI and other systems. In fact, a cooperating "experts" system might be an appropriate component of a node in a FA/C distributed system. However, as a model for a society of communicating experts, and in particular as a model for a distributed system, the sophistication of the "experts" appears far too low. Crane [1978, p. 48] in describing the cooperating "experts" approach notes:

> The effort is an attempt to develop a formal programming system by means of which effective "message passing" can be accomplished among a network of "actors." Whereas we seem to interact among ourselves so easily, such formal attempts to imitate this interaction illustrate just how sophisticated and complex our communication actually is, and that the evolution of comparably sophisticated artificial intelligence systems will likely be a long and difficult "incremental and evolutionary process."

We argue that the FA/C distributed system model is an important "mutation" in that evolution.

In review, an FA/C distributed system is composed of a collection of semi-autonomous nodes in which:

- internode interaction is expensive relative to internal processing;
- each node is capable of functioning with incomplete and inaccurate information about its environment and the state of the system;
- each node is capable of formulating a solution to the problem outside the context of other nodes in the system;
- each node is self-directed;
- errors are resolved as an integral part of problem solving activity.

2.2 The Structure of a Problem Solving System

Now that we have discussed the characteristics of distributed problem solving systems motivated by their distribution, it is appropriate to look at the characteristics motivated by the tasks which these systems are to perform. These tasks are all considered problem solving tasks and therefore the systems are termed problem solving systems. We first consider the characteristics of a centralized problem solving system.

A problem solving system is composed of three components:

- o sensors -- provide raw data about the system's environment (both external and internal);
- o effectors -- permit modification of the environment;
- o a problem solver -- uses the sensors and effectors to achieve or maintain desired goals.

A simple example of a problem solving system is a hand-eye robot which manipulates blocks on a tabletop. This system could have as sensors: a stationary video camera for viewing the tabletop, a keyboard for accepting the demands of researchers, and a thermocouple for determining internal processor temperature. As effectors, the system could have: a "hand" which can manipulate single blocks anywhere on the tabletop, a printer for complaining to researchers, and a fan which can be switched on if processor temperature rises alarmingly high. These components form a sense-decide-act cycle with the environment as illustated in Figure 1a (liberally adapted from Arbib [1972] p. 19).

The problem solving component has three main activities (Figure 1b):

- o interpretation -- developing and maintaining a model of the environment;
- o planning -- developing and evaluating alternative courses of action (plans) without actually executing them;
- o plan execution -- using effectors to apply a plan to the environment.

In the hand-eye system, interpretation would involve hypothesizing the initial positions of the blocks on the tabletop (environmental model generation) and watching to see that the hand was achieving the desired goals during plan execution (execution monitoring). Planning would involve hypothesizing a sequence of stack/unstack operations for the hand to perform (plan generation) and revising the remainder of a plan as surprises are discovered during plan execution (plan revision). Plan execution would involve the hand movement details required to pick up blocks and place them at locations specified in the plan.

2.3 The Structure of a Distributed Problem Solving System

From the constructionist viewpoint discussed in Section 2.1, a distributed problem solving system consists of a number of distributed problem solving nodes, each node having sensing, effecting, and problem solving capabilities. Continuing with the hand-eye example, a distributed two-hand problem solving system might consist of two of the single-arm, single-camera systems sharing the tabletop and linked by low-bandwidth communication.

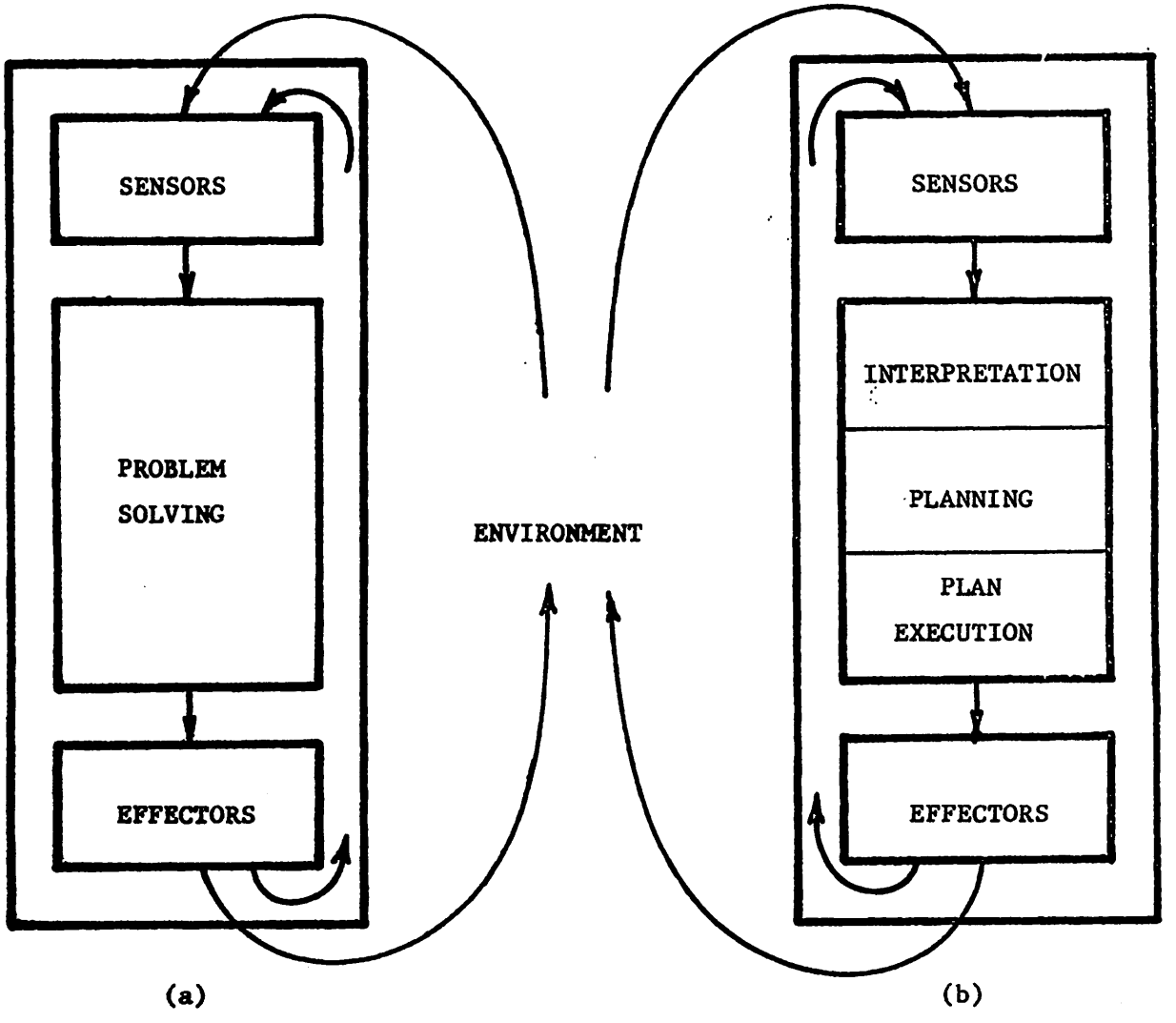

FIGURE 1: Problem Solving System Structure

Once a second hand-eye system has access to the tabletop, numerous interesting problems arise. Because each system views the tabletop via a different camera, multiple and possibly inconsistent interpretations (environmental models) are possible. Given identical goals to achieve, the two systems could potentially generate incompatible plans which, if executed, would interfere with one another through conflicts in block usages. During plan execution, the two systems must avoid crashing their arms together as they perform their various activities. Corkill [1979] discusses a number of additional issues in the distributed planning and execution of simple blocks world problems. Dealing with such domain-level issues is only one aspect of planning activity in a distributed problem solving system.

A more difficult problem is determining which node should perform what activity (what Smith [1978, p. 5] calls the "connection problem") and how that activity relates to the activities of other nodes. An important characteristic of a distributed problem solving system is that only sensing and effecting are ever fixed with respect to individual nodes. Problem solving can be performed at any node which has sufficent physical resources (processing capacity, memory), procedural resources (problem solving tactics [Sacerdoti 1979] and knowledge), and communication capability. This flexibility in problem solving locale leads to a wide range of potential distributed problem solving structures.

Localized Problem Solving. In localized problem solving, only sensing and effecting are distributed. Problem solving (interpretation, planning, and plan execution) is localized to a single node which receives sensory data from all nodes and transmits effector commands to appropriate nodes. This problem solving organization is illustrated in Figure 2.

Advantages:

- Centralized problem solving techniques can be used without modification.
- There is no internode communication cost to the problem solving process itself.

Disadvantages:

- Environmental information collected by the various nodes must be transmitted to the problem solving node.
- Effector commands must be distributed to appropriate nodes.

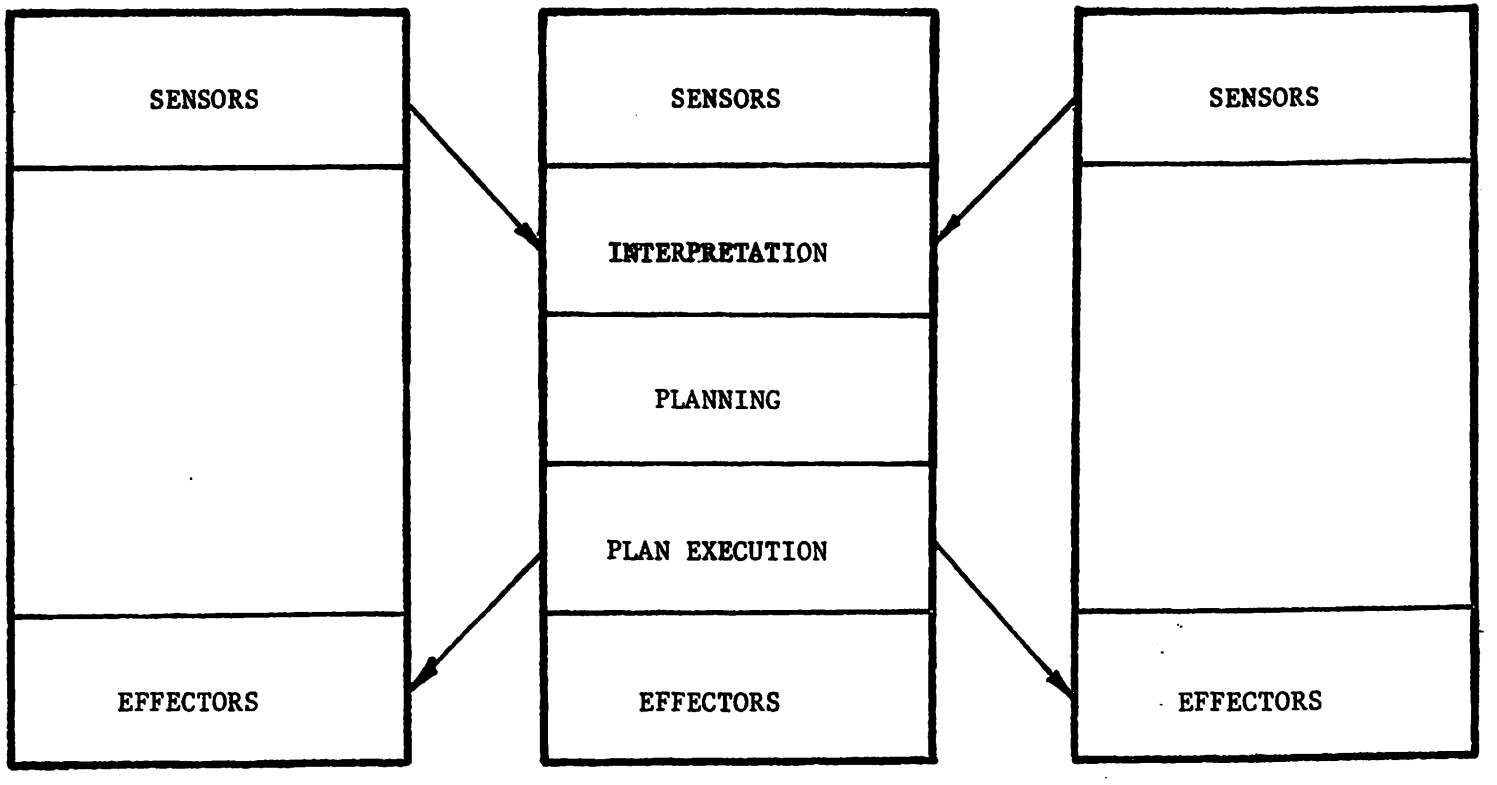

FIGURE 2: Localized Problem Solving

- o There is no internode parallelism during the problem solving process.
- o Performing all problem solving at a single node may require substantial processing capability, forcing the need for non-uniform processing capabilities in the distributed system.
- o Problem solving node failure means loss of all problem solving activity. (Redundant problem solving at other nodes further increases the communication demands on the system.)

Completely Distributed Problem Solving. In completely distributed problem solving, all aspects of the system (sensing, interpretation, planning, plan execution, and effecting) are distributed. This problem solving organization is illustrated in Figure 3.

Advantages:

- o Communication required for interpretation is reduced because some environmental data is available locally and because problem solving can proceed using an appropriate partial environmental model.
- o Communication required to distribute the plan for execution is reduced because a large portion of the generated plan may be executed locally.
- o There is the potential for internode parallelism in the problem solving process.
- o Local nodes only solve a portion of the problem, a potentially less complex and processing intensive task than solving the complete problem.
- o Failing processors only mean the loss of a portion of the problem solving.

Disadvantages:

- o FA/C techniques for coordinating this type of problem solving in distributed environments must be developed.
- o Communication is required to perform distributed problem solving.

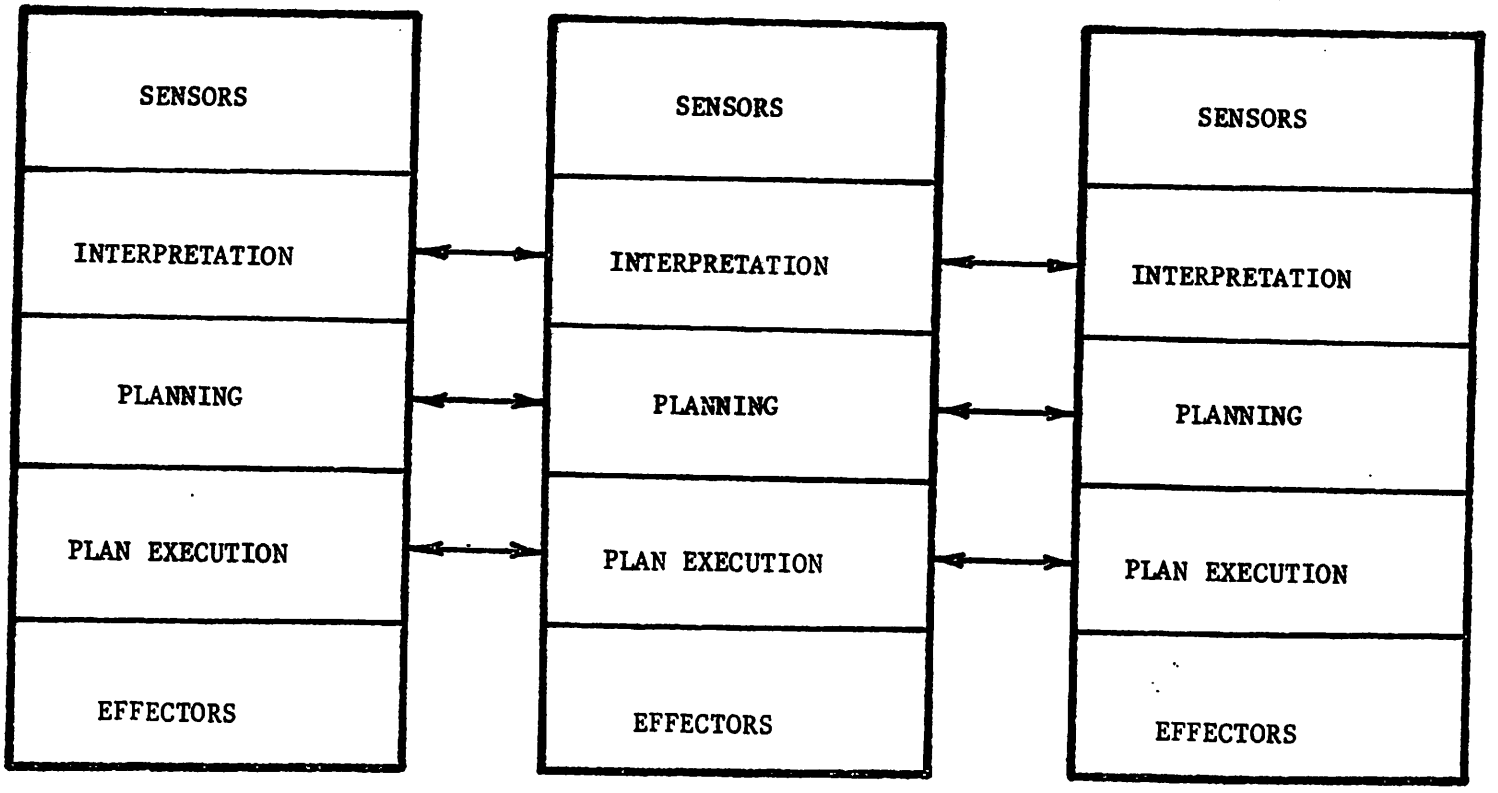

FIGURE 3: Completely Distributed Problem Solving

Of course a range of structures between localized problem solving and completely distributed problem solving are possible. By parameterizing the ratio of internode/intranode communication paths shown in Figure 4, the entire range of problem solving structures can be subsumed into one model. But which parameterization is best suited to a particular problem solving situation? This is but another difficulty in planning activity in a distributed problem solving system.

2.4 A Definition of a Distributed Problem Solving System

We have outlined the characteristics of distributed systems and have introduced the FA/C approach to the design of distributed systems. We also briefly discussed the structure of a distributed problem solving system in terms of its sensors, problem solver, and effectors. So what is a distributed problem solving system?

A distributed problem solving system, in our view, is a network of fully-functional problem solving systems (nodes), each of which contains a problem solver component which is FA/C in character. In addition, each node is capable of controlling the ratio of internode/intranode communication associated with any of its problem solving activities.

In the following section, a specific FA/C distributed problem solving architecture is introduced in the context of a more naturally distributed application than the blocks world: distributed traffic monitoring. This architecture will serve as the environment for the prototypical distributed planning system outlined in Section 4.

3.0 DISTRIBUTED TRAFFIC MONITORING

Distributed traffic monitoring is the task of generating and maintaining a model of traffic in an environment using a distributed problem solving system. The monitoring system we will consider is to locate, track, and identify vehicles and patterns of vehicles moving within a two-dimensional area. A number of processing nodes are located throughout this area, and each processor is linked to a set of (possibly non-uniform) acoustic sensors. As a vehicle moves through the monitoring area, it generates characteristic acoustic signals. Some of these signals are detected by nearby sensors which indicate the frequency and rough location of the signal source. Smith [1978] used a similar distributed vehicle monitoring task in illustrating the contract net formalism for distributed problem solving.

The traffic monitoring domain is interesting for several reasons:

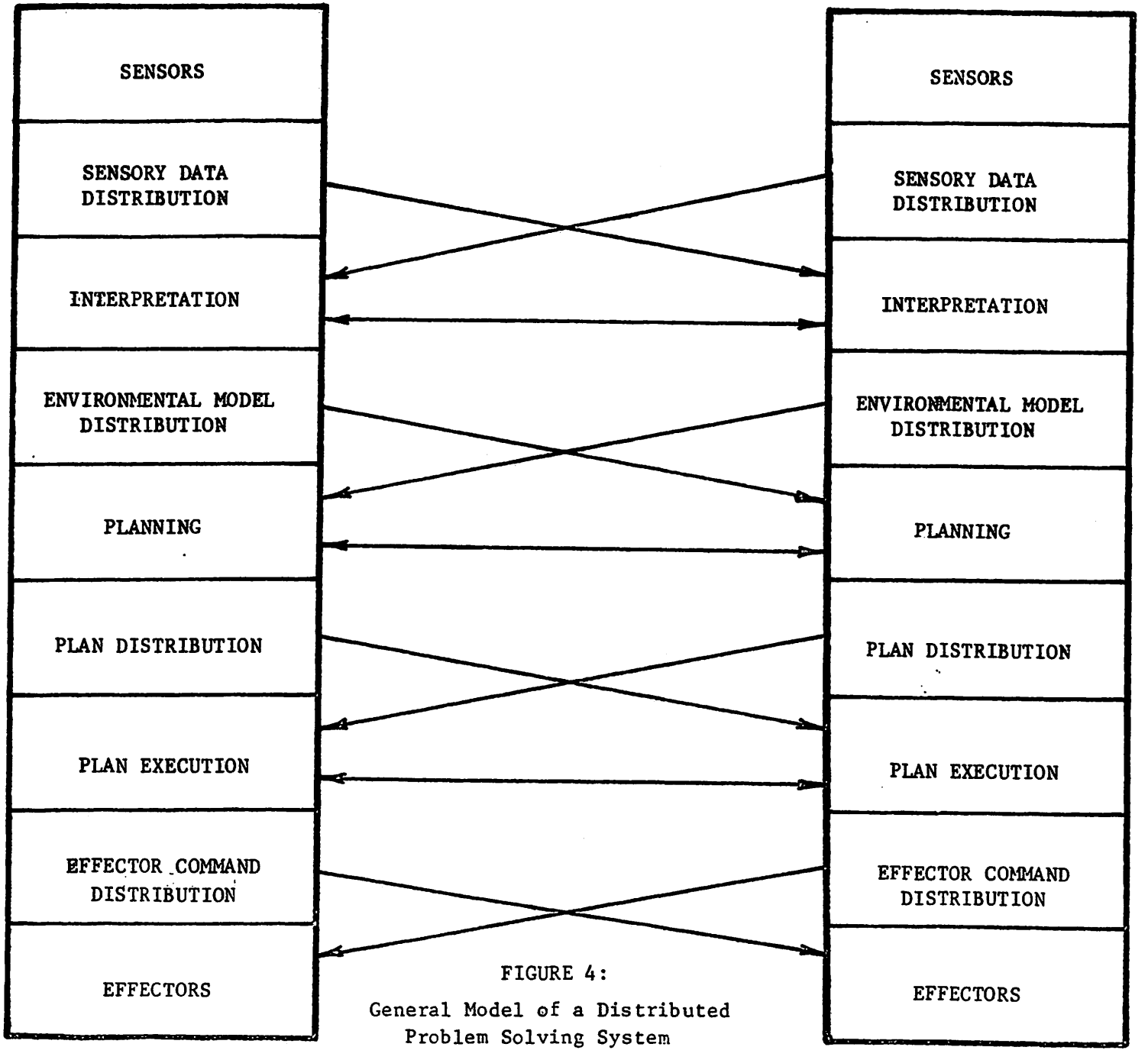

FIGURE 4:

General Model of a Distributed Problem Solving System

- It is a natural task for a distributed approach, since sensors are geographically distributed.
- Internode cooperation is required to solve the problem.
- The task can be easily varied in terms of signal complexity, temporal and spatial constraints on possible interpretations, and node-sensor configurations.
- Signals, harmonically related signals, vehicles, and movement patterns of vehicles offer a natural decomposition into disjoint levels of abstraction, with independent processing possible at each level.
- Multiple, diverse sources of knowledge must be used in order to resolve errors in the sensory data (a harmonic set formation expert, a signal tracking expert, etc.).

An FA/C distributed problem solving architecture for the traffic monitoring task is currently under development [Lesser, Reed, & Pavlin 1980]. This architecture is a generalization of the Distributed Hearsay-II architecture used by Lesser and Erman [1979] in their pilot distributed interpretation experiments.

Each node in this system is an architecturally complete Hearsay-II system [1]. "Architecturally complete" means that each node could function as a complete Hearsay-II system if it were given all of the sensory data and the required processing routines (knowledge-sources).

Two levels of planning can be indentified in the distributed traffic monitoring system. At the organizational-level, planning involves the design and construction of an appropriate organizational structure. This includes decomposing the task for distributed execution, determining an appropriate organizational structure for achieving the task, assigning individual tasks to nodes (what Smith calls the "connection problem" [Smith 1978]), allocating communication media and other shared system resources,

---

[1] "Hearsay-II" refers to the generic problem solving architecture developed as part of the Speech Understanding Project at Carnegie-Mellon University [Erman & Lesser 1975; Lesser & Erman 1977; Erman, et al. 1980] and not to its widely-known speech understanding incarnation (sometimes called "the C2 configuration"). The generic Hearsay-II architecture has been utilized in such diverse applications as protein-crystallographic analysis [Englemore & Nii 1977], image understanding [Hanson & Riseman 1978], multisensor interpretation [Nii & Feigenbaum 1978], dialogue comprehension [Mann 1979], complex learning [Soloway & Riseman 1977], and as a model of human reading [Rumelhart 1976].

and monitoring and possibly revising the structure of the organization as the task progresses. This type of planning is sometimes termed "strategic" planning.

Within the organizational structure planned at the organizational level, domain-level planning is performed. Domain-level planning involves the achievement of specific vehicle monitoring task goals and the coordination of activities within the organizational structure. This type of planning is sometimes termed "tactical" planning and involves issues of plan generation, execution monitoring, and plan revision.

An organizational-level planner will require knowledge about the suitability of various organizational structures to particular distributed problem solving scenarios. This knowledge can be developed from two sources. One source of organizational design knowledge is the operationalization of design theories "borrowed" from the field of Managment Science. The second source is an empirical evaluation of various (static) organizational structures using the distributed traffic monitoring system. Synthesis of organizational design theories (from the first source) and empirical data (from the second source) should lead to organizational design methodologies for the distributed traffic monitoring system and eventually to formal models of organizational design for distributed systems.

4.0 A PROTOTYPE DISTRIBUTED PLANNING SYSTEM

This section discusses the design of a prototype distributed planning system for the vehicle monitoring task which integrates planning at both the organizational and domain levels.

4.1 A Goal-Directed Hearsay-II Architecture

Although the data-directed behavior of the original Hearsay-II architecture has many advantages, it is severely limited in its ability to plan its interpretation activities. The scheduler (and a portion of the blackboard monitor) could be considered as a rudimentary planner which decides which knowledge sources (KSs) should be executed (and in what order) to best achieve the system goal of generating the most credible interpretation of the input data.

When a hypothesis is created or changed on the blackboard a number of KSs are added to the scheduling queue, a list of pending KS executions. Such a blackboard modification is called a blackboard event, and the KSs which are added to the scheduling queue are determined by the characteristics of that blackboard event. An evaluation function is used by the scheduler to rank the pending KS executions. This function uses a model of the current state of system processing and a model of the expected effects of executing each pending KS execution.

There are two major limitations to this purely data-directed approach. First, scheduling decisions are based on an instantaneous evaluation of the improvement a particular pending KS execution would make in the current state of processing. Scheduling has no continuity of purpose or high-level view of the reasons for executing a particular pending KS execution. As a result, many potentially superior KS execution orderings can be missed by the scheduler.

Second, the scheduler cannot choose to execute a KS on its own initiative, but must await data-directed activity to place the desired KS on the scheduling queue. Consider, for example, a pending KS execution that requires the existence of a particular type of hypothesis (a precondition hypothesis) before it can be released for execution. If that precondition hypothesis does not exist on the blackboard, either the pending KS execution must await data-directed activity to (hopefully) create the precondition hypothesis or the pending KS execution can be discarded to be (hopefully) recreated as a result of the creation of the precondition hypothesis. The first approach was used in the "problems list" of SU/X and SU/P [Nii & Feigenbaum 1978] and the second approach used in the C2 speech understanding configuration of Hearsay-II [Lesser & Erman 1977].

Clearly, what is needed is an integration of data-directed and goal-directed behavior [Lesser & Erman 1977, p.796; Nii & Feigenbaum, p. 500]. To achieve such an integration, we add to the original Hearsay-II blackboard (which will now be called the domain blackboard) a second blackboard which contains the same levels and dimensions as the domain blackboard. This new blackboard is the planning blackboard and contains data structures representing problem solving goals, plans for achieving these goals, and the scheduling queue.

Rather than adding pending KS executions to the scheduling queue, the blackboard monitor places goal/KS-action structures onto the planning blackboard. These structures explicitly represent the intended effects of each pending KS execution. In the original data-directed Hearsay-II architecture these effects were implicitly modelled. A separate set of KSs, collectively called the domain-level planner, reacts to modifications on the planning blackboard. The domain-level planner performs such activities as goal decomposition (subgoaling), planning for the achievement of precondition hypotheses (without awaiting data-directed hypothesis creation), merging of goals, generating alternative plans, etc.

The scheduling queue in the goal-directed Hearsay-II architecture is an ordering of KS actions on the planning blackboard. Because the actions are explicitly linked to the goals which they are intended to achieve, the scheduler can make a more informed decision about the execution sequence of KS actions. Additional details of the goal-directed Hearsay-II architecture are discussed in Corkill & Lesser [1980].

The addition of the planning blackboard and domain-level planner to the original Hearsay-II architecture has an added significance in distributed systems consisting of a number of architecturally complete Hearsay-II nodes. Goals provide an additional means of internode communication and control. Goals can be transmitted to other nodes for achievement, and incoming goals can be placed onto the planning blackboard to compete with locally generated goals.

Decisions of what goals to work on, whether to favor internally versus externaly generated goals, and which goals to transmit to what other nodes are organizational design issues. In the next section, we introduce an organizational-level planner into the extended Hearsay-II architecture.

4.2 An Organizational Planning System

Now that a second blackboard (the planning blackboard) has been introduced, a third called the organizational blackboard is added. The organizational blackboard contains goals which direct the activities of the domain-level planner and, indirectly, the basic interpretation system.

Goals on the organizational blackboard are manipulated by the organizational-level planner. This planner is charged with the development and maintenance of effective cooperative relationships with other nodes for the purposes of achieving planning-level goals perceived as important by the node.

Recall that planning-level goals can be generated both internally and externally. In order to process an externally generated goal in favor of an internally generated one, the node must "believe" that the externally generated goal is more important. Nodes are basically egocentric, but all nodes share the goal of accomplishing the interpretation task in the most effective manner. A node is ready to change the ratings of its (internally and externally generated) goals if "convinced" that such a change will improve system performance.

The degree to which a node needs to be convinced in order to change its goal ratings places that node at a particular point on the self-directed/message-directed continuum. A completely self-directed node pursues only those goals which can be justified by its own knowledge of the problem solving situation. A completely message-directed node pursues only those goals which are deemed important by other nodes. The message-directed approach has been advocated by Hewitt [1977], Feldman [1979], and Smith [1978].

The approach taken in this research is that neither extreme is well-suited to the uncertain environment of a distributed problem solving system. The self-directed extreme does not allow node activity to be determined by what other nodes know about the problem solving situation (e.g., "If you would do GOAL, it would really help us out."). The message-directed extreme does not allow a node to ignore requests for activities which it knows are

not appropriate (e.g., "I know doing GOAL for you won't help us at all, so I'll ignore it.") or notify the organization of that fact (e.g., "I'm telling you that we shouldn't pursue GOAL { because ... }.").

A degree of skepticism on the part of a node allows it to continue work on goals which are highly justified on the basis of local knowledge despite reception of goals for which there is negative local evidence. This skepticism can lead to an increase in the system's ability to tolerate some error in control (goal specification). Errorful or less-than-desirable goals can be ignored by skeptical nodes which possess information to the contrary. A node with a unique perspective is not necessarily stifled by an uninformed majority. The degree of skepticism exhibited by a node should dynamically change according the node's certainty as to the system importance of its own goals: as the certainty of a node's own goals decreases, it should become more receptive to externally generated goals; as a node becomes convinced of its approach, it should become more skeptical of conflicting external goals.

In distributed problem solving systems composed of large numbers of nodes, the existence of idiosyncratic degrees of skepticism can further increase the robustness of the system. In situations where there exist two competing approaches (one advocated by the organization and one apparent to some of its members) individual variances in node skepticism will insure that both approaches are pursued by the organization. The approach apparent to a portion of the node population is implicitly pursued, without the cost of making an explicit organizational decision. Of course this robustness comes at the price of uncontrolled expenditure of resources by the skeptical nodes. Reed and Lesser [1980] describe an analogous thresholding behavior theory for the selection of activity by honey bees.

The notion of skeptical nodes also has an analog in a relatively new approach to understanding the motivation of individuals in business organizations. Frequently called expectancy theory, this approach is based on four assumptions [Nadler & Lawler 1977]:

- Behavior is determined by a combination of forces in the individual and forces in the work environment -- different work environments tend to produce different behavior in similar individuals just as dissimilar individuals tend to behave differently in similar environments.

- People make decisions about their own behavior in organizations -- most of the behavior of individuals in organizations is the result of individuals' conscious decisions.

- Different individuals have different needs, desires, and goals -- these differences are not random but are based on the kinds of outcomes desired by the individuals.

- Individuals make decisions among alternative plans of behavior based on their perceptions (expectations) of the degree to which a given behavior will lead to desired outcomes.

Crane [1978, p. 104], in discussing biological and societal systems, considers the appropriateness of dual-directed control in which a balance between top-down (message-directed) processes and bottom-up (self-directed) processes are maintained. The skepticism of nodes is a mechanism which can provide such a balance. Interestingly, Crane also discusses dual-directed control in the context of the individual as a balance between conscious and subconscious processes. It may be reasonable to view goal-directed (conscious) and data-directed (subconscious) behavior as analogous dual-directed control at the level of a single system node.

Historically, the notion of decision-making using a combination of locally perceived information and decisions received from others goes back at least as far as the Kilmer, McCulloch, and Blum [1969] model of the reticular formation (an important control system in the brains of vertebrates). Their model consists of a voting system of small neural regions, each of which sees a small portion of the environment and receives the decisions of a limited number of other regions. The voting strength of each region varies with an evaluation of its decision-making performance. In their model, activity was selected only when a suitable majority of the regions arrived at a concensus.

The skepticism scheme extends this notion to allow node activity to occur without such a consensus. When there is no agreement, the organization pursues a number of activities (similar to a breadth-first search) implicitly and in parallel. As the number of agreeing nodes increases, organizational activity becomes both more specific and more explicitly controlled.

4.3 Beliefs and the Organizational Planning System

Each of the three blackboards (domain, planning, and organizational) is partitioned into a self-image and numerous peer-images. The self-image partition contains all of the blackboard data structures described thus far. Peer-image partitions represent the beliefs a node has about the domain, planning, and organizational information possessed by other nodes. Such beliefs can be expected to be more detailed in the peer-images corresponding to closely interacting nodes than in models of nodes with little or no interaction.

The use of beliefs is receiving much attention in the language understanding community. One example is the use of beliefs by Cohen [1978] and Allen [1979] in computer implementations of a model of Speech Acts [Searle 1969]. Cohen's work focused on the planning side of Speech Acts, the planning of what to say to achieve particular communication goals, and Allen's work focused on the recognition of the intended goals by the listener.

Their research is directly relevant to distributed problem solving systems in which communication is relatively expensive compared to computation. Speech Acts can provide a parsimonious high-level internode communication language [Sacerdoti 1978] by restricting the scope of the Acts to the goals and processes of the distributed problem solving system. Because communication relates to the problem solving task, much of the planning activity required for message generation and recognition already exists on the blackboards of the proposed problem solving architecture.

The domain blackboard portion of a peer-image contains beliefs about the modelled node's developing map of vehicle pattern movements. These beliefs are derived from communicated information and inferred from the behavior of the node and that node's role in the problem solving system (as perceived by the modelling node). Domain-level hypotheses and beliefs allow a node to decide who might be appropriate to ask for a particular piece of information. (Belief that a node knows a piece of information does not require a belief as to its content.) Such a decision avoids the need for announcing a general request for the information.

The planning blackboard portion of a peer-image contains beliefs about the goals and planned activities of the modelled node. These beliefs are also both derived and inferred. Planning-level hypotheses and beliefs allow a node to decide who should receive locally generated information based on the perceived goals and activities of other nodes. Such a decision avoids the need for the other node to request that the information be communicated.

The organizational blackboard portion of a peer-image contains beliefs about the organizational goals and relationships of the modelled node. Organizational-level information and beliefs allow a node to decide which tasks it should perform and which tasks will be performed by others.

In summary, domain-level beliefs describe <u>what</u> a node knows about the problem. Planning-level beliefs describe <u>how</u> that information was/is/will be used. Organizational-level beliefs describe <u>why</u> that information (and processing) is important to the overall problem solving task.

5.0 THE STUDY

5.1 Research Methodology

Development of an organizational planner for the vehicle monitoring system must begin with the design and implementation of the goal-directed Hearsay-II architecture. The architectural design is well underway [Corkill & Lesser 1980]. A part of this work is the delineation of the types of domain-level goals in the traffic monitoring domain.

A second activity is understanding the formal and informal aspects of organizational design from the perspective of Management Science. Much of this knowledge appears descriptive rather than prescriptive in character. Fox [1979a, b] was successful at transferring certain organizational notions to the design of complex software systems by human programmers.

As the distributed vehicle monitoring system becomes operational, evaluation of particular (static) organizational structures in the context of various task situations will be performed. This evaluation will require the development of organizational effectiveness measures (which are needed for the organizational-level planner as well).

Synthesis of these empirical evaluations and ideas gleaned from organizational design theories is expected to produce sufficient organizational design knowledge for the construction of the organizational-level planner. It is anticipated that the organizational-level planner itself is well within state-of-the-art planning technology. An effective domain-level planner, however, appears to require significant developments in distributed planning methodologies and will be considered outside the scope of the proposed research.

5.2 Potential Contributions of this Research

What are the potential contributions of this research? Perhaps the most significant contribution would be demonstration of both the importance and tractability of organizational planning in distributed problem solving systems via performance comparisons with static organizational structures.

A second contribution would be the detailing of a distributed problem solving architecture which integrates domain-level problem solving, domain-level planning, and organizational-level design.

A third contribution would be the development of an organizational design knowledge base for the distributed vehicle monitoring system. While some of this knowledge would be specific to the monitoring task, it is reasonable to expect that general principles for organizational planning in distributed systems would be brought to light. Such general principles would be of interest in future work on the development of formal models for organizational design in both computer science and management

science.

ACKNOWLEDGMENTS

Many of the ideas presented in this report evolved during discussions with Victor Lesser. Michael Arbib, Victor Lesser, Jasmina Pavlin, and Jack Wilden provided helpful criticism of preliminary drafts.

REFERENCES

[Allen 1979]
James F. Allen. "A Plan-Based Approach to Speech Act Recognition." Ph.D. Thesis, Technical Report 131/79, Department of Computer Science, University of Toronto, Toronto, Canada (January 1979).

[Arbib 1972]
Michael A. Arbib. The Metaphorical Brain, Wiley-Interscience, New York, New York (1972).

[Cerf & Kahn 1974]
Vinton G. Cerf and Robert E. Kahn. "A Protocol for Packet Network Intercommunication." IEEE Transactions on Communication, Vol. COM-22, No. 5 (May 1974), p. 637-648.

[Cohen 1978]
Phillip R. Cohen. "On Knowing What to Say: Planning Speech Acts." Ph.D. Thesis, Technical Report 118, Department of Computer Science, University of Toronto, Toronto, Canada (January 1978).

[Corkill 1979]
Daniel D. Corkill. "Hierarchical Planning in a Distributed Environment." Proceedings of the Sixth International Joint Conference on Artificial Intelligence, Tokyo, Japan (August 1979), p. 168-175.

[Corkill & Lesser 1980]
Daniel D. Corkill and Victor R. Lesser. "A Goal-Directed Hearsay-II Architecture." Technical Report, Department of Computer and Information Science, University of Massachusetts, Amherst, Massachussets (in preparation).

[Crane 1978]
Hewitt D. Crane. "Beyond the Seventh Synapse: The Neural Marketplace of the Mind." Research Memorandum, SRI International, Menlo Park, California (December 1978).

[Englemore & Nii 1977]
Robert S. Englemore and H. Penny Nii. "A Knowledge Based System for the Interpretation of Protein X-Ray Crystallographic Data." Technical Report Stan-CS-77-589, Stanford University, Stanford, California (February 1977).

[Erman & Lesser 1975]
Lee D. Erman and Victor R. Lesser. "A Multi-Level Organization for Problem Solving Using Many Diverse Cooperating Sources of Knowledge." Advance Papers of the Fourth International Joint Conference on Artificial Intelligence, Tbilisi, Georgia, USSR (August 1975), p. 483-490.

[Erman, et al. 1980]
Lee D. Erman, Frederick Hayes-Roth, Victor R. Lesser, and D. Raj Reddy. "The Hearsay-II Speech Understanding System: Integrating Knowledge to Resolve Uncertainty." Computing Surveys (to appear).

[Feldman 1979]
Jerome A. Feldman. "High Level Programming for Distributed Computing." Communications of the Association for Computing Machinery, Vol. 22, No. 6 (June 1979), p. 353-368.

[Fox 1979a]
Mark S. Fox. "An Organizational View of Distributed Systems." Proceedings of the International Conference on Cybernetics and Society, Denver, Colorado (October 1979), p. 354-359.

[Fox 1979b]
Mark S. Fox. "Organization Structuring: Designing Large Complex Software." Technical Report CMU-CS-79-155, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Pennsylvania (December 1979).

[Hanson & Riseman 1978]
Allen R. Hanson and Edward M. Riseman. "VISIONS: A Computer System for Interpreting Scenes." Computer Vision Systems, Allen R. Hanson and Edward M. Riseman (Editors), Academic Press, New York, New York (1978), p. 303-333.

[Hewitt 1977]
Carl Hewitt. "Viewing Control Structures as Patterns of Passing Messages." Artificial Intelligence, Vol. 8, No. 3 (Fall 1977), p. 323-364.

[Kilmer, McCulloch, & Blum 1969]
William L. Kilmer, Warren S. McCulloch, and J. Blum. "A Model of the Vertebrate Central Command System." International Journal of Man-Machine Studies, Vol. 1 (1969), p. 279-309.

[Kimbleton & Schneider 1975]
Stephen R. Kimbleton and G. Michael Schneider. "Computer Communication Networks: Approaches, Objectives, and Performance Considerations." Computing Surveys, Vol. 7, No. 3 (September 1975), p. 129-173.

[Lenat 1975]
Douglas B. Lenat. "Beings: Knowledge as Interacting Experts." Advance Papers of the Fourth International Joint Conference on Artificial Intelligence, Tbilisi, USSR (August 1975), p. 126-133.

[Lesser & Corkill 1979]
Victor R. Lesser and Daniel D. Corkill. "The Application of Artificial Intelligence Techniques to Cooperative Distributed Processing." Proceedings of the Sixth International Joint Conference on Artificial Intelligence, Tokyo, Japan (August 1979), p. 537-540.

[Lesser & Corkill 1980]
Victor R. Lesser and Daniel D. Corkill. "Functionally-Accurate Cooperative Distributed Systems." IEEE Transactions on Systems, Man, and Cybernetics (to appear, 1980).

[Lesser & Erman 1977]
Victor R. Lesser and Lee D. Erman. "A Retrospective View of the Hearsay-II Architecture." Proceedings of the Fifth International Joint Conference on Artificial Intelligence, Cambridge, Massachusetts (August 1977), p. 790-800.

[Lesser & Erman 1979]
Victor R. Lesser and Lee D. Erman. "An Experiment in Distributed Interpretation." Proceedings of the First International Conference on Distributed Computing Systems, Huntsville, Alabama (October 1979), p. 553-571.

[Lesser, Reed, & Pavlin 1980]
Victor R. Lesser, Scott Reed, and Jasmina Pavlin. "A High-Level Simulation Testbed for Cooperative Distributed Problem-Solving." Technical Report, Department of Computer and Information Science, University of Massachusetts, Amherst, Massachusetts (in preparation).

[Mann 1979]
William C. Mann. "Design for Dialogue Comprehension." Seventeenth Annual Meeting of the Association for Computational Linguistics, La Jolla, California (August 1979).

[Nadler & Lawler 1977]
David A. Nadler and Edward E. Lawler, III. "Motivation: A Diagnostic Approach." Perspectives on Behavior in Organizations, Richard Hackman, Edward E. Lawler III, and Lyman W. Porter (Editors), McGraw-Hill, New York, New York (1977), p. 26-34.

[Nii & Feigenbaum 1978]
H. Penny Nii and Edward A. Feigenbaum. "Rule Based Understanding of Signals." Pattern-Directed Inference Systems, Daniel A. Waterman and Frederick Hayes-Roth (Editors), Academic Press, New York, New York (1978), p. 483-501.

[Nilsson 1980]
Nils J. Nilsson. Principles of Artificial Intelligence, Tioga Publishing, Palo Alto, California (1980).

[Noyce 1976]
Robert N. Noyce. "From Relays to MPUs." Computer, Vol. 9, No. 12 (December 1976), p. 26-29.

[Reed & Lesser 1980]
Scott Reed and Victor R. Lesser. "The Honey Bee Colony as a Metaphor for Distributed Computing." Technical Report 80-17, Department of Computer and Information Science, University of Massachusetts, Amherst, Massachusetts (1980).

[Rumelhart 1976]
David E. Rumelhart. "Toward an Interactive Model of Reading." Technical Report 56, Center for Human Information Processing, University of California, San Diego, California (1976). International Journal of Man-Machine Studies, Vol. 9(1977), p. 537-581.

[Sacerdoti 1977]
Earl D. Sacerdoti. A Structure for Plans and Behavior, Elsevier North-Holland, New York, New York (1977).

[Sacerdoti 1978]
Earl D. Sacerdoti. "What Language Understanding Research Suggests about Distributed Artificial Intelligence." Proceedings of the Distributed Sensor Network Workshop, Carnegie-Mellon University, Pittsburgh, Pennsylvannia (December 1978), p. 8-11.

[Sacerdoti 1979]
Earl D. Sacerdoti. "Problem Solving Tactics." Proceedings of the Sixth International Joint Conference on Artificial Intelligence, Tokyo, Japan (August 1979), p. 1077-1085.

[Scientific American 1977]
Scientific American. Special Issue on Microelectronics. Scientific American, Vol. 237, No. 3 (September 1977).

[Searle 1969]
J. R. Searle. Speech Acts: An Essay in the Philosophy of Language, Cambridge University Press, New York, New York (1969).

[Simon 1957]
Herbert A. Simon. Models of Man, John Wiley & Sons, New York, New York (1957).

[Simon 1969]
Herbert A. Simon. The Sciences of the Artificial, MIT Press, Cambridge, Massachusetts (1969).

[Smith 1978]
Reid G. Smith. "A Framework for Problem Solving in a Distributed Environment." Ph.D. Thesis, Technical Report STAN-CS-78-700, Computer Science Department, Stanford University, Stanford, California (December 1978).

[Soloway & Riseman 1977]
Elliot M. Soloway and Edward M. Riseman. "Levels of Pattern Description in Learning." Proceedings of the Fifth International Joint Conference on Artificial Intelligence, Cambridge, Massachusetts (August 1977), p. 801-811.